Massachusetts Institute of Technology

Project Report

18.0851

Computational Science and Engineering I

Written by

Jered Dominguez-Trujillo

Submitted to:

Prof. Matthew Durey

May 9, 2019

1 Problem Description

Three numerical methods - Explicit Euler, Implicit Euler, and Crank-Nicolson - were developed to solve the one-dimensional heat equation and analyzed with respect to their stabilities and respective errors. As this is an analysis of numerical methods rather than an engineering solution, all values will be presented as dimensionless.

One-Dimensional Heat Equation
To correctly simulate unsteady heat transfer in one-dimension numerically, the one-dimensional Heat Equation (Eq. 1) must be discretized and solved.

$$\frac{\partial u}{\partial t} = K\frac{\partial^2 u}{\partial x^2} + q(x,t) \tag{1}$$

Where u represents the temperature, K the thermal diffusivity, and $q(x,t)$ a heat source function. The variables x and t represent space and time, respectively.

Assumptions
To simplify the numerical methods, the following assumptions were made:
1. Uniform Grid Spacing (Δx)
2. Uniform Time Steps (Δt)

Initial Conditions
The initial condition will be assumed that the temperature field, $u(x,t=0)$, is zero at all points in the domain, per Eq. 2.

$$u(x,t=0) = 0 \tag{2}$$

Boundary Conditions
The boundary conditions were defined as "fixed-free" with the fixed (Dirilecht) boundary condition applied at $x = 0$ and the free (Neumann) boundary condiiton applied at $x = L$ per Eq. 3 and Eq. 4, respectively.

Dirilecht Condition at the Left Boundary x = 0:

$$u(x=0,t) = \begin{cases} C_1 * \frac{t}{t_{ramp}} & t \leq t_{ramp} \\ C_1 & t > t_{ramp} \end{cases} \tag{3}$$

Neumann Condition at the Right Boundary x = L:

$$\frac{\partial u}{\partial x}(x=L,t) = C_2 \tag{4}$$

Heating Source Function
The source function, ($q(x,t)$ in Eq. 1) is defined as sinusoidal in time, with frequency set by ω and amplitude set by Q_{max}, and by a normal distribution in space, with a peak at $\frac{L}{2}$ (analog to the mean in a normal distribution) and standard deviation, σ, per Eq. 5.

$$q(x,t) = -Q_{max} * sin(\omega t) * exp\left(\frac{-(x-\frac{L}{2})^2}{\sigma^2}\right) \tag{5}$$

Parameters
The Heat Equation (Eq. 1) was solved on a domain of length $L = 2 * \pi$, with a Thermal Diffusivity constant of $K = 0.1$.

The Dirilecht Boundary Condition on the left ($x = 0$) was set by the constant $C_1 = 1$, and was increased linearly from the initial condition $u(x = 0, t = 0) = 0$ to C_1 over a specified time as $t_{ramp} = 1$. This resulted in the boundary condition described by Eq. 3.

The Nuemann Boundary Condition ($\frac{\partial u}{\partial x}$) on the right ($x = L$) was set by the constant $C_2 = -0.2$. This resulted in the boundary condition described by Eq. 4.

The source function (Eq. 5) is defined to have a maximum amplitude of $Q_{max} = 1$, a frequency of $\omega = q \frac{rad}{s}$ and a distribution (standard deviation) of $sigma = 0.2 * L$.

A table summarizing important parameters used is presented below.

