The University of Manchester

COMP36212-EX2

Numerical Solution of the Heat Conductance Equation

Tianpeng Huang
ID: 10653872

March 24, 2023

1 Part1: Explicit of Heat Conductance Equation

1.1 Construct the finite difference equations

Let's start with heat conductance equation:

$$\frac{\partial^2 T(x,t)}{\partial x^2} = \frac{1}{k}\frac{\partial T(x,t)}{\partial t} \tag{1}$$

Construct the finite difference equations: use a central difference approximation for the second order spatial derivative, and forward difference approximation for the first order time derivative, taught by the PPT in class[1]:

$$\frac{\partial T(x,t)}{\partial t} = \frac{T_{i,j+1} - T_{i,j}}{\Delta t} \tag{2}$$

$$\frac{\partial^2 T(x,t)}{\partial x^2} = \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} \tag{3}$$

Substitute into Equ (1) and finite difference approximations for partial derivatives at node i, j:

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} = \frac{1}{k}\frac{T_{i,j+1} - T_{i,j}}{\Delta t} \tag{4}$$

It is now possible to describe explicit update for $T_{i,j+1}$ based on values at time j:

$$T_{i,j+1} = \frac{k\Delta t}{\Delta x^2}(T_{i-1,j} + T_{i,j}(\frac{\Delta x^2}{k\Delta t} - 2) + T_{i+1,j}) \tag{5}$$

1.2 Computational molecule

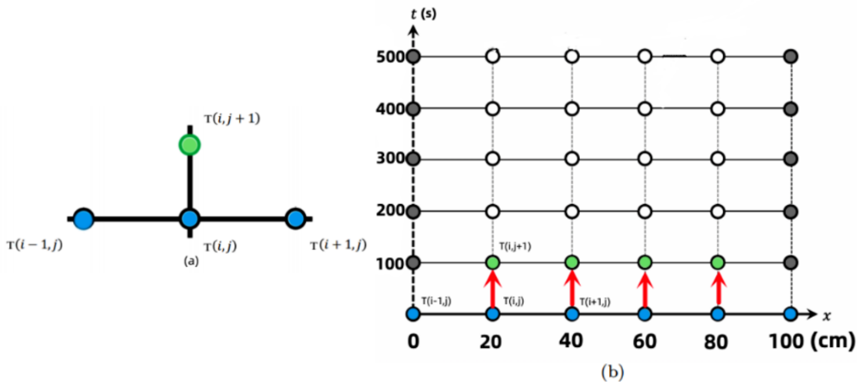


Figure 1: computational molecule on the explicit finite difference scheme

It can be known from the question, since the end of the aluminum rod is submerged in 0°C ice water the entire time, we can assume that the temperature is always 0°C no matter when i = 0 or i = 5. Therefore T(0,j) = T(5,j) = 0°C. Since the initial temperature of the rod is 500 °C, we can deduce that when $1 \leq i \leq 4$ and j = 0, the temperature is 500 °C. So T(1,0) = T(2,0) = T(3,0) = T(4,0) = 500 °C.

In figure 1, We find the computational molecule allows us to step horizontally and determine the temperatures of each node at j. After discovering the first three values, we can now calculate and update the temperat j + 1 until we reach our vertical limit by using equation 5. In this way, we can find the temperature of the bar at a certain position and at a certain time.

1.3 Python implementation

The main steps of Python code implementation:

1. Initialize the space step and time step. Set the length of the object (L), thermal conductivity (kappa), total time (T)

2. Simulate a grid(Nx*Ny) with a 2D array (like Figure1(b)), according to the step size in time and space($\Delta x, \Delta t$).

3. Initialize the initial temperature. I implemented this part using Numpy. According to the boundary conditions, both ends of the rod are always submerged in ice water at 0°C, so both T(0,j) and T(5,j) should be set to 0°C. Next, the initial temperature of bar is 500 degrees, so T(i,0) is set to 500, except for the first and last two values which are 0. Note use boundary conditions:

$$\begin{cases} T_{i,j+1} = \frac{k\Delta t}{\Delta x^2}(T_{i-1,j} + T_{i,j}(\frac{\Delta x^2}{k\Delta t} - 2) + T_{i+1,j}), \\ T_{i=0,j} = T_{i=Nx,j} = 0, \\ T_{i,j=0} = 500 \end{cases} \tag{6}$$

4. To traverse this 2-dimensional array, the parent loop is used to loop the time step (j), and the child loop is used to loop the space step (i). Then use the code to implement formula 5, and find the value of T(i, j+1) in the loop to update the 2-dimensional array, until all points are updated.

