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 2, 2025

Abstract

In this assignment, a comparison between the finite difference method and the shooting method has been conducted for the evaporation front of fuel spray. A discussion about the optimal parameters was held, and the chosen parameters were used in the results. The results show that the temperature distribution obtained by both methods is the same, however, the runtime of the shooting method is significantly shorter.

Contents

List of Figures

Listings

Nomenclature

α	mass ratio between the liquid fuel and environment oxigen
β	ratio between the latent heat of the liquid fuel and between the chemical reaction heat of liquid vapor with oxigen
Λ	empirical constant of droplet vaporation
ε	convergence limit
h	size of each cell in the domain
i	cell index
m_d	mass fraction of liquid fuel in droplets
N	number of elements
s	temporery variable
T	temperature
T_u	environment temperature of the liquid fuel upstream
T_v	vaporization temperature of the liquid fuel
x	spatial coordinate
$\boxed{\cdot}^{(n)}$	value at time step n

1 The Physical Problem

Liquid rockets are an important part of the future of rocket engine propulsion. To maximize the efficiency of the engine, it is crucial to know the location of the evaporation front. The physical problem at hand is the location of the evaporation front of a fuel spray after the atomization injector.

2 The Mathematical Model

The evaporation front of fuel spray can be described by solving the following equations:

$$\begin{gathered} \frac{d^2T}{d\zeta^2} = \Lambda e^T \left(T_v - T_u + \alpha\beta - \frac{dT}{d\zeta} \right) \\ m_d = \left(\alpha\beta\Lambda e^T \right)^{-1} \frac{d^2T}{d\zeta^2} \end{gathered} \tag{1}$$

The boundary condition of the problem:

$$\begin{array}{lccccc} \zeta \to -\infty : & m_d \to 1 & T & \to & \zeta \cdot (T_v - T_u) \\ \zeta \to +\infty : & T \to \zeta \cdot (T_v - T_u + \alpha\beta) & m_d & \to & 0 \end{array} \tag{2}$$

- According to the defenition of T: $T|_{\zeta=0} = 0$

3 The Numerical Methods

Eq.1 can be rewrite as:

$$\frac{d^2T}{d\zeta^2} + \Lambda e^T \frac{dT}{d\zeta} - \Lambda e^T (T_v - T_u + \alpha\beta) = 0$$

In our case:

- ∞ is at around 30
- $\Lambda = 0.1$
- $T_v = 0.203$
- $T_u = 0.152$
- $\alpha\beta = 0.0234$

To make sure that $T|_{\zeta=0} = 0$, we will solve the ODE in two steps, one from $\zeta \to -\infty$ to $\zeta = 0$ and the second from $\zeta = 0$ to $\zeta \to \infty$.

3.1 Finite Difference Method

Using central difference we can write the difference equations:

$$\begin{gathered} \frac{T_{i+1} - 2T_i + T_{i-1}}{h^2} + \Lambda e^{T_i} \frac{T_{i+1} - T_{i-1}}{2h} - \Lambda e^{T_i} (T_v - T_u + \alpha\beta) = 0 \qquad O\left(h^2\right) \\ i = 1, 2, \cdots, N \qquad h = \frac{\zeta|_{i=N+1} - \zeta|_{i=0}}{N + 1 - 0} \end{gathered} \tag{3}$$

We will use the 'explicit point Jacobi' method:

1. Set the filed with initial condition (linear interpolation).
2. Calculate the temperature at index i and time step *n+1* from the previous time step:

$$T_i^{n+1} = \frac{1}{2}\left(T_{i+1}^n + T_{i-1}^n\right) + \frac{h}{4}\Lambda e^{T_i^n}\left(T_{i+1}^n - T_{i-1}^n\right) - \frac{h^2}{2}\Lambda e^{T_i^n}\left(T_v - T_u + \alpha\beta\right) \tag{4}$$

3. The solution is considered converged when:

$$\left|T_i^{n+1} - T_i^n\right| < \varepsilon \qquad \forall i \in [1, N] \tag{5}$$

3.2 Shooting Method

Let's rewrite Eq.3 as a system of 2 ODE:

$$\begin{cases} \dfrac{dT}{d\zeta} = s & T|_{\zeta\to-\infty} \to \zeta \cdot (T_v - T_u) \\[2ex] \dfrac{ds}{d\zeta} = -\Lambda e^T s - \Lambda e^T\left(T_v - T_u + \alpha\beta\right) & T|_{\zeta\to+\infty} \to \zeta \cdot (T_v - T_u + \alpha\beta) \end{cases} \tag{6}$$

To solve the system of equations using the shooting method, we will guess $s_{(i=0)}^{(n)}$ and solve the system of equations using forward and backward differences (semi-implicit Euler). Namely:

$$\begin{cases} s_{i+1}^{(n)} = \left(-\Lambda e^{T_i^{(n)}} s_i^{(n)} + \Lambda e^{T_i^{(n)}}\left(T_v - T_u + \alpha\beta\right)\right)\cdot h + s_i^{(n)} & O(h) \\[2ex] T_{i+1}^{(n)} = s_i^{(n+1)} \cdot h + T_i^{(n)} & O(h) \end{cases} \quad \begin{array}{l} s_{i=0}^n = s_0^n \\ T_{i=0}^n = \zeta \cdot (T_v - T_u + \alpha\beta) \\ i = 0, 1, \cdots, N \end{array} \tag{7}$$

To correct the guess of $s_{(i=0)}^{(n)}$, let's define:

$$F_{\left(s_{(i=0)}\right)} = T_{(i=N+1)}^{(n)} - T|_{\zeta\to+\infty} \tag{8}$$

