הטכניון – מכון טכנולוגי לישראל

NUMERICAL METHODS IN AEROSPACE ENGINEERING

HOMEWORK ASSIGNMENT x

סמסטר אביב תשפ"ה

SPRING SEMESTER 2025

GRADE	OUT OF	CHAPTER
	2	ABSTRACT
	2	CONTENTS, STYLE &C.
	4	PHYSICAL PROBLEM
	4	MATHEMATICAL MODEL
	26	NUMERICAL METHODS
	20	INFLUENCE OF NUMERICAL METHODS
	20	RESULTS
	2	SUMMARY & CONCLUSIONS
	20	COMPUTER PROGRAM
	100	**TOTAL**

Almog Dobrescu ID 214254252

June 4, 2025

Abstract

Heated tubing has many applications, from water cooling in computers to transporting liquid gas. Research has been conducted to investigate the stress-strain relationships and the material properties of a tube subjected to heating and cooling. A discussion about the optimal parameters was held, and the chosen parameters were used in the results. The results show that the temperature distribution does not depend on the numerical parameter R. The error of the temperature distribution between two convergence criteria decreases exponentially with the decrease of ε.

Contents

List of Figures

Listings

Nomenclature

Δr	difference between two points in space
Δt	difference between two points in time
ε	convergence criteria
i	index in coordinate r
J	final time coordinate
j	index in time
K	diffusivity coefficient
N	number of cell
r	radiaul coordinate
T	temperature
t	time coordinate

1 The Physical Problem

Heated tubing has many applications, from water cooling in computers to transporting liquid gas. Research has been conducted to investigate the stress-strain relationships and the material properties of a tube subjected to heating and cooling.

2 The Mathematical Model

The following heat equation was in use:

$$\frac{1}{4K}\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} \quad , \quad \frac{1}{2} \leq r \leq 1 \quad , \quad 0 < t \tag{1}$$

The boundary and initial conditions of the problem:

$$\begin{aligned} T_{(0.5,t)} &= t \\ T_{(1.0,t)} &= 100 + 40t \\ T_{(r,0)} &= 200\left(r - \frac{1}{2}\right) \\ K &= 0.1 \end{aligned} \tag{2}$$

The strain I is given by the equation:

$$I = \int_{0.5}^{1} \alpha T_{(r,t)} r dr \tag{3}$$

- $\alpha = 10.7$

3 The Numerical Methods

3.1 Finite Differencing

The Heat equation can be rewritten as:

$$\frac{\partial T}{\partial t} = 4K\left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r}\right) \tag{4}$$

By using central differencing for the spatial derivatives and forward differencing for the time derivative we get the explicit scheme:

$$\frac{T_{i,j+1} - T_{i,j}}{\Delta t} + O\left(\Delta t\right) = 4K\left(\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{\Delta r^2} + O\left(\Delta r^2\right) + \frac{1}{r}\frac{T_{i+1,j} - T_{i-1,j}}{2\Delta r} + O\left(\Delta r^2\right)\right)$$

$$\Downarrow$$

$$\begin{aligned} T_{i,j+1} &= T_{i,j} + 4K\Delta t\left(\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{\Delta r^2} + \frac{1}{r}\frac{T_{i+1,j} - T_{i-1,j}}{2\Delta r}\right) + O\left(\Delta t, \Delta r^2\right) \\ T_{i,j+1} &= T_{i,j} + 4KR\left(T_{i-1,j} - 2T_{i,j} + T_{i+1,j} + \frac{\Delta r}{2r}\left(T_{i+1,j} - T_{i-1,j}\right)\right) + O\left(\Delta t, \Delta r^2\right) \end{aligned} \tag{5}$$

- $i = 1, 2, \cdots, N$
- $j = 1, 2, \cdots, J$
- $R = \dfrac{\Delta t}{\Delta r^2}$
- $T_{(i=0,j)} = j \cdot \Delta t$
- $T_{(i=N+1,j)} = 100 + 40 \cdot j \cdot \Delta t$
- $T_{(i,j=0)} = 200\left(r - \frac{1}{2}\right), \; 0 \leq r \leq N+1$

Δr was calculated as follows:

$$\Delta r = \frac{r_{\max} - r_{\min}}{N+1} \tag{6}$$

The step size in time Δt was chosen somewhat arbitrary, only to maintain stability.

The system of equation will be solved by Jacobi method for every j. This method adds an iterative index n:

$$T^{n+1}_{i,j+1} = T^{n}_{i,j} + 4KR\left(T^{n}_{i-1,j} - 2T^{n}_{i,j} + T^{n}_{i+1,j} + \frac{\Delta r}{2r}\left(T^{n}_{i+1,j} - T^{n}_{i-1,j}\right)\right) \tag{7}$$

3.2 Stability Analysis

We will use von-Neumann Stability Analysis. Let's consider the error:

$$e_{r,t} = T_{r,t} - \hat{T}_{r,t} \tag{8}$$

- $T_{r,t}$ - The solution of the differencing equation.
- $\hat{T}_{r,t}$ - The solution of the differencing equation with a small noise in the initial conditions.