Table 1: Heat Equation Parameters and Boundary Conditions

Length of Domain [L]	2π
Left Boundary Condition [C_1]	1
Right Boundary Condition [C_2]	-0.2
Thermal Diffusivity [K]	0.1
Time Ramp [t_{ramp}]	1
Maximum Heat Source [Q_{max}]	1
Frequency [ω]	$1\frac{rad}{s}$
Distribution of Heat Source [σ]	$0.2 * L$

2 Finite Difference Schemes

For Eq. 6 and Eq. 7, at time t, we define the following $u^{(n)} = \frac{\partial^n u}{\partial x^n}$

First-Order Finite Difference: Central Difference Scheme
The Central Difference Scheme for First-Order Finite Differences is used to apply the Nuemann boundary condition (Eq. 4) using a ghost node located at $L + dx$. The scheme is defined as per Eq. 6

$$u^{(1)} = \frac{u(x + \Delta x) - u(x - \Delta x)}{2\Delta x} + O(\Delta x^2) \tag{6}$$

Second Order Finite Difference
The Second-Order Finite Difference Scheme is used to discretize the spatial term $\frac{\partial^2 u}{\partial x}$ for use in the Explicit Euler, Implicit Euler, and Crank-Nicolson solvers. The scheme is defined as per Eq. 7.

$$u^{(2)}(x) = \frac{u(x + \Delta x) - 2u(x) + u(x - \Delta x)}{(\Delta x)^2} + O(\Delta x^2) \tag{7}$$

3 Explicit Euler

The Explicit Euler scheme is defined as the following, where we let n denote the time step, and i denote the grid point:

$$\frac{u_{n+1} - u_n}{\Delta t} = f(t_n, u_n) \tag{8}$$

Where the temperature across the domain at the next time step is calculated directly from the temperature field across the domain at the current time step.

Expanding Eq. 8, where $f(t_n, u_n)$ is given by the one-dimensional heat equation (Eq. 1) gives:

$$u_{n+1,i} - u_{n,1} = \Delta t\big[K\big(\frac{u_n(x+\Delta x) - 2u_n(x) + u_n(x-\Delta x)}{(\Delta x)^2}\big) + q_n(x)\big]_i \tag{9}$$

Where

$$\begin{aligned} u_n(x+\Delta x) &= u_{n,i+1} \\ u_n(x) &= u_{n,i} \\ u_n(x-\Delta x) &= u_{n,i-1} \end{aligned} \tag{10}$$

And we define the CFL Number to be:

$$CFL = K\frac{\Delta t}{(\Delta x)^2} \tag{11}$$

This gives the **explicit** final numerical solution (Eq. 12) for the temperature field at the next $(n+1)$ timestep, given the tempreature field at the current (n) timestep.

$$\boxed{u_{n+1,i} = CFL * u_{n,i+1} + (1 - 2CFL) * u_{n,i} + CFL * u_{n,i-1} + q_{n,i}\Delta t} \tag{12}$$

4 Implicit Euler

The Implicit Euler scheme is defined as the following, where we let n denote the time step, and i denote the grid point:

$$u_{n+1} = u_n + \Delta t * f(t_{n+1}, u_{n+1}) \tag{13}$$

Unlike the Explicit Euler scheme, the Implicit Euler scheme requires the simultaneous solution of all points in the domain for the next time step.

Expanding Eq. 13 gives:

$$u_{n+1,i} = u_{n,i} + \Delta t\big[K\big(\frac{u_{n+1}(x+\Delta x) - 2u_{n+1}(x) + u_{n+1}(x-\Delta x)}{\Delta x^2}\big) + q_n(x)\big]_i \tag{14}$$

Note that Eq. 10 still applies to Eq. 13 and we still define the CFL number as in Eq. 11. After some algebra, we get the following equation (Eq. 15) for the Implicit Euler Scheme.

$$\boxed{-CFL * u_{n+1,i+1} + (2CFL+1) * u_{n+1,i} - CFL * u_{n+1,i-1} = u_{n,i} + q_{n,i}\Delta t} \tag{15}$$

Boundary Conditions

1) Dirilecht Condition at the Left Boundary x = 0:

$$u(x=0,t) = \begin{cases} C_1 * \frac{t}{t_{ramp}} & t \leq t_{ramp} \\ C_1 & t > t_{ramp} \end{cases} \tag{16}$$

$$-CFL * f_{BC1} + (2CFL+1) * u_{n+1,2} - CFL * u_{n+1,3} = u_{n,2} + q_{n,2}\Delta t \tag{17}$$

$$\boxed{(2CFL+1) * u_{n+1,2} - CFL * u_{n+1,3} = u_{n,2} + q_{n,2}\Delta t + CFL * f_{BC1}} \tag{18}$$

