Numerical Solution of the Heat Conductance Equation

Sultan AlRashed

April 2022

1 Explicit Solution of Heat Conductance Equation

I Constructing Finite Difference Equations

Using the solution of the heat conductance equation:

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

We can start constructing the finite difference equations required by substituting a central difference approximation for the second order spatial derivative, and a forward difference approximation for the first order time derivative as described in page 481 of the book Heat Conduction by Hahn and Ozisik[6].

$$\frac{\partial^2 T(x,t)}{\partial x^2} = \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{20^2} \tag{2}$$

$$\frac{\partial T(x,t)}{\partial t} = \frac{T_{i,j+1} - T_{i,j}}{100} \tag{3}$$

After substituting these new equations into our original heat conductance equation, with $\kappa = 0.835$, we get:

$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x} = \frac{T_{i,j+1} - T_{i,j}}{\Delta t \times 0.835} \tag{4}$$

Explicit update for $T_{i,j+1}$ based on values at time j can now be described:

$$T_{i,j+1} = \frac{0.835 \times \Delta t}{\Delta x}(T_{i-1,j} + T_{i,j}(\frac{\Delta x}{0.835 \times \Delta t} - 2) + T_{i+1,j}) \tag{5}$$

II Computational Molecule

Since the ends of the rods remain at 0 °C throughout the process, we can assume that when $i = 0$ or $i = 5$ the temperature is at 0 °C no matter the time. Therefore $T_{0,j} = T_{5,j} = 0$.

Due to the initial temperature of the rod being 500 °C, we can deduce that when $1 \leq i \leq 4$ and $j = 0$ the temperature is at 500 °C. Therefore $T_{1,0} = T_{2,0} = T_{3,0} = T_{4,0} = 500$.

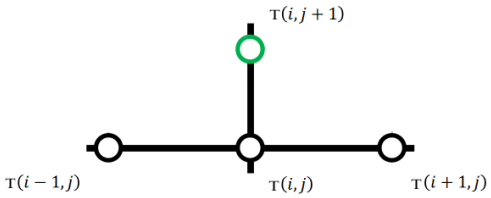


Figure 1: A computational molecule for a generic node is shown here based on the explicit finite difference scheme. If auxiliary conditions are available in any of the nodes, they are applied.

By stepping horizontally using the computational molecule, we can calculate the temperature values of each node at $j + 1$. With the previous values found, we can now step vertically and solve for $j + 2$ until we reach our vertical limit.

III Overview of Software Implementation

To generate the solution, the following was done:

1. A node class was created that holds four values: the node's i value, the node's j value, the node's temperature value. An empty list of nodes is initialized.

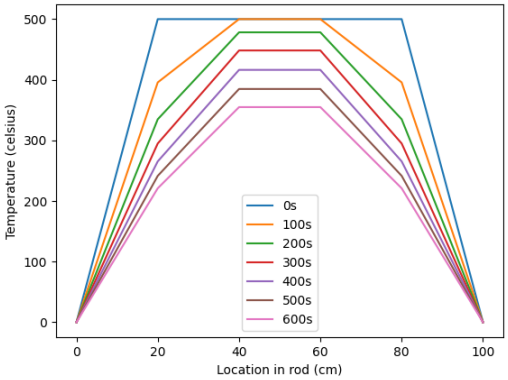
2. Auxiliary nodes were generated. This was done through two separate loops, one was responsible for generating $T_{i,j}$ where $1 \leq i \leq 4$ and $j = 0$ while the other is responsible for $T_{i,j}$ where $i = 0 or i = 5$ and $0 \leq j \leq 5$. The former is set at 500 °C while the latter is 0 °C. Auxiliary nodes generated are added to the list of nodes.

3. Interior nodes are calculated by a nested for loop, with the parent loop looping over $0 \leq j \leq 5$ (moves computational molecule vertically) and the inner loop looping over $1 \leq i \leq 5$ (moves computational molecule horizontally). Equation 5 is implemented in code and is used within the for loops to calculate the interior nodes, with them being added to the list of nodes. To get the previous nodes required for calculation, the list of nodes is searched for nodes that possess the i and j numbers required.

4. Finally, the solution is printed by looping over the list of nodes and displaying their i number, j number, and temperature value.