Let's define:

$$E_p = T_{p,0} - \hat{T}_{p,0} \tag{9}$$

We can rewrite it as:

$$E_p = \sum_{p=1}^{N} A_p e^{i\beta_n \Delta r p}, \qquad i = \sqrt{-1}, \qquad \beta_n = \frac{n\pi}{N+1} \tag{10}$$

We want to check if there is a mode that diverges. Since the equation is linear we only need one mode to diverge to consider the hole scheme as diverged:

$$E_p = A_p e^{i\beta_n \Delta r p} \tag{11}$$

We need to check how the error behaves over time and to make sure it diminishes to E_p when $t = 0$. Let's assume the error is of the form of:

$$E_{p,j} = A_p e^{i\beta \Delta r p} \cdot e^{\alpha t_j} \tag{12}$$

We are considered stable when $\Re\{\alpha\} < 0$.
We can rewrite the error:

$$E_{p,j} = A_p e^{i\beta \Delta r p} \cdot e^{\alpha \Delta t j} = A_p e^{i\beta \Delta r p} \cdot \xi^j \tag{13}$$

Therefore the stability condition is $|\xi| \leq 1$ Since the differencing equation is linear we can demand the error to satisfy the differencing equation:

$$E_{p,j+1} = E_{p,j} + 4KR\left(E_{p-1,j} - 2E_{p,j} + E_{p+1,j} + \frac{\Delta r}{2r}\left(E_{p+1,j} - E_{p-1,j}\right)\right) \tag{14}$$

After substituting the error we get:

$$e^{i\beta\Delta r p}\xi^{j+1} = e^{i\beta\Delta r p}\xi^{j} + 4KR\left(e^{i\beta\Delta r(p-1)}\xi^{j} - 2e^{i\beta\Delta r p}\xi^{j} + e^{i\beta\Delta r(p+1)}\xi^{j} + \frac{\Delta r}{2r}\left(e^{i\beta\Delta r(p+1)}\xi^{j} - e^{i\beta\Delta r(p-1)}\xi^{j}\right)\right) \tag{15}$$

Dividing by $e^{i\beta\Delta rp}\xi^j$ we get:

$$\begin{aligned}
\xi &= 1+4KR\left(e^{-i\beta\Delta r}-2+e^{i\beta\Delta r}+\frac{\Delta r}{2r}\left(e^{i\beta\Delta r}-e^{-i\beta\Delta r}\right)\right)\\
\xi &= 1+4KR\left(2\cos\left(\beta\Delta r\right)-2+\frac{\Delta r}{2r}2i\sin\left(\beta\Delta r\right)\right)\\
\xi &= 1+8KR\left(\cos\left(\beta\Delta r\right)-1+\frac{\Delta r}{2r}i\sin\left(\beta\Delta r\right)\right)\\
\xi &= 1+8KR\left(\left(1-\frac{\Delta r}{2r}\right)\cos\left(\beta\Delta r\right)-1+\frac{\Delta r}{2r}e^{i\beta\Delta r}\right)
\end{aligned} \tag{16}$$

The stability condition is $|\xi|\leq 1$ therefore:

$$\begin{gathered}
\left|1+8KR\left(\left(1-\frac{\Delta r}{2r}\right)\cos\left(\beta\Delta r\right)-1+\frac{\Delta r}{2r}e^{i\beta\Delta r}\right)\right|\leq 1\\
\left|1+8KR\left(\left(1-\frac{\Delta r}{2r}\right)\cos\left(\beta\Delta r\right)-1+\frac{\Delta r}{2r}\right)\right|\leq 1\\
\left|1+8KR\left(\left(1-\frac{\Delta r}{2r}\right)\cos\left(\beta\Delta r\right)-\left(1-\frac{\Delta r}{2r}\right)\right)\right|\leq 1\\
\left|1+8KR\left(1-\frac{\Delta r}{2r}\right)\left(\cos\left(\beta\Delta r\right)-1\right)\right|\leq 1\\
\left|1-8KR\left(1-\frac{\Delta r}{2r}\right)\sin^2\left(\frac{\beta\Delta r}{2}\right)\right|\leq 1
\end{gathered} \tag{17}$$

The first inequality:

$$\begin{gathered}
-1\leq 1-8KR\left(1-\frac{\Delta r}{2r}\right)\sin^2\left(\frac{\beta\Delta r}{2}\right)\\
\frac{1}{4}\frac{1}{\left(1-\frac{\Delta r}{2r}\right)\sin^2\left(\frac{\beta\Delta r}{2}\right)}\geq KR\\
\frac{1}{4}\frac{1}{\left(1-\frac{\Delta r}{2r}\right)}\geq KR\\
\frac{1}{4-\frac{2\Delta r}{r}}\geq KR\\
\frac{r}{4r-2\Delta r}\geq KR
\end{gathered} \tag{18}$$

