

Numerical model of one-dimensional heat conduction in a fin

Assignment one of three

Andreas Wenger | Vinzenz Götz
44267018 | 25173223

Prof. Roberto Nunhez

WS 23/24

Contents

Figures	ii
1 analytical solution of heat transfer problem	1
2 Matlab code for heat transfer problem	3

List of Figures

2.1	The matlab code for the definition of the constants used in this problem . .	3
2.2	The matlab code for the definition of further constants using the analytical solution	4
2.3	The matlab code for the calculation of T	4
2.4	The matlab code for the calculation of the analytical solution and the display of relative errors	5
2.5	The relative error of T over the length of the rod with a varying number of mesh elements.	5
2.6	The matlab code for the relative error at the last point comparing different numbers of mesh elements.	5
2.7	The matlab code for the output T_f and temperature curves of different numbers of mesh elements compared to the analytical solution.	6
2.8	The matlab code for the output temperature curves of different numbers of mesh elements compared to the analytical solution.	6

1. analytical solution of heat transfer problem

The given task is to conduct a one dimensional heat transfer analysis in a rectangular fin with a uniform cross-sectional area A . On one side the fin is fixed to a wall at a given wall temperature $T_{wall} = 300^\circ C$. At the surface of the fin there is convective heat transfer caused by a surrounding fluid with a temperature of $T_\infty = 20^\circ C$. The combined heat transfer coefficient between fin surface and fluid is given as $h = 10 \frac{W}{m^2 K}$. The heat conductivity of the fin material is given to $k = 100 \frac{W}{m K}$.

In the following section the analytical solution for the given heat transfer problem is solved. The basic equation is given to $T'' = m^2(T - T_\infty)$ according to the assignment. T can be written as $T = t + T_\infty$, which leads to $t = T - T_\infty$. From this follows $t' = T'$ and $t'' = T''$. Furthermore $m^2 = hP/kA_T$. From the equations above, follows a differential equation for t according to $t'' = m^2 t$.

As a next step to solve the differential equation the eigenvalues and eigenvectors must be calculated.

$$\begin{cases} t_1 = t \\ t_2 = t' \end{cases} \implies \vec{t}' = \begin{pmatrix} 0 & 1 \\ m^2 & 0 \end{pmatrix} \vec{t} \quad (1.1)$$

The eigenvalues result of the equation $\lambda^2 = m^2$ to $\lambda_1 = m$ and $\lambda_2 = -m$. Therefore the eigenvectors result to $\lambda_1 = \begin{pmatrix} 1 \\ m \end{pmatrix}$ and $\lambda_2 = \begin{pmatrix} 1 \\ -m \end{pmatrix}$.

This leads to the following equation where the matrix is called the fundamental matrix $Y(x)$ to $\vec{t}(x)$ which is composed of the solution vectors. The solution is therefore a linear combination of the columns of this matrix.

$$\vec{t}(x) = Y(x) \cdot \vec{c} = \begin{pmatrix} e^{mx} & e^{-mx} \\ me^{mx} & -me^{-mx} \end{pmatrix} \vec{c} \quad (1.2)$$

With $\vec{c} = (C_1 C_2)^T$ it follows that

$$t(x) = e^{mx} C_1 + e^{-mx} C_2. \quad (1.3)$$

To solve this differential equation the following boundary conditions were defined and have to be solved.

$$\begin{cases} T(0) = T_w \\ 0 = h(T(L) - T_\infty) + kT'(L) \end{cases} \implies \begin{cases} t(0) = T_w - T_\infty \\ 0 = ht(L) + kt'(L) \end{cases} \quad (1.4)$$

Considering the boundary conditions for the left and right edge, two matrices A and B as well as the vector \vec{r} are defined to the following. With A and B reflecting the boundary conditions at $x = 0$ and $x = L$ respectively. \vec{r} gives the source term from the boundary conditions.