- When $F = 0$, the guess of $s_{(i=0)}^{(n)}$ is correct

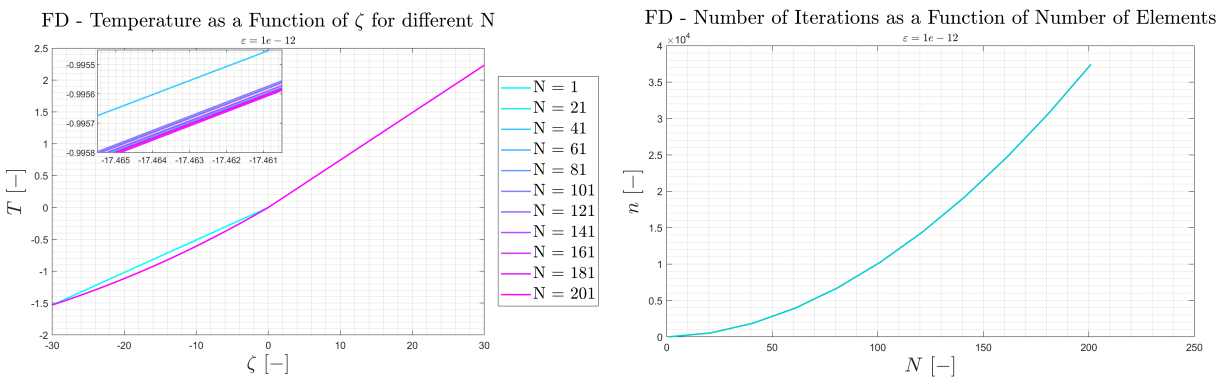
The next guess of s $s_{(i=0)}^{(n+1)}$ will be calculated numerically by using a method to find the root of an equation. Namely:

$$s_{(i=0)}^{(n+1)} = s_{(i=0)}^{(n)} - F_{\left(s_{(i=0)}^{(n)}\right)} \cdot \frac{s_{(i=0)}^{(n)} - s_{(i=0)}^{(n-1)}}{F_{\left(s_{(i=0)}^{(n)}\right)} - F_{\left(s_{(i=0)}^{(n-1)}\right)}} \qquad O(h) \tag{9}$$

4 Influence of The Numerical Methods

4.1 Finite Difference Method

4.1.1 Influence of number of elements N



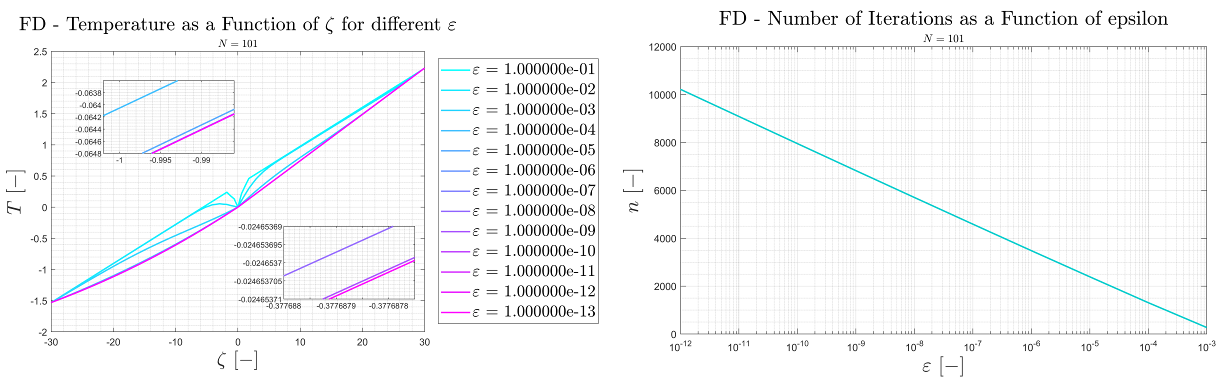
(a) Temperature as a Function of ζ for different N

(b) Number of iterations as a function of N

Figure 1: FD - Influence of the number of elements N

In Fig.1a we can see that for N bigger than 100, the solution does not really change. From Fig.1b we can see that as the number of elements increases, the number of iterations increases as well. We can conclude that $N = 101$ is a sufficient number of elements.

4.1.2 Influence of convergence criteria ε



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

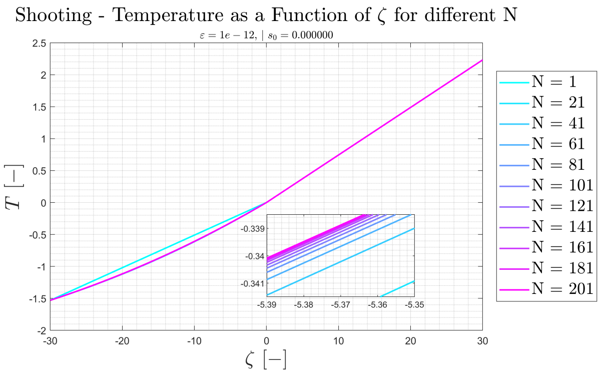
(b) Number of iterations as a function of epsilon

Figure 2: FD - Influence of the convergence criteria ε

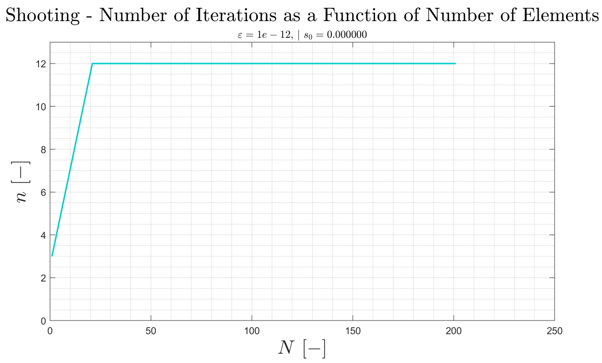
From Fig.2a we can conclude that for a convergence criteria smaller than $1e^{-8}$, the solution stays the same. From Fig.2b we can determine that the number of iterations grows exponentially with the decrease of ε. With this two insights at hand, we can determine that $\varepsilon = 1e^{-12}$ is a good choice (although it is not economical with the number of iterations).

