Scientific Computation 2020 Project 3 (updated 13/3)¶

New Project Deadline:

Due date/time: 27/3/2020, 18:00 GMT

Clarifications (edited 20/3/20)¶

part 1.3 (20/3): You should design your method specifically for *data3.npy* You do not need to consider the effectiveness of your method for generic or alternate datasets.
part 2.1 (20/3): You do not need to use output from *microbes* as input for *newdiff*. You do not need to modify *microbes* to use the method you develop in *newdiff*.
part 2.2 (20/3): Please note that the input default parameters for *microbes* were set simply to help you get started. You will almost certainly need to vary *T* and *Nt* to fully explore the dynamics in the given parameter space. You are also welcome to expand this parameter space if it helps you with your work, but this is not at all required.
part 1.2i) (15/3): The time increment for *data2.npy* should be $dt = \pi/80$, not 1 as initially stated below. If you have completed the question with $dt = 1$, that is fine, but please add a sentence to your pdf stating the *dt* that you have used (it may also then be useful to look at r0 in the range [1,100] rather than [1,5]). The data2 dataset corresponds to times:

```
t=np.arange(0,20*np.pi/80,np.pi/80)
```

> Computing the Hankel function of the first kind, $H_m^{(1)}(z)$, may produce NaNs when *m* is large. If an accurate solution requires calculations with such *m*, then consider constructing an approximate solution discarding such terms and provide a very brief comment on the quality of your approximation.
>
> Note that it may be helpful to use *complex conjugates* of Hankel functions of the first kind (which are also Hankel functions of the first kind for real-valued orders and arguments) in order to construct real-valued results. An outgoing wave has the form, $H_m^{(1)}(\omega r)exp(-ict)$ with ω, r, c all positive, and the complex conjugate, $\overline{H_m^{(1)}(\omega r)}exp(ict)$ is also an outgoing wave.

part 1 and part 2 (13/3): A number of provided functions have default values set for input parameters. Unless explicitly instructed otherwise, there is no requirement to use these and other values should be used when calling these functions as needed.
part 1 (13/3): You have been given files (*r.npy*, *theta.npy*) which contain the polar coordinates where data is provided.
part 1 (13/3): For both *data2.npy* and *data3.npy*, the data is provided at equispaced points in time, and you may assume that the time increment is 1 for data3.

part 1.1 (13/3): The “Add code here” and “modify” comments in *repair1* are guidance for what should be done in *repair2*.
part 1.2i) (13/3): The results should be computed for the times and polar angles at which data has been provided.
part 2.1 (13/3): The data to be differentiated is defined on the interval $[0, L]$ with $i = 0$ corresponding to $x = 0$ and $i = N - 1$ corresponding to $x = L$. *BC1* does not apply to *f* for this question.
part 1.3 (9/3): The functions should work with the data in *data3.npy*, not *data2.npy*.

Getting Started: Template files have been added to the course repo in the project3/ directory. You can also download the files directly from here: | part1_template.py | part2_template.py | project3_template.tex

Place these files in a folder, and create copies of the template files named *part1_dev.py* and *part2_dev.py* in this new directory. Ultimately, the final files that you submit should be titled *part1.py*, *part2.py*, and *project3.pdf*. Browse through the *dev* file; there are a few function headers, and you will have to add code for some of these functions as described below. First, however, you should *add your 8-digit college id to the docstring at the top of each python 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).

You are not required to use latex or the provided latex template. Any pdf with a simple, clear presentation is perfectly fine.

Part 1.¶

In this part, you will work with model sea-surface height data files. The data provides a view of ocean dynamics in the vicinity of a circular island. we use polar coordinates (r, θ) with $1 \leq r \leq 5$ and $0 \leq \theta \leq 2\pi$. All of the needed data files for part 1 have been collected in:

project3.zip.

Unzipping this file will extract three numpy data files, *data1.npy*, *data2.npy*, and *data3.npy*. Please be aware that the third of these files is fairly large (~80 MB).

Each of these three files can then be loaded in numpy using *np.load(‘datax.npy’)*. If you have any difficulty extracting/loading the data, please contact the instructor.