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, B = \begin{pmatrix} 0 & 0 \\ h & k \end{pmatrix}, \vec{r} = \begin{pmatrix} T_w - T_\infty \\ 0 \end{pmatrix} = \begin{pmatrix} \Delta T \\ 0 \end{pmatrix} \quad (1.5)$$

With the identities from Equation 1.5, R is given as

$$R = \begin{pmatrix} 1 & 1 \\ \underbrace{he^{mL} + mke^{mL}}_a & \underbrace{he^{-mL} - mke^{-mL}}_b \end{pmatrix} = A \cdot Y(0) + B \cdot Y(L) \quad (1.6)$$

To calculate the constants C_1 and C_2 for the differential Equation 1.3 the following system given by $R\vec{c} = \vec{r}$ of equations must be solved.

$$\begin{aligned} & \begin{pmatrix} 1 & 1 & \left| \begin{array}{c} \Delta T \\ 0 \end{array} \right. \end{pmatrix} \xrightarrow{l_2 - \frac{l_2}{b}} \begin{pmatrix} 1 - \frac{a}{b} & 0 & \left| \begin{array}{c} \Delta T \\ 0 \end{array} \right. \end{pmatrix} \\ & \xrightarrow{l_2 - \frac{l_1}{a - \frac{a}{b}}} \begin{pmatrix} 1 & 0 & \left| \begin{array}{c} \frac{\Delta T}{1 - \frac{a}{b}} \\ -\frac{\Delta T \cdot a}{1 - \frac{a}{b}} \end{array} \right. \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & \left| \begin{array}{c} \frac{\Delta T}{1 - \frac{a}{b}} \\ -\frac{\Delta T}{1 - \frac{a}{b}} \cdot \frac{a}{b} \end{array} \right. \end{pmatrix} \end{aligned} \quad (1.7)$$

$$C_1 = \frac{\Delta T}{1 - \frac{a}{b}} \quad (1.8)$$

$$C_2 = \frac{-\Delta T}{1 - \frac{a}{b}} \cdot \frac{a}{b} \quad (1.9)$$

Using the values of C_1 and C_2 in Equation 1.3 the solved differential equation for t results in

$$t(x) = e^{mx} \cdot \frac{\Delta T}{1 - \frac{a}{b}} + e^{-mx} \cdot \frac{-\Delta T}{1 - \frac{a}{b}} \cdot \frac{a}{b}. \quad (1.10)$$

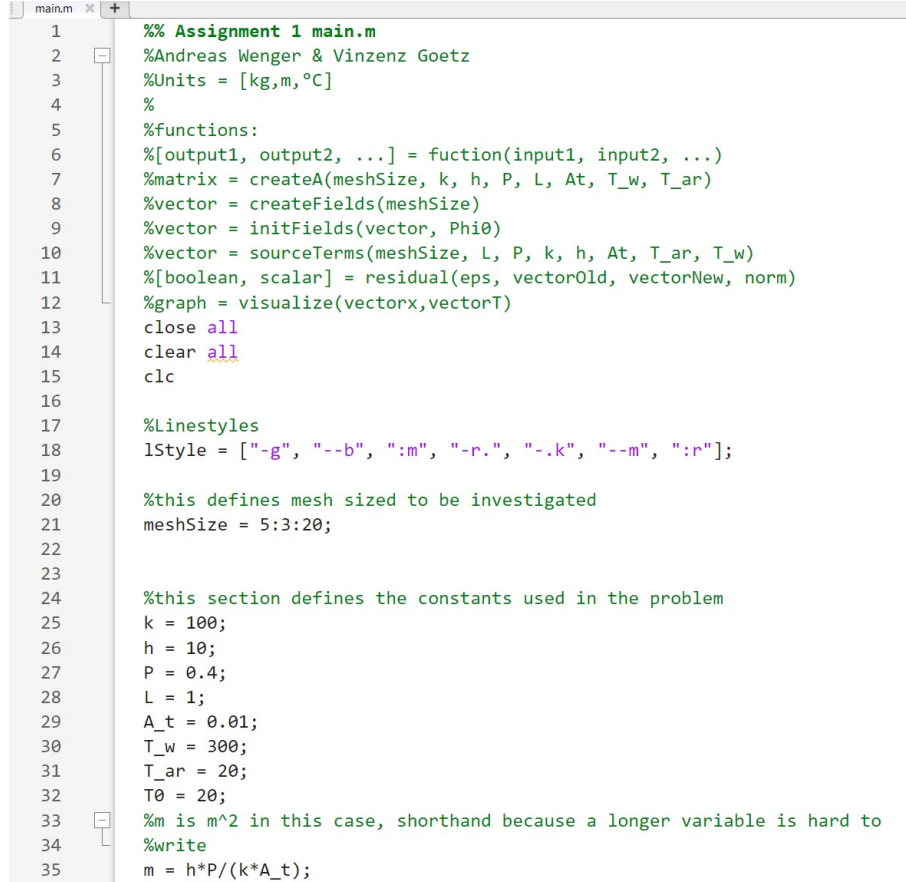
Therefore the solution of $T(x)$ follows form Equation 1.10 and $T(x) = t + T_\infty$ as

