Project 3

Amanda Lind Davíðsdóttir <amanda19@ru.is>

Bjarmi Valentino B Del Negro <bjarmib19@ru.is>

Eric Ruge <eric22@ru.is>

Fanney Einarsdóttir <fanneye22@ru.is>

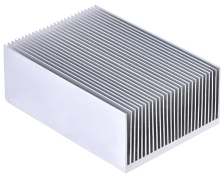
T-406-TOLU

RU Science and Engineering

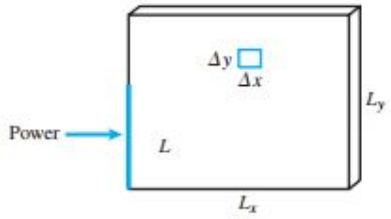
December 11, 2022

Project 3 in numerical analysis

Heatsinks are an ubiquitous design used to dissipate heat and cool certain objects which do not handle high temperatures. Examples are a car's radiator and the cooling system of a microprocessor. A typical design for a heatsink is as shown here:



which is composed of a number of thin plates. In this project we focus on one such plate and study how it reacts to being subjected to heat on one side. The plate is rectangular with dimensions L_x and L_y, and its small thickness is δ. Heat enters the plate on its left-hand side as shown here:



Our goal is to find the long-term equilibrium of the system that is compute the temperature distribution after an infinite amount of time. One can show that at an interior point the temperature $u(x, y)$ satisfies the partial differential equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{2H}{K\delta} u$$

The constant K is the thermal conductivity of the material while H is the convective heat transfer coefficient, they relate to both ways that heat can dissipate. Again δ is the thickness of the plate. On top of the equation we have boundayr conditions which encode convection into the ambient air. More precisely:

$$\frac{\partial u}{\partial n} = \frac{H}{K} u$$

which is the outward normal derivative on the sides of the plate. Since the plate is rectangular the normal derivative is simply parallel to the x and y axes with opposite directions on opposite sides:

$$\begin{aligned}
\frac{\partial u}{\partial n} &= -\frac{\partial u}{\partial y} && \text{bottom} \\
\frac{\partial u}{\partial n} &= \frac{\partial u}{\partial y} && \text{top} \\
\frac{\partial u}{\partial n} &= -\frac{\partial u}{\partial x} && \text{left} \\
\frac{\partial u}{\partial n} &= \frac{\partial u}{\partial x} && \text{right}
\end{aligned}$$

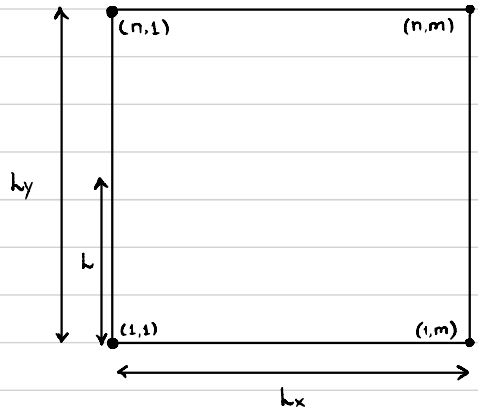
Finally we have a separate boundary condition where heat enters the plate:

$$\frac{\partial u}{\partial n} = -\frac{\partial u}{\partial x} = \frac{P}{L\delta K}$$

where P is the power og L the length along which power is applied.

Problem 1

The notation that is used for the plate is the following:



x-axis : $[1, m]$ in $m-1$ subintervals of length h, $\quad 0 = x_1 < x_2 < \dots < x_{m-1} < x_m = L_x$

y-axis : $[1, n]$ in $n-1$ subintervals of length k, $\quad 0 = y_1 < y_2 < \dots < y_{n-1} < y_n = L_y$

Robin boundary method

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{2H}{K\delta} u \quad , \quad f(x,y) = \frac{2H}{K\delta} u$$

① Discretize the equation

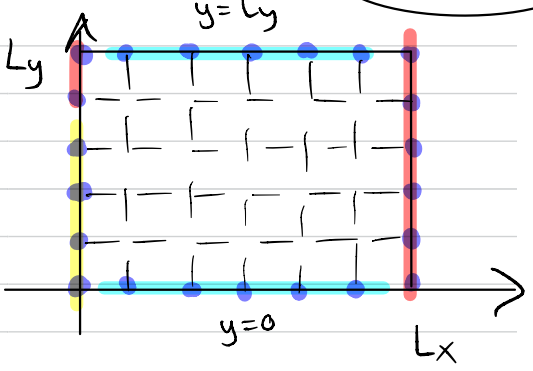
$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u(x+h,y) - 2u(x,y) + u(x-h,y)}{h^2} \longrightarrow \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{u(x,y+h) - 2u(x,y) + u(x,y+h)}{k^2} \longrightarrow \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2}$$

Eq : $$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2} = \frac{2H}{K\delta} u_{i,j}$$

works on the inner points $\boxed{\begin{matrix} 2 \le i \le m-1 \\ 2 \le j \le n-1 \end{matrix}}$

m×n matrix



Depends on L

$$\frac{u_{i+1,j}}{h^2} - \frac{2u_{i,j}}{h^2} + \frac{u_{i-1,j}}{h^2} + \frac{u_{i,j+1}}{k^2} - \frac{2u_{i,j}}{k^2} + \frac{u_{i,j-1}}{k^2} - \frac{2H u_{i,j}}{K\delta} = 0$$

$$\frac{1}{h^2} u_{i+1,j} - \frac{2}{h^2} u_{i,j} + \frac{1}{h^2} u_{i-1,j} + \frac{1}{k^2} u_{i,j+1} - \frac{2}{k^2} u_{i,j} + \frac{1}{k^2} u_{i,j-1} - \frac{2H}{K\delta} u_{i,j} = 0$$

$$\frac{1}{h^2} u_{i+1,j} + \frac{1}{h^2} u_{i-1,j} + \frac{1}{k^2} u_{i,j+1} + \frac{1}{k^2} u_{i,j-1} - \left(\frac{2}{h^2} + \frac{2}{k^2} + \frac{2H}{K\delta}\right) u_{i,j} = 0$$

$$\boxed{\frac{u_{i+1,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} + u_{i,j-1}}{k^2} - \left(\frac{2}{h^2} + \frac{2}{k^2} + \frac{2H}{K\delta}\right) u_{i,j} = 0}$$

works on the inner points $\begin{matrix} 2 \le i \le m-1 \\ 2 \le j \le n-1 \end{matrix}$

② Robin boundary conditions $\boxed{\frac{\partial u}{\partial n} = \frac{H}{K}u}$

Bottom : $\frac{\partial u}{\partial n} = -\frac{\partial u}{\partial y}$

Top : $\frac{\partial u}{\partial n} = \frac{\partial u}{\partial y}$

Left : $\frac{\partial u}{\partial n} = -\frac{\partial u}{\partial x}$

Right : $\frac{\partial u}{\partial n} = \frac{\partial u}{\partial x}$

Heat BC (left) : $\frac{\partial u}{\partial n} = -\frac{\partial u}{\partial x} = \frac{P}{L\delta K}$

Bottom : $-\frac{H}{K}u_{i,j} \approx \frac{-3u_{i,j} + 4u_{i,j+1} - u_{i,j+2}}{2k}$

$$-\frac{2kH}{K}u_{i,j} = -3u_{i,j} + 4u_{i,j+1} - u_{i,j+2}$$

$$0 = \left(\frac{2kH}{K} - 3\right)u_{i,j} + 4u_{i,j+1} - u_{i,j+2}$$

$$\boxed{0 = \left(\frac{2kH}{K} - 3\right)u_{i,1} + 4u_{i,2} - u_{i,3}} \quad \text{for } j=1,\ 2 \le i \le m-1$$

Top : $\frac{H}{K}u_{i,j} \approx \frac{-3u_{i,j} + 4u_{i,j-1} - u_{i,j-2}}{-2k}$

$$-\frac{2kH}{K}u_{i,j} = -3u_{i,j} + 4u_{i,j-1} - u_{i,j-2}$$

$$0 = \left(\frac{2kH}{K} - 3\right)u_{i,j} + 4u_{i,j-1} - u_{i,j-2}$$

$$\boxed{0 = \left(\frac{2kH}{K} - 3\right)u_{i,n} + 4u_{i,n-1} - u_{i,n-2}} \quad \text{for } j=n \quad 2 \le i \le m-1$$

Left : $-\frac{H}{K}u_{i,j} \approx \frac{-3u_{i,j} + 4u_{i+1,j} - u_{i+2,j}}{2h}$

$$-\frac{2hH}{K}u_{i,j} = -3u_{i,j} + 4u_{i+1,j} - u_{i+2,j}$$

$$0 = \left(\frac{2hH}{K} - 3\right)u_{i,j} + 4u_{i+1,j} - u_{i+2,j}$$

$$\boxed{0 = \left(\frac{2hH}{K} - 3\right)u_{1,j} + 4u_{2,j} - u_{3,j}} \quad \text{for } i=1,\ L < j \le n$$

Right : $\frac{H}{K} u_{ij} \approx \frac{-3u_{i,j} + 4u_{i-1,j} - u_{i-2,j}}{-2h}$

$$-\frac{2hH}{K} u_{i,j} = -3u_{i,j} + 4u_{i-1,j} - u_{i-2,j}$$

$$0 = \left(\frac{2hH}{K} - 3\right) u_{i,j} + 4u_{i-1,j} - u_{i-2,j}$$

$$\boxed{0 = \left(\frac{2hH}{K} - 3\right) u_{m,j} + 4u_{m-1,j} - u_{m-2,j}} \quad \text{for } i = m, \ 1 \le j \le n$$