1) (3 pts) *data1.npy* contains sea-surface data for one instance in time. As the data was being processed, it became corrupted and portions of the measurements are missing. A value of *-1000* has been placed at all locations where the data was lost. The matrix stored in the *npy* file can be visualized using the *hfield* function in the part 1 template file. Here, you will use a matrix factorization approach to build an approximate dataset which “fills in” the missing data. Say that the provided data is stored in a *a x b* matrix, *R*, while the approximate full matrix is given by $\tilde{R} = AB$ where *A* is *a x p*, *B* is *p x b* and *p* must be specified. The goal is to construct *A* and *B* which minimize the following cost, *c*:

$$c = \sum_{i,j \in K} \sum (R_{ij} - \tilde{R}_{ij})^2 + \lambda \left[\sum_{i=1}^{a} \sum_{j=1}^{p} (A_{ij})^2 + \sum_{i=1}^{p} \sum_{j=1}^{b} (B_{ij})^2 \right]$$

Here, K is the set of "coordinates" in R with valid data. The magnitudes of the elements in A and B are constrained using a l_2-regularization term. The strength of this constraint is set by λ, and larger values of λ should lead to smaller values for elements in A and B. The cost is minimized through an iterative approach which is based on the requirement that the gradient of the cost with respect to each matrix element should be zero. For example, for A we have:

$$\frac{\partial c}{\partial A_{mn}} = -2 \sum_{i,j \in K} \sum (R_{ij} - \tilde{R}_{ij}) \frac{\partial \tilde{R}_{ij}}{\partial A_{mn}} + 2\lambda A_{mn} = 0.$$

Code for updating the elements of A (one element at a time) has been provided in the function *repair1*. You will have to critically assess this code and improve it if tangible inefficiencies are present. Furthermore, you will have to add code that efficiently updates the elements of B (one element at a time). Place the complete efficient implementation in *repair2*. Add a brief (3-4 sentence) description of your work to your *pdf*. If your code has been completed correctly, it should be possible to produce an image (say, using *hfield*) that corresponds to a dataset whose repair has been at least partially successful. Include such an image in your *pdf* along with a concise (2-4 sentence) discussion which provides the parameters used to create the image ($\lambda, p, \ldots$) and comments on if/how the result could be improved further. A broad parameter variation study should be avoided.

2) (4 pts) i) Calm weather can lead to results like those found in *data2.npy* where the heights are oscillating sinusoidally in time. For reference, it is useful to compare such observations to a computed field of outgoing linear waves generated by the fluctuations at the island boundary. These waves are governed by the equations,

$$\frac{\partial^2 h}{\partial t^2} - \left(\frac{\partial^2 h}{\partial r^2} + \frac{1}{r}\frac{\partial h}{\partial r} + \frac{1}{r^2}\frac{\partial^2 h}{\partial \theta^2} \right) = 0$$
$$h(r=1, \theta, t) = f(\theta, t),$$

and $f(\theta, t)$ should be extracted from the data file. Furthermore, the radial dependence of the solution should take the form of a linear superposition of Hankel functions of the first kind, $H_m^{(1)}(\omega r)$ (see *scipy.special*). Hankel functions satisfy the Bessel equation. Complete the function, *outwave*, so that it computes the solution to these equations at a specified radial position, $h(r_0, \theta, t)$. Add a brief (3-4 sentence) description of your approach to your pdf.

ii) A much more complicated dataset is stored in *data3.npy*. There, a 3-D array contains data at 119 equally spaced points in time. Analyze and compare the height dynamics represented by this dataset on the lines, $\theta = \pi/4$, $\theta = 3\pi/4$ and $\theta = 5\pi/4$. Place the code used in your analysis in the function, *analyze1*, and add a discussion of your findings along with supporting figures to your *pdf*.

3) (3 pts) Datasets are frequently considerably larger than *data3.npy*, so it can be useful to construct a "reduced" dataset which requires less memory but which retains the key features of the original data. Design and implement a method that constructs a reduced dataset in the function, *reduce*. The function receives the

3-D numpy array (*H*) stored in *data3.npy* as input and returns one or more numpy arrays (and/or lists) from which the dataset (or an approximation to the dataset) can be reconstructed. Complete *reconstruct* so that it receives the output from *reduce* as input, and constructs a numpy array which retains the key features of *H* (and has the same shape). Add a clear and concise description of your method to your *pdf*. Include 1) an explanation of how and to what degree "key features" are retained in the reconstructed array and 2) an estimate of how much memory is saved.

Part 2.¶

Consider the following model for the competitive dynamics of two microorganisms:

$$\begin{aligned}
\frac{\partial f}{\partial t} &= \frac{\partial^2 f}{\partial x^2} + f(1-f) - \frac{fg}{f+\phi}, \\
\frac{\partial g}{\partial t} &= \frac{\partial^2 g}{\partial x^2} + \frac{\kappa fg}{f+\phi} - \mu g, \\
BC1: \frac{\partial f}{\partial x}\Big|_{x=0} = \frac{\partial f}{\partial x}\Big|_{x=L} = \frac{\partial g}{\partial x}\Big|_{x=0} &= \frac{\partial g}{\partial x}\Big|_{x=L} = 0, \\
0 \le x &\le L,
\end{aligned}$$

where $f(x,t)$ and *g(x,t)* are the concentrations of each species while ϕ, κ, and μ are model parameters. For simplicity, we are only considering dynamics in one spatial dimension.