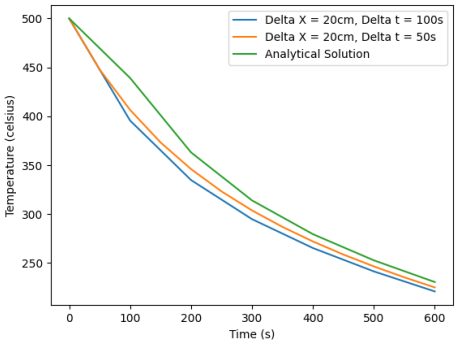
IV Experimental Results

Figure 2: Temperature distribution along the bar at 100s intervals.

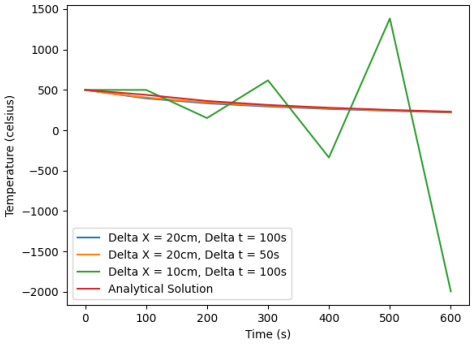


As can be easily predicted, the initial conditions of $t = 0$ and $100 > x > 0$ state that the temperature is 500 °C. While the boundary conditions of $x = 0 and x = 100$ state that the temperature is 0 °C. Both of these conditions can be seen in the graph.

As time increase, temperature decreases across the length of the bar. The temperature can be seen to decrease faster at the ends of the bar than it does in the middle.



(a) Evolution of temperature calculated with numerical solution shown against the analytical solution to view the error.



(b) Previous plot but with $\Delta x = 10$ included.

In figure (a) a noticeable inaccuracy can be seen initially (ignoring the initial condition of the temperature being 500 °C that was set in advance) that gets lower as time goes on. As discussed in an article by Douglas, Jim, Rachford, and Henry H[4], this noticeable inaccuracy can be attributed to Δt being quite large. As can be seen from figure (a), when Δt is reduced the line gets closer to the analytical solution, reducing the error.

Explicit methods present a further cause of inaccuracy due to the inherent limit of how large Δt can be or how small Δx can be, which ensures numerical stability[11]. This can be seen in figure (b) where due to how small Δx is, the results are no longer

stable and it can obviously be seen that the results are completely wrong. Much of the inaccuracy stems from explicit methods being conditionally stable.

x	t	Δt	Δx	Temperature
20cm	600s	Analytical	Analytical	230.577 °C
20cm	600s	100s	20cm	220.962 °C
20cm	600s	50s	20cm	225.047 °C
20cm	600s	100s	10cm	-1995.657 °C

To avoid issues of instability, a limit of the possible size for both Δx and Δt must be adhered to. The limit is $\Delta t \leq \frac{\Delta x^2}{2\kappa}$.

2 Implicit Crank-Nicholson Solution of Heat Conductance Equation

I Constructing Finite Difference Equations

Using the solution of heat conductance equation provided, we can use the Crank-Nicholson method of using a central difference approach to approximate the derivative at the mid-point of the interval:

$$\frac{\partial T(x,t)}{\partial t} = \frac{T_{i,j+1} - T_{i,j}}{\Delta t} \tag{6}$$

We can then calculate the spatial derivative at the temporal midpoint by calculating as the average of the approximations at the beginning and end of the interval:

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

Substituting this into the original heat conductance equation:

$$\frac{1}{2}\left(\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{\Delta x^2} + \frac{T_{i-1,j+1} - 2T_{i,j+1} + T_{i+1,j+1}}{\Delta x^2}\right) = \frac{1}{\kappa}\frac{T_{i,j+1} - T_{i,j}}{\Delta t} \tag{8}$$

We can use this equation to create a system of equations with the necessary nodes, we must step horizontally using the computational molecule to generate our system equations and then solve it as a tridiagonal matrix. This is accomplished by collecting known variables on the right hand side of the equation with the unknowns on the left, placing the coefficients in the tridiagonal matrix appropriately.