The second inequality:

$$1 - 8KR\left(1 - \frac{\Delta r}{2r}\right)\sin^2\left(\frac{\beta \Delta r}{2}\right) \leq 1$$

$$-8KR\left(1 - \frac{\Delta r}{2r}\right)\sin^2\left(\frac{\beta \Delta r}{2}\right) \leq 0 \tag{19}$$

$$\Downarrow$$

Always true

We get that the stability criteria is therefore:

$$R \leq \frac{1}{2K}\frac{r}{2r - \Delta r} \tag{20}$$

We essentially have a range of R where Δr goes to 0 and ∞:

$$0 \leq R \leq \frac{1}{4K} \tag{21}$$

For each Δr and K we have limit for R.

3.3 Convergence Criteria

In order determined if the iterative method for solving the system of equation has converged, we will check if the temperature vector at a specific time has changed from step n to step $n+1$ in the following way:

$$\left|T_{i,j}^{n+1} - T_{i,j}^{n}\right| < \varepsilon \tag{22}$$

3.4 Integral Calculation

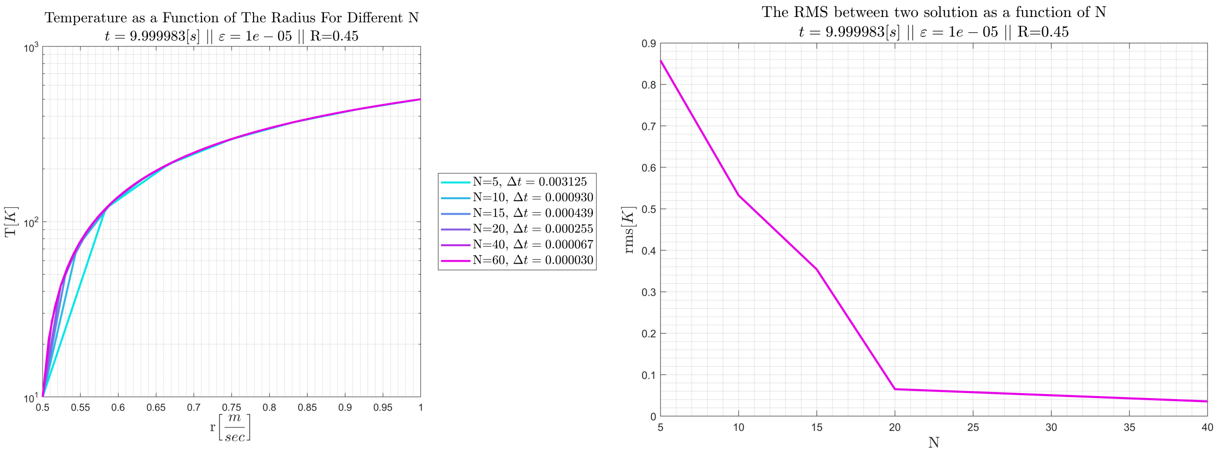
The integral to calculate the strain I will be calculated using trapezoid integration:

$$I = \alpha\frac{h}{4}\sum_{i=0}^{N}\left(T_{(i+1)} + T_{(i)}\right)\left(r_{(i+1)} + r_{(i)}\right) \tag{23}$$

4 Influence of The Numerical Methods

NOTE - the temperature will be presented in logarithmic scale.

4.1 Influence of Number of Elements N



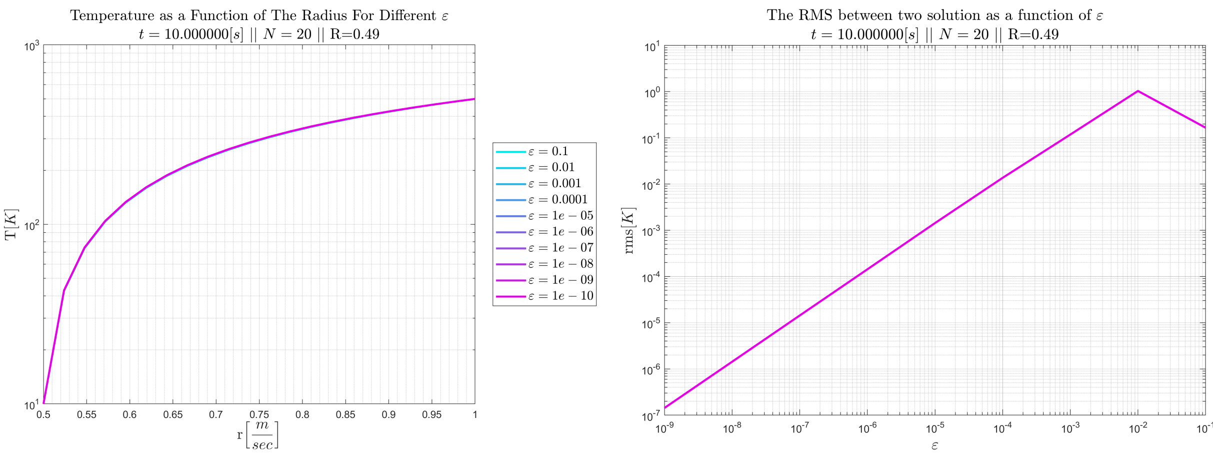
(a) Temperature as a Function of r for different N

