



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 3, 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

1	The Physical Problem	1
2	The Mathematical Model	1
3	The Numerical Methods	1
3.1	Finite Differencing	1
3.2	Convergence Criteria	2
3.3	Integral Calculation	2
4	Influence of The Numerical Methods	2
4.1	Influence of Number of Elements N	2
4.2	Influence of Convergence Criteria ε	3
4.3	Influence of The Numerical Parameter R	3
5	Results and Discussion	4
5.1	Temperature Distribution	4
5.2	Strain Calculation	5
6	Summary and Conclusion	5
A	Listing of The Computer Program	6
A.1	Parameters	6
A.2	Main Code	6

List of Figures

1	Influence of the number of elements N	2
2	FD - Influence of the convergence criteria ε	3
3	FD - Influence of the convergence criteria R	3
4	Temperature as a Function of r	4
5	Temperature as a Function of r over time	4

Listings

1	Parameters file	6
2	The main file	6

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	radial 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:

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

The boundary and initial conditions of the problem:

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

The strain I is given by the equation:

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

- $\alpha = 10.7$

3 The Numerical Methods

3.1 Finite Differencing

The Heat equation can be rewritten as:

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

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

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

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

Δr was calculated as follows:

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

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 :

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

3.2 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:

$$|T_{i,j}^{n+1} - T_{i,j}^n| < \varepsilon \quad (8)$$

3.3 Integral Calculation

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

$$I = \alpha \frac{h}{4} \sum_{i=0}^N (T_{(i+1)} + T_{(i)}) (r_{(i+1)} + r_{(i)}) \quad (9)$$

4 Influence of The Numerical Methods

NOTE - the temperature will be presented in logarithmic scale.