2) Neumann Condition at the Right Boundary x = L:

$$\frac{\partial u}{\partial x}(x=L,t) = C_2 \tag{19}$$

Central Difference

$$\frac{u_{n+1,i+1} - u_{n+1,i-1}}{2\Delta x} = C_2 \tag{20}$$

$$u_{n+1,i+1} = 2C_2\Delta x + u_{n+1,i-1} \tag{21}$$

Where $i+1 = m$ and m represents the last node (ghost node)

$$u_{n+1,m} = 2C_2\Delta x + u_{n+1,m-2} \tag{22}$$

Plugging in

$$-CFL*u_{n+1,m-2} + (1+2CFL)*u_{n+1,m-1} - CFL*(2C_2\Delta x + u_{n+1,m-2}) = u_{n,m-1} + q_{n,m-1}\Delta t \tag{23}$$

$$\boxed{-2CFL * u_{n+1,m-2} + (1 + 2CFL) * u_{n+1,m-1} = u_{n,m-1} + q_{n,m-1}\Delta t + 2CFL * C_2\Delta x} \quad (24)$$

5 Crank-Nicolson

The Crank-Nicolson scheme is defined as the following, where we let n denote the time step, and i denote the grid point

$$u_{n+1,i} = u_{n,i} + \Delta t * \Big[\frac{1}{2}f(t_n, u_n) + \frac{1}{2}f(t_{n+1}, u_{n+1}) + q_n\Big]_i \tag{25}$$

Substituting in the heat equation (Eq. 1),

$$u_{n+1,i} = u_{n,i} + \Delta t * \Big[\frac{1}{2}\big(K\frac{\partial^2 u_n}{\partial x^2}\big) + \frac{1}{2}\big(K\frac{\partial^2 u_{n+1}}{\partial x^2}\big) + q_n\Big]_i \tag{26}$$

And then discretizing,

$$\begin{aligned} u_{n+1,i} = u_{n,i} + \Delta t * \Big[&\frac{1}{2}\big(K\frac{u_n(x+\Delta x) - 2u_n(x) + u_n(x-\Delta x)}{\Delta x^2}\big) \\ &\frac{1}{2}\big(K\frac{u_{n+1}(x+\Delta x) - 2u_{n+1}(x) + u_{n+1}(x-\Delta x)}{\Delta x^2}\big) + q_n\Big]_i \end{aligned} \tag{27}$$

In addition to Eq. 10, the following (Eq. 28 are applied to Eq. 27.

$$\begin{aligned} u_{n+1}(x+\Delta x) &= u_{n+1,i+1} \\ u_{n+1}(x) &= u_{n+1,i} \\ u_{n+1}(x-\Delta x) &= u_{n+1,i-1} \end{aligned} \tag{28}$$

We still define the CFL number as in Eq. 11. After some algebra, we get the following equation (Eq. 29) for the Implicit Euler Scheme.

$$\frac{-CFL}{2} * u_{n+1,i-1} + (1+CFL) * u_{n+1,i} - \frac{-CFL}{2} * u_{n+1,i+1} = \tag{29}$$

$$\frac{CFL}{2} * u_{n,i-1} + (1-CFL) * u_{n,i} + \frac{CFL}{2} * u_{n,i+1} + q_{n,1}\Delta t \tag{30}$$