(b) RMS as a function of N

Figure 1: Influence of the number of elements N

In Fig.1 we can see that for N bigger than 20, the solution does not really change. We can conclude that $N = 20$ is a sufficient number of elements.

4.2 Influence of Convergence Criteria ε



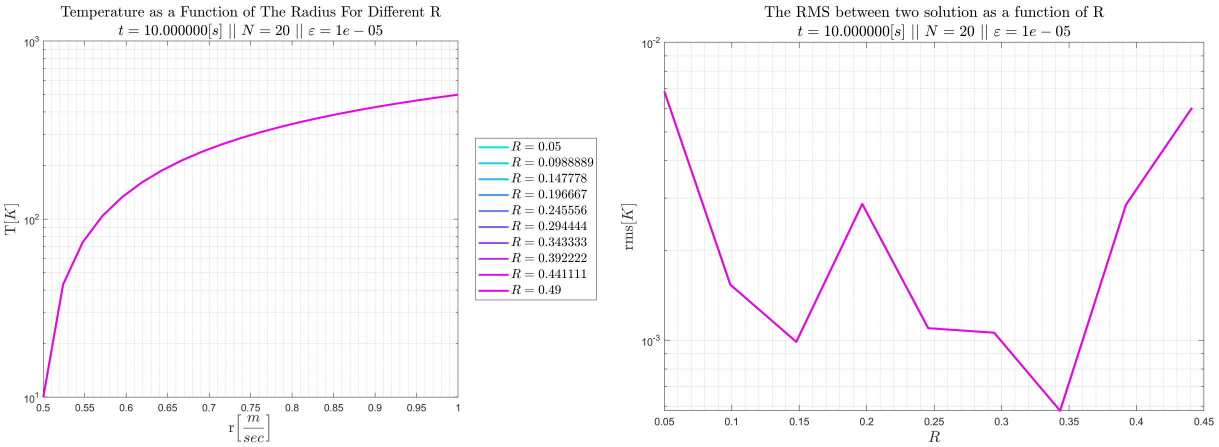
(a) Temperature as a Function of r for different ε

(b) RMS as a function of epsilon

Figure 2: FD - Influence of the convergence criteria ε

From Fig.2 we can conclude that for a convergence criteria smaller than $1e^{-5}$, the solution stays the same. We can determine that $\varepsilon = 1e^{-5}$ is a good choice.

4.3 Influence of The Numerical Parameter R



(a) Temperature as a Function of r for different R

(b) RMS as a function of R

Figure 3: FD - Influence of the convergence criteria R

From Fig.3 we can conclude that the value of the numerical parameter R has close to no effect on the error of the solution. Therefor the chosen R is 0.45.

5 Results and Discussion

5.1 Temperature Distribution

Using the parameters chosen above, the temperature distribution is therefor:

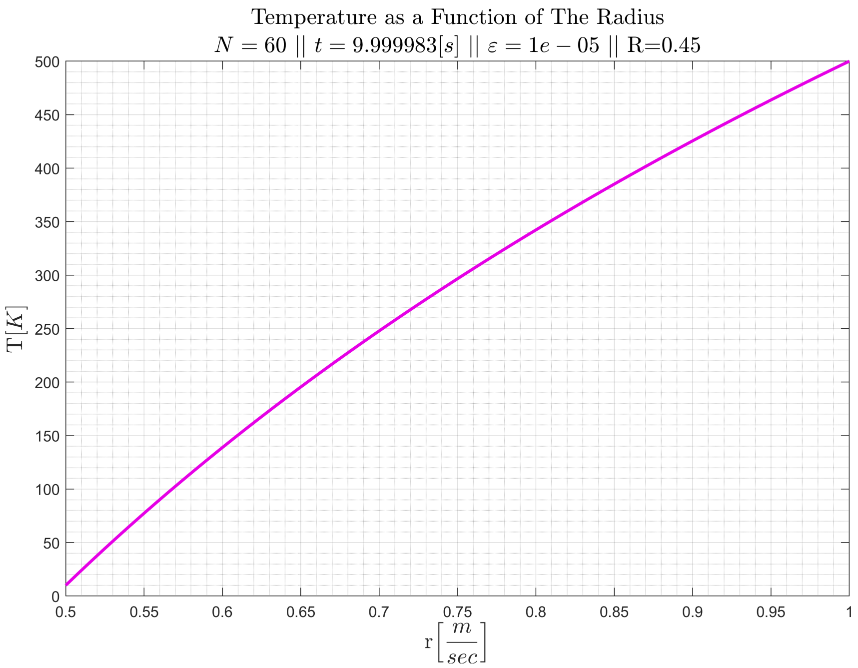


Figure 4: Temperature as a Function of r

We can also plot the temperature distribution over time for all the radiuses:

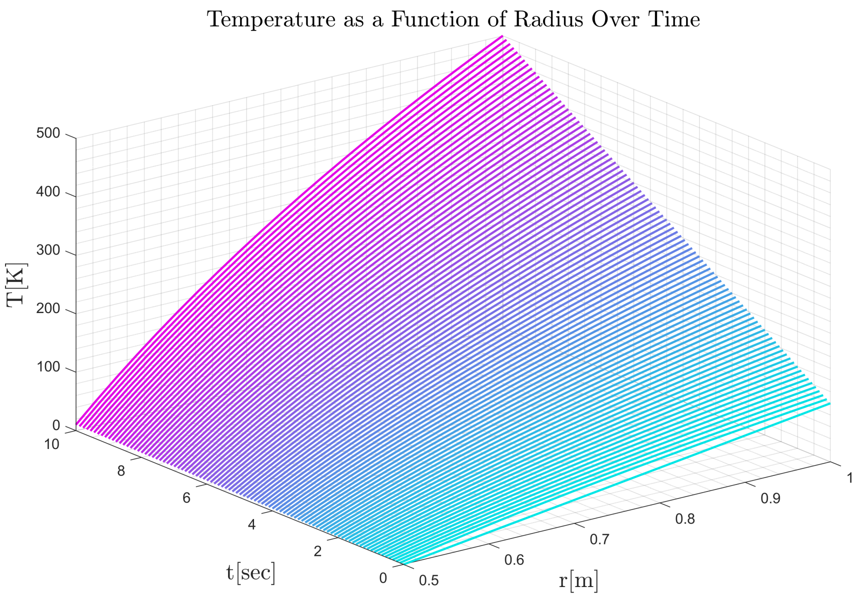


Figure 5: Temperature as a Function of r over time

We can see The temperature increases monotonically according to the boundary conditions. The change in the rate of change can be explained by the fact the the mass of the cylinder does not change linearly with the increases of the radius. The temperature increases over time to match the boundary conditions.

5.2 Strain Calculation

When using the chosen parameters, the calculated strain is:

$$I_{N=20} = 1242.8 \tag{24}$$

When using and $N = 40$ $N = 60$, the strain is:

$$I_{N=40} = 1243.2 \qquad I_{N=60} = 1243.3 \tag{25}$$

We observed that while $N = 20$ is adequate for convergence of the temperature, a higher number of elements is necessary to accurately calculate the strain. This is due to the order of the integration being $O(h)$.

6 Summary and Conclusion

In this assignment, a comprehensive analysis of the numerical parameters needed to solve the one-dimensional heat equation was conducted. The chosen parameters were used in the results section (Sec.5). The following conclusions came to light:

- 20 elements is a sufficient number of elements for convergence of the temperature distribution, however the strain value is not yet converged at such values.
- The RMS of the error decreases exponentially with the decrease of the convergence criteria.
- The numerical parameter R has near to no effect on the convergence of the temperature distribution.

A Listing of The Computer Program

A.1 Parameters

```
N             = 20;
epsilon       = 1e-5;
max_iteration = 1e6;

r_max   = 1;
r_min   = 0.5;
t_start = 0;
t_end   = 10;
K       = 0.1;
alpha   = 10.7;

h       = (r_max - r_min) / (N+1);
R       = 0.49;
% R       = delta_t / h^2;
delta_t = R * h^2;
r       = r_min+h*(0:1:N+1);
```