1.4 Experiments and Analysis

In Figure2, the initial temperature distribution is displayed at t = 0 seconds, with the interior of the rod having a temperature of 500°C and the ends having a temperature of 0°C. The temperature of the rod steadily drops over time, and the heat moves away from the rod's ends, but both ends of the bar are always 0 degrees, This meets the boundary conditions.

We can conclude above that the temperature drop in the middle of the rod is relatively slow because the heat needs to be transferred to the ends. However, areas near the boundary experience faster temperature drops due to contact with the surrounding cooling environment. Also, in the initial state, the temperature in the middle region is higher, but as time goes by, the temperature distribution becomes more and more uniform. After 600 seconds, the temperature of the whole bar dropped significantly.

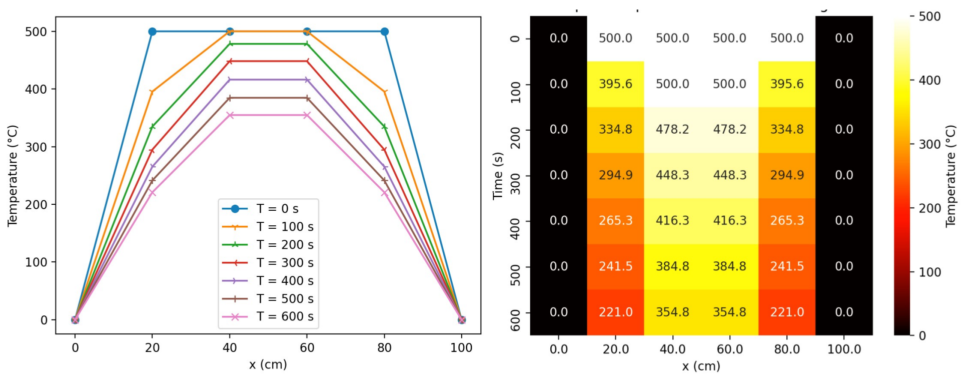


Figure 2: Temperature profile at 100-second intervals(left), heatmap(right)

In Figure 3 (left), for small time steps (such as $\Delta t = 50$ s), the results of the explicit method are closer to the analytical solution, indicating that reducing the time step size helps to improve the simulation accuracy. We can see a clear error (two absolute error line), but it gets lower and lower over time. This error can be attributed to very large Δt [2].

But when $\Delta x = 10$ cm, we found that the trend of the straight line also fluctuates abnormally (figure3(right)), and the error between the straight line and the analytical solution is very large. Errors get higher and higher over time. Since Δx =10 is too small here, the result is no longer stable. Because it exceeds the limit of Δx and dt. We can obtain $\Delta t \leq \frac{\Delta x^2}{2k}$ from the stability calculation formula[3]. In the case of Δx=10, Δt=100, we calculated that 100 > 59.88, so it is unstable.

x(cm)	**t**(s)	Δx(cm)	Δt(s)	**Temperature**(°C)	**Error**
20	600	Analytical	Analytical	230.5769	0%
20	600	20	100	220.9621	4.1699%
20	600	20	50	225.0470	2.3983%
20	600	10	100	-1995.6568	965.5059%

