

ME-450 Computer Project Report

2D Numerical Heat Transfer Analysis

March 21, 2021

Sean McGee

Instructor: Dr. Carson Pete



College of Engineering, Informatics, and Applied Sciences

1 DISCLAIMER

This report was prepared by students as part of a university course requirement. While considerable effort has been put into the project, it is not the work of licensed engineers and has not undergone the extensive verification that is common in the profession. The information, data, conclusions, and content of this report should not be relied on or utilized without thorough, independent testing and verification. University faculty members may have been associated with this project as advisors, sponsors, or course instructors, but as such they are not responsible for the accuracy of results or conclusions.

2 TABLE OF CONTENTS

1 DISCLAIMER.....	1
2 TABLE OF CONTENTS.....	2
3 INTRODUCTION	3
3.1 Project Description	3
3.2 Methodology	3
3.3 Nodal Equations	3
4 DISCUSSION OF CODE.....	4
4.1 Pseudocode.....	4
4.2 Full Code.....	4
5 CODE EXECUTION RESULTS.....	4
5.1 Heat Rate Along Edges.....	4
5.2 Surface Plot	5
5.3 Edge Temperature Profiles	5
6 CODE VALIDATION	5
6.1 Plot Comparison.....	5
6.2 Net Flow through Outer Boundaries	5
6.3 Net Flow through Individual Element Boundaries.....	5
6.4 Effect of Mesh Resolution on Results	6
6.5 Validation Conclusions.....	6
7 CONCLUSIONS.....	6
8 APPENDICES.....	7
8.1 Appendix I: References	7
8.2 Appendix II: Figures.....	8
8.3 Appendix III: Tables	14
8.4 Appendix IV: Full Code, Main Script.....	18
8.5 Appendix V: Full Code, Surface Plot	33
8.6 Full Code: Edge Profile Plots	35

3 INTRODUCTION

3.1 Project Description

The objective of this project is to calculate the temperature profile of a given geometry with prescribed boundary conditions [1]. The geometry and boundary conditions are shown in Figure 1. The following assumptions are made of the problem setup:

1. The system is at steady state
2. Heat transfer occurs in only x- and y-directions
3. Heat transfer occurs through pure conduction except along the edges with convective boundary conditions (i.e., heat transfer through radiation is negligible)
4. Elements with convective boundary conditions experience negligible changes in heat flow due to the conduction of this heat through the length of material between the element boundary and the node

3.2 Methodology

Problems involving one-dimensional conductive heat transfer can typically be solved simply enough by analytical solutions. Analytical solutions provide the exact, mathematical solution to a problem. This allows the temperature at any point within the geometry to be calculated precisely. However, moving from one to two spatial dimensions complicates this process significantly. Analytical solutions are often far more difficult to calculate, but in cases with more complex geometry, analytical solutions may not even exist. In these cases, solution methods move instead to approximate solutions. By dividing a complex shape into many smaller, simpler shapes (called *elements*), the temperature within each of these elements (assumed constant throughout the element) can be found more easily. The combination of solutions for all of these elements can provide a meaningful approximation of heat transfer through the original shape. The number of elements used can be increased or decreased to adjust the resulting speed of computation and accuracy of results.

The finite-difference method outlined in ME-450 lectures up to this point expresses *each* node temperature algebraically as a linear combination of *all* node temperatures:

$$a_{1,n}T_1 + a_{2,n}T_2 + \cdots + a_{N,n}T_N = b_n$$

The system of these equations can then be expressed in matrix form:

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,N} \\ \vdots & \ddots & \vdots \\ a_{N,1} & \cdots & a_{N,N} \end{bmatrix} \cdot \begin{bmatrix} T_1 \\ \vdots \\ T_N \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}$$

The square matrix on the left-hand side of this equation is known as the coefficient matrix and is commonly referred to as the “A” matrix (including in the attached code). The vector of constant terms on the right-hand side of the equation is commonly referred to as the “b” matrix. Solving this system produces the resulting values of T_1 - T_N , or the temperature of every node in the geometry.

3.3 Nodal Equations

The temperature of each node will depend on the temperatures of the surrounding nodes and/or any boundary conditions on the node. The precise form of this equation for some given node will then depend on where the node is located. Any node along the top edge of the geometry, for instance, will have the constant-temperature boundary condition imposed by T_2 .

From the various possible combinations of boundary conditions arise 19 different node types. Each is shown in Table 1 accompanied by a brief description, what boundary conditions (if any) are imposed on the nodes, and the resulting equation for determining node temperature. Diagrams illustrating the positions of these node variations are shown in Figure 2, Figure 3, and Figure 4.