Listing 1: Parameters file

A.2 Main Code

```
clc; clear; close all;

%% Influenc of N ==========================================================
% ========================================================================
% ========================================================================

parameters
% Ns = [5, 10, 15, 20, 40, 60, 100];
Ns = [3, 5, 7, 10];

results_vec = {};
r_vec  = {};
for Ns_index = 1:length(Ns)
    N       = Ns(Ns_index);
    h       = (r_max - r_min) / (N+1);
    R       = 0.45;
    delta_t = R * h^2;
    r       = r_min+h*(0:1:N+1);

    results_vec{Ns_index,1} = solver(r_min, r, K, h, delta_t, t_start, t_end, N, epsilon,
         max_iteration);
end
%%
load("effect_of_N_to_60.mat")

fig1 = figure('Name', '1','Position', [50, 250, 900, 600]);
size = 15;

colors = cool(length(results_vec(:,1)))*0.9;
for i = length(results_vec(:,1))
    plot(results_vec{i,1}.r, results_vec{i,1}.Ts(end,:), 'LineWidth', 2, 'Color', colors(
        i,:))
```

```
end
xlabel('r$\displaystyle\left[\frac{m}{sec}\right]$', 'FontSize',size, 'Interpreter','latex')
ylabel('T$\left[K\right]$', 'FontSize',size, 'Interpreter','latex')
title('Temperature as a Function of The Radius', 'FontSize',size, 'Interpreter','latex')
subtitle(sprintf('$N=%g$ $||$ $t=%f[s]$ $||$ $\\varepsilon=%g$ $||$ R=%g', results_vec{end,1}.N, results_vec{end,1}.t_vec(end), results_vec{end,1}.epsilon, results_vec{end,1}.R), 'FontSize', size, 'Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig1, 'images/T over r.png','Resolution',400);

% //////////////////////////////////////////////////////////////////////////
% //////////////////////////////////////////////////////////////////////////

fig2 = figure('Name', '2','Position', [150, 250, 900, 600]);
size = 15;

colors = cool(length(results_vec(:,1)))*0.9;
lg = {};
for i = 1:length(results_vec(:,1))
    semilogy(results_vec{i,1}.r, results_vec{i,1}.Ts(end,:), 'LineWidth', 2, 'Color', colors(i,:))
    hold on;
    lg{end+1} = sprintf('N=%g, $\\Delta t=%f$', results_vec{i,1}.N, results_vec{i,1}.delta_t);
end
xlabel('r$\displaystyle\left[\frac{m}{sec}\right]$', 'FontSize',size, 'Interpreter','latex')
ylabel('T$\left[K\right]$', 'FontSize',size, 'Interpreter','latex')
title('Temperature as a Function of The Radius For Different N', 'FontSize',size, 'Interpreter','latex')
subtitle(sprintf('$t=%f[s]$ $||$ $\\varepsilon=%g$ $||$ R=%g', results_vec{end,1}.t_vec(end), results_vec{end,1}.epsilon, results_vec{end,1}.R), 'FontSize', size, 'Interpreter','latex')
legend(lg, 'FontSize',size-2, 'Location','eastoutside','Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig1, 'images/T over r.png','Resolution',400); exportgraphics(fig2, 'images/Influenc of N.png','Resolution',400);

% //////////////////////////////////////////////////////////////////////////
% //////////////////////////////////////////////////////////////////////////

fig3 = figure('Name', '3','Position', [250, 250, 900, 600]);
rms_vec = [];
Ns      = [];
for i = 1:length(results_vec(:,1))-1
    rms_vec(i) = RMS(results_vec{i,1}.r, results_vec{i,1}.Ts(end,:), results_vec{i+1,1}.r, results_vec{i+1,1}.Ts(end,:));
    Ns(i) = results_vec{i,1}.N;
end

plot(Ns, rms_vec, 'LineWidth', 2, 'Color', colors(end,:))

xlabel('N', 'FontSize',size, 'Interpreter','latex')
```

```
ylabel('rms$\left[K\right]$', 'FontSize',size, 'Interpreter','latex')
title('The RMS between two solution as a function of N', 'FontSize',size, 'Interpreter','latex')
subtitle(sprintf('$t=%f[s]$ $||$ $\\varepsilon=%g$ $||$ R=%g', results_vec{end,1}.t_vec(end), results_vec{end,1}.epsilon, results_vec{end,1}.R), 'FontSize', size, 'Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig1, 'images/T over r.png','Resolution',400); exportgraphics(fig2, 'images/Influenc of N.png','Resolution',400); exportgraphics(fig3, 'images/Influenc of N - error.png','Resolution',400);

%% Influenc of epsilon ====================================================
% ==========================================================================
% ==========================================================================

parameters

epsilon_vec = logspace(-1, -10, 10);
% epsilon_vec = logspace(-1, -6, 6);

results_vec = {};
r_vec  = {};
for eps_index = 1:length(epsilon_vec)
    epsilon = epsilon_vec(eps_index);
    results_vec{eps_index,1} = solver(r_min, r, K, h, delta_t, t_start, t_end, N, epsilon, max_iteration);