Heat left : $-\frac{P}{LsK} \approx \frac{-3u_{i,j} + 4u_{i+1,j} - u_{i+2,j}}{2h}$

$$-\frac{2hP}{LsK} = -3u_{i,j} + 4u_{i+1,j} - u_{i+2,j}$$

$$\boxed{-\frac{2hP}{LsK} = -3u_{1,j} + 4u_{2,j} - u_{3,j}} \quad \text{for } i = 1, \ 1 \le j \le L$$

Problem 2

Setup $A\bar{v} = \bar{b}$, $v_{i+(j-1)m} = u_{i,j}$, $mn \times mn$ matrix

Inner points: $\frac{u_{i+1,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} + u_{i,j-1}}{k^2} - \left(\frac{2}{h^2} + \frac{2}{k^2} + \frac{2H}{K\delta}\right) u_{i,j} = 0$

$$\frac{v_{i+1+(j-1)m} + v_{i-1+(j-1)m}}{h^2} + \frac{v_{i+jm} + v_{i+(j-2)m}}{k^2} - \left(\frac{2}{h^2} + \frac{2}{k^2} + \frac{2H}{K\delta}\right) v_{i+(j-1)m} = 0$$

$$-\left(\frac{2}{h^2} + \frac{2}{k^2} + \frac{2H}{K\delta}\right) v_{i+(j-1)m} + \frac{1}{h^2} v_{i+1+(j-1)m} + \frac{1}{h^2} v_{i-1+(j-1)m} + \frac{1}{k^2} v_{i+jm} + \frac{1}{k^2} v_{i+(j-2)m} = 0$$

eq_1 : $A(t,t) = -\left(\frac{2}{h^2} + \frac{2}{k^2} + \frac{2H}{K\delta}\right)$
eq_2 : $A(t,t+1) = \frac{1}{h^2}$
eq_3 : $A(t,t-1) = \frac{1}{h^2}$
eq_4 : $A(t,t+m) = \frac{1}{k^2}$
eq_5 : $A(t,t-m) = \frac{1}{k^2}$
$b(t) = 0$

$\boxed{t = i + (j-1)m}$

for $2 \le i \le m-1$
$2 \le j \le n-1$

Bottom: $0 = \left(\frac{2kH}{K} - 3\right) u_{i,1} + 4u_{i,2} - u_{i,3}$

$$0 = \left(\frac{2kH}{K} - 3\right) v_i + 4v_{i+m} - v_{i+2m}$$

$\boxed{t = i}$

for $j = 1$, $2 \le i \le m-1$

eq_1 : $A(t,t) = \frac{2kH}{K} - 3$
eq_2 : $A(t,t+m) = 4$
eq_3 : $A(t,t+2m) = -1$
$b(t) = 0$

Top: $0 = \left(\frac{2kH}{K} - 3\right) u_{i,n} + 4u_{i,n-1} - u_{i,n-2}$

$$0 = \left(\frac{2kH}{K} - 3\right) v_{i+(n-1)m} + 4v_{i+(n-2)m} - v_{i+(n-3)m}$$

$\boxed{t = i + (n-1)m}$

for $j = n$ $2 \le i \le m-1$

eq_1 : $A(t,t) = \frac{2kH}{K} - 3$
eq_2 : $A(t,t-m) = 4$
eq_3 : $A(t,t-2m) = -1$
$b(t) = 0$

Left: $0 = \left(\frac{2hH}{K} - 3\right) u_{1,j} + 4u_{2,j} - u_{3,j}$

$0 = \left(\frac{2hH}{K} - 3\right) V_{1+(j-1)m} + 4V_{2+(j-1)m} - V_{3+(j-1)m}$

$eq_1: A(t,t) = \frac{2hH}{K} - 3$
$eq_2: A(t,t+1) = 4$
$eq_3: A(t,t+2) = -1$
$b(t) = 0$

$\boxed{t = 1 + (j-1)m}$

for $i = 1$, $L < j \leq n$

Right: $0 = \left(\frac{2hH}{K} - 3\right) u_{m,j} + 4u_{m-1,j} - u_{m-2,j}$

$0 = \left(\frac{2hH}{K} - 3\right) V_{jm} + 4V_{-1+jm} - V_{-2+jm}$

$eq_1: A(t,t) = \frac{2hH}{K} - 3$
$eq_2: A(t,t-1) = 4$
$eq_3: A(t,t-2) = -1$
$b(t) = 0$

$\boxed{t = m + (j-1)m = jm}$

for $i = m$, $1 \leq j \leq n$

Heat left: $-\frac{2hP}{LSK} = -3u_{1,j} + 4u_{2,j} - u_{3,j}$

$-\frac{2hP}{LSK} = -3V_{1+(j-1)m} + 4V_{2+(j-1)m} - V_{3+(j-1)m}$

$eq_1: A(t,t) = -3$
$eq_2: A(t,t+1) = 4$
$eq_3: A(t,t+2) = -1$
$b(t) = -\frac{2hP}{LSK}$

$\boxed{t = 1 + (j-1)m}$

for $i = 1$, $1 \leq j \leq L$

Problem 3

In this section, the equations from Problems 1 and 2 are transformed into code. The function set_up_matrix_3_4 constructs both the outer and the inner boundaries of the system, which is then used in code prob3 to calculate the heat distribution of the plate. The system is described according to the following table 1, using n, m = 10, an ambient temperature of 20 ℃, and heat entering the plate on the whole side meaning L = L_y.

Table 1: System parameters of problem 3

L_x [cm]	L_y [cm]	δ [cm]	P [W]	K [W/cm℃]	H [W/cm^2℃]
2	2	0.1	5	1.68	0.005

Set_up_matrix_3_4 is generating a $\bar{A}\bar{v} = \bar{b}$ system. The matrix $\bar{A}$ has mn x mn dimensions, and the vector $\bar{b}$ has 1 x mn dimensions. The inner systems equations and boundary conditions are generated in for-loops with n and m conditions as shown in problem 2 and section B.

The main code defines the values of n and m and uses set_up_matrix_3_4 to compute $\bar{v}$ as shown in line 7 of figure 1. The vector is then reshaped and plotted with x and y coordinates displaying the dimensions of the height of the plate, z which shows the distribution of temperature.

```
1  close all; clear all; clc;
2  m = 10;
3  n = 10;
4
5  [A,b] = set_up_matrix_3_4(m,n);
6
7  v = (A\b)+20;
8  v_matrix = reshape(v(1:m*n),m,n); %here we reshape from vector to matrix so it matches the X,Y
       ↪ boundaries
9
10 [X,Y] = meshgrid(1:m,1:n);
11
12
13 mesh(X,Y,v_matrix)
14 colorbar
```

Figure 1: Main code problem 3

The results are shown in figure 2. The highest temperature is 164.9626 ℃ (yellow), both in (1,1) and (1,n). These are the corners of the side where the heat is entering the plate.

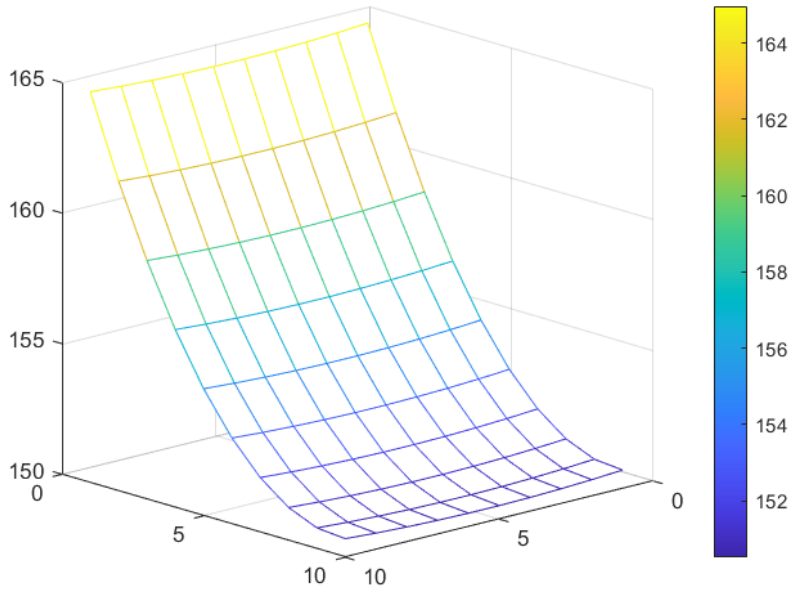


Figure 2: The heat distribution of the system in problem 3

Error Analysis

Problem 4

Problem 4 is about minimizing the computed error of the estimated temperature while finding suitable values for m and n to keep the running time short. This is done by comparing systems with $m, n \in (10, 20, ..., 90)$ with a very accurate but slow computation of $m = n = 100$. The values will then be used in the following problem sections.

