M345SC 2019 Homework 3

Due date/time: 21/3/2019, 23:59 GMT

Clarifications (edited 18/3/19)

18/3/19

- Part 2.1: In each question, you are asked to find the maximum growth of, $(Numerator)/e(t=0)$, where three different numerators have been specified. The denominator, $e(t=0)$, is defined in i) and is the same for ii) and iii). For iii), the numerator may be positive or negative. Larger growth corresponds to larger positive values of $\left[\sum_{i=0}^{N-1}\left(S_i(t=T)V_i(t=T)\right)\right]/e(t=0)$.
- Part 2.2: Say that the covariance matrix for this problem is defined as $C=AA^T$ with A defined appropriately. Then, the "total variance" is trace(C), and you can assume that "important" nodes will be successfully identified if 1) $\hat{I}$ is computed using an appropriate product involving A and a unit vector in $\mathbf{R}^M$ and 2) the variance of $\hat{I}$ approximates the total variance of the dataset as closely as possible.

.

14/3/19

- Part 1: The note for part 1 had a typo – the listed modules can be used for all questions in part 1.

13/3/19

- Part 1: You may use the time module.
- Part 1.3: The one-sided finite difference method to use is:

$$f_0'+\alpha f_1' = \frac{1}{h}(af_0+bf_1+cf_2+df_3)\,,$$
$$f_{N-1}'+\alpha f_{N-2}' = -\frac{1}{h}(af_{N-1}+bf_{N-2}+cf_{N-3}+df_{N-4})\,,$$

with $\alpha=3, a=-17/6, b=3/2, c=3/2, d=-1/6$. The "primes" were missing from the LHS in the lecture slides.

11/3/19

- Part 2: You may use scipy.linalg and scipy.integrate in part 2.
- Part 2.1 iii): The input *params* in *growth3* has too many elements, you should remove the 0 and use:

```
params=(2,2.8,1,1.0,0.5)
```

This project consists of two parts – in the first, you will be simulating/analyzing nonlinear waves, and in the second you will work further on the spread of infectious agents in a network using a model similar to the one from homework 2.

Getting Started: Template files have been added to the course repo in the hw3/ directory. You can also download the files directly from here: | p1_template.py | p2_template.py

Place these files in a folder, and create copies of the template files named *p1_dev.py* and *p2_dev.py* in this new directory. Ultimately, the final files that you submit should be titled *p1.py* and *p2.py*. Browse through the *dev* file; there are a few function headers, and you will have to add code for each of these functions as described below. First, however, you should *add your name and 8-digit college id to the docstring at the top of each file.* Please make

regular backups of your files. You are allowed to make multiple submissions on blackboard (the last one before the deadline will be marked).

Part 1. (10 pts)

Consider the following model for the evolution of nonlinear waves:

$$\frac{\partial g}{\partial t} + \beta \mathcal{N}(g) = \alpha \frac{\partial^2 g}{\partial x^2} + g,$$

where $g(x,t)$ is a complex periodic function with period, $L = 100$, α and β are complex model parameters, and $\mathcal{N}(g)$ is a nonlinear function. A mostly-complete function (*nwave*) for computing solutions to this model has been provided in *p1_template.py*. Starting from a provided initial condition, the solution is marched forward in time using *odeint* which calls *RHS*. *RHS* is incomplete.

1) Complete *RHS* so that it uses discrete Fourier transforms to compute $\frac{\partial^2 g}{\partial x^2}$. Code has been provided which you can use to display the computed solution.

2) Analyze and compare the simulation results for the following two cases, A: $\alpha = 1 - 2i, \beta = 1 + 2i$ and B: $\alpha = 1 - i, \beta = 1 + 2i$. There will be an initial transient as the solution adjusts away from the initial condition which should be discarded (say, for t<50). The input parameters Nt,Nx, and T may need to be varied as you develop your analysis. You should focus on the most energetic wavenumbers and frequencies and also carefully consider if/to what degree each case is chaotic. Your analysis should be structured around a few figures which are generated by code placed in the function, *analyze*. Save these figures and submit them with your codes. Add the corresponding discussion to the docstring of the function.

3) Consider whether a compact finite difference method would be superior to Fourier differentiation for nonlinear waves such as those generated by this model. Critically compare DFT-based calculation of $\frac{\partial g}{\partial x}$ with a tridiagonal finite-difference (fd) method with coefficients,

$$\alpha = 3/8, \beta = 0, a = 25/16, b = 1/5, c = -1/80.$$

(see the general form of implicit fd schemes introduced in lecture 16, **these coefficients are unrelated to the parameters in the model PDE above**) At the boundaries ($x = 0, x = L - h$), use the one-sided 4th order scheme from lecture. Take simulation results for case B at t=100 as "test waves" for determining which of the two methods is superior. You should generate a few figures to support your conclusion and add a discussion to the docstring of *wavediff* which carefully describes both your reasoning and the contents of your figures. The code in *wavediff* should generate these figures, and you should also save the figures and submit them with your codes.