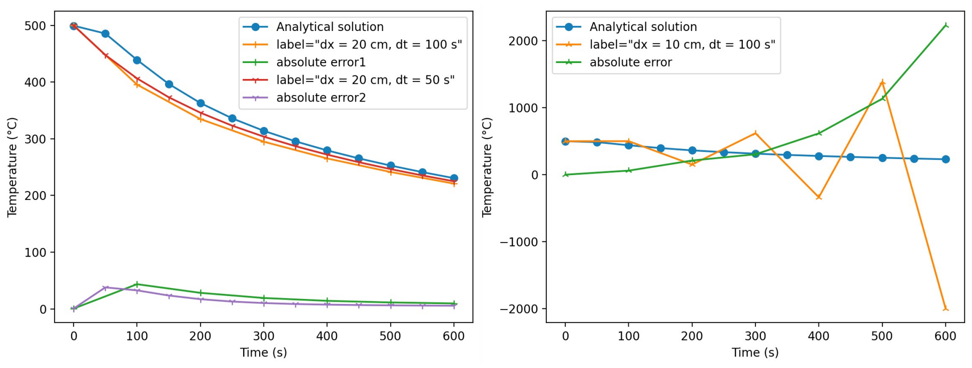


Figure 3: Temperature at x=20cm with error,usual(left), unusual(right)

According to the table above I have compiled (bar at 600s, temperature of x=20cm), we seem to be able to draw that the error of the explicit method increases relatively with the increase of the time step and space step, which shows that the explicit method is in Stability and accuracy are lower in these cases.

Because the stability of the explicit finite difference method is limited by the time step size and the space step size. According to the Courant-Friedrichs-Lewy (CFL) condition[4], the stability of an explicit method requires $\Delta t \leq \frac{\Delta x^2}{2k}$. When the time step and space step exceed this limit, the simulation results may produce unstable fluctuations and errors.

In this experiment, reducing the time step size (e.g., $\Delta t = 50$ s) helps to improve the stability and accuracy of the simulation, as this allows the explicit method to satisfy the CFL condition. However, when the spatial step size is small (such as $\Delta x = 10$ cm), the accuracy of the explicit method is still limited even if the CFL condition is satisfied. This is because explicit methods have inherent errors, especially at higher spatial resolutions.

2 Part2: Implicit of Heat Conductance Equation

2.1 Construct the finite difference equations

We may approximate the derivative at the midpoint of the interval using the solution of the heat conduction equation (Equ(1)) and the Crank-Nicholson method (central difference approach):

$$\frac{\partial T(x,t)}{\partial t}|_{i,j+\frac{1}{2}} = \frac{T_{i,j+1} - T_{i,j}}{\Delta t} \tag{7}$$

The average of the approximations at the start and end of the interval can be used to assess the spatial derivative at the temporal midpoint[1]:

$$\frac{\partial^2 T(x,t)}{\partial x^2}|_{i,j+\frac{1}{2}} = \frac{1}{2}(\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{\Delta x^2} + \frac{T_{i-1,j+1} - 2T_{i,j+1} + T_{i+1,j+1}}{\Delta x^2}) \tag{8}$$

By assuming that the PDE fits the computing grid at both the nodal midpoints and the nodes, the central difference approximation for the time derivative increases accuracy to second order:

$$\frac{\partial T(x,t)}{\partial t}|_{i,j+\frac{1}{2}} = \frac{1}{k}\frac{\partial^2 T(x,t)}{\partial x^2}|_{i,j+\frac{1}{2}} \tag{9}$$

Integrating Equations (1), (7), (8), (9):

$$\frac{1}{2}(\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{\Delta x^2} + \frac{T_{i-1,j+1} - 2T_{i,j+1} + T_{i+1,j+1}}{\Delta x^2}) = \frac{1}{k}\frac{T_{i,j+1} - T_{i,j}}{\Delta t} \tag{10}$$

Then we get:

$$T_{i,j+1} - \frac{T_{i+1,j+1}}{\frac{2\Delta x^2}{k\Delta t} + 2} - \frac{T_{i-1,j+1}}{\frac{2\Delta x^2}{k\Delta t} + 2} = \frac{T_{i-1,j}}{\frac{2\Delta x^2}{k\Delta t} + 2} + \frac{T_{i,j}(\frac{2\Delta x^2}{k\Delta t} - 2)}{\frac{2\Delta x^2}{k\Delta t} + 2} + \frac{T_{i+1,j}}{\frac{2\Delta x^2}{k\Delta t} + 2} \tag{11}$$

We may use this equation to build a network of equations with the required nodes. To solve it as a tridiagonal matrix (using Thomas algorithm), we must first step horizontally across the computational molecule to form our network of equations. To do this, group the known variables on the right side of the equation and the unknowns on the left, then correctly place the coefficients in the tridiagonal matrix.