4.2 Shooting Method

4.2.1 Influence of number of elements N



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

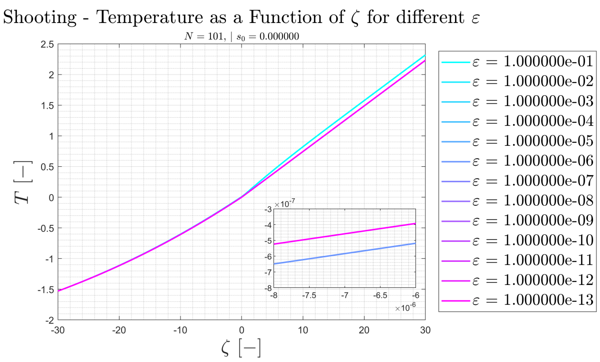


(b) Number of iterations as a function of N

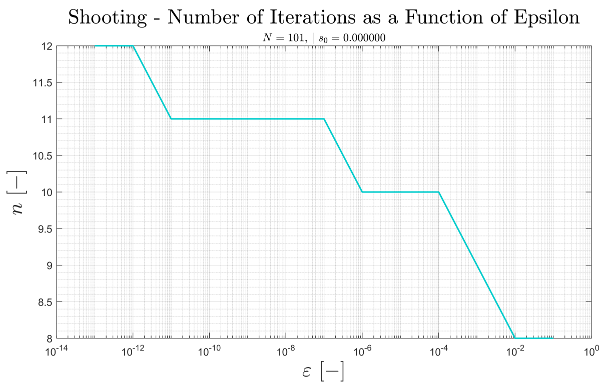
Figure 3: Shooting - Influence of the number of elements N

In Fig.3a we can see that for N bigger than 100, the solution does not really change. From Fig.3b we see that for more than 25 elements, the number of iterations stays constant for a certain convergence criteria and initial condition. We can conclude that $N = 101$ is a sufficient number of elements.

4.2.2 Influence of convergence criteria ε



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

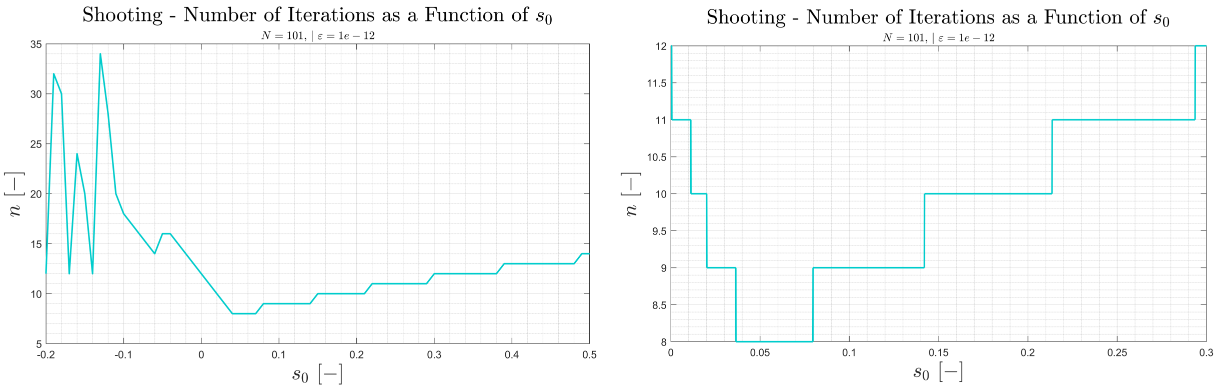


(b) Number of iterations as a function of epsilon

Figure 4: Shooting - Influence of the convergence criteria ε

Figure 4a shows that for a convergence criteria smaller than $1e^{-6}$, the differences between the solutions are because of rounding errors. From Fig.4b we can learn that the number of iterations does not change much for Different convergence criteria. We can determine that $\varepsilon = 1e^{-12}$ is a good choice.

4.2.3 Influence of initial guess s_0



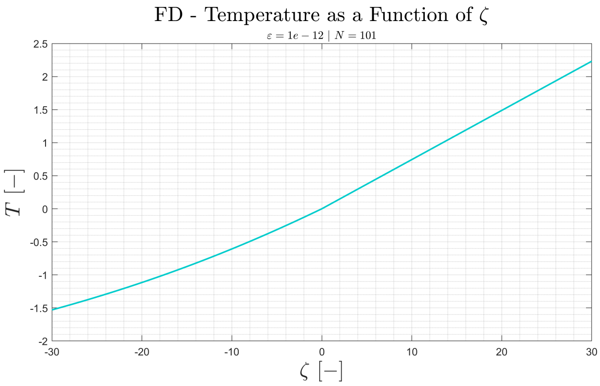
(a) (b)

Figure 5: Number of iterations as a function of initial guess

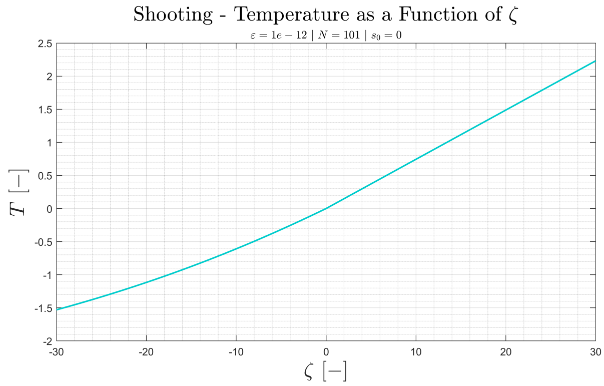
Figure 5 shows that for different negative initial conditions, the number of iterations is not stable. Moreover, we can learn that the real initial slope is around 0.05, as this is the condition for which it took the least amount of steps to converge.

5 Results and Discussion

In the following section, the numerical solution for the ODE will be presented using the parameters chosen in the previous section (Sec.4).