end
%%
load("effect_of_epsilon_-1_to_-10.mat")

fig4 = figure('Name', '4','Position', [350, 250, 900, 600]);
size = 15;

colors = cool(length(results_vec(:,1)))*0.9;
lg = {};
for i = 1:length(results_vec(:,1))
    semilogy(results_vec{i,1}.r, results_vec{i,1}.Ts(end,:), 'LineWidth', 2, 'Color', colors(i,:))
    hold on;
    lg{end+1} = sprintf('$\\varepsilon=%g$', results_vec{i,1}.epsilon);
end
xlabel('r$\displaystyle\left[\frac{m}{sec}\right]$', 'FontSize',size, 'Interpreter','latex')
ylabel('T$\left[K\right]$', 'FontSize',size, 'Interpreter','latex')
title('Temperature as a Function of The Radius For Different $\varepsilon$', 'FontSize', size, 'Interpreter','latex')
subtitle(sprintf('$t=%f[s]$ $||$ $N=%g$ $||$ R=%g', results_vec{end,1}.t_vec(end), results_vec{end,1}.N, results_vec{end,1}.R), 'FontSize', size, 'Interpreter','latex')
legend(lg, 'FontSize',size-2, 'Location','eastoutside','Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig4, 'images/Influenc of epsilon.png','Resolution',400);

fig5 = figure('Name', '5','Position', [450, 250, 900, 600]);
rms_vec = [];
```

```
eps_vec = [];
for i = 1:length(results_vec(:,1))-1
    rms_vec(i) = RMS(results_vec{i,1}.r, results_vec{i,1}.Ts(end,:), results_vec{i+1,1}.r
        , results_vec{i+1,1}.Ts(end,:));
    eps_vec(i) = results_vec{i,1}.epsilon;
end

loglog(eps_vec, rms_vec, 'LineWidth', 2, 'Color', colors(end,:))

xlabel('$\varepsilon$', 'FontSize',size, 'Interpreter','latex')
ylabel('rms$\left[K\right]$', 'FontSize',size, 'Interpreter','latex')
title('The RMS between two solution as a function of $\varepsilon$', 'FontSize',size, '
    Interpreter','latex')
subtitle(sprintf('$t=%f[s]$ $||$ $N=%g$ $||$ R=%g', results_vec{end,1}.t_vec(end),
    results_vec{end,1}.N, results_vec{end,1}.R), 'FontSize', size, 'Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig4, 'images/Influenc of epsilon.png','Resolution',400); exportgraphics
    (fig5, 'images/Influenc of epsilon - error.png','Resolution',400);

%% Influenc of R ==========================================================
% ========================================================================
% ========================================================================

parameters

% Rs = linspace(0.05,0.49,10);
Rs = linspace(0.45, 0.3, 4);

results_vec = {};
r_vec  = {};
for R_index = 1:length(Rs)
    R = Rs(R_index);
    delta_t = R * h^2;
    results_vec{R_index,1} = solver(r_min, r, K, h, delta_t, t_start, t_end, N, epsilon,
        max_iteration);
end
%%
load("effect_of_R_0.05_to_0.49.mat")

fig6 = figure('Name', '6','Position', [550, 250, 900, 600]);
size = 15;

colors = cool(length(results_vec(:,1)))*0.9;
lg = {};
for i = 1:length(results_vec(:,1))
    semilogy(results_vec{i,1}.r, results_vec{i,1}.Ts(end,:), 'LineWidth', 2, 'Color',
        colors(i,:))
    hold on;
    lg{end+1} = sprintf('$R=%g$', results_vec{i,1}.R);
end
xlabel('r$\displaystyle\left[\frac{m}{sec}\right]$', 'FontSize',size, 'Interpreter','
    latex')
ylabel('T$\left[K\right]$', 'FontSize',size, 'Interpreter','latex')
title('Temperature as a Function of The Radius For Different R', 'FontSize',size, '
    Interpreter','latex')
```

```
subtitle(sprintf('$t=%f[s]$ $||$ $N=%g$ $||$ $\\varepsilon=%g$', results_vec{end,1}.t_vec(end), results_vec{end,1}.N, results_vec{end,1}.epsilon), 'FontSize', size, 'Interpreter','latex')
legend(lg, 'FontSize',size-2, 'Location','eastoutside','Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig6, 'images/Influenc of R.png','Resolution',400);

fig7 = figure('Name', '7','Position', [650, 250, 900, 600]);
rms_vec = [];
R_vec = [];
for i = 1:length(results_vec(:,1))-1
    rms_vec(i) = RMS(results_vec{i,1}.r, results_vec{i,1}.Ts(end,:), results_vec{i+1,1}.r, results_vec{i+1,1}.Ts(end,:));
    R_vec(i) = results_vec{i,1}.R;
end

semilogy(R_vec, rms_vec, 'LineWidth', 2, 'Color', colors(end,:))

xlabel('$R$', 'FontSize',size, 'Interpreter','latex')
ylabel('rms$\left[K\right]$', 'FontSize',size, 'Interpreter','latex')
title('The RMS between two solution as a function of R', 'FontSize',size, 'Interpreter','latex')
subtitle(sprintf('$t=%f[s]$ $||$ $N=%g$ $||$ $\\varepsilon=%g$', results_vec{end,1}.t_vec(end), results_vec{end,1}.N, results_vec{end,1}.epsilon), 'FontSize', size, 'Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig6, 'images/Influenc of R.png','Resolution',400); exportgraphics(fig7, 'images/Influenc of R-error.png','Resolution',400);