1) Dirilecht Condition at the Left Boundary x = 0:

$$u(x=0,t) = u_{n+1,1} = \begin{cases} C_1 * \frac{t}{t_{ramp}} & t \le t_{ramp} \\ C_1 & t > t_{ramp} \end{cases} = f_{BC1} \tag{31}$$

$$\begin{aligned} \frac{-CFL}{2} * f_{BC1} + (1+CFL) * u_{n+1,i} - \frac{-CFL}{2} * u_{n+1,i+1} = \\ \frac{CFL}{2} * u_{n,i-1} + (1-CFL) * u_{n,i} + \frac{CFL}{2} * u_{n,i+1} + q_{n,1}\Delta t \end{aligned} \tag{32}$$

$$(1+CFL) * u_{n+1,i} - \frac{CFL}{2} * u_{n+1,i+1} = \tag{33}$$

$$\frac{CFL}{2} * u_{n,i-1} + (1-CFL) * u_{n,i} + \frac{CFL}{2} * u_{n,i+1} + q_{n,1}\Delta t + \frac{CFL}{2} * f_{BC1} \tag{34}$$

2) Neumann Condition at the Right Boundary x = L:

$$\frac{\partial u}{\partial x}(x = L, t) = C_2 \tag{35}$$

Central Difference

$$\frac{u_{n+1,i+1} - u_{n+1,i-1}}{2\Delta x} = C_2 \tag{36}$$

$$u_{n+1,i+1} = 2C_2\Delta x + u_{n+1,i-1} \tag{37}$$

Where $i + 1 = m$ and m represents the last node (ghost node)

$$u_{n+1,m} = 2C_2\Delta x + u_{n+1,m-2} \tag{38}$$

Plugging in

$$\begin{split}\frac{-CFL}{2} * u_{n+1,i-1} + (1 + CFL) * u_{n+1,i} - \frac{CFL}{2} * \big(2C_2\Delta x + u_{n+1,i-1}\big) = \\ \frac{CFL}{2} * u_{n,i-1} + (1 - CFL) * u_{n,i} + \frac{CFL}{2} * u_{n,i+1} + q_{n,1}\Delta t\end{split} \tag{39}$$

$$\boxed{\begin{aligned} -CFL * u_{n+1,i-1} + (1 + CFL) * u_{n+1,i} = \quad (40)\\ \frac{CFL}{2} * u_{n,i-1} + (1 - CFL) * u_{n,i} + \frac{CFL}{2} * u_{n,i+1} + q_{n,1}\Delta t + \frac{CFL}{2} * \big(2C_2\Delta x\big) \quad (41)\end{aligned}}$$

6 Stability

Explicit Euler

Table 2: Explicit Euler Method Stability Tests

Nodes [NX]	Time Steps [NT]	Thermal Diffusivity [K]	Δt	Δx	CFL	Stability
20	500	0.1	0.			Stable
20	200	0.1	0.2	0.31416	0.20264	Stable
20	100	0.1				
20	50	0.1				

Implicit Euler

Table 3: Implicit Euler Method Stability Tests

Nodes [NX]	Time Steps [NT]	Thermal Diffusivity [K]	Δt	Δx	CFL	Stability
20	500	0.1	0.			Stable
20	200	0.1	0.2	0.31416	0.20264	Stable
20	100	0.1				
20	50	0.1				

Crank-Nicolson

Table 4: Crank-Nicolson Method Stability Tests

Nodes [NX]	Time Steps [NT]	Thermal Diffusivity [K]	Δt	Δx	CFL	Stability
20	500	0.1	0.			Stable
20	200	0.1	0.2	0.31416	0.20264	Stable
20	100	0.1				
20	50	0.1				