$$T(x) = e^{mx} \cdot \frac{\Delta T}{1 - \frac{a}{b}} - e^{-mx} \cdot \frac{\Delta T}{1 - \frac{a}{b}} \cdot \frac{a}{b} + T_\infty. \quad (1.11)$$

2. Matlab code for heat transfer problem

In the upcoming chapter the matlab code programmed for the heat transfer problem will be presented and discussed.

The following Figure 2.1 shows the beginning of the matlab code used for the heat transfer problem.



```
1 %% Assignment 1 main.m
2 %Andreas Wenger & Vinzenz Goetz
3 %Units = [kg,m,°C]
4 %
5 %functions:
6 %[output1, output2, ...] = fuction(input1, input2, ...)
7 %matrix = createA(meshSize, k, h, P, L, At, T_w, T_ar)
8 %vector = createFields(meshSize)
9 %vector = initFields(vector, Phi0)
10 %vector = sourceTerms(meshSize, L, P, k, h, At, T_ar, T_w)
11 %[boolean, scalar] = residual(eps, vectorOld, vectorNew, norm)
12 %graph = visualize(vectorx,vectorT)
13 close all
14 clear all
15 clc
16
17 %Linestyles
18 lStyle = ["-g", "--b", ":m", "-r.", "-.k", "--m", ":r"];
19
20 %this defines mesh sized to be investigated
21 meshSize = 5:3:20;
22
23
24 %this section defines the constants used in the problem
25 k = 100;
26 h = 10;
27 P = 0.4;
28 L = 1;
29 A_t = 0.01;
30 T_w = 300;
31 T_ar = 20;
32 T0 = 20;
33 %m is m^2 in this case, shorthand because a longer variable is hard to
34 %write
35 m = h*P/(k*A_t);
```

Figure 2.1: The matlab code for the definition of the constants used in this problem

At first all existing windows are closed, the storage of variables is erased and the command window is cleared. The mesh size is choosen to vary from 5 to 20 elements using a step size of 3. In total 6 different mesh sizes are compared in this study. As described later the relative error using more than 20 mesh elements will not drop significantly. Therefore the maximum mesh size is limited to 20 in order to save computing time.

The next section of the code defines the constants used in the problem as given in the assignment. For convenience of the reader and writer the variable used for m^2 is m .

As the analytical solution is already obtained in chapter 1, the subsequent Figure 2.2 depicts a further constant being defined, which is given as $\gamma = \frac{\Delta T}{1-\frac{a}{b}}$. Furthermore the error vector is initialized with the right length.

```

36
37 %this is defining some constants which help in the analytical solution of
38 %the problem
39 mprime=sqrt(m);
40 a = h*exp(mprime*L)+mprime*k*exp(mprime*L);
41 bprime = h*exp(-mprime*L)-mprime*k*exp(-mprime*L);
42 gam = (T_w-T_ar)/((1-a/bprime));
43
44
45 %this is for no convection on the face
46 %gam = (T_w-T_ar)/(1+exp(-2*sqrt(m)*L));
47
48 %this initializes the error vectors with the right length
49 errRel = zeros(1,length(meshSize));
50

```

Figure 2.2: The matlab code for the definition of further constants using the analytical solution