The system for $m = n = 100$ was solved in the same way as the previous problem 3 by using set_up_matrix_3_4 (section B). In order to show the temperature distribution on the plate, the v vector needed to be reshaped to a matrix, so it would match the X,Y boundaries used in the figure 3.

```
%% Problem 4.1
close all; clear all; clc;

m = 100;
n = 100;
[A,b] = set_up_matrix_3_4(m,n);
v = (A\b)+20;
v_matrix = reshape(v(1:m*n),m,n); %here we reshape from vector to matrix so it maches the X,Y
    ↪ boundaries

[X,Y] = meshgrid(1:m,1:n);

mesh(X,Y,v_matrix)
colorbar

refrence_val = v_matrix(1,1); %keep the refrence value for comparision
```

Figure 3: $m = n = 100$ code problem 4

The heat distribution of the system can be seen in figure 4. The highest temperature is 164.8084 ℃ (yellow), both in (1,1) and (1,n), as expected, and that value is kept to use as a reference and comparison.

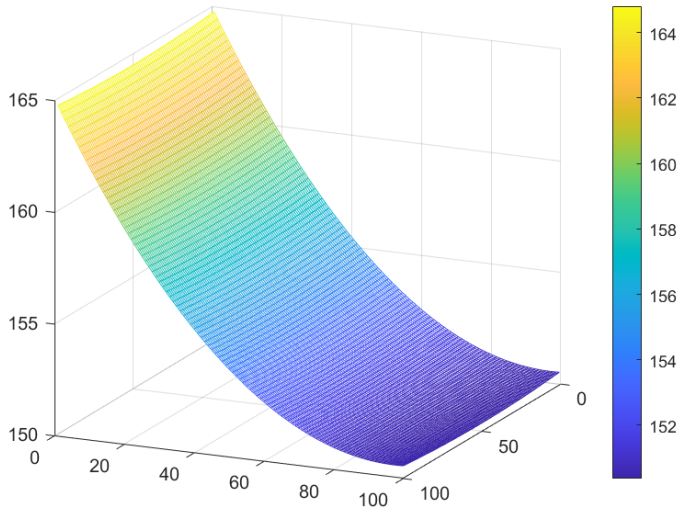


Figure 4: The heat distribution of the system in problem 4

Then, the system was solved for $m, n \in (10, 20, ..., 90)$, a total of 9^2 times. Each time the deviation from the reference value $m = n = 100$ was computed and the answers were saved in a matrix. As seen in figure 5 two for-loops were used to look at m as a constant while running n through 9 different value outcomes and calculating the deviation from the reference value, then the value of m was increased, and the same procedure was used. This was repeated 9 times for a total of 81 different values for comparison.

```
1  %% 4.2 Compare the values to reference value
2
3  m = 10;
4  n = 10;
5  val_count = 1;
6  m_count = 1;
7  n_count = 1;
8  for i = 1:9
9      for j = 1:9
10         [A,b] = set_up_matrix_3_4(m,n);
11         v = (A\b)+20;
12         v_matrix = reshape(v(1:m*n),m,n);
13         comp_val = abs(v_matrix(1,1) - refrence_val);
14         compare_vec(j,i) = comp_val;
15
16         if comp_val < 0.01
17             wanted_error(val_count) = comp_val;
18             m_vec(m_count) = m;
19             n_vec(n_count) = n;
20
21             val_count = val_count+1; m_count = m_count+1; n_count = n_count+1;
22         end
23         n = n+10;
24     end
25     m = m+10;
26     n = 10;
27 end
28
29 wanted_values = [m_vec' n_vec' wanted_error'];
```

Figure 5: Comparison code problem 4

A conclusion was found when comparing the outcome of increasing m and increasing n, from the matrix which is computed in line 14 (figure 5), the deviation from the maximum temperature and the reference temperature was larger when m was increased. The largest deviations in each case are shown in tables 2 and 3.

Table 2: Deviation from the maximum temperature and the reference temperature when increasing n

	$m = 10, n = 10$	$m = 10, n = 90$	difference
error:	0.1541	0.1458	**0.0083**

Table 3: Deviation from the maximum temperature and the reference temperature when increasing m

	$m = 10, n = 10$	$m = 90, n = 10$	difference
error:	0.1541	0.0090	**0.1451**

Two variables are used to determine the most efficient values of n and m. The running time should be less than 0.5 sec (rather much lower) and the deviation from the reference temperature needs to be less than 0.01 ℃. The deviation of the values are computed in line 16 figure 5. Those values are then compared for the shortest

running time, as shown in figure 6 (line 13). The results show that $m = 50$ and $n = 20$ are the ideal values. These values result in the smallest running time of 0.0360s while at the same time ensuring a temperature deviation of 0.0062℃, which is smaller than the tolerance of 0.01℃ and 0.5s.

```
1  %% 4.3 Get the most time efficient values
2
3  counter = 1;
4  for i = 1:length(wanted_values)
5      m = wanted_values(i,1);
6      n = wanted_values(i,2);
7      tic
8      [A,b] = set_up_matrix_3_4(m,n);
9      v = (A\b)+20;
10     line = toc;
11     v_matrix = reshape(v(1:m*n),m,n);
12
13     if line < 0.5
14         time(counter,1) = i; %column one displays in which line the most efficient values appears in
15         time(counter,2) = line;
16         counter = counter + 1;
17     end
18
19 end
20
21 efficient_val = wanted_values(9,1:2);
```

Figure 6: Most efficient value code problem 4

Varying power intake

Problem 5

This section analyzes the system of problem 3 when the power is entering the plate on the lower half of the left-hand side, as visualized on page 2. To solve this system, the code of problem 3 is modified and used with the m,n values of problem 4. The same system parameters from problem 3 were used, but this time with

$$L_x = L_y = 4cm$$

and therefore $L = 2cm$.

When changing the boundary condition of the power intake on the left side, the matrix of the system changes. As seen in section E, the left boundary condition is generated with a for-loop by using the ratio of the y-axis with n to determine the length. The rest of the code remains the same as in problem 3 without having any if-conditions to move the power source. The main code of this section uses Set_up_matrix5 like in problem 3 to display the heat distribution as shown below in 7.

```
close all; clear all; clc;
m = 10;
n = 10;

[A,b] = set_up_matrix_3_4(m,n);

v = (A\b)+20;
v_matrix = reshape(v(1:m*n),m,n); %here we reshape from vector to matrix so it matches the X,Y
    ↪ boundaries

[X,Y] = meshgrid(1:m,1:n);


mesh(X,Y,v_matrix)
colorbar
```

Figure 7: Main code problem 5

The results in figure 8 show that the highest temperature is 72.6226℃ located at (1,1) which is the initial point in the project's setup. From there, the heat distributes evenly through the plate in all directions.

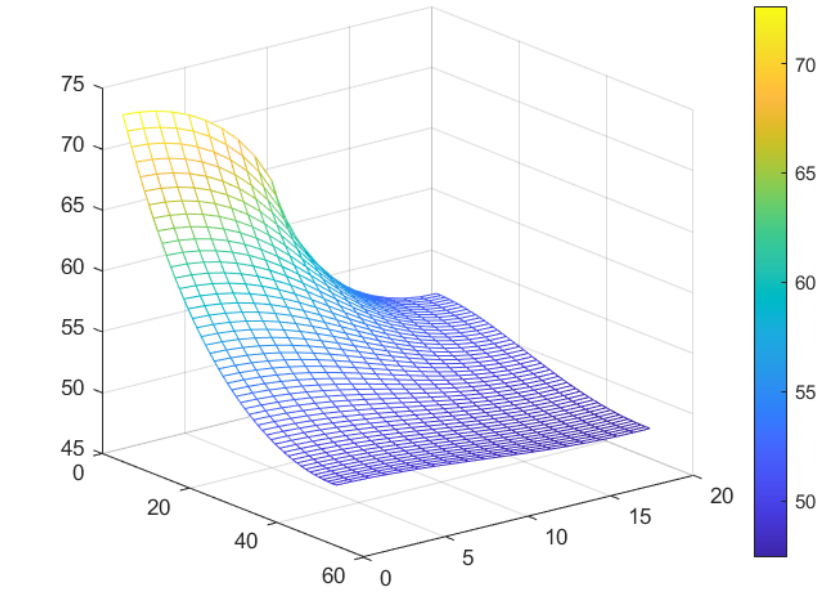


Figure 8: The heat distribution of the system in problem 5

Problem 6

In this section, the position of the power source is analyzed by moving the processor along the left side of the system of Problem 5. By moving the processor from (1,1) to $(1,\frac{n}{2})$ the maximum temperature of each run is compared to find the ideal position with the lowest maximum temperature.