4 DISCUSSION OF CODE

4.1 Pseudocode

Code used to solve this problem involves many inputs, processes, and outputs. As such, the structure of the code is complex and can become confusing to write. It is helpful to begin by identifying the major tasks which must be accomplished in the form of *pseudocode*. This pseudocode should describe the general problem-solving strategy and major steps required to calculate the desired results. The pseudocode generated for this problem is as follows:

```
| Initialize input values (geometry, h, k, boundary condition values, etc.)
| Get number of rows, columns to subdivide geometry into finite elements
| Initialize node type matrix
| For each node:
|   | Depending on position, assign corresponding node type label
| Initialize coefficient matrix A and constant vector b
| For each node in node type matrix:
|   | Depending on label, populate A and b with corresponding values per equation
| Solve  $A \setminus b$  to find vector of temperatures for each node
| Place temperature vector results into corresponding location of new matrix with original geometry
| Plot resulting temperature surface
| Plot temperature profile along each edge
| For each node:
|   | Calculate net heat flow out of node
|   | Sum/average resulting values
| For each node along outer geometry boundary:
|   | Calculate net heat flow through boundary
|   | Sum/average resulting values
```

This is a brief, high-level summary of tasks to perform in code. Each of these items requires many different commands and steps for computation, which could be outlined in more detailed pseudocode. However, many of these tasks can be accomplished in multiple different ways, requiring the pseudocode to begin outlining specific MATLAB commands to use to perform tasks. At this state of preliminary, high-level planning, it is typically unknown which method will prove superior, so forming increasingly detailed pseudocode has diminishing returns. Instead, the process of writing actual code begins here. As the code evolves over this process, the results of an earlier step may be stored in a format which is inconvenient to work with in a later step. Rather than being “committed” to a particular solution process described by pseudocode, a more convenient or appropriate method may be implemented while still adhering to the general pseudocode outlined above.

4.2 Full Code

The full code produces for this project spans nearly 1,000 lines of code across three MATLAB scripts (one for calculation of results, one for generating a surface plot of the temperature profile, and one for plotting the temperature profiles of each edge), and as such does not fit appropriately in the body of this report. It is instead displayed in full in Appendix 8.4 . The script is also submitted alongside this document for verification.

5 CODE EXECUTION RESULTS

5.1 Heat Rate Along Edges

Table 2 shows results for the heat rate per unit depth along the outer edges of the geometry for 10x20, 20x40, and 40x80 elements. Note that, because the bottom boundary has an adiabatic boundary condition, heat rate across this boundary is always zero.

5.2 Surface Plot

Surface plots of the resulting node temperatures for 10x20, 20x40, and 40x80 elements can be seen in Figure 5, Figure 6, and Figure 7, respectively. Yellow lines around the perimeter of the geometry indicate the value of the constant-temperature boundary conditions at those locations. The boundaries of the generation region (and the center of this region) are extended vertically through the temperature surface to better illustrate its location on this surface.

5.3 Edge Temperature Profiles

Temperature profiles along each edge of the geometry are shown in Figure 8. Plots resulting from different numbers of elements are nearly indistinguishable, so only the 40x80 case is illustrated. Constant-temperature boundary conditions are shown by dotted lines overlaid on appropriate plots.

6 CODE VALIDATION

The MATLAB script file generates a matrix representing the physical layout of elements in the geometry. These elements were assigned their corresponding node types and equations which were in turn used to calculate their respective temperatures. The results of this code must be verified to be accurate before they can be used to form meaningful conclusions about the problem at hand. This verification process will use multiple strategies to identify inaccuracy: visual inspection of the temperature profile, calculation of net heat flow through the outer boundaries of the geometry, calculation of net heat flow through the boundaries of each element, and observation of changes in calculated results arising from changes to element size.

6.1 Plot Comparison

The code results are first verified by inspection. Firstly, the temperature of a given node should be close to the temperatures of its surrounding nodes. As a result, a plot of temperature profile should vary gradually and continuously over small distances. Observing Figure 7 shows no irregularities, discontinuities, nor abrupt changes in temperature profile. Additionally, comparing Figure 7 to the provided validation plot (Figure 9) shows no obvious differences [2].

6.2 Net Flow through Outer Boundaries

The results can be further validated by summing the heat flow through all external boundaries. Because the system has been assumed to be at steady state, the amount of heat flowing out of the boundaries should be equal in magnitude (though opposite in sign) to the amount of heat generated within the boundaries. This generated heat q_g is given by the rate of generation \dot{q} and the area of the generation region A_g :

$$q_g = \dot{q} \cdot A_g = 50,000 \frac{\text{W}}{\text{m}^2\text{K}} \cdot 0.32 \text{ m}^2 = 16 \text{ kW}$$

As such, summing the results from Table 2 should differ negligibly from this value. Indeed, as shown in Table 3, the sums of heat flow through all boundaries are almost exactly equal to 16 kW for all numbers of elements. However, the error for 20x40 elements is quite close to the maximum stated in the project description, and the error for 40x80 elements narrowly exceeds this maximum.

6.3 Net Flow through Individual Element Boundaries

Similarly, the heat flow across the edges of each individual element should be equal to the amount of heat generated within that element. Elements with node types 9-12 generate heat through half of their area, node type 13 generate heat through their entire area, and all other node types generate no heat. Heat flow (less the heat generated within the element) through boundaries for all nodes is calculated, and the resulting average and maximum error from zero is shown in Table 4.