We can time advance and use the j + 1 nodes as our new j nodes after the j + 1 nodes have been solved for. The technique is then repeated in order to solve and update for all nodes.

2.2 Computational molecule

The Crank-Nicolson method, as illustrated by Equations 10 and 11, calculates the numerator for a given time step. The nodes involved in this calculation are T(i-1, j), T(i, j), T(i+1, j), T(i-1, j+1), T(i, j+1), and T(i+1, j+1). This is a result of using the temporal midpoint, which generates the necessary nodes for computation.

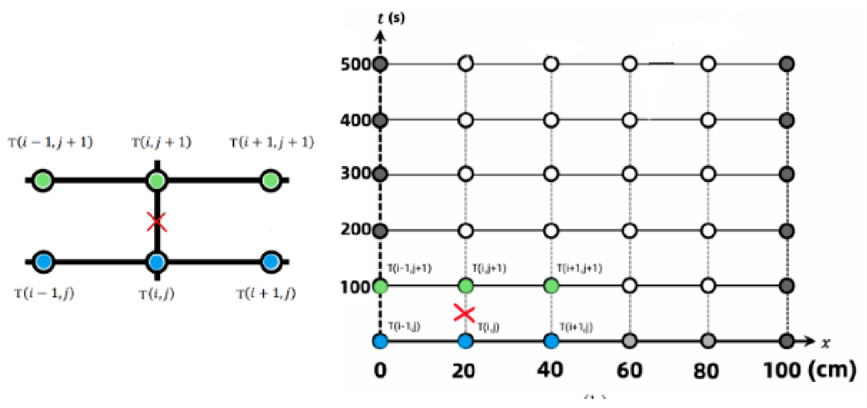


Figure 4: computational molecule on the Crank-Nicolson finite difference scheme

We can obtain the required variables to construct a tridiagonal matrix by using the computational molecule, which can then be utilized to solve for the temperatures of the unknown nodes. Subsequently, we can progress upwards and solve for j + 1 until we reach our vertical boundary, using solve and update all nodes.

In summary, the Crank-Nicolson method provides an efficient and accurate approach to solving the 1D heat conduction equation by iteratively constructing and solving a tridiagonal matrix system. This enables us to determine the temperature distribution across the domain at various time steps.

2.3 Python implementation

The main steps of Python code implementation:

1. Initialize the space step and time step. Set the length of the object (L), thermal conductivity (kappa), total time (T)

2. Simulate a grid with a 2D array (like Figure1(b)), according to the step size in time and space($\Delta x, \Delta t$).

3. Initialize the initial temperature. I implemented this part using Numpy. According to the boundary conditions, both ends of the rod are always submerged in ice water at 0°C, so both T(0,j) and T(5,j) should be set to 0°C. Next, the initial temperature of bar is 500 degrees, so T(i,0) is set to 500, except for the first and last two values which are 0.

4. Using a time-stepping loop, solve the linear equation Ax = Bx(t) sequentially. where A and B are matrices and x is a vector of length representing temperatures at discrete points.

5. According to Equations 12, 13, and 14, apply the equations for the boundary conditions that each computational molecule satisfies. The right side of the equation contains known variables, while the left side contains unknown variables. Calculate the right side as a numerical value, and collect the unknown coefficients on the left side into a list. In order to meet the boundary conditions given by the title, we will encounter the following three situations when dealing with T(n, j+1)($\gamma = \frac{2\Delta x^2}{k\Delta t}$):

(a) When a boundary condition is met by the left node $T_{i-1,j+1}$:

$$T_{i,j+1} - \frac{T_{i+1,j+1}}{\gamma+2} = \frac{T_{i-1,j}}{\gamma+2} + \frac{T_{i,j}(\gamma-2)}{\gamma+2} + \frac{T_{i+1,j}}{\gamma+2} + \frac{T_{i-1,j+1}}{\gamma+2} \tag{12}$$