Considering the fact that initial conditions can solve for nodes at j we can focus on nodes at $j+1$. Although boundary conditions can sometimes account for certain nodes at $j+1$, an interior group of nodes must be dealt with. Therefore, these three equations can solve for each $j+1$ node:

When node $T_{i-1,j+1}$ meets a boundary condition:

$$T_{i,j+1} - \frac{T_{i-1,j+1}}{\frac{2\Delta x^2}{k\Delta t} + 2} = \frac{T_{i-1,j}}{\frac{2\Delta x^2}{k\Delta t} + 2} + \frac{T_{i,j}\left(\frac{2\Delta x^2}{k\Delta t} - 2\right)}{\frac{2\Delta x^2}{k\Delta t} + 2} + \frac{T_{i+1,j}}{\frac{2\Delta x^2}{k\Delta t} + 2} + \frac{T_{i+1,j+1}}{\frac{2\Delta x^2}{k\Delta t} + 2} \tag{9}$$

When node $T_{i+1,j+1}$ meets a boundary condition:

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

When no boundary conditions are met:

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

After $j+1$ nodes are solved for, we can timestep up and use the $j+1$ nodes as our new j nodes. We then repeat the process to solve for all nodes.

II Computational Molecule

The nodes present in the implicit Crank-Nicholson method are $T_{i-1,j}$, $T_{i,j}$, $T_{i+1,j}$, $T_{i-1,j+1}$, $T_{i,j+1}$, $T_{i,j+1}$, $T_{i+1,j+1}$. This is due to taking the temporal midpoint, which generates the nodes necessary for calculation.

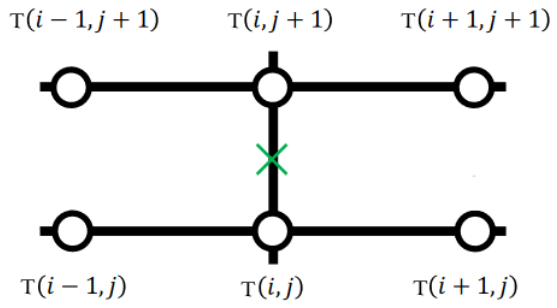


Figure 4: A computational molecule for a generic node is shown here based on the Crank-Nicholson method. If auxiliary conditions are available in any of the nodes, they are applied.

By stepping horizontally using the computational molecule, we can generate the variables required to create a tridiagonal matrix with which we can solve the unknown nodes' temperatures. With the previous values found, we can now step vertically and solve for $j+2$ until we reach our vertical limit.

III Overview of Software Implementation

To generate the solution, the following was done:

1. A node class is created containing the node's i and j value, representing their position on the node graph and the node's temperature.

2. The boundary conditions and initial conditions are all initialised as nodes into a list of nodes.

3. Two nested for loops were responsible for generating the known and unknown variables for a single instance of the computational molecule. The inner loop was responsible for $i-1$, i, and $i+1$ while the parent loop was responsible for j and $j+1$.

4. To calculate the known and unknown variables, a check is performed against the list of nodes if any of the nodes in the computational molecule already exist. If they do, add it to the list of known variables.

5. The two nested loops contains a further parent loop, responsible for moving the computational molecule horizontally. Steps 2 and 3 are repeated until all known variables are specified for each computational molecule.

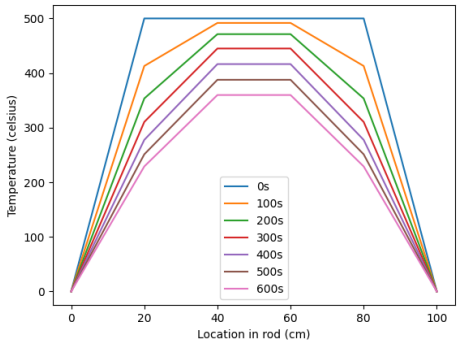
6. Equations 9, 10, and 11 are implemented in software. The equations are applied according to the boundary conditions met by each computational molecule. The right hand side of the equation contain known variables, while the left hand side contains the unknowns. The right hand side is calculated into a number, while the left hand side's unknowns' coefficients are collected into a list.

7. Equations 9, 10, and 11 are used to generate the coefficients and definite values necessary to construct a tridiagonal matrix and a list of definite values, respectively, out of all the computational molecules.