The following fragment of our matlab code initializes a for loop to calculate the given heat transfer problem for multiple mesh elements sequentially. This is shown in Figure 2.3.

```

50
51 for i = 1:length(meshSize)
52 %here, the system matrix as well as source terms are initialized
53 dx = L/meshSize(i);
54 %system matrix is created
55 A = createA(meshSize(i), k, h, P, L, A_t, T_w, T_ar);
56 %source terms are created
57 b = sourceTerms(meshSize(i), L, P, k, h, A_t, T_ar, T_w);
58 x = dx/2:dx:L-dx/2;
59
60 %system is solved and T obtained
61 T = A\b;
62 T_p = T(end);
63 T_f = ((2*k/dx)*T_p+h*T_ar)/((2*k/dx)+h);
64 T(2:end+1) = T;
65 T(1) = T_w;
66 T(end+1) = T_f;
67 x(2:end+1) = x;
68 x(1) = 0;
69 x(end+1) = L;
70
71 figure(1)
72 plot(x,T,lStyle(i),LineWidth=1.5);
73 hold on
74

```

Figure 2.3: The matlab code for the calculation of T

For the calculation to work properly, at first dx is defined. Afterwards the system matrix as well as the source terms are initialized with the functions $A = \text{createA}()$ and $b = \text{sourceTerms}()$.

createA takes as arguments the mesh elements, k , h , P , L , and the area of the fin tip. Where k is the heat conductiveness, h the heat transfer coefficient, P the perimeter of the fin and L the length. It returns the sparse system matrix for the heat conduction problem, which was defined in the assignment.

sourceTerms takes the mesh elements, L , P , k , h , the area of the tip, the temperature of the surrounding fluid and the temperature of the wall. L , p , k and h are defined as in createA . This function returns a vector of source terms also defined in the assignment.

The next step is calculating the numerical solution for T at the cell midpoints. This is done by the built-in function of Matlab to solve a system of linear equations which solves $A\vec{x} = b$ by $x = A \backslash b$. Moreover the temperature T_f at the end of the rod is calculated and the vector T shifted and the first entry set to T_{wall} . Also, the vector x of cell centroid positions is shifted by 1 to account for the face of the cell of the wall and also lengthened by 1 to account for the face of the tip.

The temperature T is then plotted into figure(1) as seen in Figure 2.3.

Upcoming Figure 2.4 shows matlab code for calculating the analytical solution as well as displaying the relative errors.

```

75
76
77
78 %calculate analytical solution for given mesh size
79 anaSol = gam.*exp(mprime.*x)-gam.*a./bprime.*exp(-mprime*x)+T_ar;
80
81 %fill error vector
82 errRel(i) = norm(anaSol-T)/norm(anaSol)/length(T);
83
84 errLen = abs(anaSol-T') ./ anaSol;
85 figure(2)
86 plot(x, errLen, lStyle(i), LineWidth=1.5)
87 hold on
88
89 ltext(i) = strcat("Mesh elements = ", num2str(meshSize(i)));
90
91 end
92

```

Figure 2.4: The matlab code for the calculation of the analytical solution and the display of relative errors

The error vectors shown in figure Figure 2.5 display the relative error between the numerical solution and the analytical solution over the length of the rod at different numbers of mesh elements.

The relative error at the last point of the rod is displayed in figure Figure 2.6. As shown, the relative error decreases with increasing numbers of mesh elements.