4.1 Influence of Number of Elements N

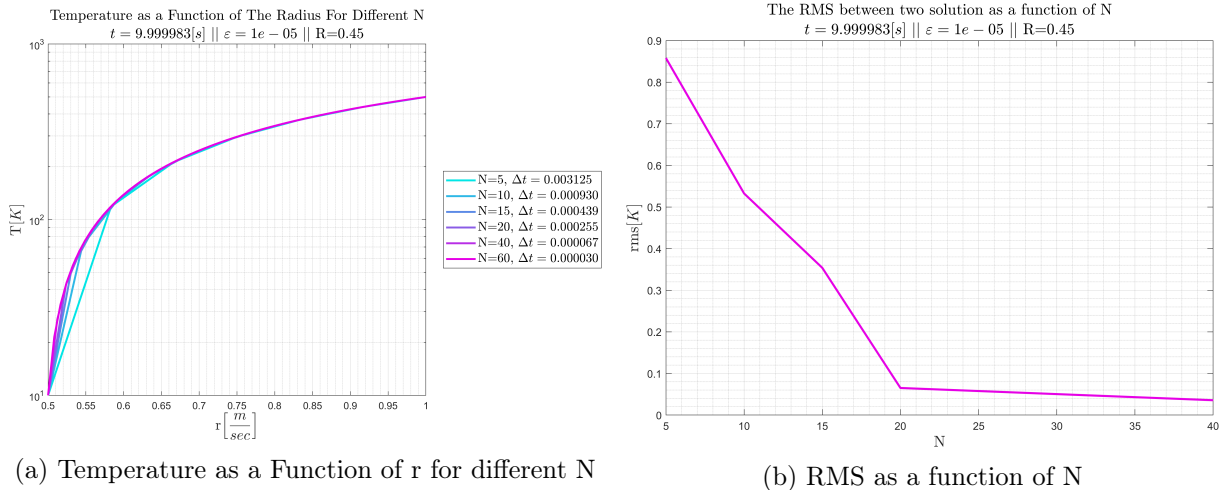


Figure 1: Influence of the number of elements N

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

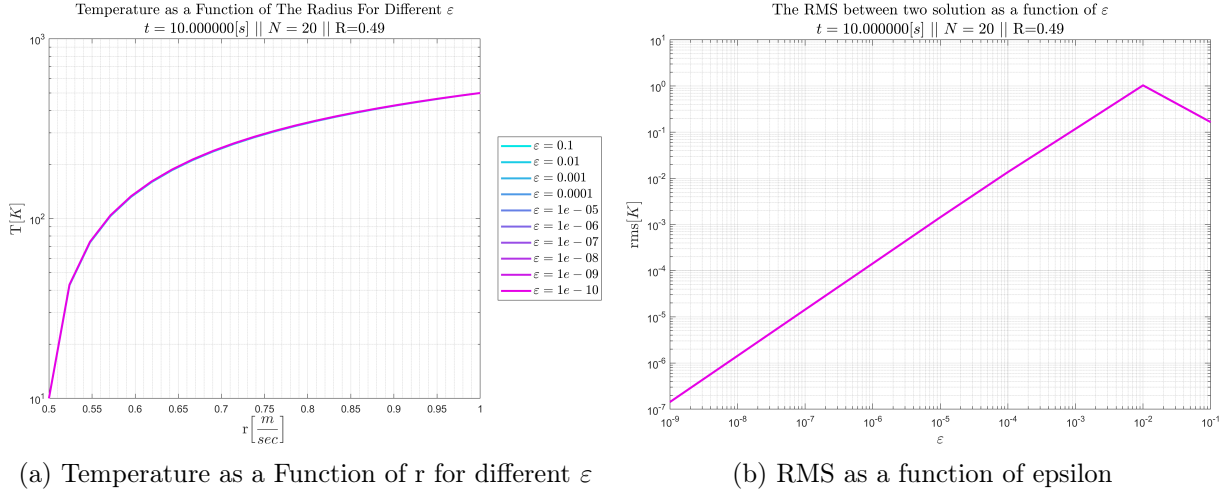


Figure 2: FD - Influence of the convergence criteria ε

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

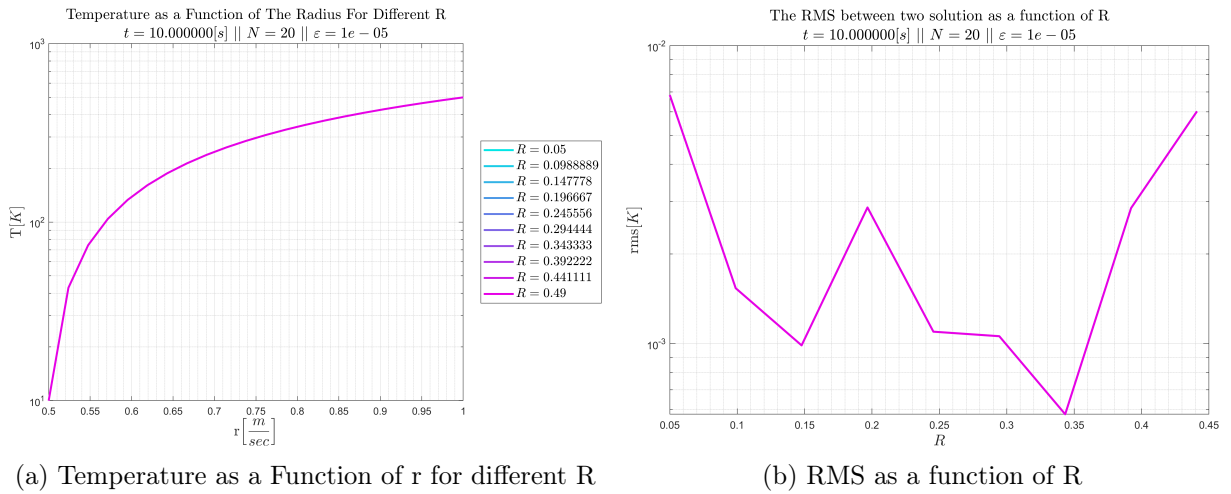


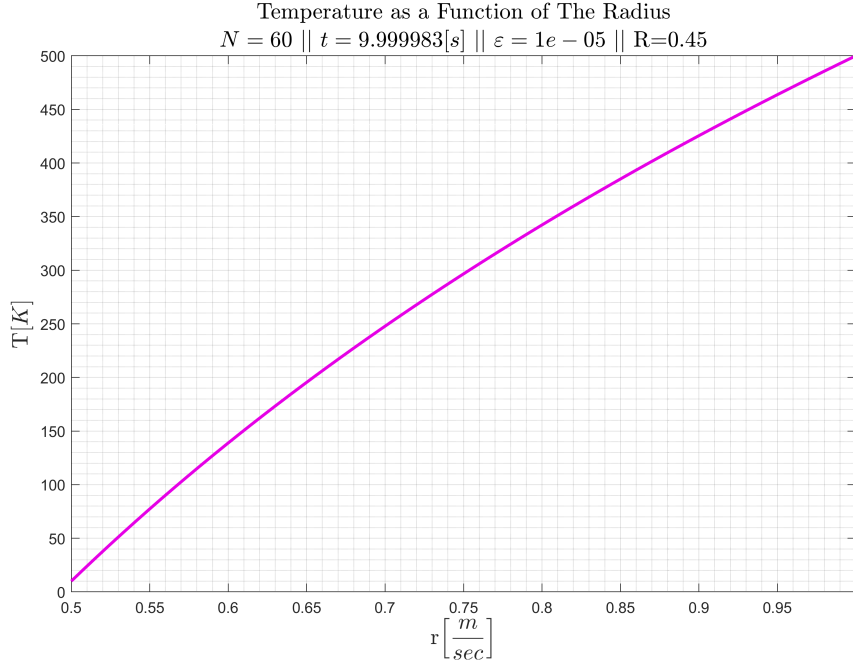
Figure 3: FD - Influence of the convergence criteria R

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. Therefore the chosen R is 0.45.

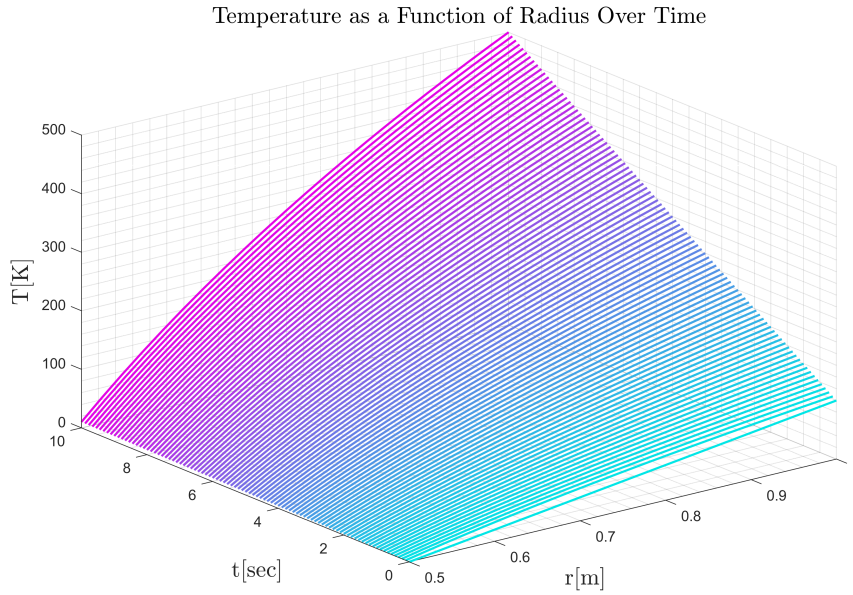
5 Results and Discussion

5.1 Temperature Distribution

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



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





5.2 Strain Calculation

When using the chosen parameters, the calculated strain is:

$$I_{N=20} = 1242.8 \quad (10)$$

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

$$I_{N=40} = 1243.2 \quad I_{N=60} = 1243.3 \quad (11)$$

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

```

1  N          = 20;
2  epsilon    = 1e-5;
3  max_iteration = 1e6;
4
5  r_max      = 1;
6  r_min      = 0.5;
7  t_start    = 0;
8  t_end      = 10;
9  K          = 0.1;
10 alpha      = 10.7;
11
12 h          = (r_max - r_min) / (N+1);
13 R          = 0.49;
14 % R        = delta_t / h^2;
15 delta_t    = R * h^2;
16 r          = r_min+h*(0:1:N+1);