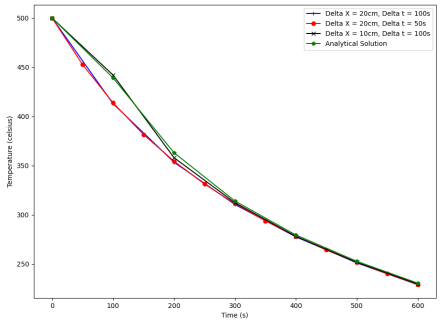
8. The tridiagonal matrix is solved by implementing the Thomas algorithm. The solved values for the temperature of each node discovered is set as the value of newly instantiated node classes. The nodes are then added to the list of discovered nodes.

9. An even further parent loop preceding the other three is responsible for moving the computational molecule vertically upwards, using the solutions generated from before as the basis. Steps 2 - 8 are repeated until the vertical limit is reached, thereby getting a solution for each node.

IV Experimental Results



(a) Temperature distribution along the bar at 100s intervals.



(b) Temperature evolution of the bar with time at $x = 20$cm

x	t	Δt	Δx	Temperature
20cm	600s	Analytical	Analytical	230.577 °C
20cm	600s	100s	20cm	228.955 °C
20cm	600s	50s	20cm	229.318 °C
20cm	600s	100s	10cm	229.712 °C

For figure (a), compared to the previous figure in section one the results seem smoother. Similarly to the figure in part one, the boundary and initial conditions seem to be met completely.

The most interesting finding is figure (b), where the explicit method's instability that caused $\Delta x = 10, \Delta t = 100$ to veer far off the correct path is no longer present. Since the implicit Crank-Nicholson method is unconditionally stable[12][8], the error from before does not exist here.

Similarly to the explicit method, the result converges to the analytical solution the more timesteps are taken. The result also converges closer to the analytical solution the smaller Δx and Δt are, with the $\Delta t = 50s$ line almost matching up completely with the analytical line.

The results also converge closer to the analytical solution, showing that the implicit Crank-Nicholson method is more accurate and precise than explicit methods[8]. In fact we can compare the accuracy of the Crank-Nicholson method against the explicit method by comparing their error margins:

x	t	Δt	Δx	Crank-Nicholson Error	Explicit Error
20cm	600s	100s	20cm	0.7035%	4.17%
20cm	600s	50s	20cm	0.546%	2.3983%
20cm	600s	100s	10cm	0.37515%	965.5%

With the error shown clearly, it is easy to see that the Crank-Nicholson method is much more accurate than the explicit method even when we ignore the instability

caused at $\Delta x = 10$cm. Is it worth the computational effort required? For applications that do not require an extreme degree of accuracy, the explicit method requires relatively low computational effort to solve the problem and should be used in those situations (avoiding situations of instability while doing so). For applications that require fine control and a high degree of accuracy, the Crank-Nicholson method provides a relatively high computational effort way of meeting those demands.

3 Modelling the Initial Temperature Change

I Choice of Approach with Initial Reasoning

Due to the time period explored being lower, Δt will also be lower. This gives us an opportunity to use the explicit finite difference method since the limit is $\Delta t \leq \frac{\Delta x^2}{2\kappa}$, and by making Δt a factor of 10 smaller (to match the change in $t_{\max}$) we are unlikely to encounter problems of stability. We can therefore reduce the computational effort exerted by using the explicit finite difference method.

On the other hand, problems of stability were not the only ones encountered with the explicit finite difference method. Accuracy was also lower when compared to the Crank-Nicholson method. Since there is no analytical result to compare to, it is best served to stick with the implicit Crank-Nicholson method to maintain as close of a result as possible to a theoretical analytical solution. When it comes to manufacturing precision and accuracy are highly sought after, even if it is at the cost of computational effort. Therefore, it is best to use the Crank-Nicholson method. An important note is that the Crank-Nicholson method produces an oscillatory response to sharp initial transients[3], but due to the lack of sharp initial transients here this oscillatory response is unlikely to be reproduced.