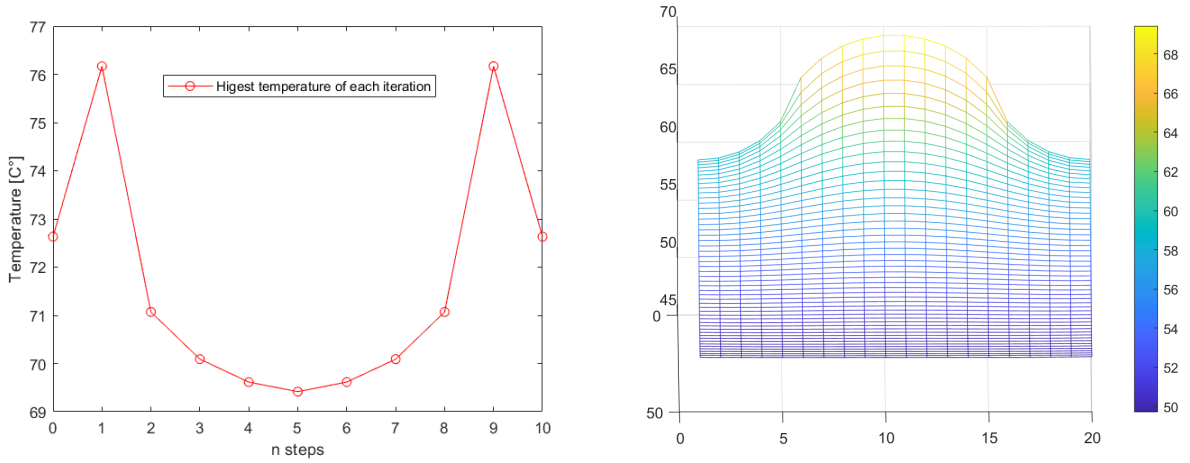
The power source is moved with a for-loop as seen in the main code of problem 6, figure 9. The set_up_the_ matrix function takes the loop value to initialize if-conditions. The if-conditions are used to move the power source along the left-hand side. This way it covers the scenarios when the power is not moved or moved all the way from the bottom (the initial point at $(1,\frac{n}{2})$), and if the power is in between these situations. After the matrix is set up, the main code solves the system, stores the maximum temperature value in one vector in each iteration, and plots a 3D graph of the heat distribution. Once the for loop has reached $\frac{n}{2}$ iterations, the maximum temperature is plotted over the distance to the origin (1,1).

```
%% Problem 6
close all; clear all; clc;
P = 5; %W
H = 0.005; %W/cm^2 * C
K = 1.68; %W/cm^2 * C
m = 50;
n = 20; %values from Problem 4
Lx = 4;
Ly = 4;
L = 2;

counter = 1;
for i = 0:n/2
    [A,b] = set_up_the_matrix(m,n,P,H,K,L,Lx,Ly,i);
    v = (A\b)+20;
    v_matrix = reshape(v(1:m*n),m,n); %here we reshape from vector to matrix so it maches the X,Y
    ↪ boundaries
    v_matrix = v_matrix';
    v_highest(counter,1) = max(v);
    v_highest(counter,2) = i;

    figure
    [X,Y] = meshgrid(1:m,1:n);
    mesh(X,Y,v_matrix)
    colorbar
    %view(2)
    counter = counter + 1;
end

figure
plot(0:10,v_highest(:,1),'r-o')
ylabel('Temperature [C ]')
xlabel('n steps')
legend('Higest temperature of each iteration','Location','Best')
```

Figure 9: Main code problem 6

The results show that the lowest maximum temperature occurs when the processor is positioned in the middle of the left side. This leaves five points without power intake from the processor near the top and bottom of the plate, as shown in figure 10 (b). Graph (a) shows the highest temperature of the plate according to the position of the processor, with x-axis being the distance of the beginning of the processor to the edge point of the plate

(1,1). The code of problem 3 is used to generate the picture of the heat distribution on the right. The maximum heat at the optimal position when the processor is in the middle of the left side is 69.4034 ℃.



(a) Maximum temperature with respect to the n steps, which is the distance from the initial point (1,1) to the position where the heat starts to enter the plate

(b) Heat distribution with processor position at (1,5) to (1,15)

Figure 10: The heat distribution of the system when the position of the processor changes

Problem 7

In this Problem, using the position found in Problem 6 (that gave the lowest maximum temperature of the plate), the bisection method was used to find the maximum power intake, P, so the plate temperature never exceeded 100℃. In order to use the bisection method, a new function was made, calc_max_temp.m in figure 11, to calculate the maximum temperature for the bisection method.

```
function max_temp = calc_max_temp(m,n,P,H,K,L,Lx,Ly,move)

   [A,b] = test(m,n,P,H,K,L,Lx,Ly,move);
    v = (A\b)+20;
   max_temp = max(v)-100;

end
```

Figure 11: Code that calculates the maximum temperature for the bisection method

In order to use the bisection method, values a and b need to be stated as starting points, and, tol (tolerance) which is how accurate the answer should be. In this problem, P was found with 2 significant digits. The minimum power, a, is set to be 5 W since that is the value that has been used in previous problems and that value has been giving a temperature value of less than 80℃, so it is clear that the power has to increase. The maximum value, b, is set to 10 W which was found by testing as that gave a temperature value exceeding 100℃. The bisection function is then called in the main file, seen in figure 12. The bisect3 function can be seen in section H.

```
%% Problem 7 with bisection method
close all; clear all; clc;


H = 0.005; %W/cm^2 * C
K = 1.68; %W/cm^2 * C
m = 50;
n = 20; %values from Problem 4
Lx = 4;
Ly = 4;
L = 2;
move = 5; %posision where the heat starts to enter
a = 5; %minimum power
b = 10; %maximum power
tol = 10^-2;

max_power_allowed = bisect3(a,b,tol,m,n,H,K,L,Lx,Ly,move);

T = calc_max_temp(m,n,max_power_allowed,H,K,L,Lx,Ly,move)+100;
%have to add 100 to the T value since it is a midpoint (zero)

fprintf('\nThe maximum power allowed is: %.2f W\nWhich results in maximum plate temperature: %.2f C
    ↪ \n',max_power_allowed,T)
```

Figure 12: Main code for problem 7

The results are shown in table 4, where the maximum power intake, P, was determined and the maximum temperature it produces.

Table 4: The maximum power intake found by using the bisection method, without exceeding 100 °C

Maximum power allowed	Maximum plate temperature
8.09 W	99.89 °C

Varying material

Problem 8

In this Problem the same restrictions are used for the temperature using the same position for the processor as in Problems 6 and 7. However, it is assumed that the thermal conductivity K varies from 1 to 5 W/cm ℃. Each value of K, therefore, results in a specific highest power intake, which keeps the plate under 100 ℃. This value is computed and plotted as a function of K on the interval [1,5] with 25 iterations.

The main code of Problem 8 is defining K as a vector from 1 to 5 with 25 values. From there, a for-loop iterates K over all the values and computes the maximum power allowed at the maximum temperature of 100 ℃ as explained in Problem 7 and stores the values in one vector for K and one for the power P, as shown in figure 13. After the for-loop, the vectors are plotted.

```
%% Problem 8
close all; clear all; clc;

H = 0.005;
m = 50;
n = 20; %values from Problem 4
Lx = 4;
Ly = 4;
L = 2;
move = 5; %posision where the heat starts to enter
a = 5; %minimum power
b = 12; %maximum power
tol = 10^-2;

K = linspace(1,5,25);

for j = 1:length(K)
    max_power_allowed = bisect3(a,b,tol,m,n,H,K(j),L,Lx,Ly,move);
    T = calc_max_temp(m,n,max_power_allowed,H,K(j),L,Lx,Ly,move)+100;
    P_vec(j) = max_power_allowed;
    K_vec(j) = K(j);
end


plot(K_vec,P_vec,'LineWidth',1.5)
ylabel('Maximum Power allowed [W]')
xlabel('Thermal Conductivity of material K [W/cm C]')
```

Figure 13: Main code for problem 8

Figure 14 shows when K is increased, the highest power intake also increases. This seems reasonable due to the increasing ability to conduct heat. When more heat is conducted, the system can take in more energy without exceeding 100℃.

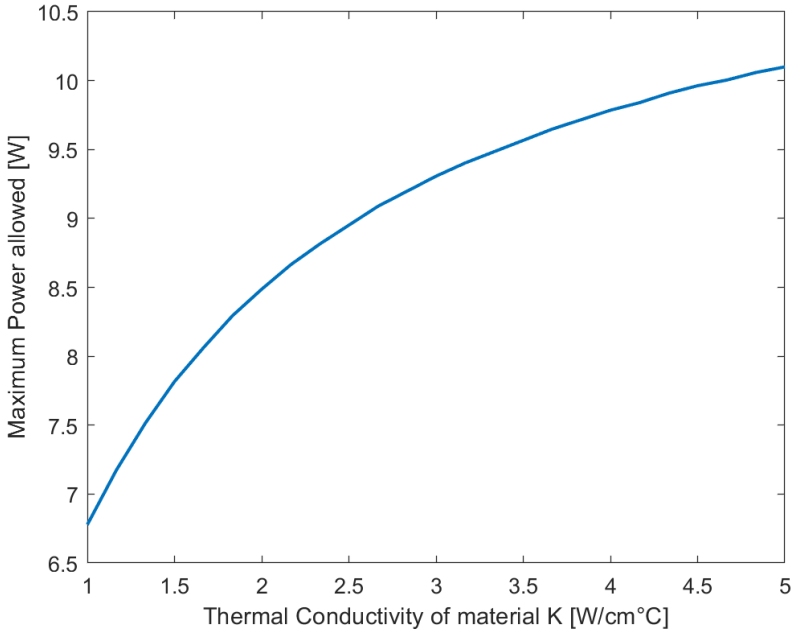


Figure 14: Relationship between each value of K and the highest power intake

Problem 9

In this section, the thermal conductivity value of K = 1.68 W/cm ℃ was used again. A small rectangular notch $(10m * 10n)$ was cut off the plate in the upper right corner. Therefore, the boundary conditions needed to be updated so that they would reflect the situation.

