Numerical Solution of the Heat Conductance Equation

Jonathan Hackett

April 2024

1 Explicit Solution

1.1 Finite Difference Approximations

$$\left.\frac{\partial^2 T(x,t)}{\partial x^2}\right|_{i,j} \approx \frac{T(x_{i+1},t_j) - 2T(x_i,t_j) - T(x_{i-1},tj)}{\Delta x^2}$$
$$\left.\frac{\partial T(x,t)}{\partial t}\right|_{i,j} \approx \frac{T(x_i,t_{j+1}) - T(x_i,t_j)}{\Delta t}$$

1.2 Update Equation

$$T(x_i,t_{j+1}) = rT(x_{i+1},t_j) - (1-2r)T(x_i,t_j) - rT(x_{i-1},t_j)$$

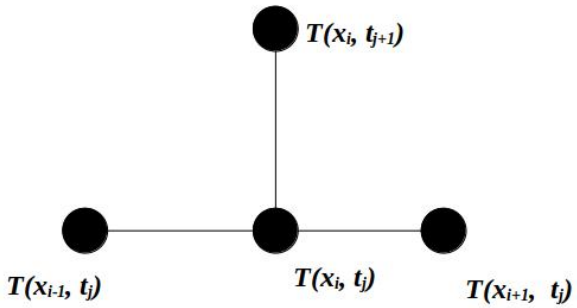
$$\begin{aligned} \text{where } x_i &= i\Delta x \\ t_m &= m\Delta t \\ r &= \frac{k\Delta t}{\Delta x^2} \end{aligned}$$

1.3 Implementation

Results are stored in a 2d array A of size $\frac{600}{\Delta t}+1$ by $\frac{L}{\Delta x}+1$ where $A[j][i] = T(x_i,t_j)$. The boundary conditions are applied by first setting $A[0] = [0,500,...,500,0]$ and then $A[j][0], A[j][L] = 0$ for $1 \le j \le \frac{600}{\Delta t}$.

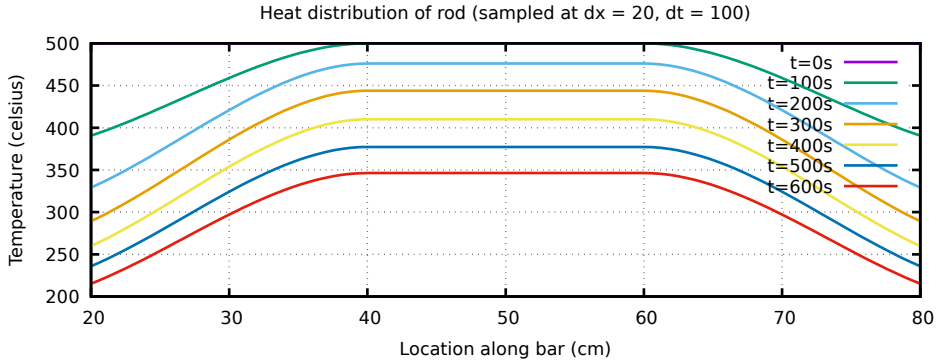
After this, the main program loop can begin, during which the elements $1 \le x \le \frac{L}{\Delta x}$ of each $A[j]$ for $1 \le j \le \frac{100}{\Delta x}$ are iterated over using the update equation above.