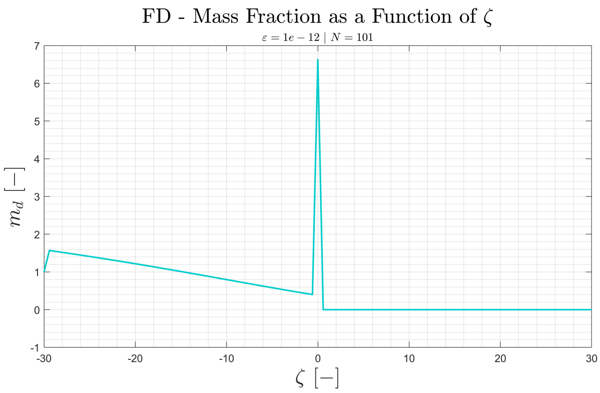
(a) FD - temperature as a function of ζ



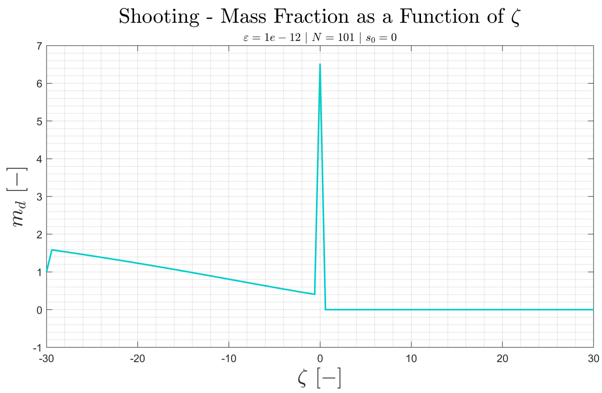
(b) shooting - temperature as a function of ζ

Figure 6: Temperature as a function of ζ

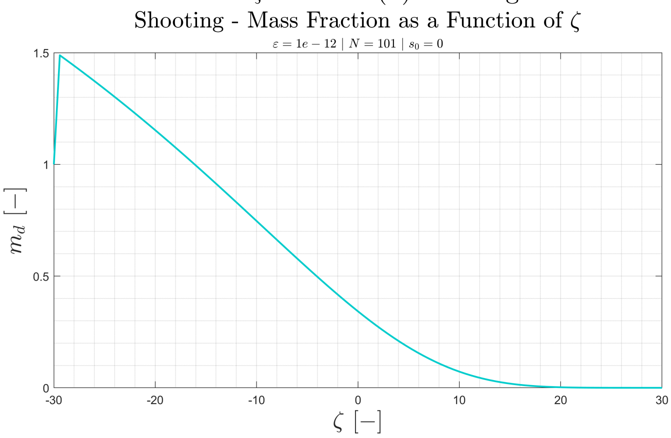
We can see in Fig.6 that there are no differences in the final result between the two methods. With the finite differences method, it took about 10,000 iterations to converge, while with the shooting method, it took only around 10 steps. On the left side of $\zeta = 0$, the solution increases monotonically and faster than linearly. On the right side of $\zeta = 0$, the solution increases linearly with ζ.



(a) FD - mass fraction as a function of ζ



(b) shooting - mass fraction as a function of ζ



(c) shooting - mass fraction as a function of ζ - no zero

Figure 7: Mass fraction as a function of ζ

In Fig.7 we can see that indeed for $\zeta > 0$, the temperature is linear as the mass fraction is constant and depends on the second derivative of the temperature. Moreover, the forced $T|_{\zeta=0} = 0$ creates an artificial peak in the mass fraction, which causes it to have a single discontinuity. Additionally, the left boundary condition is not met as the analytically calculated boundary conditions for the temperature are only correct when $\zeta \to \pm\infty$.

6 Summary and Conclusion

In this assignment, a comparison between the finite difference method and the shooting method has been conducted. A discussion about the optimal parameters was held, and the chosen parameters were used in the result section (Sec.5). The following conclusions came to light:

- The shooting method is more efficient in terms of run time but has a smaller order of local error.
- The number of iterations of the finite difference method increases exponentially with the decrease of the convergence criteria.
- For more than 25 elements, the number of iterations of the shooting methods stays constant.
- The size of the convergence criteria has a minor effect on the number of iterations and the temperature distribution.
- Negative initial guesses of the slope for the shooting method result in drastically changing number of iterations.
- The boundary conditions of the temperature are correct only when $\zeta \to \pm\infty$.
- Forcing $T|_{\zeta=0} = 0$ results in discontinuity in the mass fraction and the second order derivative of the temperature.

A Listing of The Computer Program

A.1 Parameters

```

N       = 101;
epsilon = 1e-12;

lambda     = 0.1;
T_v        = 0.203;
T_u        = 0.152;
alpha_beta = 0.0234;
infinity   = 30;
zeta_max   = infinity;
zeta_min   = -infinity;
h          = (zeta_max - zeta_min) / (N+1);
T_start    = zeta_min * (T_v - T_u);
T_end      = zeta_max * (T_v - T_u + alpha_beta);
md_start   = 1;
md_end     = 0;
```

