ME535 Winter 2023: Homework 4

This is an individual assignment, so the answers you submit must be your own work. Before you upload the file to Canvas, execute the final version of the file to make sure that it runs properly and produces your intended answers. Insert your code to implement the functions as specified (without changing function names and/or arguments). Do not import from libraries other than those that are already imported into the template (such as numpy).

> Please DO NOT use this file as the basis for your homework submission.

> Add your code to `hw4_template.py` to create the file that you submit.

1) In class, we will see that that analytic and Fourier Transform techniques are applicable for "nice" geometries whose boundaries lie on constant coordinate curves. For this problem, you will show that finite difference methods can be applied to more general geometries. (This problem is constructed so regularly sample grid points lie on the boundary. For boundaries that do not align nicely with a regular grid, only a slightly more complicated formulation is required.)

In particular, solve Laplace's equation $u_{xx} + u_{yy} = 0$ to determine the steady-state temperature distribution $u(x, y)$ in a flat triangular plate with insulated faces (to ensure no heat flow in the z-direction) and vertices at $[[0, 0], [0, 1], [1, 0]]$.

The boundary conditions are as follows:

- On the edge along the y-axis, $u(0, y) = y$.
- On the edge along the x-axis, $u(x, 0) = x^2$.
- On the hypotenuse (lying on $y = 1 - x$), $u(x, 1 - x) = 1$.

Choose an initial guess of the temperature distribution (e.g. $u \equiv 0$), then interatively update (using the 5-point stencil scheme on a uniform grid with spacing 0.05) until the maximum change caused by an update is less than `1e-2`.

Implement this solution scheme by filling in code for the functions `initialize_1`, `update` and `iter_solve`. `initialize_1` should do what its name implies by imposing the boundary conditions on an input grid. `update` should do a single update step of the values at the grid points internal to the triangle. `iter_solve` should create an array to represent the relevant grid, repeatedly call `update` until the termination condition is met, and return the array of computed values (for plotting etc.).

2) In class, we consider applying finite difference methods to the heat equation and numerically determine a stability criteria relating the spatial and temporal stepsizes. Here you do likewise with the wave equation.

Apply a central difference approximation to obtain a discrete version of the wave equation $u_{tt} = c^2 u_{xx}$ on the interval $-100 < x < 100, 0 < t < 35$ with wavespeed $c = 1$ and gridspacing $\Delta x = 0.4$ (i.e. with 501 points uniformly distributed across the domain). The boundary conditions are $u(0, t) = u(1, t) = 0$, and the initial conditions are $u(x, 0) = 1 - |x/10|$ on $-10 < x < 10$ and $u(x) = 0$ elsewhere (corresponding to displacing the central portion of a taut string into a "V"-shape and releasing it from rest).

Implement functions `initialize_2` (to set up the 2D numpy array for storing the solution), `single_step` (to compute the displacement at the next timestep) and `step_solve` (to compute a simulation over the discrete space-time grid and return the simulation as a 2D numpy array).

Using your functions, compute solutions for timesteps $\Delta t = 0.999 \Delta x$ and $\Delta t = 1.001 \Delta x$.

Based on your numerical solutions, comment on the stability of this numerical solution method. How do your results relate to the CFL condition?

> Note that the central difference approximation of the second time derivative requires displacement data at 2 consecutive times. Here we have chosen a simple case where the string is released from rest (with velocity zero so the displacement is not changing initially) so you can set both the initial and fictitious previous displacements to correspond to the specified initial displacement.