ECE4960 Assignment 5 Report

In this project, our team plan to solve 1 dimensional and 2 dimensional PDE of heat function using Finite Difference Method. The heat equation we choose is

$$\frac{\partial u}{\partial t} = \kappa \frac{\partial^2 u}{\partial x^2}$$

Equation.1D Heat Equation

$$u_t = c^2 \nabla^2 u = c^2 (u_{xx} + u_{yy})$$

Equation.2D Heat Equation

Top-down organization

To solve the PDE problem, we first write a `PDESolver()` which can receive parameters and carry out different operations. Because of that the parameter matrix structure will be different in different dimensions, we need to specify in which space dimension (1D or 2D) we want to solve the PDE problem, then call different `ParameterMatrix`() functions to build up the parameter matrix. Based on the parameter matrix we obtain, we can call `NewT` () to update the temperature distribution by solving matrix problem.

Design details

(1) In `PDESolver()`, we can choose in which space dimension (1D or 2D) as well as by which method (Forward/Backward/Trapezoidal Euler) we want to solve the PDE problem.
(2) If the problem is 1D, `ParameterMatrix`_1D() will be called to construct the parameter matrix. Similarly, `ParameterMatrix`_2D() will be called if the problem is 2D. The main difference between these two constructors are the rough structure of the matrix.
(3) In every parameter matrix constructor, the matrix elements are different if different time discretization methods are chose (by `int choice`).
(4) Once the parameter matrixes are determined, we can update the temperature distribution by the formula $[\mathrm{A}]\lfloor n_{i,j+1} \rfloor = [\mathrm{B}]\lfloor n_{i,j} \rfloor$ (Forward Euler)) or $[\mathrm{B}]\lfloor n_{i,j+1} \rfloor = [\mathrm{A}]\lfloor n_{i,j} \rfloor$ (Backward/Trapezoidal Euler).

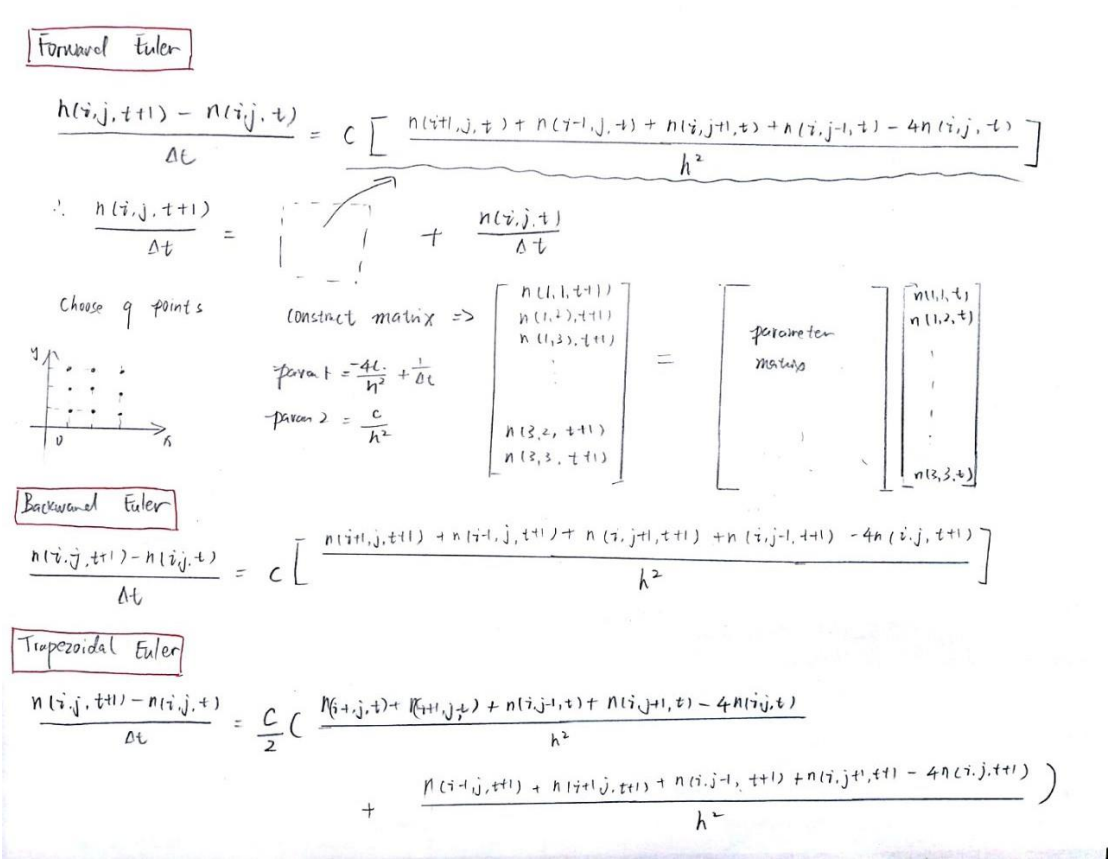


Fig. Formula

Code reuse

If we want to implement higher dimension solver, we only need to add another parameter matrix constructor which will realize different matrix structure.

Or if we want to use more methods to update the temperature distribution, we only need to add more switch cases and use the same matrix structure in the `ParameterMatrix`().