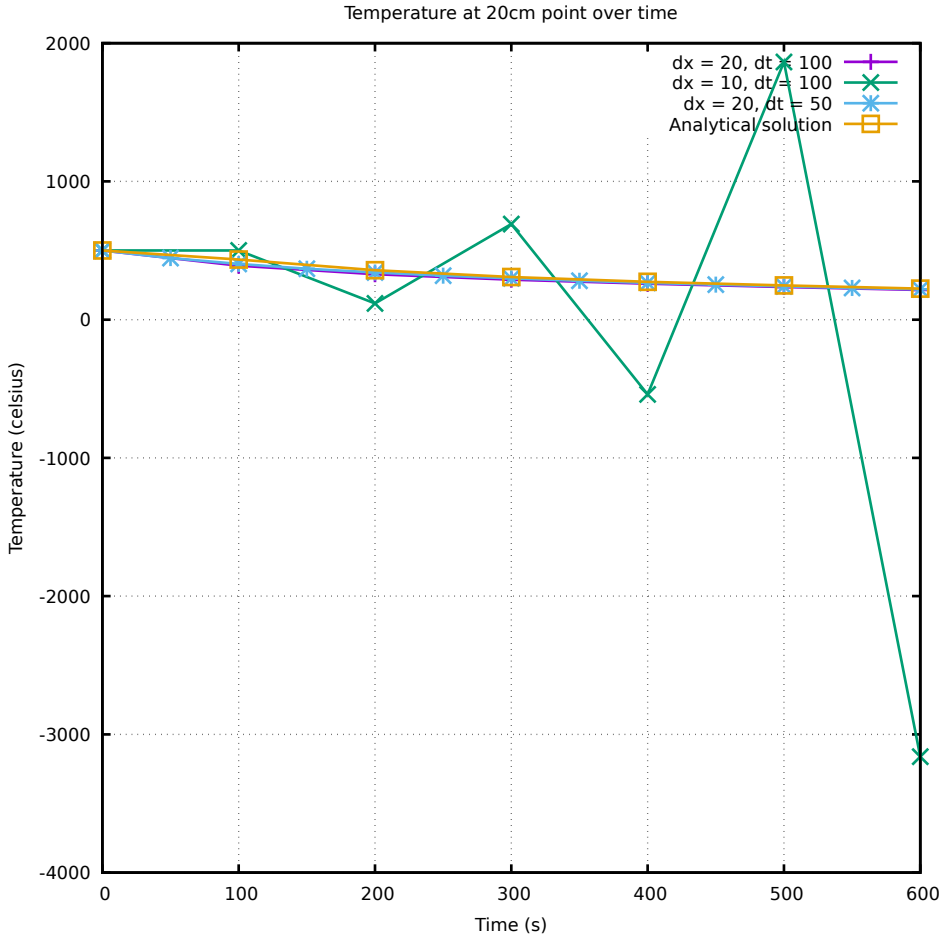
Figure 1: Computational Molecule for Explicit FDM



1.4 Results

Results show that $\Delta x = 20, \Delta t = 50$ yields the most accurate results. $\Delta x = 10, \Delta t = 100$ yields highly inaccurate results. This is because the explicit FD method used is unstable for $r > \frac{1}{2}$. To remedy this, one can set Δt to a smaller value.





Time(s)	Temperature(C) at 20cm along bar			
	$\Delta x = 20, \Delta t = 100$	$\Delta x = 20, \Delta t = 50$	$\Delta x = 10, \Delta t = 100$	Analytical
0	500.00	500.00	500.00	500.00
50	-	445.31	-	-
100	390.63	402.59	500.00	438.66
150	-	368.56	-	-
200	329.10	340.87	117.19	351.94
250	-	317.87	-	-
300	289.26	298.33	691.41	306.71
350	-	281.40	-	-
400	259.82	266.46	-540.77	272.42
450	-	253.05	-	-
500	235.85	240.86	1863.77	246.06
550	-	229.64	-	
600	215.19	219.22	-3161.11	223.92

2 Implicit Solution

2.1 Finite Difference Approximations

$$\left.\frac{\partial^2 T(x,t)}{\partial x^2}\right|_{i,j+\frac{1}{2}} \approx \frac{1}{2}\left(\frac{T(x_{i-1},t_j) - 2T(x_i,t_j) + T(x_{i+1},t_j)}{\Delta x^2} + \frac{T(x_{i-1,j+1},t_{j+1}) - 2T(x_i,t_{j+1}) + T(x_{i+1},t_{j+1})}{\Delta x^2}\right)$$

$$\left.\frac{\partial T(x,t)}{\partial t}\right|_{i,j+\frac{1}{2}} \approx \frac{T(x_i,t_{j+1}) - T(x_i,t_j)}{\Delta t}$$

2.2 Update Equation

$$-\frac{r}{2}T(x_{i-1},t_{j+1}) + (1+r)T(x_i,t_{j+1}) - \frac{r}{2}T(x_{i+1},t_{j+1}) \qquad r = \frac{k\Delta t}{\Delta x^2}$$
$$= \frac{r}{2}T(x_{i-1},t-j) + (1-r)T(x_i,t_j) + \frac{r}{2}T(x_{i+1},t_j)$$