A histogram of element errors in Figure 10 reveals that errors appear to be distributed normally, showing no skew nor peaks except at the median value of 0. A plot of these errors over the face of the geometry in Figure 11 reveals only a faint suggestion that elements near the generation region seem to have errors of larger magnitude with greater frequency. If this indicated an error within generation region nodal equations, it should be expected that this trend be more dramatic or consistent. Instead, it is assumed that these errors arise from the floating-point arithmetic used throughout the solution process due to the normal distribution of error, and the inconsistent locations of large errors. Because of the combination of large and small values in the generation region nodal equations, these small errors may be exaggerated in this region producing the aforementioned results.

6.4 Effect of Mesh Resolution on Results

Results in Table 3 show the effect of quadrupling the total number of elements used in solution computation on the *error* of net heat flow through geometry boundaries. From 10x20 to 20x40 elements, the error increases by about 1.5%, while the doubling the 20x40 elements to 40x80 increases error by about 42%. However, it is important to note that these are changes in the *magnitude of the errors*, not in the magnitude of the net heat flow through the boundaries. As the errors shown in Table 3 are errors from the expected heat flow of 16 kW, the percent changes in calculated heat flows are expected to be truly miniscule. Indeed, Table 5 shows that the changes in calculated heat flow are on the order of $(10^{-12})\%$.

6.5 Validation Conclusions

Validation results illustrate that the calculated solution generally agrees with anticipated results. The temperature profile appears as expected and seems to differ negligibly from the temperature profile provided for validation. Net heat flows across individual element boundaries and across outer geometry boundaries differ from the heat generated within these regions by at most 10^{-9} W. These errors appear to be normally distributed and are perhaps attributable to floating-point arithmetic errors. This can be reduced (or simply verified) by increasing the working precision within MATLAB at the expense of demanding greater computational time and power.

7 CONCLUSIONS

The results from the MATLAB code appear to be accurate and representative of the expected results for the problem. Increasing the number of elements generally increases the accuracy of calculated temperature values at some given point but appear to also increase the error in net heat flow through the boundaries of the entire geometry. As such, this method requires striking a balance between these two factors. Future work may include identification of the optimal such node density which minimizes both heat flow error and temperature error.

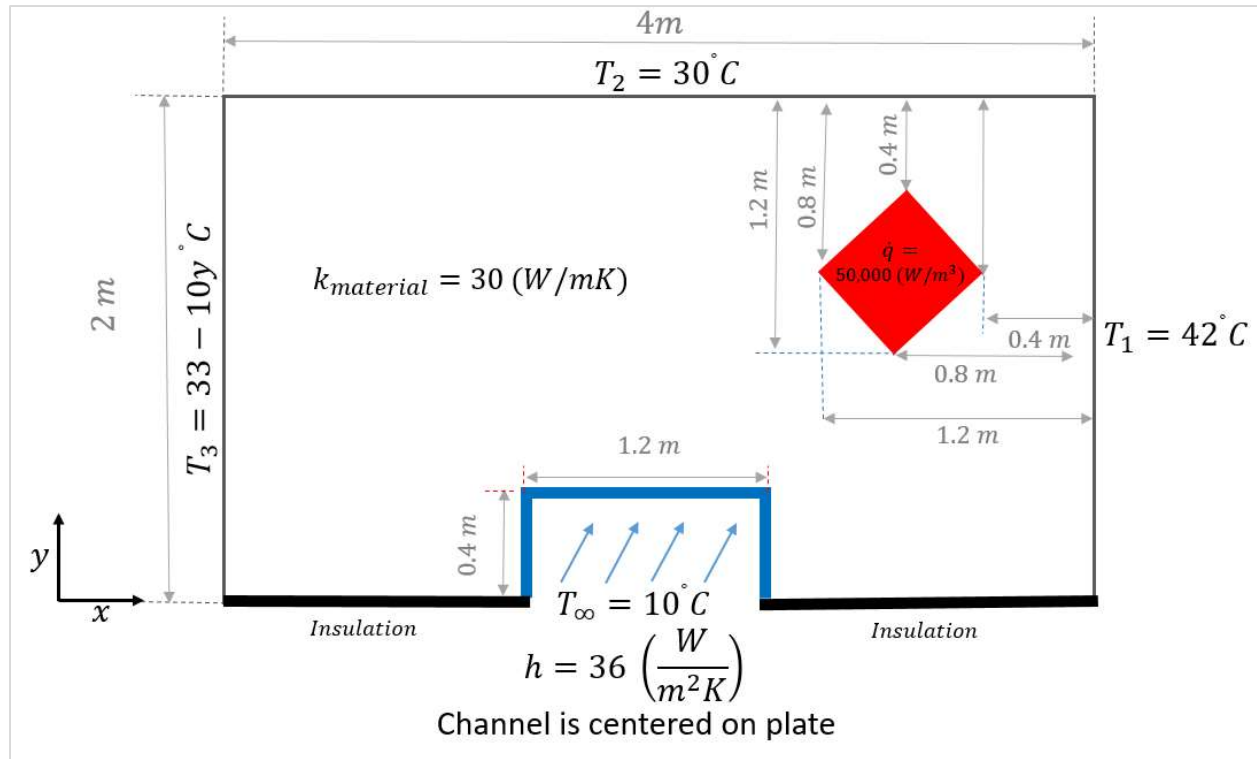
Furthermore, an aspect which was neglected throughout this analysis was the heat transfer by conduction between a convective boundary and the adjacent node. Increasing node density decreases this distance at the cost of increasing the number of nodes which experience this inaccuracy. It is hypothesized that accounting for this change may reduce the errors associated with increasing mesh density.