```

Listing 1: Parameters file

A.2 Main Code

```

1  clc; clear; close all;
2
3  %% Influenc of N =====
4  % =====
5  % =====
6
7  parameters
8  % Ns = [5, 10, 15, 20, 40, 60, 100];
9  Ns = [3, 5, 7, 10];
10
11 results_vec = {};
12 r_vec = {};
13 for Ns_index = 1:length(Ns)
14     N      = Ns(Ns_index);
15     h      = (r_max - r_min) / (N+1);
16     R      = 0.45;
17     delta_t = R * h^2;
18     r      = r_min+h*(0:1:N+1);
19
20     results_vec{Ns_index,1} = solver(r_min, r, K, h, delta_t, t_start, t_end, N, epsilon,
        max_iteration);
21 end
22 %%
23 load("effect_of_N_to_60.mat")
24
25 fig1 = figure('Name', '1', 'Position', [50, 250, 900, 600]);
26 size = 15;
27
28 colors = cool(length(results_vec(:,1)))*0.9;
29 for i = length(results_vec(:,1))
30     plot(results_vec{i,1}.r, results_vec{i,1}.Ts(end,:), 'LineWidth', 2, 'Color', colors(
        i,:))

```



```

31 end
32 xlabel('r$\displaystyle\left[\frac{m}{sec}\right]$', 'FontSize',size, 'Interpreter','
    latex')
33 ylabel('T$\left[K\right]$', 'FontSize',size, 'Interpreter','latex')
34 title('Temperature as a Function of The Radius', 'FontSize',size, 'Interpreter','latex')
35 subtitle(sprintf('$N=%g$ $||$ $t=%f[s]$ $||$ $\backslash\varepsilon=%g$ $||$ $R=%g$', 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')
36 grid on
37 grid minor
38 box on
39 % exportgraphics(fig1, 'images/T over r.png','Resolution',400);
40
41 % //////////////////////////////////////
42 % //////////////////////////////////////
43
44 fig2 = figure('Name', '2','Position', [150, 250, 900, 600]);
45 size = 15;
46
47 colors = cool(length(results_vec(:,1))*0.9;
48 lg = {};
49 for i = 1:length(results_vec(:,1))
50     semilogy(results_vec{i,1}.r, results_vec{i,1}.Ts(end,:), 'LineWidth', 2, 'Color',
        colors(i,:))
51     hold on;
52     lg{end+1} = sprintf('N=%g, $\Delta t=%f$', results_vec{i,1}.N, results_vec{i,1}.
        delta_t);
53 end
54 xlabel('r$\displaystyle\left[\frac{m}{sec}\right]$', 'FontSize',size, 'Interpreter','
    latex')
55 ylabel('T$\left[K\right]$', 'FontSize',size, 'Interpreter','latex')
56 title('Temperature as a Function of The Radius For Different N', 'FontSize',size, '
    Interpreter','latex')
57 subtitle(sprintf('$t=%f[s]$ $||$ $\backslash\varepsilon=%g$ $||$ $R=%g$', results_vec{end,1}.t_vec(
    end), results_vec{end,1}.epsilon, results_vec{end,1}.R), 'FontSize', size, '
    Interpreter','latex')
58 legend(lg, 'FontSize',size-2, 'Location','eastoutside','Interpreter','latex')
59 grid on
60 grid minor
61 box on
62 % exportgraphics(fig1, 'images/T over r.png','Resolution',400); exportgraphics(fig2, '
    images/Influenc of N.png','Resolution',400);
63
64 % //////////////////////////////////////
65 % //////////////////////////////////////
66
67 fig3 = figure('Name', '3','Position', [250, 250, 900, 600]);
68 rms_vec = [];
69 Ns = [];
70 for i = 1:length(results_vec(:,1))-1
71     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,:));
72     Ns(i) = results_vec{i,1}.N;
73 end
74
75 plot(Ns, rms_vec, 'LineWidth', 2, 'Color', colors(end,:))
76
77 xlabel('N', 'FontSize',size, 'Interpreter','latex')