2.3 Implementation

Figure 2: Matrix representation of Update Equation

$$\underbrace{\begin{bmatrix} \beta_1 & \gamma_1 & 0 & 0 & ... & 0 & 0 & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & 0 & ... & 0 & 0 & 0 \\ 0 & \alpha_3 & \beta_3 & \gamma_3 & ... & 0 & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & ... & \alpha_{n-1} & \beta_{n-1} & \gamma_{n-1} \\ 0 & 0 & 0 & 0 & ... & 0 & \alpha_n & \beta_n \end{bmatrix}}_{A} \times \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ ... \\ x_{n-1} \\ x_n \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ ... \\ s_{n-1} \\ s_n \end{bmatrix}}_{s}$$

$$\alpha_i = -\frac{r}{2} \qquad n = \frac{100}{\Delta x} - 1$$
$$\beta_i = 1 + r \qquad s_i = \frac{r}{2}T(x_{i-1}, t-j) + (1-r)T(x_i, t_j) + \frac{r}{2}T(x_{i+1}, t_j)$$
$$\gamma_i = -\frac{r}{2}$$

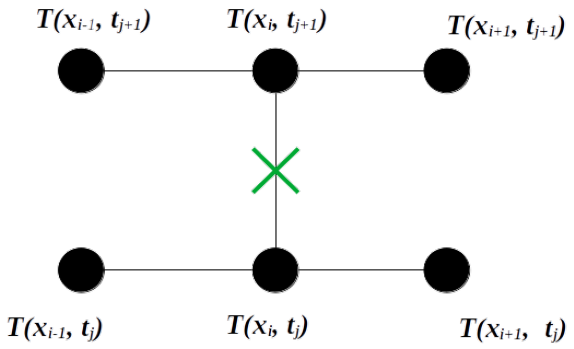
The update equation can be represented as a tridiagonal matrix system, as in figure 2, which can be solved for x using the Thomas algorithm.

In software, arrays a, b, y each of size $n+2$ are used to store each value of α, β, γ respectively. Boundary conditions of x are applied by setting indexes $0, n$ to 0.0 for each of these arrays. Values of T are stored in array $x[n+2]$ where $x[i]$ is the temperature at point i at the current time. $x = [0.0] + [500] * n + [0.0]$ is initially set for t's boundary condition.

Once this is done, the values of a, b, y can be decomposed in accordance with Thomas's algorithm. Then, the main program loop can begin.

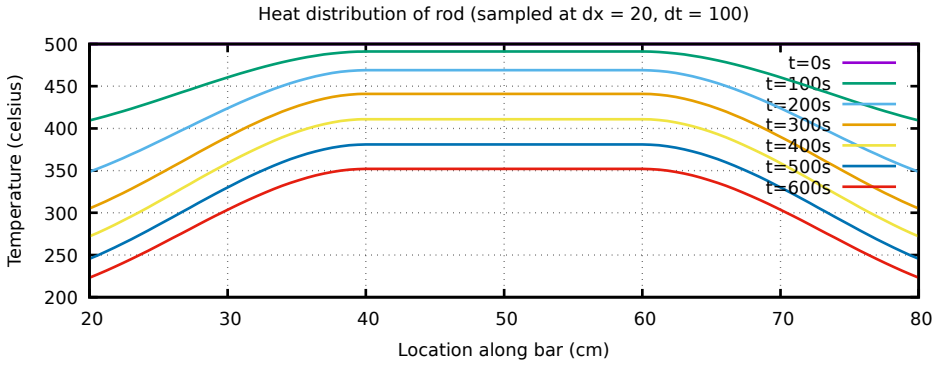
At each step, s is evaluated using the values of x. Then, the substitution steps of Thomas's algorithm are applied. The resulting values are stored in x. This repeats for $1 < t \leq \frac{100}{\Delta x} - 1$.