The boundaries for the inner points from the original setup were divided into two different sections this time to take into account the new boundaries that occur when the rectangular notch is cut off one corner. The rectangular notch was used and set up as so-called "ghost values" since they no longer are a part of the plate, so they were set equal to zero. When applying the new boundaries for the corner that was cut off $(x * y)$, it is important to dedicate the new corner to only one boundary condition. As it was set up before, the 90-degree corner is dedicated to the right boundary condition, resulting in shortening the x-side boundary condition by one m.

Figure 15 shows the highest temperature of the plate according to the position of the processor after the notch has been cut off, with the x-axis being the distance of the processor to the initial point (1,1). The graph shows that the lowest maximum temperature occurs when the processor is positioned in the middle of the left side (n=5). The temperatures on the left side of the graph are lower than on the right side and that is due to the location of the rectangle notch that was cut off. The area which is cut off is missing to radiate heat, and therefore the temperature when n=9 is now higher.

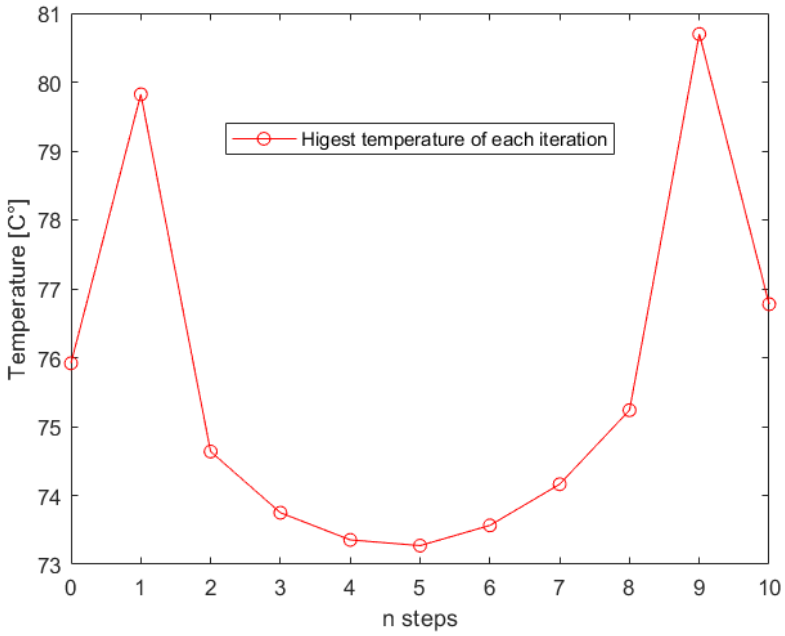
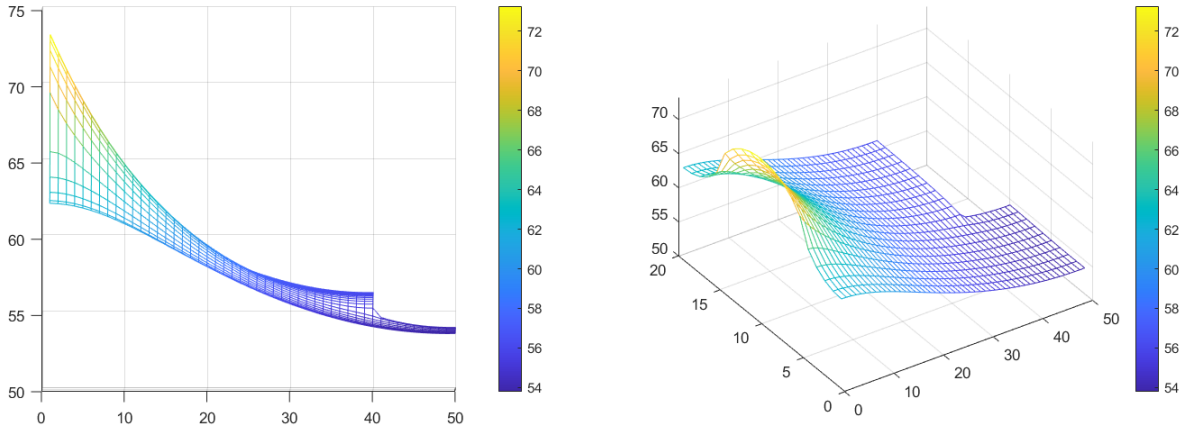


Figure 15: Maximum temperature with respect to the n steps, which is the distance from the initial point (1,1) to the position where the heat starts to enter the plate

Figure 16 shows the heat distribution of the system after the rectangular notch was cut off the plate. Figure 16a clearly shows how at the point near the rectangular notch (m = 50, n = 20) the temperature is higher than at the same distance away from the heat source on the opposite edge at (m = 50, n = 1). The maximum temperature of the plate is 80.6979℃. This underlines how the material is connected through the points and how important one small surface like this is to radiate heat away from the power source.



(a) Heat distribution of xy view

(b) Heat distribution of xyz view

Figure 16: The heat distribution of the system when a rectangular notch is cut off the plate in the upper right corner.

Using the position (n=5) from figure 15, which gave the lowest maximum temperature of the plate, the bisection method was used (like in Problem 7) to determine the maximum allowed power intake, P, so the temperature of the plate would never exceed 100℃. The results are shown in table 5, where the maximum power intake, P, was determined and the maximum temperature it produces.

Table 5: The maximum power intake found by using the bisection method, without exceeding 100 ℃

Maximum power allowed	Maximum plate temperature
7.50 W	99.91℃

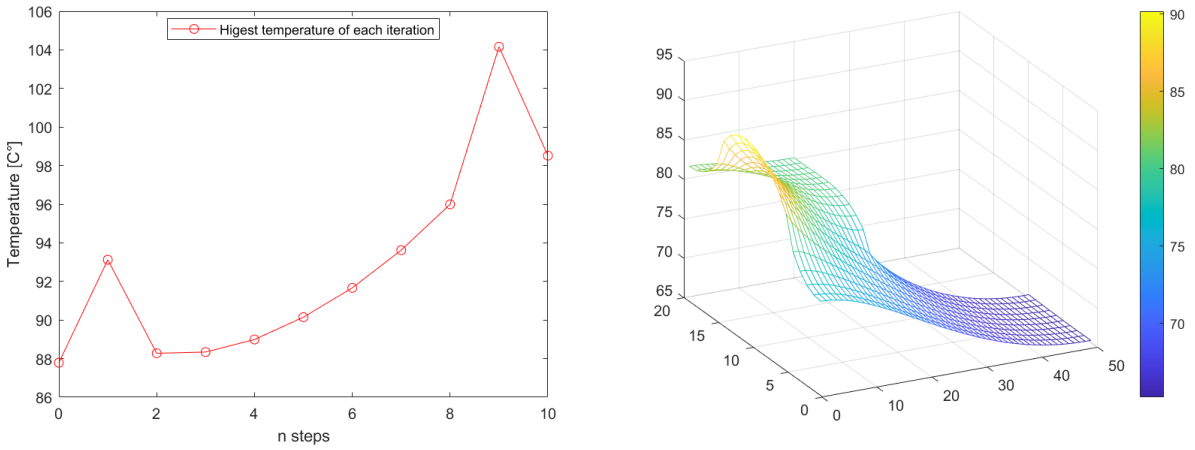
Comparing the results from table 5 with table 4, it can be concluded that less power is needed to heat up the plate with less surface area to 100℃, since the heat distribution isn't as good as the plate which has a larger surface area.

Independent Work

Two different experiments were made in the independent work section. The behavior of the system is analyzed when the rectangular notch that is cut off is increased, and then the influence of the convective heat transfer coefficient H on the system.

Analyzing the behavior of the system with decreased surface area

The rectangular notch that is cut off affects the whole system as seen in Problem 9. By increasing the notch, the system loses surface area to radiate heat. As expected, this results in higher temperatures when the removed surface area is larger. Figure 17 shows the system when the notch $(30m * 10n)$ cuts away 30% of the total surface area, which is three times the surface area of problem 9 notch, resulting in a maximum temperature of 104.162℃.

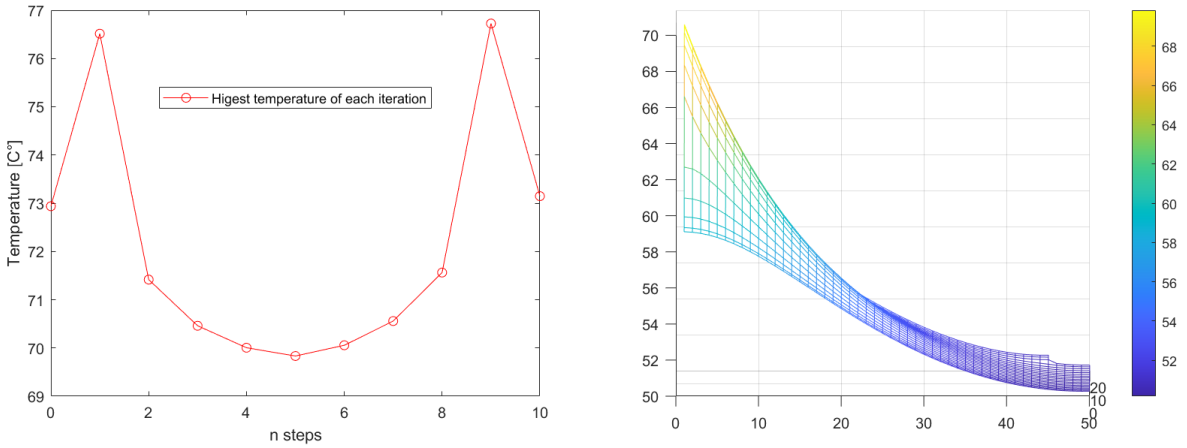


(a) Maximum temperature with respect to the n steps, which is the distance from the initial point (1,1) to the position where the heat starts to enter the plate

(b) Heat distribution with processor position at (1,5) to (1,15)

Figure 17: The heat distribution of the system when a rectangular notch is cut off the plate in the upper right corner when $(30m * 20n)$.

