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,θr,θ) with 1≤r≤51≤r≤5 and 0≤θ≤2π0≤θ≤2π.

1) *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 R~=ABR~=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*:

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

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 l2l2-regularization term. The strength of this constraint is set by λλ, and larger values of λλ 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:

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

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 (λ,p,...λ,p,...) 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(θ,t)f(θ,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(1)m(ωr)Hm(1)(ωr) (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(r0,θ,t)h(r0,θ,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, θ=π/4θ=π/4, θ=3π/4θ=3π/4 and θ=5π/4θ=5π/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:

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

where f(x,t)f(x,t) and *g(x,t)* are the concentrations of each species while ϕϕ, κκ, and μμ 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:

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

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)f(x−L)=f(x)=f(x+L). The coefficients, α,a,b,cα,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 ϕ=0.3,L=1024,Nx=1024,μ/κ=0.4ϕ=0.3,L=1024,Nx=1024,μ/κ=0.4, and with $\kappa\kappa$ in the range, 1.5≤κ≤21.5≤κ≤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 κ=1.5,1.7,2κ=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≤t≤T0≤t≤T for some sensible *T*, and for long time, t→∞t→∞.