7 Error

7.1 Spatial Resolution

7.2 Temporal Resolution

8 Conclusions

9 Appendix

MATLAB Code for Numerical Solution of Heat Equation

Listing 1: Numerical Heat Equation

```
%% 18.0851 Project
% Author     : Jered Dominguez-Trujillo
% Date       : May 2, 2019
% Description : Numerical Solution to Heat Equation

% SCHEME = 0 -> EXPLICIT
% SCHEME = 1 -> IMPLICIT
% SCHEME = 2 -> CRANK_NICOLSON

function U = NumHT(SCHEME, BC1, BC2, KT, L, NX, TM, NT, TR, SOURCE_FLAG, W, SIGMA, QMAX)
    clc;

    % Boundary Conditions
    LENGTH = L;                         % Length of Domain
    TIME_RAMP = TR;                     % Time Ramp at X = 0 for BC to Go from 0 to C1
    C1 = BC1;                           % Dirilecht Condition U(x) = C1 at X = 0
    C2 = BC2;                           % Neumann Condition at dU/dX = C2 at X = L
    K = KT;                             % Thermal Diffusivity

    % Default Arguments
    if nargin <= 12
        QMAX = 1;                       % Default Maximum Value of Heat Source Function

        if nargin <= 11
            SIGMA = 0.2 .* LENGTH;       % Default Standard Deviation of Heat Source Function
        end

        if nargin <= 10
            W = 1;                      % Default Frequency of Heat Source Function (rad/s)
        end
    end

    % Spatial Domain
    NODES = NX;                         % Nodes
    DX = LENGTH ./ NODES;                % DX Calculation
    X = linspace(0, LENGTH + DX, NODES + 2);% X Vector with Ghost Node

    % Time Domain
    TMAX = TM;                          % End Time of Simulation
    DT = TMAX ./ NT;                     % DT Calculation
    TIMESTEPS = TMAX ./ DT + 1;          % Number of Time Steps
    T = linspace(0, TMAX, TIMESTEPS);   % Time Vector

    % Calculate CFL Number
    CFL = (DT .* K) ./ (DX .* DX);      % Multiplication Factor K (DT / DX^2)

    % Print Out Simulation Info
    if SCHEME == 0
        fprintf('Explicit Method:\n');
    elseif SCHEME == 1
        fprintf('Implicit Method:\n');
    elseif SCHEME == 2
        fprintf('Crank-Nicolson Method:\n');
    end

    fprintf('Thermal Diffusivity [K]: %.3f\n', K);
```

```
fprintf('BCs:\n(1) U(x) = %.2f At X = 0\n(2) dU/dX = %.2f at X = L = %.2f\n\n', C1, C2, L);
fprintf('Length: %.2f\t\tDX: %.5f\t\tNodes: %.0f\n', LENGTH, DX, NODES);
fprintf('Max Time: %.2f\t\tDT: %.5f\t\t\n\n', TMAX, DT);
if SOURCE_FLAG == 1
    fprintf('Q(x) = - QMAX * SIN(OMEGA * T) * EXP(-(X - L/2)^2 / SIGMA ^ 2)\n');
    fprintf('QMAX: %.2f\t\tOMEGA: %.2f rad/s\t\tSIGMA: %.2f\t\t L: %.2f\n\n', QMAX, W, SIGMA,
         LENGTH);
end
fprintf('CFL Number: %.5f\n', CFL);

% Initialize Matrices
U = zeros(TIMESTEPS, NODES + 2);
Q = zeros(TIMESTEPS, NODES + 2);

% Time Tolerance
eps = 10^-7;

% If Heat Source Flag is True
if SOURCE_FLAG == 1
    for ii = 1:TIMESTEPS
        Q(ii, :) = - QMAX .* (sin((W .* T(ii)))) .* exp(-((X - (LENGTH ./ 2)) .^ 2) ./ (SIGMA .^
             2));
    end
end

% Initialize
