18.337 Spring 2026 Homework 1
HW is due Wednesday February 11th, 11:59pm, but you should have at least parts 1 and 2 done earlier.

In this homework we will build a physics-informed neural network in Julia from scratch in order to solve the 2D Poisson equation.

A reference that might be useful is https://ieeexplore.ieee.org/document/712178

Note that no libraries should be used except for LinearAlgebra. This includes no libraries for automatic differentiation, neural networks, optimization, or machine learning.

Part 1

Define a 3-layer multilayer perceptron that takes in 2 values and returns a single value. This is a neural network NN(x,y) = z. Make two hidden layers and choose the size of that hidden layer size to be both size 16 and use tanh as the activation function for all layers except the last which should just be a linear layer (identity activation function)

This should be given as a single Julia function. For the definition of the multilayer perceptron, see the lecture notes.

https://book.sciml.ai/notes/03-Introduction_to_Scientific_Machine_Learning_through_Physics-Informed_Neural_Networks/

Part 2

Define the function NN_derivative(x,y,dx,dy) that computes the Jacobian of the NN(x,y) function, i.e. the matrix of derivatives, in the same manner as automatic differentiation. I.e. create the function that would be equivalent to inlining all forward-mode automatic differentiation calculations for the function call NN(x+eps*dx, y+eps*dx) = z + eps*dz. Do not use automatic differentiation or build an automatic differentiation library, instead write the functions that directly computes the output (z, dz).

For example, if f(x,y)=sin(xy), f_derivative(x,y,dx,dy) could be cos(xy)*(y*dx+x*dy)

Part 3

Write a function that performs gradient descent on a function f in order to achieve a minimum, i.e. gradient_descent(f, df, u0, alpha) which starts from u0 and uses the update relation u_{n+1} = u_n - df(u_n) * alpha with a chosen learning rate alpha, where df is the gradient of the function f with respect to its inputs. Demonstrate this function works by finding the minimum of f(x,y) = x^2 + y^2 starting with the guess u0 = [1.0,1.0].

Part 4

Solve the Poisson equation u_xx + u_yy = - sin(pi*x) * sin(pi*y) with Dirchlet zero boundary conditions on [0,1] x [0,1] (i.e. u(0,y) = u(1,y) = u(x,0) = u(x,1) = 0) using a physics-informed neural network. Follow the lecture notes for how to do the process. Use the neural network defined in Part 1, choose randomized initial parameters for the neural network using the randn function for the weight matrices W and start with zero initializations for the bias vectors b, and use the gradient descent function from Part 3 with the Jacobian function defined in Part 2.

Hint: This can be simpler to solve if you define a function approximator that enforces the boundary conditions to always be satisfied. Remember for ODEs we enforced the boundary condition via g(t) = u0 + t*NN(t), is there something similar that can be done for this example?