8 APPENDICES

8.1 *Appendix I: References*

- [1] Northern Arizona University, *ME450 - Heat Transfer, Spring 2021 Computer Project*, Flagstaff, AZ, 2021.
- [2] Northern Arizona University, *ME450 - Spring 2021 Computer Project Validation Plots*, Flagstaff, AZ, 2021.

8.2 Appendix II: Figures



← T2 →						
1	2	2	...	2	2	3
8	0	0	...	0	0	4
8	0	0	...	0	0	4
...
8	0	0	...	0	0	4
8	0	0	...	0	0	4
7	6	6	...	6	6	5
← Adiabatic →						

Figure 2: Edge of geometry node types

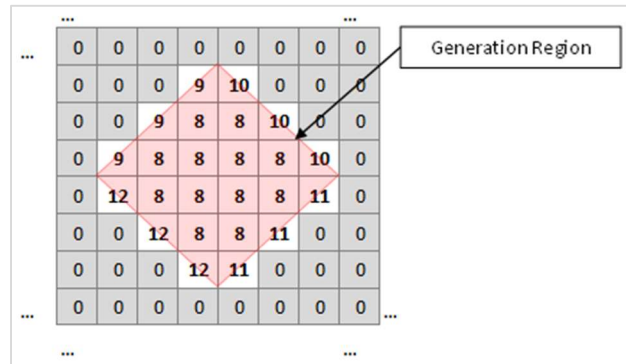


Figure 3: Generation region node types

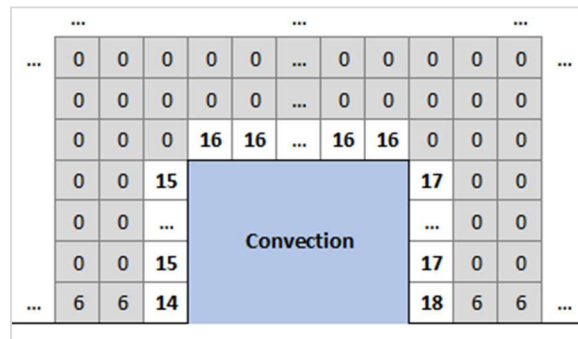


Figure 4: Convection region node types

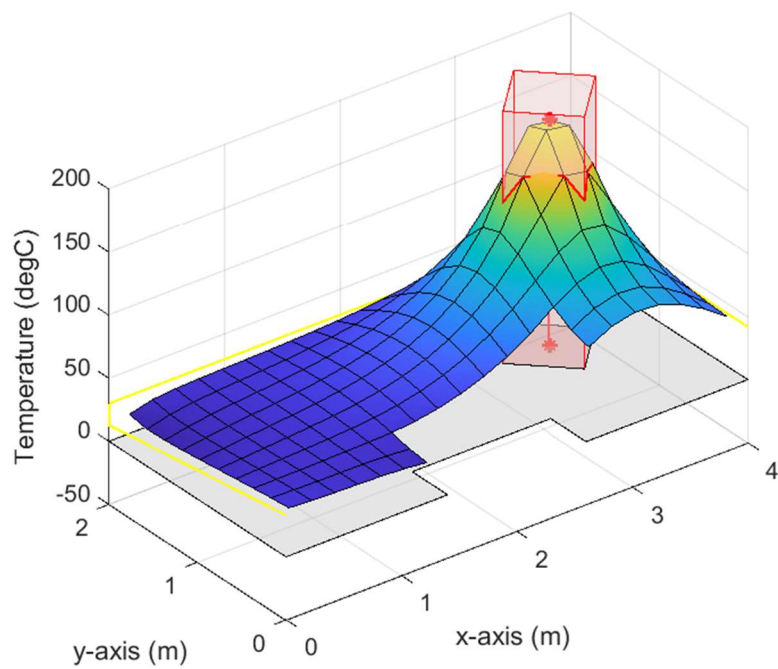


Figure 5: Temperature surface plot for 10x20 elements

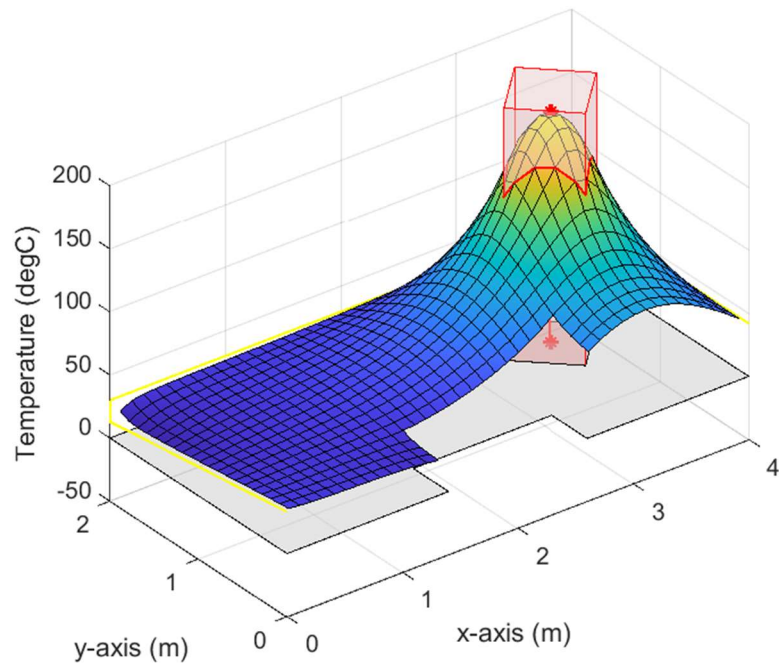