II Overview of Software Implementation

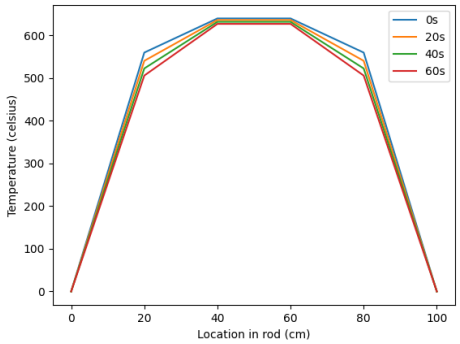
The implementation here follows the same footsteps as section two's, with the only change being the generation of initial conditions. Instead of setting the initial conditions as 500 °C, the initial settings are adjusted according to the following equation (where $g = 0.1, c = 400.$):

$$T(x, 0) = -xg(x - L) + c \tag{12}$$

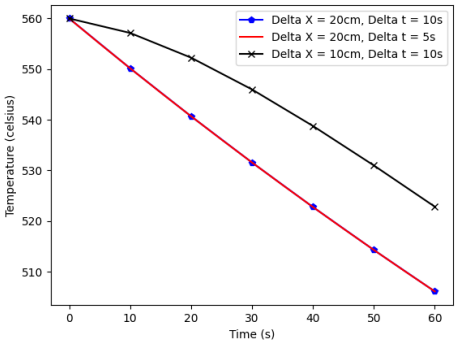
Different sizes of Δx change how these variables are generated, but the code was consistently made modular to handle these types of changes with ease.

For $\Delta x = 20$cm, the four initial condition nodes' temperatures were: $T_{1,0} = 560$ °C, $T_{2,0} = 640$ °C, $T_{3,0} = 640$ °C, $T_{4,0} = 560$ °C. These values seem to align with expected values from the equation given and do not unreasonable.

III Verification of Accuracy



(a) Temperature distribution along the bar at 100s intervals.



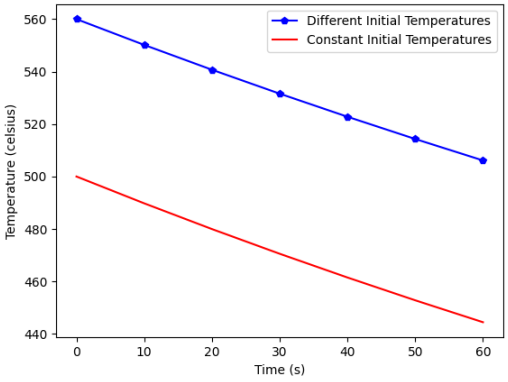
(b) Temperature evolution of the bar with time at $x = 20$cm, with $\Delta x = 10$cm

From the results generated, we can see that they align with expectations. Boundary conditions can be seen to be met in figure (a), while the initial conditions of $x = 20$cm and $t = 0$s can be seen to be met in figure (b) (with the initial condition being 560 °C as discussed above).

To verify the accuracy of the implementation, we can use the initial conditions of part two and compare the results of this implementation against the implementation in section two, considering it had a high degree of accuracy.

Since the implicit Crank-Nicholson method is devoid of instability, we can reliably reduce Δt. To make the results applicable, we will only check for the range $0 \leq t \leq 60$:

Figure 7: Temperature evolution of the bar with time at $x = 20$cm. $\Delta x = 20$cm and $\Delta t = 10$s.



By deriving both lines to find the rate of temperature change we get:

$$\text{Different Initial Temperature Line: } \frac{540.698 - 560}{20 - 0} = -0.9651 \frac{^\circ C}{s}$$

$$\text{Constant Initial Temperature Line: } \frac{479.964 - 500}{20 - 0} = -1.0018 \frac{^\circ C}{s}$$

With the rate of change being similar (but not the same), we can make the assumption that the accuracy is quite high with the slight difference accounting for the differing initial conditions. This is because the constant initial temperature line has been verified for accuracy, and by generalising it with a comparison against the different initial temperature (using the gradient) we can verify the different initial temperature's accuracy. Since the timestep is small ($\Delta t = 10$s), the rate of change in temperature should in fact be quite similar. The lines also seem to be quite similar with just an offset in the y-axis, showing the similarity in gradient.

x	t	Δt	Δx	Different Initial Temperature	Constant Initial Temperature
20cm	60s	10s	20cm	506.122 °C	444.454 °C
20cm	60s	5s	20cm	506.127 °C	444.454 °C
20cm	60s	10s	10cm	522.850 °C	444.454 °C

Furthermore the table above shows when $\Delta t = 5$s it shares a similar result to $\Delta t = 10$s. When $\Delta x = 10$cm however, the result is no longer similar. In fact when looking at figure 6 (b) the $\Delta x = 10$cm line has a different gradient for temperature reduction. This can be attributed to Δx just being smaller.