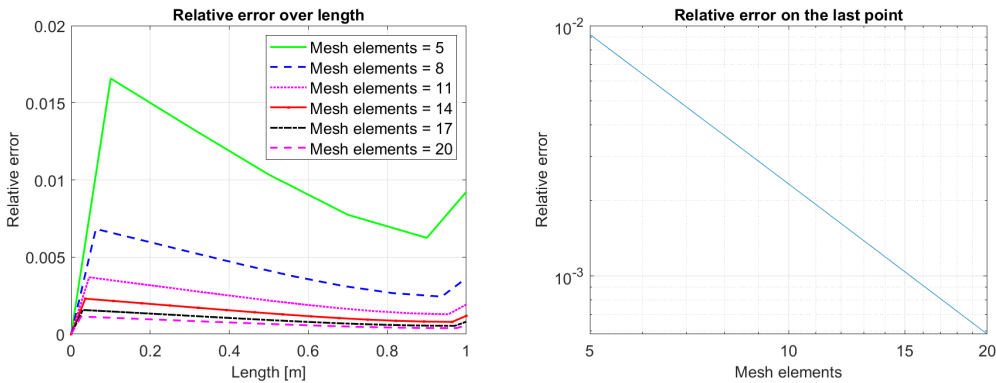


Figure 2.5: The relative error of T over the length of the rod with a varying number of mesh elements.

Figure 2.6: The matlab code for the relative error at the last point comparing different numbers of mesh elements.


```

92
93 %output of temperature at the end of the fin
94 fprintf('The temperature at the tip of the fin = %.3f °C for a meshsize of %d elements \n', T_f, meshSize(end))
95
96 %plot errors and the analytical solution
97 figure(2)
98 title("Relative error over length")
99 xlabel("Length [m]");
100 ylabel("Relative error");
101 legend(ltext);
102 grid
103 fontsize(13,"points")
104 saveas(2, "errLen.png")
105
106
107 figure(1)
108 grid
109 plot(x, anaSol, LineWidth=1.5)
110 xlabel("x [m]");
111 ylabel("T [°C]");
112 ltext(end+1) = "analytical solution";
113 legend(ltext);
114 %the fontsize command requires version R2022a or later
115 fontsize(13,"points")
116 saveas(1, "T.png")
117
118 figure(3)
119 loglog(meshSize, errRel)
120 title("Relative error on the last point")
121 xlabel("Mesh elements");
122 ylabel("Relative error");
123 grid
124 fontsize(13,"points")
125 saveas(3, "errElems.png")
126
127 hold off

```

Figure 2.7: The matlab code for the output T_f and temperature curves of different numbers of mesh elements compared to the analytical solution.

Figure 2.7 depicts the matlab code used for the output.

The temperature at the tip of the fin T_f is posted into the command window of matlab for the biggest number of mesh elements investigated.

Furthermore the temperature curve for the analytical solution is displayed in Figure 2.8. It can be seen that the difference between varying numbers of mesh elements and the analytical solution is small.

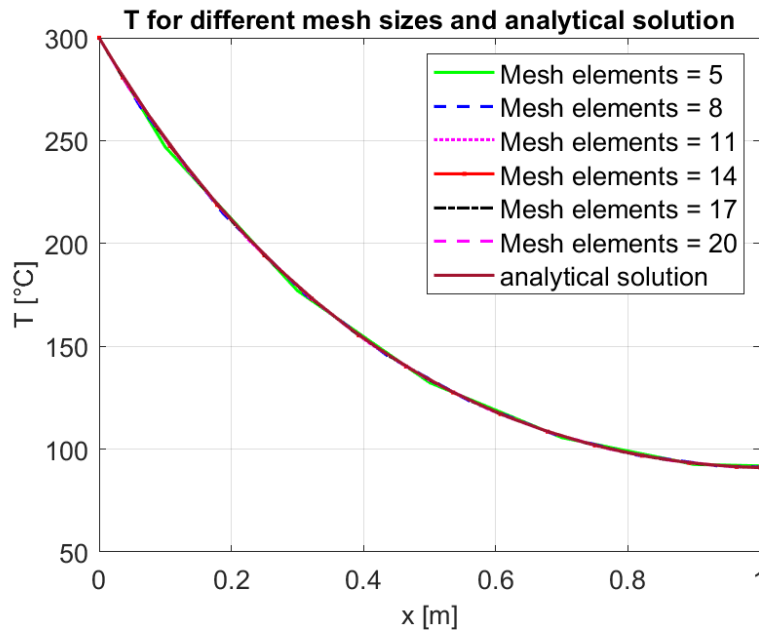


Figure 2.8: The matlab code for the output temperature curves of different numbers of mesh elements compared to the analytical solution.