(b) with no boundary conditions met

$$T_{i,j+1} - \frac{T_{i+1,j+1}}{\gamma+2} - \frac{T_{i-1,j+1}}{\gamma+2} = \frac{T_{i-1,j}}{\gamma+2} + \frac{T_{i,j}(\gamma-2)}{\gamma+2} + \frac{T_{i+1,j}}{\gamma+2} \tag{13}$$

(c) When a boundary condition is met by the right node $T_{i+1,j+1}$:

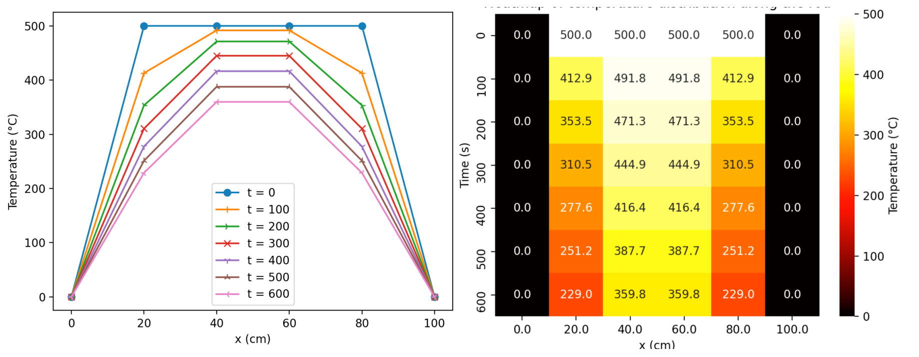
$$T_{i,j+1} - \frac{T_{i-1,j+1}}{\gamma+2} = \frac{T_{i-1,j}}{\gamma+2} + \frac{T_{i,j}(\gamma-2)}{\gamma+2} + \frac{T_{i+1,j}}{\gamma+2} + \frac{T_{i+1,j+1}}{\gamma+2} \tag{14}$$

6. Generate the coefficients and constants required to construct the tridiagonal matrices (A, B) and the constant list, using all calculated numerators. Solve the tridiagonal matrices using the numpy.linalg.solve() function. Assign the solved temperature values for each node to the corresponding attributes of the newly instantiated node class instances. Then, add these instances to the list of discovered nodes.

7. Then repeat steps 4 - 6 until each point is updated.

2.4 Experiments and Analysis

The graph on the left in Figure 5, compared with the corresponding graph in Part 1. The result looks more reasonable and more in line with real physics. The boundary and initial conditions are also fully satisfied.



Time (s) \ x (cm)	0.0	20.0	40.0	60.0	80.0	100.0
0	0.0	500.0	500.0	500.0	500.0	0.0
100	0.0	412.9	491.8	491.8	412.9	0.0
200	0.0	353.5	471.3	471.3	353.5	0.0
300	0.0	310.5	444.9	444.9	310.5	0.0
400	0.0	277.6	416.4	416.4	277.6	0.0
500	0.0	251.2	387.7	387.7	251.2	0.0
600	0.0	229.0	359.8	359.8	229.0	0.0

Figure 5: Temperature profile at 100-second intervals(left), heatmap(right)

In Figure 6 we find that the instability of the explicit method in Part 1 where $\Delta x = 10$ and $\Delta t = 100$ deviate significantly from the correct path no longer exists. Since the implicit Crank-Nicolson method is stable, the previously observed errors do not error here.

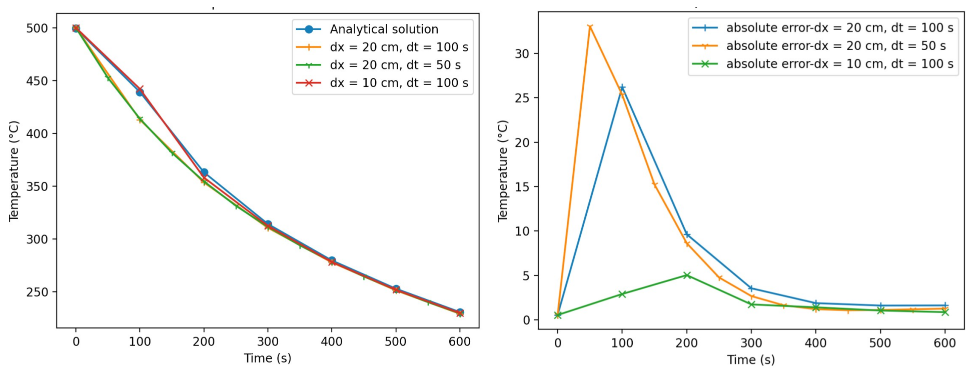


Figure 6: Temperature at x=20cm(left), absolute error(right)