```




```

126 eps_vec = [];
127 for i = 1:length(results_vec(:,1))-1
128     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,:));
129     eps_vec(i) = results_vec{i,1}.epsilon;
130 end
131
132 loglog(eps_vec, rms_vec, 'LineWidth', 2, 'Color', colors(end,:))
133
134 xlabel('$\varepsilon$', 'FontSize',size, 'Interpreter','latex')
135 ylabel('rms$\left[K\right]$', 'FontSize',size, 'Interpreter','latex')
136 title('The RMS between two solution as a function of $\varepsilon$', 'FontSize',size, '
    Interpreter','latex')
137 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')
138 grid on
139 grid minor
140 box on
141 % exportgraphics(fig4, 'images/Influenc of epsilon.png','Resolution',400); exportgraphics
    (fig5, 'images/Influenc of epsilon — error.png','Resolution',400);
142
143 %% Influenc of R =====
144 % =====
145 % =====
146
147 parameters
148
149 % Rs = linspace(0.05,0.49,10);
150 Rs = linspace(0.45, 0.3, 4);
151
152 results_vec = {};
153 r_vec = {};
154 for R_index = 1:length(Rs)
155     R = Rs(R_index);
156     delta_t = R * h^2;
157     results_vec{R_index,1} = solver(r_min, r, K, h, delta_t, t_start, t_end, N, epsilon,
        max_iteration);
158 end
159 %%
160 load("effect_of_R_0.05_to_0.49.mat")
161
162 fig6 = figure('Name', '6','Position', [550, 250, 900, 600]);
163 size = 15;
164
165 colors = cool(length(results_vec(:,1)))*0.9;
166 lg = {};
167 for i = 1:length(results_vec(:,1))
168     semilogy(results_vec{i,1}.r, results_vec{i,1}.Ts(end,:), 'LineWidth', 2, 'Color',
        colors(i,:))
169     hold on;
170     lg{end+1} = sprintf('$R=%g$', results_vec{i,1}.R);
171 end
172 xlabel('$r\displaystyle\left[\frac{m}{sec}\right]$', 'FontSize',size, 'Interpreter','
    latex')
173 ylabel('$T\left[K\right]$', 'FontSize',size, 'Interpreter','latex')
174 title('Temperature as a Function of The Radius For Different R', 'FontSize',size, '
    Interpreter','latex')

```



```

175 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')
176 legend(lg, 'FontSize',size-2, 'Location','eastoutside','Interpreter','latex')
177 grid on
178 grid minor
179 box on
180 % exportgraphics(fig6, 'images/Influenc of R.png','Resolution',400);
181
182 fig7 = figure('Name', '7','Position', [650, 250, 900, 600]);
183 rms_vec = [];
184 R_vec = [];
185 for i = 1:length(results_vec(:,1))-1
186     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,:));
187     R_vec(i) = results_vec{i,1}.R;
188 end
189
190 semilogy(R_vec, rms_vec, 'LineWidth', 2, 'Color', colors(end,:))
191
192 xlabel('$R$', 'FontSize',size, 'Interpreter','latex')
193 ylabel('rms$\\left[K\\right]$', 'FontSize',size, 'Interpreter','latex')
194 title('The RMS between two solution as a function of R', 'FontSize',size, 'Interpreter','
    latex')
195 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')
196 grid on
197 grid minor
198 box on
199 % exportgraphics(fig6, 'images/Influenc of R.png','Resolution',400); exportgraphics(fig7,
    'images/Influenc of R - error.png','Resolution',400);
200
201 %% Integrate =====
202 % =====
203 % =====
204
205 parameters
206 result = solver(r_min, r, K, h, delta_t, t_start, t_end, N, epsilon, max_iteration);
207
208 % integrate N=20
209 sum_N20 = 0;
210 for i = [0:result.N+1-1]+1
211     sum_N20 = sum_N20 + (result.Ts(end,i)+result.Ts(end,i+1))*result.r(i);
212 end
213 sum_N20 = sum_N20 * 10.7 * 0.5 * result.h
214
215 % integrate N=40
216 load("effect_of_N_to_60.mat")
217 result = results_vec{end-1};
218 sum_N40 = 0;
219 for i = [0:result.N+1-1]+1
220
221     sum_N40 = sum_N40 + (result.Ts(end,i)+result.Ts(end,i+1))*result.r(i);
222 end
223 sum_N40 = sum_N40 * 10.7 * 0.5 * result.h
224
225 % integrate N=60

