# MF850: Advanced Computational Methods

## Problem Set 2

If you run into problems, try to verify each component of your code on simple cases to make sure they work independently.

**Due date:** Monday October 10, 2022, at noon Boston time.

**Instructions:** You submit on Blackboard. You may solve this assigment in groups of two. A submission is constituted by answers to the problems along with the code used. A file called `hw2.jl` should contain your code, or your entry point if you separate your code into multiple files. This file should run without errors from a fresh Julia instance/REPL. The code must be formatted by loading the package `JuliaFormatter` and running `format` on the submission files. In other words, submissions in notebook format are not accepted (but you may of course develop in them before creating the submission).

*Hint:* Running `format(".")` runs the formatter on every `.jl` file recursively in the current directory.

Please contact the instructor or a TA if you have questions regarding these instructions or if you find the problem formulation unclear.

**Problem 2.1** Consider the following equation:

$$v'' + v' = x^2 \quad \text{in } [0,1]. \tag{1}$$

(a) Numerically solve (1) with the Dirichlet boundary conditions

$$v(0) = 0 \quad \text{and} \quad v(1) = -1.$$

*Hint:* Solve it analytically to verify that the solution is correct.

(b) Estimate the order of convergence as the grid size decreases.

(c) Solve (1) analytically and plot the errors as a function of grid size in such a way that you can see the order of convergence in the figure.

(d) Numerically solve (1) with the Robin boundary conditions

$$v(0) + v'(0) = 1 \quad \text{and} \quad 2v(1) + v'(1) = 0.$$

**Problem 2.2** Solve

$$\Delta v = 0 \quad \text{in } \mathcal{D} = (-1,1) \times (-1,1)$$

with the Dirichlet boundary condition

$$v(x,y) = \cos(\pi(x + y^2)) \quad \text{on } \partial\mathcal{D}.$$

Make a surface plot of the solution (using the `surface` command).

What value do you find for $v(-0.5, 0)$?

*Note:* If you want a fine discretization, you must use a sparse matrix for the linear system. Julia lets you do this with the built in `SparseArrays` module.