When the notch is relatively small, the effect is still significant, as seen in figure 18. With a notch of $5m * 2n$, both values represent 10% of the plate edge, representing then 1% of the total surface area.



(a) Maximum temperature with respect to the n steps, which is the distance from the initial point (1,1) to the position where the heat starts to enter the plate

(b) Heat distribution with processor position at (1,5) to (1,15)

Figure 18: The heat distribution of the system when a rectangular notch is cut off the plate in the upper right corner when $(5m * 2n)$.

The small notch results in a maximum temperature of 76.7259℃ which is more than 7 degrees higher than without notch as seen in table 6.

Table 6: The relationship between the size of the notch and the maximum temperature

Size of rectangular notch	% of total surface area	Maximum temperature
$(0m * 0n)$	0%	69.4034 ℃
$(5m * 2n)$	1%	76.7259 ℃
$(10m * 10n)$	10%	80.6979 ℃
$(30m * 10n)$	30%	104.162 ℃

Figure 19 shows the temperature over the size of the notch relative to the surface area with the values of table 6. The graph shows how the temperature increases when the notch increases in four data points.

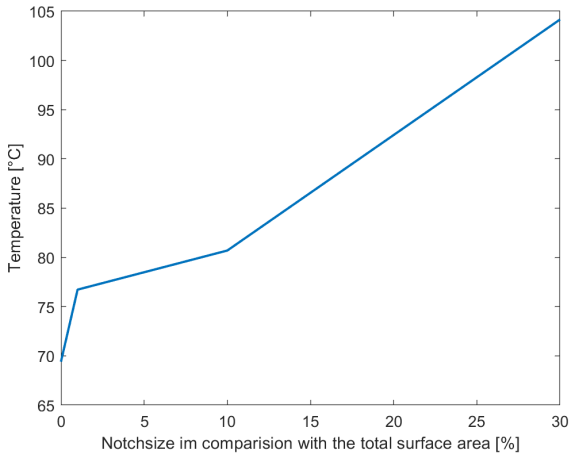


Figure 19: Maximum temperature with respect to the size of the notch relative to the total surface area.

Effect of the constant H on the heating of the plate

The same experiment as in Problem 8 was done but instead of assuming the thermal conductivity K is variable it is assumed that the convective heat transfer coefficient H is variable from 0.005 to 0.1 $\frac{W}{cmC}$. The bisection method was used to get the maximum allowable power so the maximum plate temperature would not exceed 100℃. The maximum allowable power was plotted with respect to the convective heat transfer coefficient in figure 20.

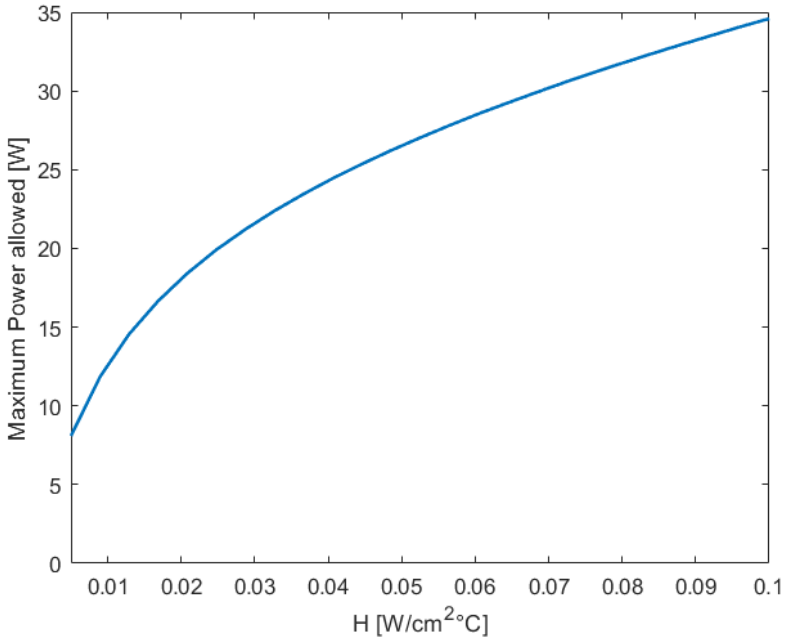


Figure 20: The influence of the convective heat transfer coefficient has on the maximum allowable power

What can be interpreted from the graph is that if the plate is surrounded by air which has $H = 0.005$, less power is needed to heat the plate up to 100℃. And on the other hand, if the plate is surrounded by water which has $H = 0.1$ significantly more power is needed to heat up the plate to 100℃.

A Main code Problem 3

```
close all; clear all; clc;
m = 10;
n = 10;

[A,b] = set_up_matrix_3_4(m,n);

v = (A\b)+20;
v_matrix = reshape(v(1:m*n),m,n); %here we reshape from vector to matrix so it matches the X,Y
    ↪ boundaries

[X,Y] = meshgrid(1:m,1:n);


mesh(X,Y,v_matrix)
colorbar
```

B Set_up_matrix_3_4

```
function [A b] = set_up_matrix_3_4(m,n)
P = 5; %W
L = 2; %cm
Lx = 2; Ly = 2; delta = 0.1; %cm
h = Lx/(m-1); k = Ly/(n-1);
H = 0.005; %W/cm^2 * C
K = 1.68; %W/cm^2 * C
A = zeros(m*n,m*n); b = zeros(m*n,1);

%INNER MATRIX
for i = 2:m-1
    for j = 2:n-1
        eq = i + (j-1)*m;
        A(eq,eq) = -((2/h^2) + (2/k^2) + (2*H/(K*delta))); %%%%
        A(eq,eq+1) = 1/h^2;
        A(eq,eq-1) = 1/h^2;
        A(eq,eq+m) = 1/k^2;%%%%%
        A(eq,eq-m) = 1/k^2;
    end
end

%Left boundary (All power)
for j = 1:n   %LATER USE n = L
   i = 1; eq = i+(j-1)*m;
   A(eq,eq) = -3;
   A(eq,eq+1) = 4;
   A(eq,eq+2) = -1;
   b(eq) = (-2*h*P)/(L*delta*K);
end

%RIGHT BOUNDARY
for j = 1:n
   i = m; eq = i+(j-1)*m;
   A(eq,eq) = -3+((2*h*H)/K);
   A(eq,eq-1) = 4;
   A(eq,eq-2) = -1;
   b(eq) = 0;
```

```
end

%BOTTOM BOUNDARY
for i = 2:m-1
   j = 1; eq = i+(j-1)*m;
   A(eq,eq) = -3+((2*k*H)/K);
   A(eq,eq+m) = 4;
   A(eq,eq+2*m) = -1;
   b(eq) = 0;
end

%TOP BOUNDARY
for i = 2:m-1
   j = n; eq = i+(j-1)*m;
   A(eq,eq) = -3+((2*k*H)/K);
   A(eq,eq-m) = 4;
   A(eq,eq-2*m) = -1;
   b(eq) = 0;
end
```

C Main code Problem 4

```
%% Problem 4.1
close all; clear all; clc;

m = 100;
n = 100;
[A,b] = set_up_matrix_3_4(m,n);
v = (A\b)+20;
v_matrix = reshape(v(1:m*n),m,n); %here we reshape from vector to matrix so it maches the X,Y
    ↪ boundaries

[X,Y] = meshgrid(1:m,1:n);

mesh(X,Y,v_matrix)
colorbar

refrence_val = v_matrix(1,1); %keep the refrence value for comparision


%% 4.2 Compare the values to refrence value

m = 10;
n = 10;
val_count = 1;
m_count = 1;
n_count = 1;
for i = 1:9
    for j = 1:9
        [A,b] = set_up_matrix_3_4(m,n);
        v = (A\b)+20;
        v_matrix = reshape(v(1:m*n),m,n);
        comp_val = abs(v_matrix(1,1) - refrence_val);
        compare_vec(j,i) = comp_val;

        if comp_val < 0.01
            wanted_error(val_count) = comp_val;
```

```
            m_vec(m_count) = m;
            n_vec(n_count) = n;

            val_count = val_count+1; m_count = m_count+1; n_count = n_count+1;
        end
        n = n+10;
    end
    m = m+10;
    n = 10;
end

wanted_values = [m_vec' n_vec' wanted_error'];

%% 4.3 Get the most time efficient values


counter = 1;
for i = 1:length(wanted_values)
    m = wanted_values(i,1);
    n = wanted_values(i,2);
    tic
    [A,b] = set_up_matrix_3_4(m,n);
    v = (A\b)+20;
    line = toc;
    v_matrix = reshape(v(1:m*n),m,n);

    if line < 0.5
        time(counter,1) = i; %column one displays in which line the most efficient values appears in
    ↪ ,in
        time(counter,2) = line;
        counter = counter + 1;
    end

end

efficient_val = wanted_values(9,1:2);

%% 4.4 printing correct values

n_matter1 = compare_vec(1,1);

n_matter2 = compare_vec(end,1);

m_matter1 = compare_vec(1,1);
m_matter2 = compare_vec(1,end);

chosen_temperature_deviation = wanted_values(9,3);
%best_runing_time =

fprintf('\nWhen m = 10 and n = 10 we get the error: %.4f',n_matter1)
fprintf('\nWhen m = 10 and n = 90 we get the error: %.4f',n_matter2)

fprintf('\nWhen n = 10 and m = 10 we get the error: %.4f',m_matter1)
fprintf('\nWhen n = 10 and m = 90 we get the error: %.4f',m_matter2)


fprintf('\n\nThe most efficient values were\n m = %.f\n n = %.f\n',efficient_val(1), efficient_val
    ↪ (2))
```

```
%%%%%%%%%%%%%%%%%%% FPRINF --> SHOW ALSO THE TEMPERATURE CHANGE AND RUNNING TIME WE CHOSE %%%%%%%%%%%%%%%%%%%%%
fprintf('\nWhere these m,n values give the temperature change: %.4f C\nAnd the running time is: %.4fs\n',chosen_temperature_deviation,min(time(:,2)))
```