A function (*microbes*) for computing solutions to this model has been provided in *part2_template.py*. Starting from a provided initial condition, the solution is marched forward in time using *odeint* which calls *RHS*. *RHS* uses 2nd-order centered finite differences to approximate the spatial derivatives in the equations above. Code has also been provided which you can use to display the computed solution.

1) (4pts) Consider the following compact finite difference scheme for computing the second derivative:

$$\begin{aligned}
\alpha f''_{i-1} + f''_i + \alpha f''_{i+1} &= \frac{1}{h^2}\left[\frac{c}{9}(f_{i+3} - 2f_i + f_{i-3}) + \frac{b}{4}(f_{i+2} - 2f_i + f_{i-2}) + a(f_{i+1} - 2f_i + f_{i-1})\right] \\
f''_0 + 10f''_1 &= \frac{1}{h^2}\left[\frac{145}{12}f_0 - \frac{76}{3}f_1 + \frac{29}{2}f_2 - \frac{4}{3}f_3 + \frac{1}{12}f_4\right] \\
f''_{N-1} + 10f''_{N-2} &= \frac{1}{h^2}\left[\frac{145}{12}f_{N-1} - \frac{76}{3}f_{N-2} + \frac{29}{2}f_{N-3} - \frac{4}{3}f_{N-4} + \frac{1}{12}f_{N-5}\right] \\
i &\in \{0, 1, 2, \ldots, N-1\},
\end{aligned}$$

where h is the grid spacing (for an equispaced grid). The first equation is used for all *i* except *i=0* and *i=N-1*.

i) Complete *newdiff* so that it efficiently implements this scheme to compute the 2nd derivative of a function using the array provided as input. The function should be assumed to be periodic, $f(x-L)=f(x)=f(x+L)$. The coefficients, α, a, b, c , have been included in *newdiff*.

ii) Use computations to analyze the accuracy and cost of the method and the efficiency of your implementation. Critically compare the suitability of this method and the 2nd-order centered scheme for multiscale problems. Carefully explain if/when the compact finite difference scheme should be selected. Add your discussion and accompanying figures to your pdf. Place the code used for the analysis in *analyzefd*

2) (4 pts) Analyze and compare results for simulations with $\phi=0.3, L=1024, Nx=1024, \mu/\kappa=0.4$, and with κ in the range, $1.5 \leq \kappa \leq 2$. Typically simulations contain an initial transient as the system responds to the initial conditions followed by a settled dynamical state. Discard the transient in your analysis, and focus on the global qualitative dynamics (e.g. the system is steady (no time-dependence), simple sinusoidal oscillations in time, …). For the cases $\kappa=1.5, 1.7, 2$ carefully analyze if/to what degree the dynamics correspond to chaos. Also consider if the dynamics in *data3.npy* from part 1 are chaotic. Add the code used in your analysis to *dynamics*, and add the analysis with accompanying figures to your *pdf*.

Note: It is fine for simulations to take several minutes, but if they are requiring an hour or more, consider reducing the timespan and/or number of grid points.

3) (2 pts) This model contains the steady spatially-homogenous solutions,

$$f_0 = \frac{\mu\phi}{\kappa-\mu},$$
$$g_0 = (1-f_0)(\phi+f_0)$$

Consider the dynamics of small-amplitude spatially-sinusoidal perturbations to this steady state. In your *pdf*, carefully explain how you would investigate these dynamics both for finite time spans, $0 \leq t \leq T$ for some sensible *T*, and for long time, $t \to \infty$.

Further Notes:

1. Marking will consider both the correctness and efficiency of your code as well as the soundness of your analysis. Code used to produce figures or generate input for functions is unlikely to be closely examined for efficiency.

2. All figures created by your code should be well-made and properly labeled.

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

5. You may use numpy, scipy, matplotlib, time, and timeit as needed in this assignment. Do not use any other modules without the instructor’s permission.

6. Please be sensible with the time you allocate to the open-ended aspects of the assignment and to the assignment overall. You should aim for your report to be 10 pages (or less) in length with 15 or less figures. These numbers are provided as guidance and marks will not be deducted for exceeding them.

Submitting the assignment¶

To submit the assignment for assessment, go to the course blackboard page, and find the link for “Project 3” Click on “Project 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, *part1.py*, *part2.py*, *project3.pdf* . Finally, click “Submit”.