Figure 6: Temperature surface plot for 20x40 elements

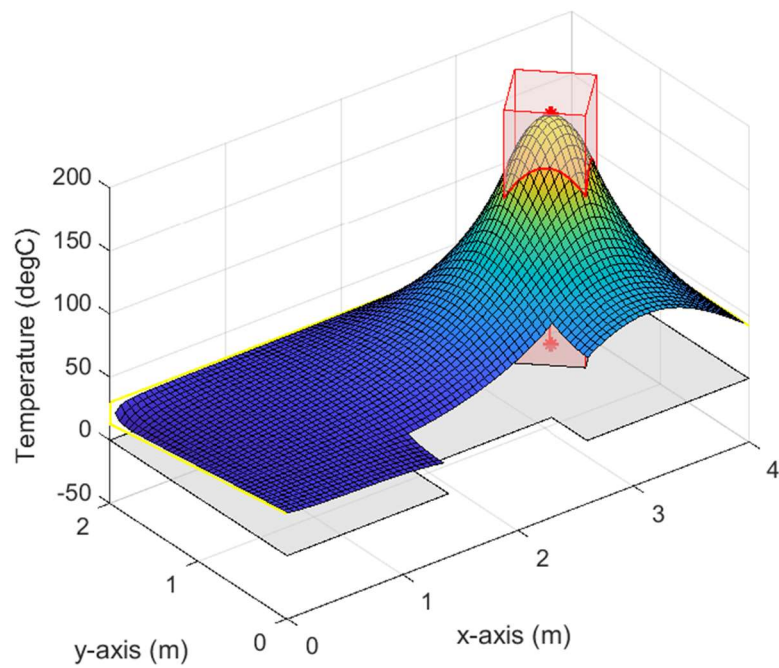


Figure 7: Temperature surface plot for 40x80 elements

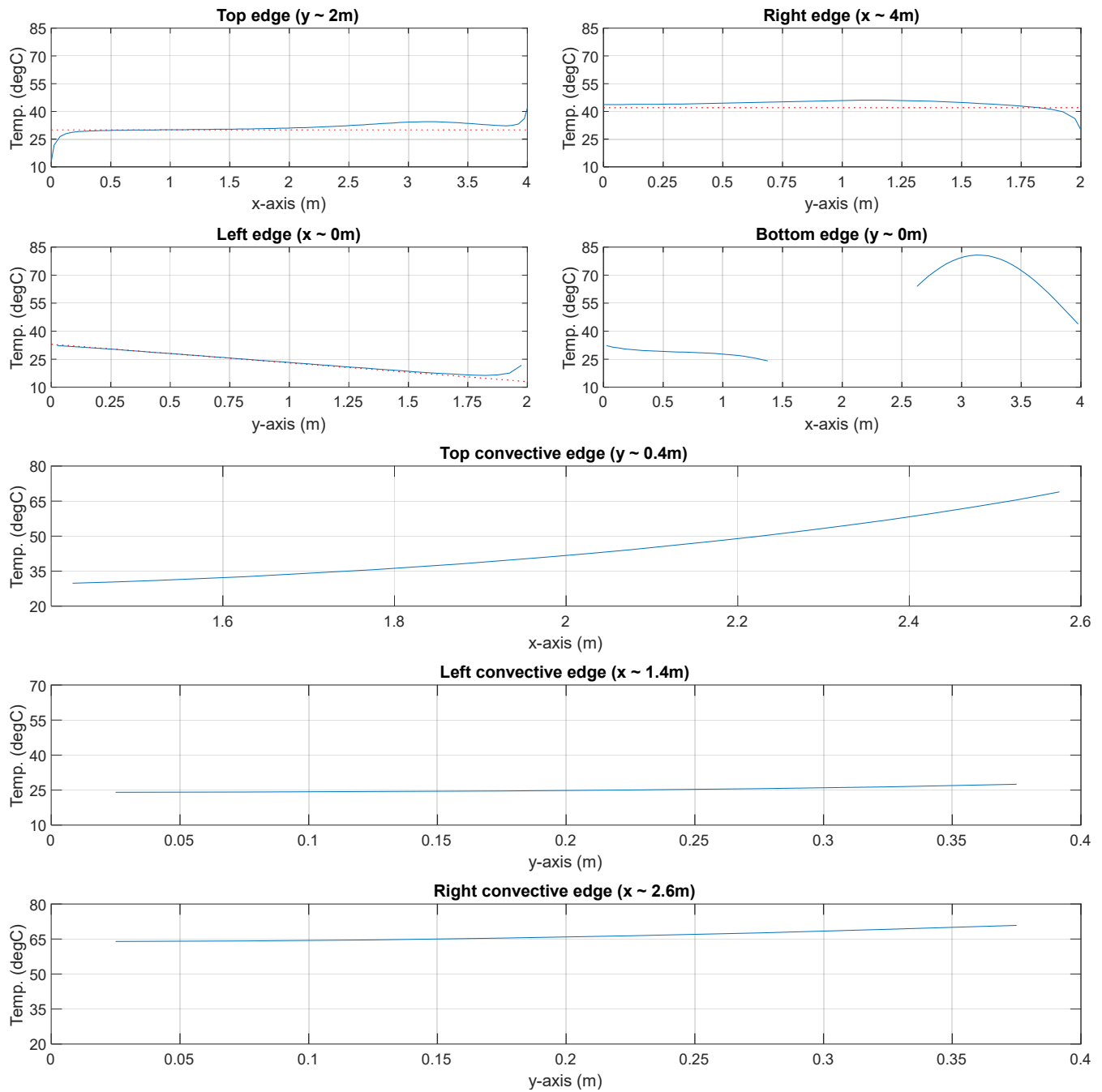


Figure 8: Temperature profile along outer boundaries, 40x80 elements

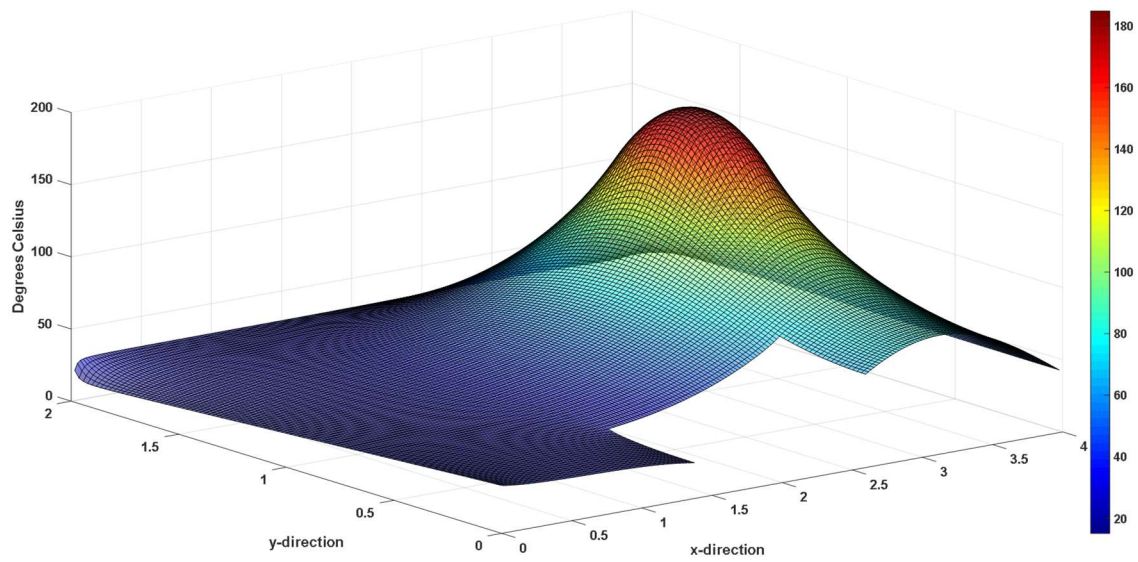


Figure 9: Temperature surface plot provided for validation [2]