Listing 1: Parameters file

A.2 Main Code

```
clc; clear; close all;

%%
% Influence of Parameters
% @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
parameters

fig1 = figure ('Position',[0 50 900 500]);
hold all

Ns = 1:20:201;
result = {};
lg = {};
for i = 1:length(Ns)
    N = Ns(i);
    colors = cool(length(Ns));
    h = (zeta_max - zeta_min) / (N+1);
    result{end+1} = finite_difference_method(T_start, T_end, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
    plot(linspace(zeta_min, zeta_max, N+2), result{end}{end,:}, '-', 'LineWidth', 1.5, 'Color', colors(i,:))
    lg{end+1} = sprintf('N = %d', N);
end

size = 20;
title('FD - Temperature as a Function of $\zeta$ for different N','FontSize',size, 'Interpreter','latex');
subtitle(sprintf('$\\varepsilon=%g$', epsilon), 'Interpreter','latex')
ylabel('$T$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$\zeta$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
```

```
box on
legend(lg,'FontSize',size-3 ,'Location','eastoutside', 'Interpreter','latex')

zoom = axes('position',[0.175 0.6 0.275 0.275]);
box on % put box around new pair of axes
hold all
for i = 1:length(Ns)
    N = Ns(i);
    colors = cool(length(Ns));
    h = (zeta_max - zeta_min) / (N+1);
    plot(linspace(zeta_min, zeta_max, N+2), result{i}{end,:}, '-', 'LineWidth', 1.5, 'Color', colors(i,:))
end
zoom.XLim = [-17.4655, -17.4605];
zoom.YLim = [-0.9958, -0.99545];
grid on
grid minor
% exportgraphics(fig1, 'images/FD - T vs zeta for diff N.png','Resolution',400);

%%

parameters

fig2 = figure ('Position',[150 50 900 500]);
hold all

Ns = 1:20:201;
ns = [];
for i = 1:length(Ns)
    N = Ns(i);
    colors = cool(length(Ns));
    h = (zeta_max - zeta_min) / (N+1);
    T_history_FD = finite_difference_method(T_start, T_end, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
    ns(i) = length(T_history_FD(:,1));
end
size = 20;
plot(Ns, ns, '-', 'LineWidth', 1.5, 'Color', cool(1)*0.8)
title('FD - Number of Iterations as a Function of Number of Elements','FontSize',size, 'Interpreter','latex');
subtitle(sprintf('$\\varepsilon=%g$', epsilon), 'Interpreter','latex')
ylabel('$n$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$N$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
box on

% exportgraphics(fig2, 'images/FD - n vs N.png','Resolution',400);

%%

parameters

fig3 = figure ('Position',[300 50 900 500]);

epss = logspace(-3,-12,10);
ns = [];
for i = 1:length(epss)
```

```
    epsilon = epss(i);
    colors = cool(length(epss));
    T_history_FD = finite_difference_method(T_start, T_end, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
    ns(i) = length(T_history_FD(:,1));
end
size = 20;
semilogx(epss, ns, '-', 'LineWidth', 1.5, 'Color', cool(1)*0.8)
title('FD - Number of Iterations as a Function of epsilon','FontSize',size, 'Interpreter','latex');
subtitle(sprintf('$N=%d$', N), 'Interpreter','latex')
ylabel('$n$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$\varepsilon$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
box on

% exportgraphics(fig3, 'images/FD - n vs epsilon.png','Resolution',400);

%%
parameters

fig4 = figure ('Position',[450 50 900 500]);
hold all

epss = logspace(-1,-13,13);
result = {};
lg = {};
for i = 1:length(epss)
    epsilon = epss(i);
    colors = cool(length(epss));
    result{end+1} = finite_difference_method(T_start, T_end, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
    plot(linspace(zeta_min, zeta_max, N+2), result{end}{end,:}, '-', 'LineWidth', 1.5, 'Color', colors(i,:))
    lg{end+1} = sprintf('$\\varepsilon$ = %d', epsilon);
end

size = 20;
title('FD - Temperature as a Function of $\zeta$ for different $\varepsilon$','FontSize',size, 'Interpreter','latex');
subtitle(sprintf('$N=%d$', N), 'Interpreter','latex')
ylabel('$T$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$\zeta$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
box on
legend(lg,'FontSize',size-3 ,'Location','eastoutside', 'Interpreter','latex')

zoom = axes('position',[0.45 0.2 0.2 0.2]);
box on % put box around new pair of axes
for i = 1:length(epss)
    epsilon = epss(i);
    colors = cool(length(epss));
    plot(linspace(zeta_min, zeta_max, N+2), result{i}{end,:}, '-', 'LineWidth', 1.5, 'Color', colors(i,:))
    hold all
end
```

```
zoom.XLim = [-0.377688, -0.37768775];
zoom.YLim = [-0.02465371, -0.02465369];
grid on
grid minor

zoom = axes('position',[0.175 0.6 0.2 0.2]);
box on % put box around new pair of axes
for i = 1:length(epss)
    epsilon = epss(i);
    colors = cool(length(epss));
    plot(linspace(zeta_min, zeta_max, N+2), result{i}{end,:}, '-', 'LineWidth', 1.5, 'Color', colors(i,:))
    hold all
end
zoom.XLim = [-1.002, -0.986];
zoom.YLim = [-0.0648, -0.0636];
grid on
grid minor
% exportgraphics(fig4, 'images/FD - T vs zeta for diff epsilon.png','Resolution',400);

%%
parameters

fig5 = figure ('Position',[0 200 900 500]);
hold all

Ns = 1:20:201;
s0 = 0;
result = {};
lg = {};
for i = 1:length(Ns)
    N = Ns(i);
    colors = cool(length(Ns));
    h = (zeta_max - zeta_min) / (N+1);
    result{end+1} = shooting_method_with_zero(T_start, T_end, s0, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
    plot(linspace(zeta_min, zeta_max, N+2), result{end}{end,:}, '-', 'LineWidth', 1.5, 'Color', colors(i,:))
    lg{end+1} = sprintf('N = %d', N);
end

size = 20;
title('Shooting - Temperature as a Function of $\zeta$ for different N','FontSize',size, 'Interpreter','latex');
subtitle(sprintf('$\\varepsilon=%g$, $|$ $s_0=%f$', epsilon, s0), 'Interpreter','latex')
ylabel('$T$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$\zeta$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
box on
legend(lg,'FontSize',size-3 ,'Location','eastoutside', 'Interpreter','latex')

zoom = axes('position',[0.43 0.2 0.22 0.22]);
box on % put box around new pair of axes
hold all
for i = 1:length(Ns)
    N = Ns(i);
    colors = cool(length(Ns));