IV Further Discussion.

The implicit Crank-Nicholson can be seen to avoid problems of instability in this new situation due to its unconditional stability while also maintaining a high degree of precision and accuracy[7][13]. From our experimentation, the results seemed to align with both expectations and previous experimental results in section two.

However with these strengths come limitations[5], with the main one being the extensive computational effort required to implement the implicit Crank-Nicholson and to generate results[7]. The implementation takes a while due to multiple factors: the increase in amount of nodes evaluated compared to the explicit finite difference method, the separation of known and unknown nodes, creation of the tridiagonal matrix, and implementing a solver for the sparse tridiagonal matrix[13]. Generating results takes longer than explicit methods due to the same reasons stated above.

The implicit Crank-Nicholson method can be shown to use more memory and take more time to compute results compared to other methods[9][1]. In the explicit finite difference method, one only needs to find the value of a single node with all the nodes required for calculation being known, making it a simple equation solver that loops over the node graph.

Most importantly though is seeing how the implicit Crank-Nicholson method can be generalised to other problems. Implementations of the implict Crank-Nicholson method have been shown to generate numerical solutions to problems such as the simulation of water hammer[2] and the simulation of dispersive structures[9]. With some modifications the Crank-Nicholson method can be further extended to even more problems. For example, by using an alternating Crank-Nicholson method one can decouple the Ginzburg-Landau equations[10].

However, the problems that this specific implementation can handle are limited specifically to ones of heat conductance in a two dimensional node graph. With modifications to the equations used to generate the known and unknown variables, a person can feasibly extend the current implementation to any other problem that can be solved using the unmodified implicit Crank-Nicholson method (which includes the problems cited above that require no modifications).

The analysis provided in this section cannot be confirmed as definite. Without an analytical solution to compare against, it can be difficult to ascertain with 100% certainty that the approach used was successful. However by gauging the similarity of results compared to previous ones (while accounting for the change in initial conditions), it can be assumed that the result is not wholly inaccurate. In fact, we can say that the result is most likely accurate.

References

[1] Alla Tareq Balasim and Norhashidah Hj Mohd Ali. A rotated crank–nicolson iterative method for the solution of two-dimensional time-fractional diffusion equation. *Indian Journal of Science and Technology*, 8(32):1–7, 2015.

[2] Abdol Mahdi Behroozi and Mohammad Vaghefi. Numerical simulation of water hammer using implicit crank-nicolson local multiquadric based differential quadrature. *International Journal of Pressure Vessels and Piping*, 181:104078, 2020.

[3] Dieter Britz, Ole Østerby, and Jörg Strutwolf. Damping of crank–nicolson error oscillations. *Computational Biology and Chemistry*, 27(3):253–263, 2003.

[4] 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.

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

[6] David W Hahn and M Necati Özisik. *Heat conduction*. John Wiley & Sons, 2012.

[7] Willem Hundsdorfer. Unconditional convergence of some crank-nicolson lod methods for initial-boundary value problems. *mathematics of computation*, 58(197):35–53, 1992.

[8] 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.

[9] Shi-Yu Long, Wei-Jun Chen, Qi-Wen Liang, and Min Zhao. A general ade-fdtd with crank-nicolson scheme for the simulation of dispersive structures. *Progress In Electromagnetics Research Letters*, 86:1–6, 2019.

[10] Mo Mu and Yunqing Huang. An alternating crank–nicolson method for decoupling the ginzburg–landau equations. *SIAM journal on numerical analysis*, 35(5):1740–1761, 1998.

[11] 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.

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

[13] Guilin Sun and Christopher W Trueman. Efficient implementations of the crank-nicolson scheme for the finite-difference time-domain method. *IEEE transactions on microwave theory and techniques*, 54(5):2275–2284, 2006.