036049: Applied ML4SE, Winter 2023/2024
Homework 4

Due: March 19, 2024

This homework is related to Physics Informed Neural Networks (PINNs) being discussed in the class.

Computation

1. In this exercise, you will modify a PINNs code to solve the 1D unsteady diffusion equation to instead solve the 1D Allen-Cahn differential equation[1]:

$$u_t - 0.0001u_{xx} + 5u^3 - 5u = 0, \quad t \in [0,1], x \in [-1,1], \tag{1}$$

where u is the dependent variable to be solved using PINNs. The problem is subject to following initial and boundary conditions:

$$\begin{aligned} \text{ICs:} \quad & u(0,x) = x^2\cos(\pi x) \\ \text{Boundary condition-1:} \quad & u(t,-1) = u(t,1) \\ \text{Boundary condition-2:} \quad & u_x(t,-1) = u_x(t,1) \end{aligned} \tag{2}$$

   (a) Rather than provide you with the code to simply run (too easy) or make you write your own code from scratch (too hard), it is more educational for you to type in the code yourself line-by-line from a source. Specifically, we have prepared a video tutorial which presents the code to you line-by-line. After you have your code file ready and working for the **1D unsteady diffusion equation**, modify the diffusion coefficient to consider the following values: 0.01, 0.1, and 1.0 and run the code to compare the final time result graphically for each case.

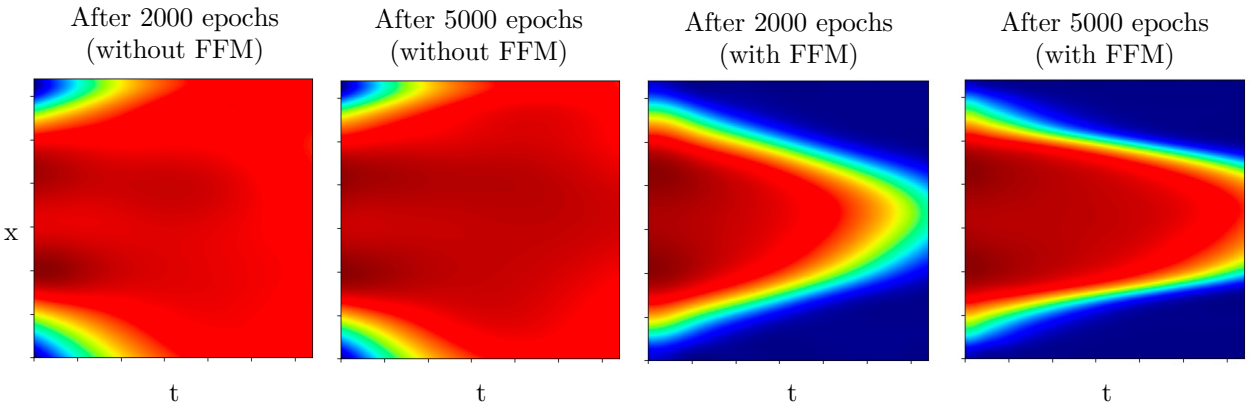
   (b) Next, modify the *initial conditions*, *boundary conditions*, and *the differential equation residual*, etc., to solve the above-described **1D Allen-Cahn equation**. The program uses a standard Multi-Layer Perceptron (MLP) style neural network to map a relationship between the input coordinates (x,t) and the output u. Use three hidden layers containing 64 units each, Tanh activation, Adam optimizer, learning rate of 0.001, and `Nx=Nt=128`.

   (c) Run the network for 20,000 epochs and report: (i) Four loss history curves - equation loss, boundary loss, initial condition loss, and total loss, (ii) The final solution using the `plt.imshow()` command at the end of 5,000, 15,000, and 20,000 epochs. (If you have a desktop computer or are able to run longer, feel free to run the code up to 100,000 epochs; it might take quite a few hours). Multiply the initial condition loss, boundary condition loss, and differential equation loss by 20, 20, and 1, respectively, to compute the total loss before the backward pass - this helps convergence.

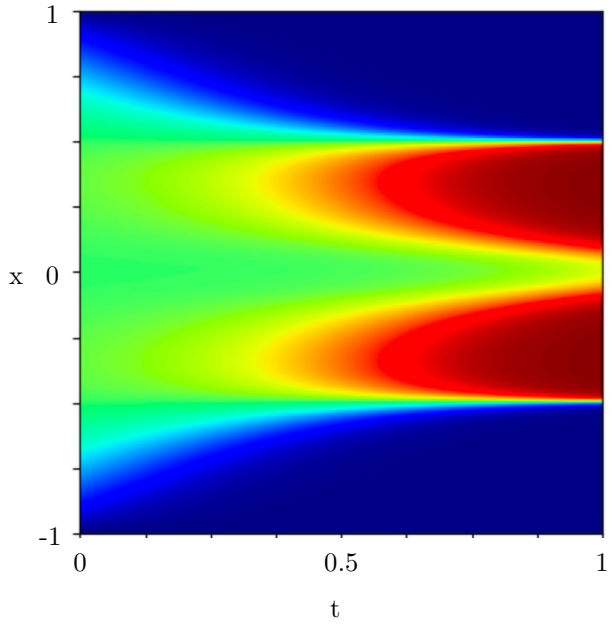
[1] The Allen–Cahn equation (after John W. Cahn and Sam Allen) is a reaction–diffusion equation of mathematical physics which describes the process of phase separation in multi-component alloy systems, including order-disorder transitions.

(d) Use the Fourier Feature Mapping (FFM) algorithm described in the tutorial video and the code provided to you, to transform your input coordinates into a series of sinusoidal mappings of vector length equal to the length of adjacent downstream hidden layer. Repeat exercise (b) and run the program with FFM. Report your loss history and solution again and comment on them. Make sure to use a standard deviation parameter equal to '2.0' in the FFM class definition.

Following are the images of solution for your reference, after 2000 and 5000 epochs with and without FFM:



Following is the solution with `Nx=Nt=200`, Number of hidden layers = 4, Dimension of each hidden layer = 256, with Fourier Feature Mapping, after 100,000 epochs ran on NVIDIA A100 GPU at CFDLAB Technion in ≈71 minutes:



Submission Guidelines

Upload a PDF file to the moodle page for grading with screenshots of your code, plots, and suitable explanations written as a short report.