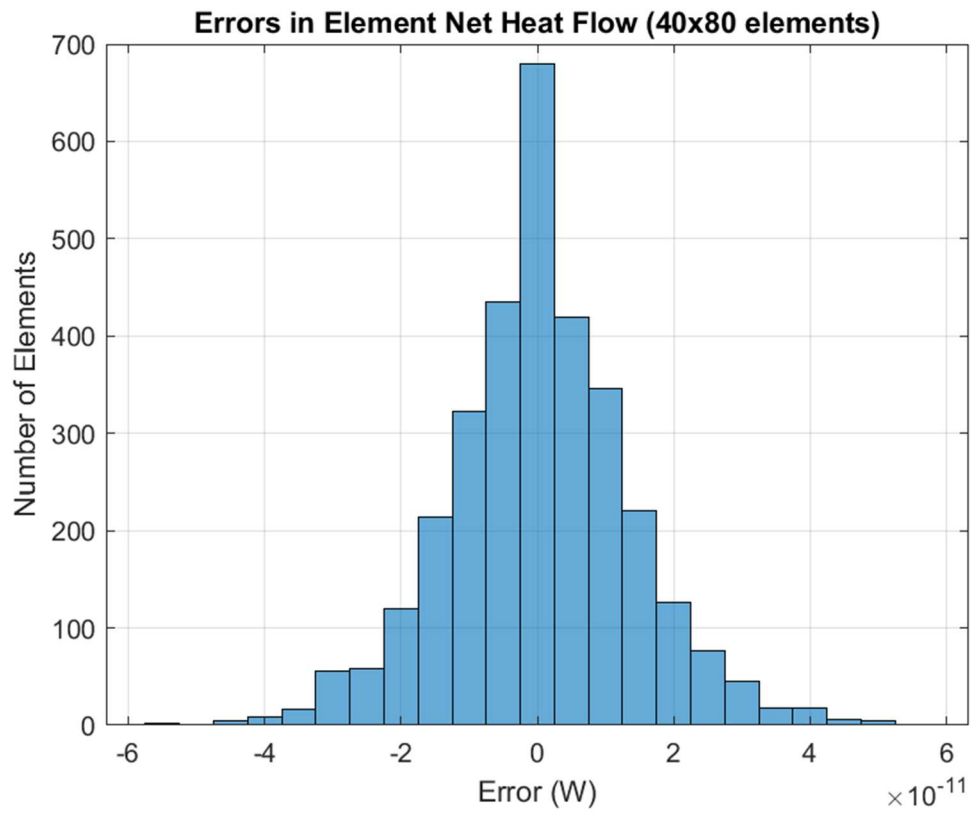


Figure 10: Distribution of nodal boundary errors, 40x80 elements

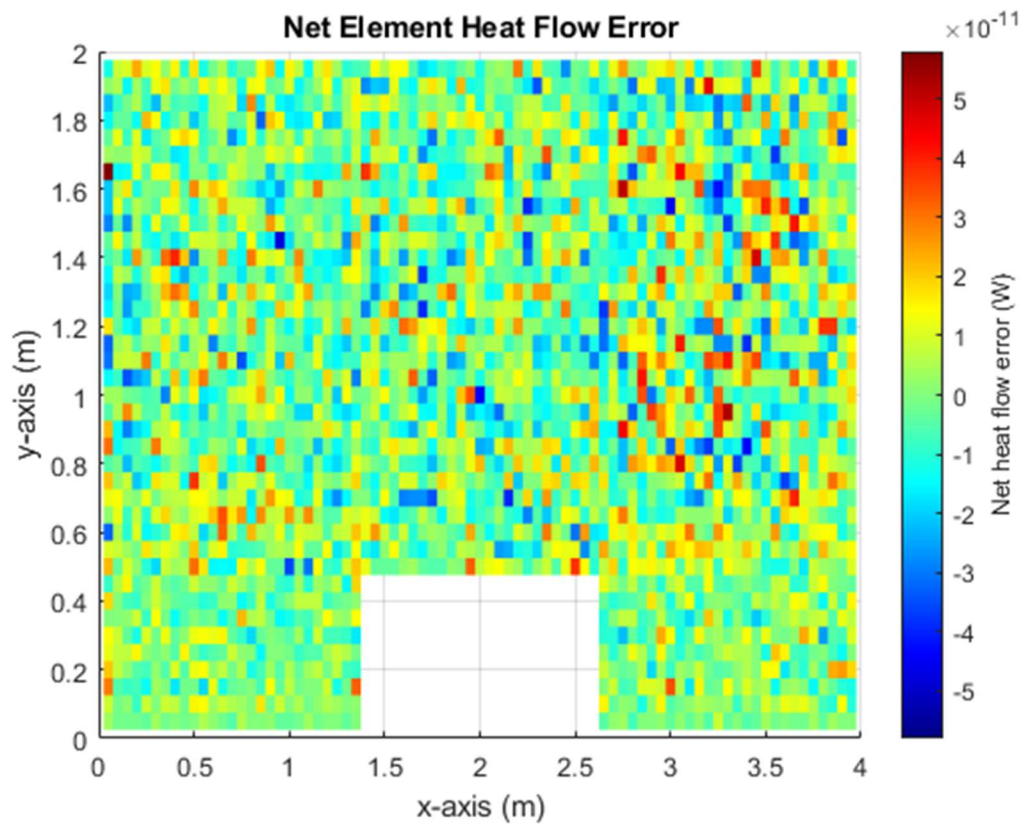


Figure 11: Error in net heat flow through element boundaries

8.3 Appendix III: Tables

Table 1: Node types and equations

Node type 0	Description:	Not along any edges, internal to geometry
	Top BC:	N/A
	Bottom BC:	N/A
	Left BC:	N/A
	Right BC:	N/A
	Equation:	$T_N + T_W + T_E + T_S - 4T = 0$
Node type 1	Description:	Top-left corner of geometry
	Top BC:	T_2
	Bottom BC:	N/A
	Left BC:	T_3
	Right BC:	N/A
	Equation:	$T_E + T_S - 6T = -2T_2 - 2T_3$
Node type 2	Description:	Top edge of geometry
	Top BC:	T_2
	Bottom BC:	N/A
	Left BC:	N/A
	Right BC:	N/A
	Equation:	$T_W + T_E + T_S - 5T = -2T_2$
Node type 3	Description:	Top-right corner of geometry
	Top BC:	T_2
	Bottom BC:	N/A
	Left BC:	N/A
	Right BC:	T_1
	Equation:	$T_W + T_S - 6T = -2T_1 - 2T_2$
Node type 4	Description:	Right edge of geometry
	Top BC:	N/A
	Bottom BC:	N/A
	Left BC:	N/A
	Right BC:	T_1
	Equation:	$T_N + T_W + T_S - 5T = -2T_1$
Node type 5	Description:	Bottom-right corner of geometry
	Top BC:	N/A
	Bottom BC:	Adiabatic
	Left BC:	N/A
	Right BC:	T_1
	Equation:	$T_N + T_W - 4T = -2T_1$
Node type 6	Description:	Bottom edge of geometry
	Top BC:	N/A
	Bottom BC:	Adiabatic
	Left BC:	N/A
	Right BC:	N/A
	Equation:	$T_N + T_W + T_E - 3T = 0$

Node type 7	Description:	Bottom-left corner of geometry
	Top BC:	N/A
	Bottom BC:	Adiabatic
	Left BC:	T_3
	Right BC:	N/A
	Equation:	$T_N + T_E - 4T = -2T_3$
Node type 8	Description:	Left edge of geometry
	Top BC:	N/A
	Bottom BC:	N/A
	Left BC:	T_3
	Right BC:	N/A
	Equation:	$T_N + T_E + T_S - 5T = -2T_3$
Node type 9	Description:	Top-left edge of generation block
	Top BC:	N/A
	Bottom BC:	N/A
	Left BC:	N/A
	Right BC:	N/A
	Equation:	$T_N + T_W + T_E + T_S - 4T = -g \frac{(\Delta x)^2}{2k}$
Node type 10	Description:	Top-right edge of generation block