```

```matlab
    h = (zeta_max - zeta_min) / (N+1);
    plot(linspace(zeta_min, zeta_max, N+2), result{i}{end,:}, '-', 'LineWidth', 1.5, 'Color', colors(i,:))
end
zoom.XLim = [-5.39, -5.35];
zoom.YLim = [-0.3415, -0.3385];
grid on
grid minor
% exportgraphics(fig5, 'images/shooting - T vs zeta for diff N.png','Resolution',400);

%%

parameters

fig6 = figure ('Position',[150 200 900 500]);
hold all

Ns = 1:20:201;
s0 = 0;
ns = [];
for i = 1:length(Ns)
    N = Ns(i);
    colors = cool(length(Ns));
    h = (zeta_max - zeta_min) / (N+1);
    T_history_FD = shooting_method_with_zero(T_start, T_end, s0, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
    ns(i) = length(T_history_FD(:,1));
end
size = 20;
plot(Ns, ns, '-', 'LineWidth', 1.5, 'Color', cool(1)*0.8)
title('Shooting - Number of Iterations as a Function of Number of Elements','FontSize', size, 'Interpreter','latex');
subtitle(sprintf('$\\varepsilon=%g$, $|$ $s_0=%f$', epsilon, s0), 'Interpreter','latex')
ylabel('$n$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$N$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
box on
ylim([0,13])

% exportgraphics(fig6, 'images/shooting - n vs N.png','Resolution',400);

%%

parameters

fig7 = figure ('Position',[300 200 900 500]);

epss = logspace(-1,-13,13);
s0 = 0;
ns = [];
for i = 1:length(epss)
    epsilon = epss(i);
    colors = cool(length(epss));
    T_history_shooting = shooting_method_with_zero(T_start, T_end, s0, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
    ns(i) = length(T_history_shooting(:,1));
end
```

```
size = 20;
semilogx(epss, ns, '-', 'LineWidth', 1.5, 'Color', cool(1)*0.8)
title('Shooting - Number of Iterations as a Function of Epsilon','FontSize',size, 'Interpreter','latex');
subtitle(sprintf('$N=%d$, $|$ $s_0=%f$', N, s0), 'Interpreter','latex')
ylabel('$n$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$\varepsilon$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
box on

% exportgraphics(fig7, 'images/shooting - n vs epsilon.png','Resolution',400);

%%
parameters

fig8 = figure ('Position',[450 200 900 500]);
hold all

epss = logspace(-1,-13,13);
s0 = 0;
result = {};
lg = {};
for i = 1:length(epss)
    epsilon = epss(i);
    colors = cool(length(epss));
    result{end+1} = shooting_method_with_zero(T_start, T_end, s0, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
    plot(linspace(zeta_min, zeta_max, N+2), result{end}{end,:}, '-', 'LineWidth', 1.5, 'Color', colors(i,:))
    lg{end+1} = sprintf('$\\varepsilon$ = %d', epsilon);
end

size = 20;
title('Shooting - Temperature as a Function of $\zeta$ for different $\varepsilon$','FontSize',size, 'Interpreter','latex');
subtitle(sprintf('$N=%d$, $|$ $s_0=%f$', N, s0), 'Interpreter','latex')
ylabel('$T$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$\zeta$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
box on
legend(lg,'FontSize',size-3 ,'Location','eastoutside', 'Interpreter','latex')

zoom = axes('position',[0.43 0.2 0.22 0.22]);
box on % put box around new pair of axes
hold all
for i = 1:length(epss)
    epsilon = epss(i);
    colors = cool(length(epss));
    plot(linspace(zeta_min, zeta_max, N+2), result{i}{end,:}, '-', 'LineWidth', 1.5, 'Color', colors(i,:))
end
zoom.XLim = [-8e-6, -6e-6];
zoom.YLim = [-8e-7, -3e-7];
grid on
grid minor
```

```matlab
% exportgraphics(fig8, 'images/shooting - T vs zeta for diff epsilon.png','Resolution',400);

%%
parameters

fig9 = figure ('Position',[600 200 900 500]);
hold all

s0s = -0.5:0.1:0.5;
result = {};
lg = {};
for i = 1:length(s0s)
    s0 = s0s(i);
    colors = cool(length(s0s));
    result{end+1} = shooting_method_with_zero(T_start, T_end, s0, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
    plot(linspace(zeta_min, zeta_max, N+2), result{end}{end,:}, '-', 'LineWidth', 1.5, 'Color', colors(i,:))
    lg{end+1} = sprintf('$s_0$ = %7.4f', s0);
end

size = 20;
title('Shooting - Temperature as a Function of $\zeta$ for different $s_0$','FontSize', size, 'Interpreter','latex');
subtitle(sprintf('$N=%d$, $|$ $\\varepsilon=%g$', N, epsilon), 'Interpreter','latex')
ylabel('$T$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$\zeta$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
box on
legend(lg,'FontSize',size-3 ,'Location','eastoutside', 'Interpreter','latex')

zoom = axes('position',[0.43 0.2 0.22 0.22]);
box on % put box around new pair of axes
hold all
for i = 1:length(s0s)
    s0 = s0s(i);
    colors = cool(length(s0s));
    plot(linspace(zeta_min, zeta_max, N+2), result{i}{end,:}, '-', 'LineWidth', 1.5, 'Color', colors(i,:))
end
zoom.XLim = [-6.905842e-6, -6.905838e-6];
zoom.YLim = [-4.5194355e-7, -4.519434e-7];
grid on
grid minor
% exportgraphics(fig9, 'images/shooting - T vs zeta for diff s0.png','Resolution',400);
% exportgraphics(fig8, 'images/shooting - T vs zeta for diff epsilon.png','Resolution',400); exportgraphics(fig7, 'images/shooting - n vs epsilon.png','Resolution',400); exportgraphics(fig6, 'images/shooting - n vs N.png','Resolution',400); exportgraphics(fig5, 'images/shooting - T vs zeta for diff N.png','Resolution',400); exportgraphics(fig4, 'images/FD - T vs zeta for diff epsilon.png','Resolution',400); exportgraphics(fig3, 'images/FD - n vs epsilon.png','Resolution',400); exportgraphics(fig2, 'images/FD - n vs N.png','Resolution',400); exportgraphics(fig1, 'images/FD - T vs zeta for diff N.png','Resolution',400);

%%
parameters
```

```

fig10 = figure ('Position',[750 200 900 500]);

ns = [];
s0s = 0:0.0001:0.3;
lg = {};
for i = 1:length(s0s)
    s0 = s0s(i);
    colors = cool(length(s0s));
    T_history_shooting = shooting_method_with_zero(T_start, T_end, s0, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
    ns(i) = length(T_history_shooting(:,1));
end
size = 20;
plot(s0s, ns, '-', 'LineWidth', 1.5, 'Color', cool(1)*0.8)
title('Shooting - Number of Iterations as a Function of $s_0$','FontSize',size, 'Interpreter','latex');