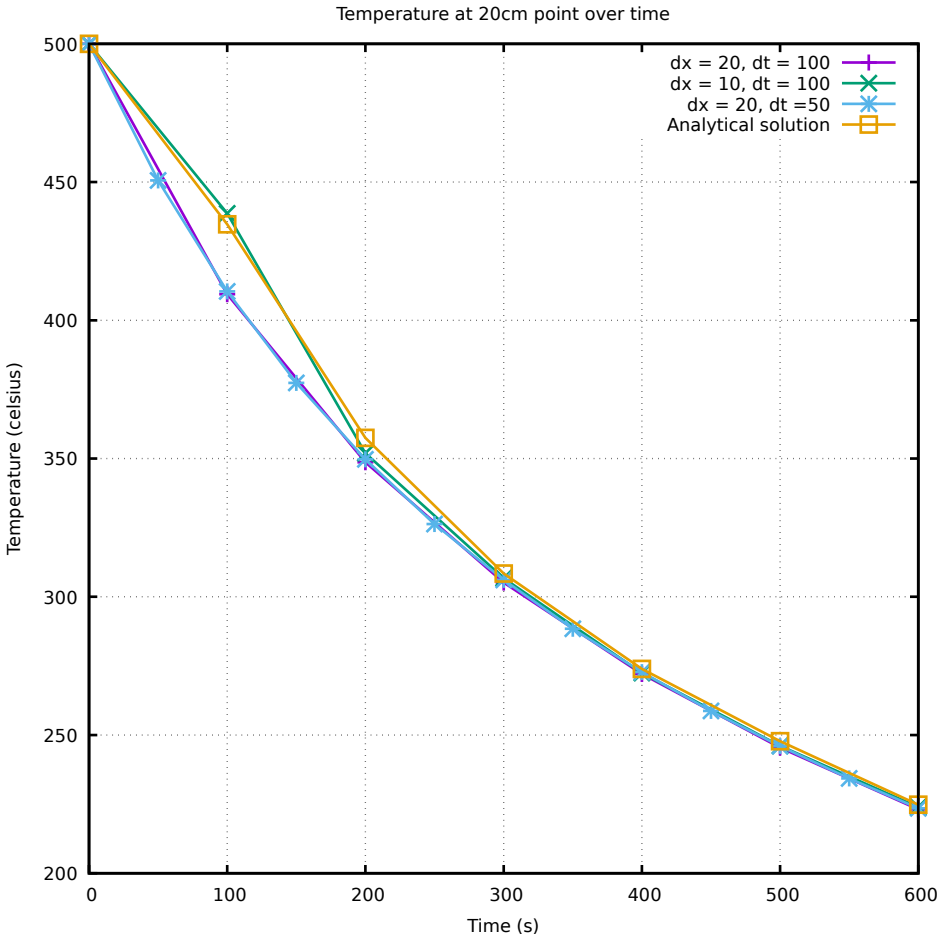
Figure 3: Computational Molecule for Implicit Crank-Nicholson



2.4 Results

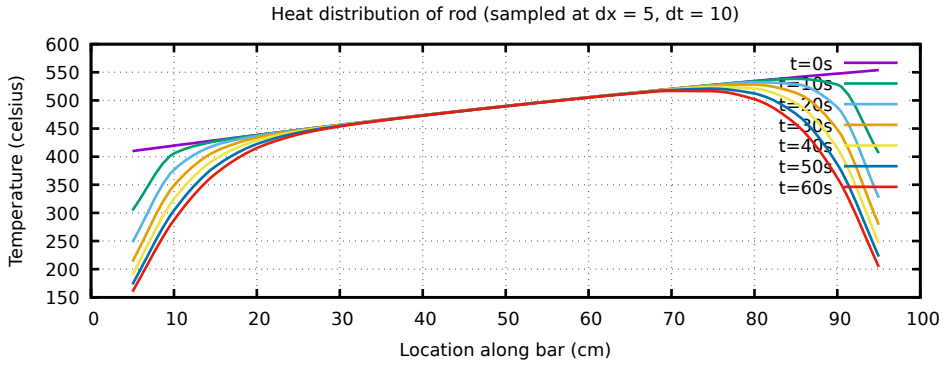
Results show overall improved accuracy over explicit FD. $\Delta x = 10, \Delta t = 100$ yields results closest to those of the analytical method, much closer than EFD for any of the tested variables. The method is unconditionally stable for the heat equation - unlike EFD - which explains it's stability for $\Delta x = 10, \Delta t = 100$ versus EFD's instability. The trade-off for these benefits is a more complex algorithm that uses more memory and takes longer to run.