```



```

226 load("effect_of_N_to_60.mat")
227 result = results_vec{end};
228 sum_N60 = 0;
229 for i = [0:result.N+1-1]+1
230
231     sum_N60 = sum_N60 + (result.Ts(end,i)+result.Ts(end,i+1))*result.r(i);
232 end
233 sum_N60 = sum_N60 * 10.7 * 0.5 * result.h
234
235
236
237
238 %% Functions =====
239 % =====
240 % =====
241
242 function T = init_fuild(r_min, h, N)
243     i = 0:1:N+1;
244     r = r_min+h * i;
245     T = 200 * (r - 0.5);
246 end
247
248 function T = set_BC(T, t)
249     T(1) = t;
250     T(end) = 100 + 40 * t;
251 end
252
253 function T_next = step_space_jacobi(T_current, r, K, h, delta_t, N)
254     T_next(1) = T_current(1);
255     T_next(N+1+1) = T_current(N+1+1);
256     for i = [1:N]+1
257         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));
258         % 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))
                ;
259     end
260 end
261
262 function converged = check_convergence(T_next, T_current, epsilon, N)
263     converged = true;
264     for i = [1:N]+1
265         if abs(T_next(i) - T_current(i)) > epsilon
266             converged = false;
267             return
268         end
269     end
270 end
271
272 function T_next_t = solve_for_specific_t_jacobi(T_current_t, r, K, h, delta_t, N, epsilon
    , max_iteration)
273     T_current = T_current_t;
274     for n = 1:max_iteration
275         T_next = step_space_jacobi(T_current, r, K, h, delta_t, N);
276         if check_convergence(T_next, T_current, epsilon, N)
277             break
278         end

```



```

279     T_current = T_next;
280 end
281 T_next_t = T_next;
282 end
283
284 function results = solver(r_min, r, K, h, delta_t, t_start, t_end, N, epsilon,
    max_iteration)
285     T_current_t = init_fuild(r_min, h, N);
286     t = t_start;
287     Ts(1, :) = T_current_t;
288
289     while t <= t_end
290         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)
291         T_current_t = set_BC(T_current_t, t);
292         T_next_t = solve_for_specific_t_jacobi(T_current_t, r, K, h, delta_t, N, epsilon,
            max_iteration);
293         Ts(end+1, :) = T_next_t;
294
295         t = t + delta_t;
296         T_current_t = T_next_t;
297     end
298     results.Ts      = Ts;
299     results.N       = N;
300     results.r       = r;
301     results.h       = h;
302     results.delta_t = delta_t;
303     results.epsilon = epsilon;
304     results.R       = delta_t / h^2;
305     results.t_vec   = delta_t*(0:1:t_end/delta_t);
306 end
307
308 function y = interpolate(x, x_vec, y_vec)
309 % This function assume an ordered x_vec from low to high
310 if x > x_vec(end) || x < x_vec(1)
311     fprintf('value out of bounds\n');
312     return;
313 end
314
315 index_i = 0;
316 for i = 1:length(x_vec)
317     if x <= x_vec(i)
318         index_i = i;
319         break;
320     end
321 end
322
323 if x == x_vec(index_i)
324     y = y_vec(index_i);
325     return;
326 end
327
328 m = (y_vec(index_i+1) - y_vec(index_i)) / (x_vec(index_i+1) - x_vec(index_i));
329 b = y_vec(index_i) - x_vec(index_i) * m;
330
331 y = m * x + b;
332 end
333

```



```
334 function rms = RMS(x1, y1, x2, y2)
335 % This functions calculatlates the RMS value for 1 compared to 2
336     y2_len_y1 = [];
337     for i = 1:length(x1)
338         y2_len_y1(i) = interpolate(x1(i), x2, y2);
339     end
340
341     rms = sqrt(sum((y1-y2_len_y1).^2)/length(y1));
342 end
```

Listing 2: The main file