subtitle(sprintf('$N=%d$, $|$ $\\varepsilon=%g$', N, epsilon), 'Interpreter','latex')
ylabel('$n$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$s_0$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
box on

% exportgraphics(fig10, 'images/shooting - n vs s0 - no negative.png','Resolution',400);

%%
parameters

fig11 = figure ('Position',[750 300 900 500]);

ns = [];
s0s = -0.2:0.01:0.5;
lg = {};
for i = 1:length(s0s)
    s0 = s0s(i);
    colors = cool(length(s0s));
    T_history_shooting = shooting_method_with_zero(T_start, T_end, s0, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
    ns(i) = length(T_history_shooting(:,1));
end
size = 20;
plot(s0s, ns, '-', 'LineWidth', 1.5, 'Color', cool(1)*0.8)
title('Shooting - Number of Iterations as a Function of $s_0$','FontSize',size, 'Interpreter','latex');
subtitle(sprintf('$N=%d$, $|$ $\\varepsilon=%g$', N, epsilon), 'Interpreter','latex')
ylabel('$n$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$s_0$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
box on

% exportgraphics(fig11, 'images/shooting - n vs s0 - with negative.png','Resolution',400);

%%
% Results
% @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
parameters

fig12 = figure ('Position',[0 100 900 500]);

T_history = finite_difference_method(T_start, T_end, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
plot(linspace(zeta_min, zeta_max, N+2), T_history{end,:}, '-', 'LineWidth', 1.5, 'Color', cool(1)*0.8)

size = 20;
title('FD - Temperature as a Function of $\zeta$','FontSize',size, 'Interpreter','latex');
subtitle(sprintf('$\\varepsilon=%g$ $|$ $N=%d$', epsilon, N), 'Interpreter','latex')
ylabel('$T$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$\zeta$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig12, 'images/FD - T vs zeta.png','Resolution',400);

%%
parameters

s0 = 0;

fig13 = figure ('Position',[150 100 900 500]);

T_history = shooting_method_with_zero(T_start, T_end, s0, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
plot(linspace(zeta_min, zeta_max, N+2), T_history{end,:}, '-', 'LineWidth', 1.5, 'Color', cool(1)*0.8)

size = 20;
title('Shooting - Temperature as a Function of $\zeta$','FontSize',size, 'Interpreter','latex');
subtitle(sprintf('$\\varepsilon=%g$ $|$ $N=%d$ $|$ $s_0=%g$', epsilon, N, s0), 'Interpreter','latex')
ylabel('$T$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$\zeta$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig13, 'images/shooting - T vs zeta.png','Resolution',400);

%%
parameters

fig14 = figure ('Position',[300 100 900 500]);

T_history = finite_difference_method(T_start, T_end, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
md = calc_md(T_history{end,:}, md_start, md_end, alpha_beta, lambda, h, N);
plot(linspace(zeta_min, zeta_max, N+2), md, '-', 'LineWidth', 1.5, 'Color', cool(1)*0.8)

size = 20;
title('FD - Mass Fraction as a Function of $\zeta$','FontSize',size, 'Interpreter','latex');
subtitle(sprintf('$\\varepsilon=%g$ $|$ $N=%d$', epsilon, N), 'Interpreter','latex')
```

```
ylabel('$m_d$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$\zeta$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig14, 'images/FD - md vs zeta.png','Resolution',400);

%%
parameters

s0 = 0;

fig15 = figure ('Position',[450 100 900 500]);

T_history = shooting_method_with_zero(T_start, T_end, s0, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
md = calc_md(T_history{end,:}, md_start, md_end, alpha_beta, lambda, h, N);
plot(linspace(zeta_min, zeta_max, N+2), md, '-', 'LineWidth', 1.5, 'Color', cool(1)*0.8)

size = 20;
title('Shooting - Mass Fraction as a Function of $\zeta$','FontSize',size, 'Interpreter','latex');
subtitle(sprintf('$\\varepsilon=%g$ $|$ $N=%d$ $|$ $s_0=%g$', epsilon, N, s0), 'Interpreter','latex')
ylabel('$m_d$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$\zeta$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig15, 'images/shooting - md vs zeta.png','Resolution',400);

%%
parameters

s0 = 0;

fig16 = figure ('Position',[600 100 900 500]);

T_history = shooting_method(T_start, T_end, s0, lambda, alpha_beta, T_v, T_u, epsilon, h, N);
md = calc_md(T_history{end,:}, md_start, md_end, alpha_beta, lambda, h, N);
plot(linspace(zeta_min, zeta_max, N+2), md, '-', 'LineWidth', 1.5, 'Color', cool(1)*0.8)

size = 20;
title('Shooting - Mass Fraction as a Function of $\zeta$','FontSize',size, 'Interpreter','latex');
subtitle(sprintf('$\\varepsilon=%g$ $|$ $N=%d$ $|$ $s_0=%g$', epsilon, N, s0), 'Interpreter','latex')
ylabel('$m_d$ $[-]$','FontSize',size, 'Interpreter','latex')
xlabel('$\zeta$ $[-]$','FontSize',size, 'Interpreter','latex')
grid on
grid minor
box on
% exportgraphics(fig16, 'images/shooting - md vs zeta - no zero.png','Resolution',400);
% exportgraphics(fig16, 'images/shooting - md vs zeta - no zero.png','Resolution',400); exportgraphics(fig15, 'images/shooting - md vs zeta.png','Resolution',400); exportgraphics(fig14, 'images/FD - md vs zeta.png','Resolution',400); exportgraphics(fig13, 'images/shooting - T vs zeta.png','Resolution',400); exportgraphics(fig12, '
```

```
    images/FD - T vs zeta.png','Resolution',400); exportgraphics(fig11, 'images/shooting - n vs s0 - with negative.png','Resolution',400); exportgraphics(fig10, 'images/shooting - n vs s0 - no negative.png','Resolution',400); exportgraphics(fig8, 'images/shooting - T vs zeta for diff epsilon.png','Resolution',400); exportgraphics(fig7, 'images/shooting - n vs epsilon.png','Resolution',400); exportgraphics(fig6, 'images/shooting - n vs N.png','Resolution',400); exportgraphics(fig5, 'images/shooting - T vs zeta for diff N.png','Resolution',400); exportgraphics(fig4, 'images/FD - T vs zeta for diff epsilon.png','Resolution',400); exportgraphics(fig3, 'images/FD - n vs epsilon.png','Resolution',400); exportgraphics(fig2, 'images/FD - n vs N.png','Resolution',400); exportgraphics(fig1, 'images/FD - T vs zeta for diff N.png','Resolution',400);