CURRENT_T = 0;
TIMESTEP = 0;

% Plot Initial Conditions
fTemperature = figure('Name', 'Temperature History', 'NumberTitle', 'off');
figure(fTemperature);

% Temperature Profile
subplot(211);
plot(X, U(1, :), 'o-');
xlabel('X', 'FontSize', 18); ylabel('T', 'Fontsize', 18);
axis([0 LENGTH -2 2]);

% Heat Source Profile
subplot(212);
plot(X, Q(1, :), 'o-');
xlabel('X', 'FontSize', 18); ylabel('Q', 'Fontsize', 18);
axis([0 LENGTH -QMAX QMAX]);

suptitle(['Time = ', num2str(CURRENT_T), ' s']);
% pause();

% Explicit Scheme
if SCHEME == 0

    % Iterate Until TMAX
    while CURRENT_T < TMAX - eps
        CURRENT_T = CURRENT_T + DT;
        TIMESTEP = TIMESTEP + 1;

        n = TIMESTEP;

        % Explicit Euler Method
        for ii = 2:NODES + 1
            U(n + 1, ii) = CFL .* U(n, ii + 1) + (1 - 2 .* CFL) .* U(n, ii) + CFL .* U(n, ii -
                 1) + Q(n, ii) .* DT;
```

```
        end

        % Boundary Conditions
        U(n + 1, 1) = C1 .* (CURRENT_T / TIME_RAMP) .* (CURRENT_T <= TIME_RAMP) + C1 .*
             (CURRENT_T > TIME_RAMP);
        U(n + 1, NODES + 2) = U(n + 1, NODES) + 2 .* DX .* C2;

        % Plot New Temperature Profile
        figure(fTemperature);

        % Temperature Profile
        subplot(211);
        plot(X, U(n + 1, :), 'o-');
        xlabel('X', 'FontSize', 18); ylabel('T', 'Fontsize', 18);
        axis([0 L -2 2]);

        % Heat Source Profile
        subplot(212);
        plot(X, Q(n + 1, :), 'o-');
        xlabel('X', 'FontSize', 18); ylabel('Q', 'Fontsize', 18);
        axis([0 LENGTH -QMAX QMAX]);

        suptitle(['Time = ', num2str(CURRENT_T), ' s']);

        pause(0.01);
    end

% Implicit Scheme
elseif SCHEME == 1
    LOWER = zeros(1, NODES);
    DIAG  = zeros(1, NODES + 1);
    UPPER = zeros(1, NODES);

    LOWERI = 2:NODES + 1; LOWERJ = 1:NODES;
    DIAGI = 1:NODES + 1; DIAGJ = 1:NODES + 1;
    UPPERI = 1:NODES; UPPERJ = 2:NODES + 1;

    DIAG(1) = 1; UPPER(1) = 0;

    for ii = 2:NODES
        LOWER(ii) = -CFL;
        DIAG(ii) = 2 .* CFL + 1;
        UPPER(ii) = -CFL;
    end

    DIAG(NODES + 1) = 2 .* CFL + 1;

    % Iterate Until TMAX
    while CURRENT_T < TMAX - eps
        CURRENT_T = CURRENT_T + DT;
        TIMESTEP = TIMESTEP + 1;

        n = TIMESTEP;

        RHS = zeros(NODES + 1, 1);

        % Implicit Euler Method
        for ii = 2:NODES + 1
            RHS(ii) = U(n, ii) + Q(n, ii) .* DT;
        end

        % Boundary Conditions
```

```
        fBC1 = C1 .* (CURRENT_T / TIME_RAMP) .* (CURRENT_T <= TIME_RAMP) + C1 .* (CURRENT_T >
            TIME_RAMP);
        LOWER(1) = 0;
        RHS(1) = fBC1;
        RHS(2) = RHS(2) + CFL .* fBC1;

        % Forward Difference Scheme
        % DIAG(NODES + 1) = CFL + 1;
        % RHS(NODES + 1) = RHS(NODES + 1) + C2 .* CFL .* DX;

        % Central Difference Scheme
        L(NODES) = -2 .* CFL;
        RHS(NODES + 1) = RHS(NODES + 1) + 2 .* C2 .* CFL .* DX;