Similar to the explicit method, the result converges to the analytical solution as more timesteps are taken. The outcome also converges more closely to the analytical solution as both Δx and Δt decrease in size. In particular, the $\Delta t = 50$s line almost perfectly aligns with the analytical line.

x(cm)	**t**(s)	Δx(cm)	Δt(s)	**Temperature**(°C)	**Error**
20	600	Analytical	Analytical	230.5769	0%
20	600	20	100	228.9552	0.7033%
20	600	20	50	229.3180	0.5460%
20	600	10	100	229.7124	0.3750%

We compared the table of Part2 with the table of Part1, and we found that the error between each line and the analytical solution, the error of the Crank-Nicolson method is smaller than that of the explicit method. This shows that the Crank-Nicolson method is more accurate and precise than the explicit method[5].

But is the required computational work worthwhile? The explicit method gives a relatively minimal computational effort to solve the problem and should be utilised in cases that do not require exceptional accuracy (while avoiding instances of instability). The Crank-Nicolson method offers a more computationally demanding solution that satisfies the requirements for applications that demand fine control and high levels of precision[6].

3 Part3: Modelling the Initial Temperature Change

3.1 Introduction and Choice of method

This part I plan to use the same method as Part2 - Crank Nicholson. Since this part requires accurate results for temperature changes in bar within 60 seconds, this means that Δt is smaller. We know that the stability limit is $\Delta t \leq \frac{\Delta x^2}{2k}$, we are unlikely to encounter stability problems, which means we can also use the explicit finite difference method of Part1. Compared with the Crank Nicholson method, it is simpler to implement, and the demand for calculation is not high.

However, it can be seen from the above two parts that the accuracy of the explicit finite difference method is not as good as that of the Crank-Nicolson method. In addition, it is not suitable for complex boundary conditions (the boundary conditions in this part are more complex than before)[7]. Furthermore, since there is no analytical solution for this part as a reference, in order to make the result as close as possible to the theoretical truth, I finally chose the Crank-Nicholson method.

3.2 Computational molecule and Python implementation

Since this part follows the same approach as the second part, the Computational molecule and Python implementation is not repeated here. But the only change is the generation of initial conditions. The initial temperature distribution is no longer 500 degrees but uneven distribution, adjust the initial settings according to the following equation 13:

$$T(x,0) = -xg(x-L) + c \qquad (g = 0.1, c = 400) \tag{15}$$

But it should be noted that with the change of dx of different sizes, the generation methods of these variables are also different, so the temperature of the bar should be initialized with the temperature of the correct position.

As an example, for Δx = 20cm, the temperatures of the four initial condition nodes are: T(0,0) = 0 °C, T(20,0) = 560 °C, T(40,0) = 640 °C, T(60,0) = 640 °C, T(80,0) = 560 °C, T(100,0) = 0 °C.

3.3 Preliminary research

On the left side of Figure 7, and then compare the same type of diagrams of Part 1 and 2, we can see that they meet expectations, indicating the correctness of the code implementation. On the right side of Figure 7, I compared multiple experiments with different combinations of Δx and Δt. It is found that the change rule is related to the size of Δx. Lines with the same Δx can be seen as a class, they almost overlap.