%% Integrate ==============================================================
% =========================================================================
% =========================================================================

parameters
result = solver(r_min, r, K, h, delta_t, t_start, t_end, N, epsilon, max_iteration);

% integrate N=20
sum_N20 = 0;
for i = [0:result.N+1-1]+1
    sum_N20 = sum_N20 + (result.Ts(end,i)+result.Ts(end,i+1))*(result.r(i)+result.r(i+1))/2;
end
sum_N20 = sum_N20 * alpha * 0.5 * result.h

% integrate N=40
load("effect_of_N_to_60.mat")
result = results_vec{end-1};
sum_N40 = 0;
for i = [0:result.N+1-1]+1

    sum_N40 = sum_N40 + (result.Ts(end,i)+result.Ts(end,i+1))*(result.r(i)+result.r(i+1))/2;
end
sum_N40 = sum_N40 * alpha * 0.5 * result.h
```

```
% integrate N=60
load("effect_of_N_to_60.mat")
result = results_vec{end};
sum_N60 = 0;
for i = [0:result.N+1-1]+1

    sum_N60 = sum_N60 + (result.Ts(end,i)+result.Ts(end,i+1))*(result.r(i)+result.r(i+1))/2;
end
sum_N60 = sum_N60 * alpha * 0.5 * result.h




%% Functions ==============================================================
% ========================================================================
% ========================================================================

function T = init_fuild(r_min, h, N)
    i = 0:1:N+1;
    r = r_min+h * i;
    T = 200 * (r - 0.5);
end

function T = set_BC(T, t)
    T(1)   = t;
    T(end) = 100 + 40 * t;
end

function T_next = step_space_jacobi(T_current, r, K, h, delta_t, N)
    T_next(1) = T_current(1);
    T_next(N+1+1) = T_current(N+1+1);
    for i = [1:N]+1
        T_next(i) = T_current(i) + 4 * K * delta_t * ((T_current(i+1) - 2 * T_current(i) + T_current(i-1)) / (h^2) + 1 / r(i) * (T_current(i+1) - T_current(i-1)) / (2*h));
        % T_next(i) = T_current(i) + 4 * K * delta_t * ((T_current(i+1) - 2 * T_current(i) + T_next(i-1)) / (h^2) + 1 / r(i) * (T_current(i+1) - T_next(i-1)) / (2*h));
    end
end

function converged = check_convergence(T_next, T_current, epsilon, N)
    converged = true;
    for i = [1:N]+1
        if abs(T_next(i) - T_current(i)) > epsilon
            converged = false;
            return
        end
    end
end

function T_next_t = solve_for_specific_t_jacobi(T_current_t, r, K, h, delta_t, N, epsilon, max_iteration)
    T_current = T_current_t;
    for n = 1:max_iteration
        T_next = step_space_jacobi(T_current, r, K, h, delta_t, N);
```

```
        if check_convergence(T_next, T_current, epsilon, N)
            break
        end
        T_current = T_next;
    end
    T_next_t = T_next;
end

function results = solver(r_min, r, K, h, delta_t, t_start, t_end, N, epsilon,
    max_iteration)
    T_current_t = init_fuild(r_min, h, N);
    t = t_start;
    Ts(1, :) = T_current_t;

    while t <= t_end
        fprintf('N: %d || R: %g || epsilon: %g || delta_t: %g || t: %6.4f/%g\n', N,
            delta_t / h^2, epsilon, delta_t, t, t_end)
        T_current_t = set_BC(T_current_t, t);
        T_next_t = solve_for_specific_t_jacobi(T_current_t, r, K, h, delta_t, N, epsilon,
             max_iteration);
        Ts(end+1, :) = T_next_t;

        t = t + delta_t;
        T_current_t = T_next_t;
    end
    results.Ts      = Ts;
    results.N       = N;
    results.r       = r;
    results.h       = h;
    results.delta_t = delta_t;
    results.epsilon = epsilon;
    results.R       = delta_t / h^2;
    results.t_vec   = delta_t*(0:1:t_end/delta_t);
end

function y = interpulate(x, x_vec, y_vec)
% This function assume an ordered x_vec from low to high
    if x > x_vec(end) || x < x_vec(1)
        fprintf('value out of bounds\n');
        return;
    end

    index_i = 0;
    for i = 1:length(x_vec)
        if x <= x_vec(i)
            index_i = i;
            break;
        end
    end

    if x == x_vec(index_i)
        y = y_vec(index_i);
        return;
    end

    m = (y_vec(index_i+1) - y_vec(index_i)) / (x_vec(index_i+1) - x_vec(index_i));
    b = y_vec(index_i) - x_vec(index_i) * m;

```

```
    y = m * x + b;
end

function rms = RMS(x1, y1, x2, y2)
% This functions calcutlates the RMS value for 1 compared to 2
    y2_len_y1 = [];
    for i = 1:length(x1)
        y2_len_y1(i) = interpulate(x1(i), x2, y2);
    end

    rms = sqrt(sum((y1-y2_len_y1).^2)/length(y1));
end
```

Listing 2: The main file