%%
% Functions
% @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

function [T_history] = finite_difference_method(T_start, T_end, lambda, alpha_beta, T_v, T_u, epsilon, h, N)
    % check that number of elements is odd
    if mod(N,2) ~= 1
        fprintf('N must be odd. It is: %d\n', N);
        T_history = 0;
        return
    end
    % calc index of zeta = 0
    index_of_zero = (N + 1) / 2 + 1; % the plus one after the fraction is because of matlab
    % set initial guess of the temperature
    T_current = T_start + (T_end - T_start) / (N+1) * [0:N+1];
    % add the current temperature into the temperature history
    T_history{1,:} = T_current;
    % the main loop of the solver
    for i = 1:1e6
        i
        % getting the updated temprature vector
        T_next = finite_difference_method_step(T_current, lambda, alpha_beta, T_v, T_u, index_of_zero, h, N);
        % add the current temperature into the temperature history
        T_history{end+1,:} = T_next;
        % check convergence
        if check_convergence_finite_difference_method(T_next, T_current, epsilon, N)
            break
        end
        % updating the temperature field
        T_current = T_next;
    end
end

function converged = check_convergence_finite_difference_method(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] = finite_difference_method_step(T_current, lambda, alpha_beta, T_v, T_u, index_of_zero, h, N)
    % setting boundary conditions
    T_next(1) = T_current(1);
    T_next(N+1+1) = T_current(N+1+1);

    % setting the temperature at zeta = 0 to be zero
    T_next(index_of_zero) = 0;

    % preforming the step
    for i = [1:N]+1
        % keeping the temperature at zeta = 0 to be zero
        if i == index_of_zero
            continue;
        end
        T_next(i) = 0.5*(T_current(i+1) + T_current(i-1)) + 0.25 * h * lambda * exp(T_current(i)) * (T_current(i+1) - T_current(i-1)) - 0.5 * h^2 * lambda * exp(T_current(i)) * (T_v - T_u + alpha_beta);
    end
end

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

function [T] = shooting_method_step(T0, s0, lambda, alpha_beta, T_v, T_u, h, N)
    T(1) = T0;
    s(1) = s0;
    for i = [0:N]+1 % semi implicit Euler
        s(i+1) = lambda * exp(T(i)) * ((T_v - T_u + alpha_beta) - s(i)) * h + s(i);
        T(i+1) = s(i+1) * h + T(i);
    end
end

function [s0_next] = guess_next_s(s0_current, s0_past, F_current, F_past)
    s0_next = s0_current - F_current * (s0_current - s0_past) / (F_current - F_past);
end

function [T_history] = shooting_method(T_start, T_end, s0, lambda, alpha_beta, T_v, T_u, epsilon, h, N)
    F = [];
    s0_s = [];
    % setting initial guess
    s0_s(1) = s0;
    % the main loop of the solver
    for i = 1:1e6
        i
        % getting temperature vector according to the latest initial guess
        T = shooting_method_step(T_start, s0_s(end), lambda, alpha_beta, T_v, T_u, h, N);
        % add the current temperature into the temperature history
        if i ==1
            T_history{1,:} = T;
```

```
        else
            T_history{end+1,:} = T;
        end
        % calculating the temperature difference at the end
        F(end+1) = T(end) - T_end;
        % checking if the size of the difference is small enought
        if abs(F(end)) < epsilon
            break
        end
        % updating the initial guess of s
        if i == 1
            s0_s(end+1) = guess_next_s(s0_s(end), s0_s(end)-1, F(end), F(end)-1); % gussing initial slop of 1
        else
            s0_s(end+1) = guess_next_s(s0_s(end), s0_s(end-1), F(end), F(end-1));
        end
    end
end

function [T_history] = shooting_method_with_zero(T_start, T_end, s0, lambda, alpha_beta, T_v, T_u, epsilon, h, N)
    % check that number of elements is odd
    if mod(N,2) ~= 1
        fprintf('N must be odd. It is: %d\n', N);
        T_history = 0;
        return
    end
    % calc index of zeta = 0
    index_of_zero = (N + 1) / 2 + 1; % the plus one after the fraction is because of matlab
    % shooting from the initial temperature to zeta = 0 and setting T = 0
    % there
    [T_history_to_zero] = shooting_method(T_start, 0, s0, lambda, alpha_beta, T_v, T_u, epsilon, h, (N+1)/2-1);
    % shooting from zeta = 0 wher T = 0 to the final temperature
    [T_history_from_zero] = shooting_method(0, T_end, s0, lambda, alpha_beta, T_v, T_u, epsilon, h, (N+1)/2-1);
    % setting the number of steps for each side
    len_to_zero = length(T_history_to_zero);
    len_from_zero = length(T_history_from_zero);
    % zipping the history of the temperature vectors according to the number of steps it
    % took
    if len_to_zero > len_from_zero
        for i = 1:len_from_zero
            T_history{i,:} = zip_two_arrays(T_history_to_zero{i,:}, T_history_from_zero{i,:}, 1);
        end
        for i = len_from_zero+1:len_to_zero
            T_history{i,:} = zip_two_arrays(T_history_to_zero{i,:}, T_history_from_zero{end,:}, 1);
        end
    elseif len_from_zero > len_to_zero
        for i = 1:len_to_zero
            T_history{i,:} = zip_two_arrays(T_history_to_zero{i,:}, T_history_from_zero{i,:}, 1);
        end
        for i = len_to_zero+1:len_from_zero
```

```
            T_history{i,:} = zip_two_arrays(T_history_to_zero{end,:}, T_history_from_zero{i,:}, 1);
        end
    else
        for i = 1:len_to_zero
            T_history{i,:} = zip_two_arrays(T_history_to_zero{i,:}, T_history_from_zero{i,:}, 1);
        end
    end
end

function [array] = zip_two_arrays(a1, a2, num_of_overlap)
    array = [a1,a2(num_of_overlap+1:end)];
end

function md = calc_md(T, md_start, md_end, alpha_beta, lambda, h, N)
    md(1) = md_start;
    md(N+1+1) = md_end;
    for i = [1:N]+1
        md(i) = (alpha_beta * lambda * exp(T(i)))^(-1) * (T(i+1) - 2 * T(i) + T(i-1)) / h^2;
    end
end
```

Listing 2: The main file