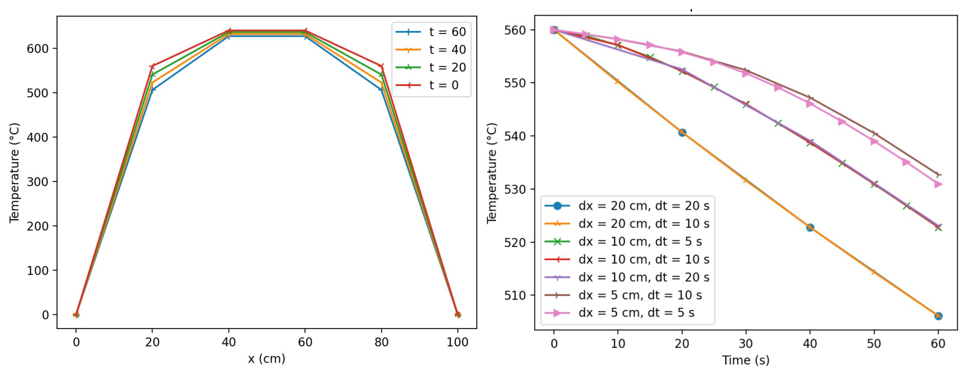


Figure 7: Temperature profile at 20-second intervals(left), T change at x=20(right)

3.4 Proving Experiment

3.4.1 Define Hypothesis

Hypothesis: Under the Crank-Nicholson method, the bar with an non-uniform initial temperature distribution is accurate when the temperature changes in the ice water 60s experiment.

3.4.2 Design Experiment

Choose a reasonable set of Δx and Δt. First repeat the experiment of Part2 - use a bar with a uniform initial temperature (500 degrees), use Crank-Nicholson to calculate the temperature change within 60 seconds when x=20cm, and record the data. Next, use the bar with uneven initial temperature (Part3), use Crank-Nicholson to calculate the temperature change within 60 seconds when x=20cm, and record the data. Then use the temperature difference for the same time period to calculate the rate of change of the temperature drop of the two bars. If the values of the two rates of change do not differ much, the hypothesis holds, otherwise, the hypothesis does not hold.

3.4.3 Prepare Experiment

We need to choose a reasonable set of Δx and Δt. Due to the high stability of the rank-Nicholson method and only in the time range $0 \leq t \leq 60$, we can safely reduce Δt. We want to reduce the error as much as possible. On the right side of Figure 7, lines with the same Δx can be seen as a class, they almost overlap. It seems that we can take Δx at will. But we observe that when Δx=20, the lines for different Δt values overlap very much. This means that when Δx=20 the value of Δt is less affected. So we choose **Δx=20, Δt=10**.

3.4.4 Conduct Experiment and Collect Data

Figure 8 is Δx=20, Δt=10 under two kinds of bars. Temperature change at position x=20 and time within 60 seconds. The rate of change (slope) is calculated every 20 seconds.

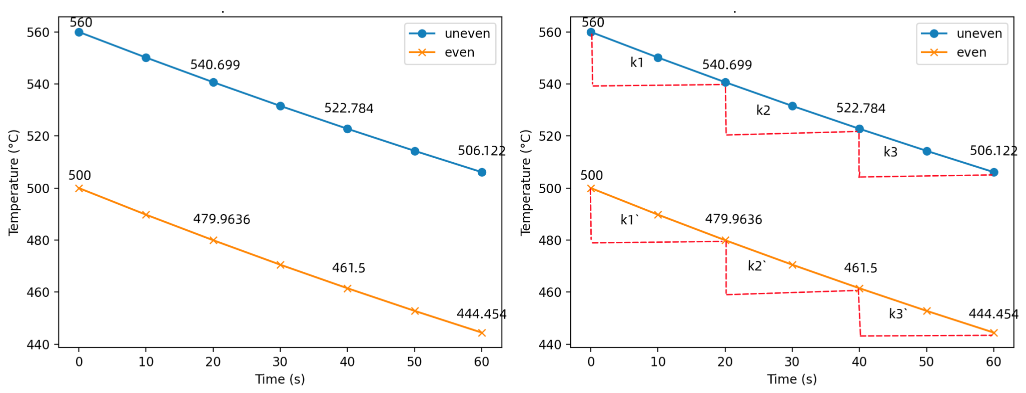


Figure 8: Temperature change at x=20cm(left), change of rate(right)

3.4.5 Analyze Data

Calculate the rate of change of each bar at each same time interval and record it:

Bar	K1(0-20s)	K2(20-40s)	K3(40-60s)	Average K
Uneven initial temperature	0.96505	0.89575	0.83310	0.89800
Even initial temperature	1.00182	0.92318	0.85230	0.92580

I calculated the difference in the rate of change (slope) of two bars at the same time interval. As shown in the table below:

Bar	K1(0-20s)	K2(20-40s)	K3(40-60s)	Average K
Difference(K-K‘)	0.03677	0.02743	0.01920	0.02780

3.4.6 Test Hypothesis

According to the two table in 3.4.5, we found that the difference in rate of change (slope) is small, the highest is 0.03677, the smallest is 0.0192, and the average is 0.0278. This is due to the error caused by the difference in the initial temperature. But compared with the whole, this difference is negligible. It can be said that the two lines (figure 8) have almost the same slope.

3.4.7 Result

According to the results of data analysis and hypothesis testing, we can conclude that the temperature change rates of the two bars are still very the same under the influence of errors. Combined with the purpose of our designed experiment, we can conclude that under the Crank-Nicholson method, the rod with a non-uniform initial temperature distribution is accurate when the temperature changes in the ice water 60s experiment.

4 Conclusion

From this CW, we know that the Crank-Nicolson method is a numerical method for solving partial differential equations, especially when solving time-dependent problems (Heat Conductance Equation). The Crank-Nicolson method has better stability when the time step is large[8], while the explicit method may cause the numerical solution to be unstable when the time step is large. The Crank-Nicolson method has second-order time convergence[7], which means that the error decreases at a quadratic rate as the time step decreases. In contrast, explicit methods usually only have first-order time convergence, which means that explicit methods may require smaller time steps to achieve the same accuracy.

However, the Crank-Nicolson method also has some shortcomings. The Crank-Nicolson method is an implicit method, which means that each time step needs to solve a linear system. This can lead to higher computational complexity and computational cost compared to explicit methods[9]. The implementation of the Crank-Nicolson method is usually more complicated compared to the explicit method. A system of linear equations needs to be solved, possibly using advanced numerical techniques such as matrix factorization or iterative solvers.

The Crank-Nicholson method has a wide range of applications and is widely used in various fields, including engineering, physics, biology, and economics. Its high numerical accuracy and stability make it a versatile tool for solving various types of time-dependent problems, such as heat conduction equations, topological conduction equations, and linear and nonlinear time-dependent equations[10]. The method’s ability to effectively handle complex and challenging problems has made it a popular choice in many industries and research fields.

References

[1] Dr Oliver Rhodes. Comp36212 week 8–implicit vs explicit. University Lecture, 2023.

[2] Jim Douglas and Henry H Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society*, 82(2):421–439, 1956.

[3] Mohamad Muhieddine, Edouard Canot, and Ramiro March. Various approaches for solving problems in heat conduction with phase change. *International Journal on Finite Volumes*, page 19, 2009.

[4] Carlos A De Moura and Carlos S Kubrusly. The courant–friedrichs–lewy (cfl) condition. *AMC*, 10(12), 2013.

[5] Omowo Babajide Johnson and Longe Idowu Oluwaseun. Crank-nicolson and modified crank-nicolson scheme for one dimensional parabolic equation. *International Journal of Applied Mathematics and Theoretical Physics*, 6(3):35–40, 2020.

[6] Bangti Jin, Buyang Li, and Zhi Zhou. An analysis of the crank–nicolson method for subdiffusion. *IMA Journal of Numerical Analysis*, 38(1):518–541, 2018.

[7] Michael Giles and Rebecca Carter. Convergence analysis of crank-nicolson and rannacher time-marching. 2005.

[8] C Sun and CW Trueman. Unconditionally stable crank-nicolson scheme for solving two-dimensional maxwell's equations. *Electronics Letters*, 39(7):595–597, 2003.

[9] Daniel J Duffy. A critique of the crank nicolson scheme strengths and weaknesses for financial instrument pricing. *The Best of Wilmott*, page 333, 2004.

[10] Alexei Lozinski, Marco Picasso, and Virabouth Prachittham. An anisotropic error estimator for the crank–nicolson method: application to a parabolic problem. *SIAM Journal on Scientific Computing*, 31(4):2757–2783, 2009.