D Main code Problem 5

```
close all; clear all; clc;
m = 50;
n = 20; %values from Problem 4
L = 2; %cm
Lx = 4;
Ly = 4;

[A,b] = set_up_matrix5(m,n,L,Lx,Ly);

v = (A\b)+20;
v_matrix = reshape(v(1:m*n),m,n); %here we reshape from vector to matrix so it maches the X,Y boundaries
v_matrix = v_matrix';
[X,Y] = meshgrid(1:m,1:n);


mesh(X,Y,v_matrix)
colorbar
```

E Set_up_matrix Problem 5

```
function [A b] = set_up_matrix5(m,n)
P = 5; %W
ratio = L/Ly *n; %scaling the y axis to position the power source
delta = 0.1; %cm
h = Lx/(m-1);
k = Ly/(n-1);
H = 0.005; %W/cm^2 * C
K = 1.68; %W/cm^2 * C
A = zeros(m*n,m*n);
b = zeros(m*n,1);

%INNER MATRIX
for i = 2:m-1
    for j = 2:n-1
        eq = i + (j-1)*m;
        A(eq,eq) = -((2/h^2) + (2/k^2) + (2*H/(K*delta))); %%%%
        A(eq,eq+1) = 1/h^2;
        A(eq,eq-1) = 1/h^2;
        A(eq,eq+m) = 1/k^2;%%%%%
        A(eq,eq-m) = 1/k^2;
    end
end


%Left boundary no power
for j = ratio:n
```

```
   i = 1; eq = i+(j-1)*m;
   A(eq,eq) = -3+((2*h*H)/K);
   A(eq,eq+1) = 4;
   A(eq,eq+2) = -1;
end


%Left boundary (All power)
for j = 1:ratio   %LATER USE n = L
   i = 1; eq = i+(j-1)*m;
   A(eq,eq) = -3;
   A(eq,eq+1) = 4;
   A(eq,eq+2) = -1;
   b(eq) = (-2*h*P)/(L*delta*K);
end

%RIGHT BOUNDARY
for j = 1:n
   i = m; eq = i+(j-1)*m;
   A(eq,eq) = -3+((2*h*H)/K);
   A(eq,eq-1) = 4;
   A(eq,eq-2) = -1;
   b(eq) = 0;
end

%BOTTOM BOUNDARY
for i = 2:m-1
   j = 1; eq = i+(j-1)*m;
   A(eq,eq) = -3+((2*k*H)/K);
   A(eq,eq+m) = 4;
   A(eq,eq+2*m) = -1;
   b(eq) = 0;
end

%TOP BOUNDARY
for i = 2:m-1
   j = n; eq = i+(j-1)*m;
   A(eq,eq) = -3+((2*k*H)/K);
   A(eq,eq-m) = 4;
   A(eq,eq-2*m) = -1;
   b(eq) = 0;
end
```

F Main code Problem 6

```
%% Problem 6
close all; clear all; clc;
P = 5; %W
H = 0.005; %W/cm^2 * C
K = 1.68; %W/cm^2 * C
m = 50;
n = 20; %values from Problem 4
Lx = 4;
Ly = 4;
L = 2;

counter = 1;
```

```
for i = 0:n/2
    [A,b] = set_up_the_matrix(m,n,P,H,K,L,Lx,Ly,i);
    v = (A\b)+20;
    v_matrix = reshape(v(1:m*n),m,n); %here we reshape from vector to matrix so it maches the X,Y
    ↪ boundaries
    v_matrix = v_matrix';
    v_highest(counter,1) = max(v);
    v_highest(counter,2) = i;

    figure
    [X,Y] = meshgrid(1:m,1:n);
    mesh(X,Y,v_matrix)
    colorbar
    %view(2)
    counter = counter + 1;
end

figure
plot(0:10,v_highest(:,1),'r-o')
ylabel('Temperature [C ]')
xlabel('n steps')
legend('Higest temperature of each iteration','Location','Best')
```

G calc_max_temp

```
function max_temp = calc_max_temp(m,n,P,H,K,L,Lx,Ly,move)

   [A,b] = test(m,n,P,H,K,L,Lx,Ly,move);
    v = (A\b)+20;
   max_temp = max(v)-100;

end
```

H bisect3

```
function xc = bisect3(a,b,tol,m,n,H,K,L,Lx,Ly,move)

f=@(P)calc_max_temp(m,n,P,H,K,L,Lx,Ly,move);
if sign(f(a))*sign(f(b)) >= 0
  error('f(a)f(b)<0 not satisfied!') %ceases execution
end
fa=f(a);
while (b-a)/2>tol
  c=(a+b)/2;
  fc=f(c);
  if fc == 0              %c is a solution, done
    break
  end
  if sign(fc)*sign(fa)<0  %a and c make the new interval
    b=c;
  else                    %c and b make the new interval
    a=c;fa=fc;
  end
end
xc=a; %takes the left side of the point (lower boundary to not get a number higher than 100 C )
```

```
%new midpoint is best estimate
```

I Main code Problem 7

```
%% Problem 7 with bisection method
close all; clear all; clc;


H = 0.005; %W/cm^2 * C
K = 1.68; %W/cm^2 * C
m = 50;
n = 20; %values from Problem 4
Lx = 4;
Ly = 4;
L = 2;
move = 5; %posision where the heat starts to enter
a = 5; %minimum power
b = 10; %maximum power
tol = 10^-2;

max_power_allowed = bisect3(a,b,tol,m,n,H,K,L,Lx,Ly,move);

T = calc_max_temp(m,n,max_power_allowed,H,K,L,Lx,Ly,move)+100;
%have to add 100 to the T value since it is a midpoint (zero)

fprintf('\nThe maximum power allowed is: %.2f W\nWhich results in maximum plate temperature: %.2f C
    ↪ \n',max_power_allowed,T)
```

J Main code Problem 8

```
%% Problem 8
close all; clear all; clc;

H = 0.005;
m = 50;
n = 20; %values from Problem 4
Lx = 4;
Ly = 4;
L = 2;
move = 5; %posision where the heat starts to enter
a = 5; %minimum power
b = 12; %maximum power
tol = 10^-2;

K = linspace(1,5,25);

for j = 1:length(K)
    max_power_allowed = bisect3(a,b,tol,m,n,H,K(j),L,Lx,Ly,move);
    T = calc_max_temp(m,n,max_power_allowed,H,K(j),L,Lx,Ly,move)+100;
    P_vec(j) = max_power_allowed;
    K_vec(j) = K(j);
end


plot(K_vec,P_vec,'LineWidth',1.5)
```

```
ylabel('Maximum Power allowed [W]')
xlabel('Thermal Conductivity of material K [W/cm C]')
```

K calc_max_temp9

```
function max_temp = calc_max_temp9(m,n,P,H,K,L,Lx,Ly,move,x,y,External_temp)

   [A,b] = set_up_matrix9(m,n,P,H,K,L,Lx,Ly,move,x,y);
    v = (A\b);
    for index = 1:length(v)
        if v(index) < 0.00001
            v(index) = NaN;
        else
            v(index) = v(index)+External_temp;
        end
    end;
   max_temp = max(v)-100;

end
```

L bisect9

```
function xc = bisect9(a,b,tol,m,n,H,K,L,Lx,Ly,move,x,y,External_temp)

f=@(P)calc_max_temp9(m,n,P,H,K,L,Lx,Ly,move,x,y,External_temp);
if sign(f(a))*sign(f(b)) >= 0
  error('f(a)f(b)<0 not satisfied!') %ceases execution
end
fa=f(a);
while (b-a)/2>tol
  c=(a+b)/2;
  fc=f(c);
  if fc == 0              %c is a solution, done
    break
  end
  if sign(fc)*sign(fa)<0  %a and c make the new interval
    b=c;
  else                    %c and b make the new interval
    a=c;fa=fc;
  end
end
xc=a; %takes the left side of the point (lower boundary to not get a number higher than 100 C )
%new midpoint is best estimate
```