Notes for Part 1: i) You may use time, numpy, matplotlib, and scipy for part 1. Do not use any other modules without the instructor's permission.

ii) If your code for 1.1 is not working, you may use a finite difference calculation of the 2nd derivative instead for the purpose of completing 1.2 and 1.3, though you will then only receive partial credit for 1.3.

iii) Add code to the name==main portion of the *p1_dev.py* to call *analyze* and *wavediff* and generate the figures you are submitting.

Part 2. (10 pts)

2.1 Here, you will consider a model for the spread of small perturbations to the number of *S*, *I*, and *V* cells in a network. The model equations are:

$$\frac{dS_i}{dt} = \alpha I_i - (\gamma + \kappa) S_i + \sum_{j=0}^{N-1} (F_{ij} S_j - F_{ji} S_i)$$
$$\frac{dI_i}{dt} = \theta S_i - (\kappa + \alpha) I_i + \sum_{j=0}^{N-1} (F_{ij} I_j - F_{ji} I_i)$$
$$\frac{dV_i}{dt} = \kappa V_i - \theta S_i + \sum_{j=0}^{N-1} (F_{ij} V_j - F_{ji} V_i)$$

Now, θ is a constant parameter like α, γ, κ, and τ. The *flux matrix*, F_{ij}, is defined as before, $F_{ij} = \tau \frac{q_i A_{ij}}{\sum_{k=0}^{N-1} q_k A_{kj}}$ where A_{ij} is the adjacency matrix of the network (which is unweighted and undirected), q_i is the degree of node *i*. You may assume that each node has at least one link to another node, and with this modified model, S, I, and V may take on positive or negative values.

i. We are interested in the growth of perturbations, and we define a "perturbation energy" as, $e(t) = \sum_{i=0}^{N-1} \left(S_i^2(t) + I_i^2(t) + V_i^2(t) \right)$. Complete *growth1* in *p2_dev.py* so that the maximum growth, $e(t = T)/e(t = 0)$, is computed with the model parameters, *T*, and a networkx graph, G, provided as input. The function should return both the growth and the initial condition that generates the growth. The initial condition should be returned as a 3N-element array. See the function documentation for further details.

ii. Now, complete *growth2* so that it finds the maximum growth of $\left[\sum_{i=0}^{N-1} I_i^2(t = T) \right] / e(t = 0)$. Aside from this change, the functionality should be the same as in *growth1*.

iii. Complete *growth3* so that it finds the maximum growth of $\left[\sum_{i=0}^{N-1} (S_i(t = T) V_i(t = T)) \right] / e(t = 0)$. You do not need to return the corresponding initial condition. Aside from these changes, the functionality should be the same as in *growth1*.

2. You are now tasked with identifying the most "important" network nodes based on measurements at one time of I_i for *M* different organisms in the early stages of infection (each with N-node networks). The data is provided as an *N* x *M* matrix, and the aim is to construct an array, $\hat{I}_i$, which best approximates the total variance of I_i in the dataset. Here, the variance corresponds to variations across the *N* nodes in a network, not variations in time. Complete the function *Inew* so that it computes and returns $\hat{I}_i$ as an N-element array (or list).

Test data for this question can be downloaded here: | q22test.txt This data is saved as a 100 x 4 matrix which can be loaded using *np.loadtxt('q22test.txt')*.

For each question above, add a concise description of your approach to the docstring of the appropriate function.

Note for Part 2: You may use numpy, networkX, scipy.linalg, and scipy.integrate for part 2. Do not use any other modules without the instructor's permission.

Further Notes:

1. Marking will consider both the correctness and efficiency of your code as well as the soundness of your analysis.

2. 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.

3. In order to assign partial credit, markers must be able to understand how your code is organized and what you are trying to do.

4. You are allowed to discuss general aspects of Python and the problem statement with others, however you should not discuss your particular implementations/solutions with other students or show your code to other students.

5. You may create additional functions as needed. Please do not modify input/output from provided functions without permission.

Submitting the assignment

To submit the assignment for assessment, go to the course blackboard page, and find the link for “Homework 3” Click on the link for Homework 3, and then 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, *p1.py*, *p2.py*, *fig1.png*, *fig2.png*, etc… . Finally, click “Submit”.