Validation and Testing

(1) validation

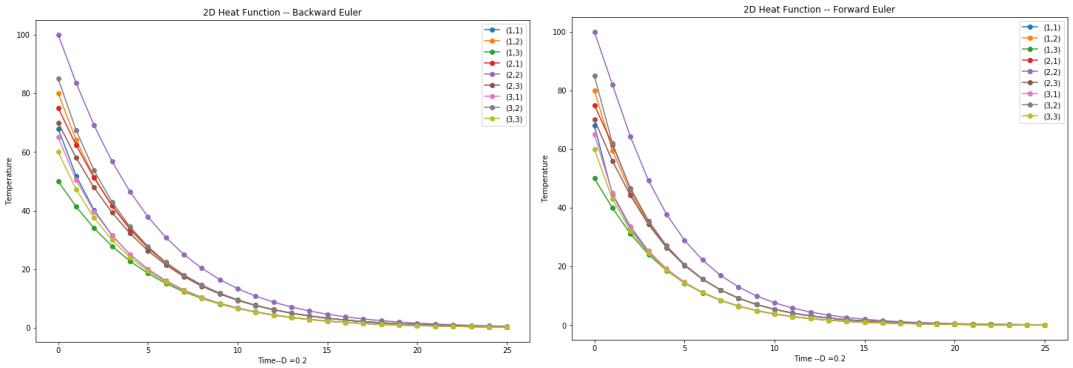
As for the validation of our PDE Solver, we implement Wilkinson Validation by comparing the temperature in different coordinates calculated through Forward Euler, Backward Euler and Trapezoidal Euler in FDM.

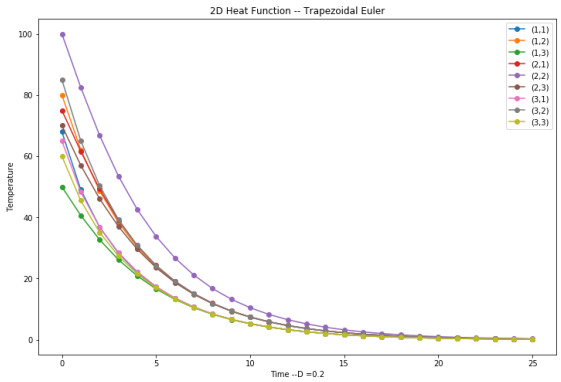
- One Dimensional heat equation

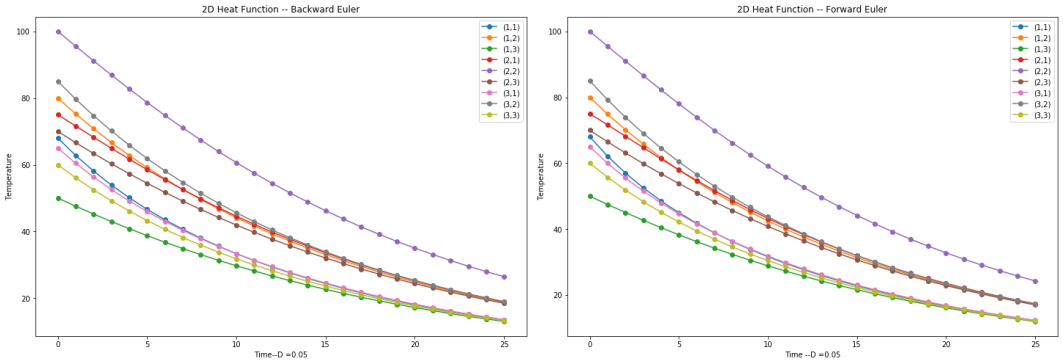
We compared the result from three different methods and then calculate the second norm. The second norm ranges from 10-1 to 10-2, which in turn shows that the 1D PDE solver could be correct. The exact value can be seen invalidation.txt.

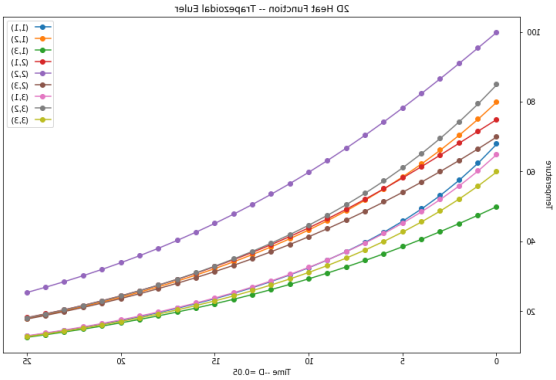
- Two Dimensional heat equation

We plot the variation of temperature in 9 different points with respect to time.
According to the plot, the variation trends are similar, and the curve could be almost the same with smaller C. The exact value can be seen in output.txt.









(2) Testing

In this part, we test all necessary helper functions.

- Validation and testing of matrix solver

We use iterative matrix solver this time. And we validate it using ill-conditioned matrix, then we test it in the case of knowing ground truth. The result shows that our matrix solver is robust and correct.

- Testing of parameter matrix construction

In this project, the main difference between one dimensional problem and two dimensional problem is how we construct our parameter matrix. Knowing the ground truth, we call these methods and then print corresponding matrix. The result shows that out matrix construction is correct.