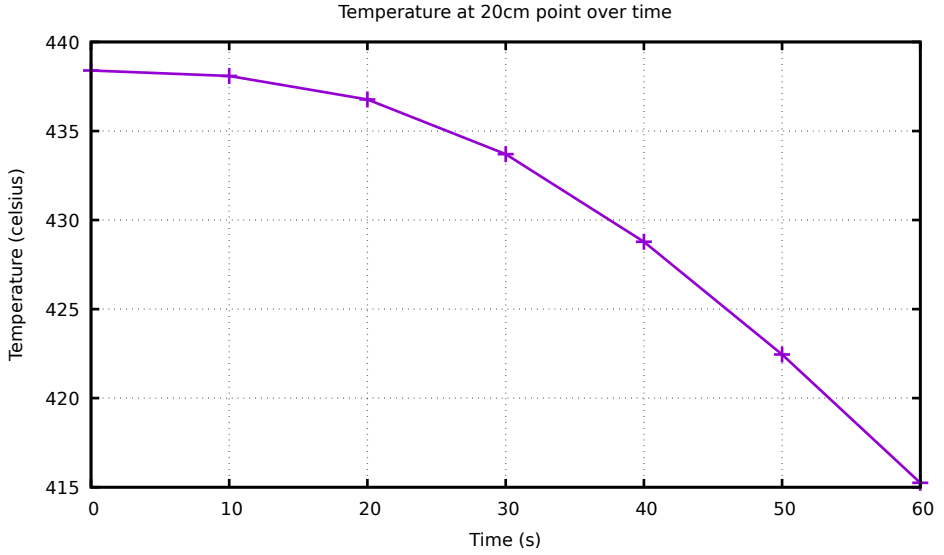

Heat distribution of rod (sampled at dx = 20, dt = 100)
Temperature (celsius)
Location along bar (cm)
t=0s
t=100s
t=200s
t=300s
t=400s
t=500s
t=600s
500
450
400
350
300
250
200
20
30
40
50
60
70
80


Temperature at 20cm point over time
Temperature (celsius)
Time (s)
dx = 20, dt = 100
dx = 10, dt = 100
dx = 20, dt =50
Analytical solution
500
450
400
350
300
250
200
0
100
200
300
400
500
600

Time(s)	Temperature(C) at 20cm along bar			
	$\Delta x = 20, \Delta t = 100$	$\Delta x = 10, \Delta t = 100$	$\Delta x = 20, \Delta t = 50$	Analytical
0	500.00	500.00	500.00	500.00
50	-	-	450.58	-
100	409.46	438.66	410.43	438.66
150	-	-	377.37	-
200	348.63	351.94	349.73	351.94
250	-	-	326.27	-
300	305.14	306.71	306.08	306.71
350	-	-	288.42	-
400	272.06	272.42	272.76	272.42
450	-	-	258.72	-
500	245.46	246.06	245.97	246.06
550	-	-	234.28	-
600	223.12	223.92	223.48	223.92

3 Modelling initial temperature change





Time(s)	Temperature(C) at 20cm along bar
0	438.40
10	438.09
20	436.78
30	433.70
40	428.78
50	422.45
60	415.24

My proposed model is based upon the Implicit Crank-Nicholson method, due to it's unconditional stability over the heat equation. The initial value of x is simply calculated using the supplied distribution, then the program continues the same as the uniform ICN model.

Since the amount of time the temperature is modelled over is significantly shorter than that of the previous two models, Δt is likely to be smaller than Δx. Thus, an argument could be made that the benefits of ICN over EFD are unnecessary, since time steps are likely to be relatively small compared to size steps. Therefore, the EFD method could be argued for due to it's smaller resource footprint and faster execution,

However, I made the decision to use ICN in the interest of versatility and the potential of scaling with different lengths of bar - particularly smaller ones where $\frac{k\Delta t}{\Delta x^2} > \frac{1}{2}$, for which an EFD based model would be unstable.