M Set_up_matrix Problem 9

```
function [A b] = set_up_matrix9(m,n,P,H,K,L,Lx,Ly,move, x, y)
ratio = (L/Ly)*n; %scaling the y axis to position the power source
delta = 0.1; %cm
h = Lx/(m-1);
k = Ly/(n-1);
%H = 0.005; %W/cm^2 * C
%K = 1.68; %W/cm^2 * C
```

```
A = zeros(m*n,m*n);
b = zeros(m*n,1);

%INNER MATRIX 2
for i = 2:m-1
    for j = 2:n-y-1
        eq = i + (j-1)*m;
        A(eq,eq) = -((2/h^2) + (2/k^2) + (2*H/(K*delta)));
        A(eq,eq+1) = 1/h^2;
        A(eq,eq-1) = 1/h^2;
        A(eq,eq+m) = 1/k^2;
        A(eq,eq-m) = 1/k^2;
    end
end

%INNER MATRIX for top piece 1

for i = 2:m-1-x
    for j = n-y : n-1
        eq = i + (j-1)*m;
        A(eq,eq) = -((2/h^2) + (2/k^2) + (2*H/(K*delta)));
        A(eq,eq+1) = 1/h^2;
        A(eq,eq-1) = 1/h^2;
        A(eq,eq+m) = 1/k^2;
        A(eq,eq-m) = 1/k^2;
    end
end

%INNER MATRIX for x-y
for i = m-x+1:m
    for j = n-y+1:n
        eq = i + (j-1)*m;
        A(eq,eq) = -((2/h^2) + (2/k^2) + (2*H/(K*delta)));
        A(eq,eq+1) = 0;
        A(eq,eq-1) = 0;
        A(eq,eq+m) = 0;
        A(eq,eq-m) = 0;

    end
end




if move == 0
 %Left boundary no power
    for j = ratio+1:n %changed ratio to ratio + 1 (works the same)
       i = 1; eq = i+(j-1)*m;
       A(eq,eq) = -3+((2*h*H)/K);
       A(eq,eq+1) = 4;
       A(eq,eq+2) = -1;
    end


    %Left boundary POWER
    for j = 1:ratio   %LATER USE n = L
       i = 1; eq = i+(j-1)*m;
       A(eq,eq) = -3;
       A(eq,eq+1) = 4;
```

```
        A(eq,eq+2) = -1;
        b(eq) = (-2*h*P)/(L*delta*K);
    end
end


if move > 0 || move < L
     %Left boundary no power (UPPER SIDE)
    for j = ratio + move + 1 : n
        i = 1; eq = i+(j-1)*m;
        A(eq,eq) = -3+((2*h*H)/K);
        A(eq,eq+1) = 4;
        A(eq,eq+2) = -1;
    end

     %Left boundary no power (LOWER SIDE)
    for j = 1:move
        i = 1; eq = i+(j-1)*m;
        A(eq,eq) = -3+((2*h*H)/K);
        A(eq,eq+1) = 4;
        A(eq,eq+2) = -1;
    end


    %Left boundary POWER (MIDDLE)
    for j = 1 + move : ratio + move%%%%%% %here we move power up and we have to delete the top so it
    ↪  doesn't get longer than L
        i = 1; eq = i+(j-1)*m;
        A(eq,eq) = -3;
        A(eq,eq+1) = 4;
        A(eq,eq+2) = -1;
        b(eq) = (-2*h*P)/(L*delta*K);
    end

end

if move == ratio

    %Left boundary no power
    for j = 1:ratio %changed ratio to ratio + 1 (works the same)
        i = 1; eq = i+(j-1)*m;
        A(eq,eq) = -3+((2*h*H)/K);
        A(eq,eq+1) = 4;
        A(eq,eq+2) = -1;
    end


    %Left boundary POWER
    for j = ratio+1:n  %LATER USE n = L
        i = 1; eq = i+(j-1)*m;
        A(eq,eq) = -3;
        A(eq,eq+1) = 4;
        A(eq,eq+2) = -1;
        b(eq) = (-2*h*P)/(L*delta*K);
    end


end

```

```
125
126
127
128
129
130
131
132 %RIGHT BOUNDARY until y
133 for j = 1:n-y
134    i = m; eq = i+(j-1)*m;
135    A(eq,eq) = -3+((2*h*H)/K);
136    A(eq,eq-1) = 4;
137    A(eq,eq-2) = -1;
138    b(eq) = 0;
139 end
140
141 %RIGHT BOUNDARY for y
142 for j = n-y:n
143    i = m-x; eq = i+(j-1)*m;
144    A(eq,eq) = -3+((2*h*H)/K);
145    A(eq,eq-1) = 4;
146    A(eq,eq-2) = -1;
147    b(eq) = 0;
148 end
149
150
151
152
153 %BOTTOM BOUNDARY
154 for i = 2:m-1
155    j = 1; eq = i+(j-1)*m;
156    A(eq,eq) = -3+((2*k*H)/K);
157    A(eq,eq+m) = 4;
158    A(eq,eq+2*m) = -1;
159    b(eq) = 0;
160 end
161
162 %TOP BOUNDARY until x
163 for i = 2:m-1-x
164    j = n; eq = i+(j-1)*m;
165    A(eq,eq) = -3+((2*k*H)/K);
166    A(eq,eq-m) = 4;
167    A(eq,eq-2*m) = -1;
168    b(eq) = 0;
169 end
170
171 %TOP BOUNDARY for x
172 for i = m+1-x:m-1
173    j = n-y; eq = i+(j-1)*m;
174    A(eq,eq) = -3+((2*k*H)/K);
175    A(eq,eq-m) = 4;
176    A(eq,eq-2*m) = -1;
177    b(eq) = 0;
178 end
```

N Main code Problem 9

```
%% Problem 9
close all; clear all; clc;
P = 5; %W
H = 0.005; %W/cm^2 * C
K = 1.68; %W/cm^2 * C
m = 50;
n = 20; %values from Problem 4
Lx = 4;
Ly = 4;
L = 2;
x = 10;
y = 10;
Extrernal_temp = 20;

counter = 1;
for i = 0:n/2
    [A,b] = set_up_matrix9(m,n,P,H,K,L,Lx,Ly,i, x, y);
    v = (A\b);
    for index = 1:length(v)
        if v(index) < 0.00001
            v(index) = NaN;
        else
            v(index) = v(index)+Extrernal_temp;
        end
    end

    v_matrix = reshape(v(1:m*n),m,n); %here we reshape from vector to matrix so it maches the X,Y
    ↪ boundaries
    v_matrix = v_matrix';
    v_highest(counter,1) = max(v);
    v_highest(counter,2) = i;

    figure
    [X,Y] = meshgrid(1:m,1:n);
    mesh(X,Y,v_matrix)
    colorbar
    %view(2)
    counter = counter + 1;
end

figure
plot(0:10,v_highest(:,1),'r-o')
ylabel('Temperature [C ]')
xlabel('n steps')
legend('Higest temperature of each iteration','Location','Best')
```

O Main code Independent work

```
%% Independent problem PART 1
close all; clear all; clc;
T = [69.4034,76.7259,80.6979, 104.162]; %Temperatures from the examples
m = 50;
n = 20;

Atotal = m*n; %total surface area
A = [0,5*2/Atotal*100,10*10/Atotal*100,30*10/Atotal*100];

plot(A,T, 'linewidth', 1.5)

xlabel('Notchsize in comparision with the total surface area [%]')
ylabel('Temperature [ C ]')









%% Independent problem PART 2
close all; clear all; clc;

%H = 0.005;
K = 1.68;
m = 50;
n = 20; %values from Problem 4
Lx = 4;
Ly = 4;
L = 2;
move = 5; %posision where the heat starts to enter
a = 1; %minimum power
b = 70; %maximum power
tol = 10^-2;

H = linspace(0.005,0.1,25);

for j = 1:length(H)
    max_power_allowed = bisect3(a,b,tol,m,n,H(j),K,L,Lx,Ly,move);
    T = calc_max_temp(m,n,max_power_allowed,H(j),K,L,Lx,Ly,move)+100;
    P_vec(j) = max_power_allowed;
    H_vec(j) = H(j);
end


plot(H_vec,P_vec,'LineWidth',1.5)
axis([0.005 0.1 0 35])
ylabel('Maximum Power allowed [W]')
xlabel('H [W/cm^{2} C ]')

%xlabel('Convective heat transfer (H [W/cm^{2}* C ])')

%%  Independent problem PART 3 (WHICH WE DID NOT WRITE ABOUT IN THE REPORT)
close all; clear all; clc;
P = 5;
H = 0.005; %W/cm^2 * C
K = 1.68; %W/cm^2 * C
m = 50;
```

```
58 n = 20; %values from Problem 4
59 Lx = 4;
60 Ly = 4;
61 L = 2;
62 move = 0; %posision where the heat starts to enter
63
64 [A,b] = set_up_the_matrix_independent(m,n,P,H,K,L,Lx,Ly,move);
65
66 v = (A\b)+20;
67 v_matrix = reshape(v(1:m*n),m,n); %here we reshape from vector to matrix so it maches the X,Y
       ↪ boundaries
68 v_matrix = v_matrix';
69 [X,Y] = meshgrid(1:m,1:n);
70
71
72 mesh(X,Y,v_matrix)
73 colorbar
74 %view(2)
75 highest_temp = v_matrix(1,1);
76 fprintf('\nThe highest temperature is: %.4f',highest_temp)
```