        MA = sparse([LOWERI, DIAGI, UPPERI], [LOWERJ, DIAGJ, UPPERJ], [LOWER, DIAG, UPPER],
            NODES + 1, NODES + 1);

        U(n + 1, 1:end-1) = MA \ RHS;

        % Forward Difference Scheme
        % U(n + 1, end) = U(n + 1, end - 1) + C2 .* DX;

        % Central Difference Scheme
        U(n + 1, end) = U(n + 1, end - 2) + 2 .* C2 .* DX;

        % Plot New Temperature Profile
        figure(fTemperature);

        % Temperature Profile
        subplot(211);
        plot(X, U(n + 1, :), 'o-');
        xlabel('X', 'FontSize', 18); ylabel('T', 'Fontsize', 18);
        axis([0 LENGTH -2 2]);

        % Heat Source Profile
        subplot(212);
        plot(X, Q(n + 1, :), 'o-');
        xlabel('X', 'FontSize', 18); ylabel('Q', 'Fontsize', 18);
        axis([0 LENGTH -QMAX QMAX]);

        suptitle(['Time = ', num2str(CURRENT_T), ' s']);

        pause(0.01);
    end

% Crank-Nicolson Scheme
elseif SCHEME == 2
    LOWER = zeros(1, NODES);
    DIAG  = zeros(1, NODES + 1);
    UPPER = zeros(1, NODES);

    LOWERI = 2:NODES + 1; LOWERJ = 1:NODES;
    DIAGI = 1:NODES + 1; DIAGJ = 1:NODES + 1;
    UPPERI = 1:NODES; UPPERJ = 2:NODES + 1;

    DIAG(1) = 1; UPPER(1) = 0;

    for ii = 2:NODES
        LOWER(ii) = -CFL ./ 2;
        DIAG(ii) = CFL + 1;
        UPPER(ii) = -CFL ./ 2;
    end
```

```
DIAG(NODES + 1) = CFL + 1;

% Iterate Until TMAX
while CURRENT_T < TMAX - eps
    CURRENT_T = CURRENT_T + DT;
    TIMESTEP = TIMESTEP + 1;

    n = TIMESTEP;

    RHS = zeros(NODES + 1, 1);

    % Crank-Nicolson Method
    for ii = 2:NODES + 1
        RHS(ii) = CFL .* U(n, ii - 1) ./ 2 + (1 - CFL) .* U(n, ii) + CFL .* U(n, ii + 1) ./
             2 + Q(n, ii) .* DT;
    end

    % Boundary Conditions
    fBC1 = C1 .* (CURRENT_T / TIME_RAMP) .* (CURRENT_T <= TIME_RAMP) + C1 .* (CURRENT_T >
         TIME_RAMP);
    LOWER(1) = 0;
    RHS(1) = fBC1;
    RHS(2) = RHS(2) + CFL .* fBC1 ./ 2;

    % Forward Difference Scheme
    % DIAG(NODES + 1) = CFL ./ 2 + 1;
    % RHS(NODES + 1) = RHS(NODES + 1) + C2 .* CFL .* DX ./ 2;

    % Central Difference Scheme
    L(NODES) = -CFL;
    RHS(NODES + 1) = RHS(NODES + 1) + C2 .* CFL .* DX;

    MA = sparse([LOWERI, DIAGI, UPPERI], [LOWERJ, DIAGJ, UPPERJ], [LOWER, DIAG, UPPER],
         NODES + 1, NODES + 1);

    U(n + 1, 1:end-1) = MA \ RHS;

    % Forward Difference Scheme
    % U(n + 1, end) = U(n + 1, end - 1) + C2 .* DX;

    % Central Difference Scheme
    U(n + 1, end) = U(n + 1, end - 2) + 2 .* C2 .* DX;

    % Plot New Temperature Profile
    figure(fTemperature);

    % Temperature Profile
    subplot(211);
    plot(X, U(n + 1, :), 'o-');
    xlabel('X', 'FontSize', 18); ylabel('T', 'Fontsize', 18);
    axis([0 LENGTH -2 2]);

    % Heat Source Profile
    subplot(212);
    plot(X, Q(n + 1, :), 'o-');
    xlabel('X', 'FontSize', 18); ylabel('Q', 'Fontsize', 18);
    axis([0 LENGTH -QMAX QMAX]);

    suptitle(['Time = ', num2str(CURRENT_T), ' s']);

    pause(0.01);
```

```
        end
    end

    close all;
end
```