	Top BC:	N/A
	Bottom BC:	N/A
	Left BC:	N/A
	Right BC:	N/A
	Equation:	$T_N + T_W + T_E + T_S - 4T = -g \frac{(\Delta x)^2}{2k}$
Node type 11	Description:	Bottom-right edge of generation block
	Top BC:	N/A
	Bottom BC:	N/A
	Left BC:	N/A
	Right BC:	N/A
	Equation:	$T_N + T_W + T_E + T_S - 4T = -g \frac{(\Delta x)^2}{2k}$
Node type 12	Description:	Bottom-left edge of generation block
	Top BC:	N/A
	Bottom BC:	N/A
	Left BC:	N/A
	Right BC:	N/A
	Equation:	$T_N + T_W + T_E + T_S - 4T = -g \frac{(\Delta x)^2}{2k}$
Node type 13	Description:	Entirely within generation block
	Top BC:	N/A
	Bottom BC:	N/A
	Left BC:	N/A
	Right BC:	N/A
	Equation:	$T_N + T_W + T_E + T_S - 4T = -g \frac{(\Delta x)^2}{k}$

Node type 14	Description:	Bottom-left corner of convection boundary
	Top BC:	N/A
	Bottom BC:	N/A
	Left BC:	N/A
	Right BC:	N/A
	Equation:	$T_N + T_W - \left(2 + \frac{h \Delta x}{k}\right) T = -T_\infty \left(\frac{h \Delta x}{k}\right)$
Node type 15	Description:	Left edge of convection boundary
	Top BC:	N/A
	Bottom BC:	N/A
	Left BC:	N/A
	Right BC:	N/A
	Equation:	$T_N + T_W + T_S - \left(3 + \frac{h \Delta x}{k}\right) T = -T_\infty \left(\frac{h \Delta x}{k}\right)$
Node type 16	Description:	Top edge of convection boundary
	Top BC:	N/A
	Bottom BC:	N/A
	Left BC:	N/A
	Right BC:	N/A
	Equation:	$T_N + T_W + T_E - \left(3 + \frac{h \Delta x}{k}\right) T = -T_\infty \left(\frac{h \Delta x}{k}\right)$
Node type 17	Description:	Right edge of convection boundary
	Top BC:	N/A
	Bottom BC:	N/A
	Left BC:	N/A
	Right BC:	N/A
	Equation:	$T_N + T_E + T_S - \left(3 + \frac{h \Delta x}{k}\right) T = -T_\infty \left(\frac{h \Delta x}{k}\right)$
Node type 18	Description:	Bottom-right corner of convection boundary
	Top BC:	N/A
	Bottom BC:	N/A
	Left BC:	N/A
	Right BC:	N/A
	Equation:	$T_N + T_E - \left(2 + \frac{h \Delta x}{k}\right) T = -T_\infty \left(\frac{h \Delta x}{k}\right)$

Table 2: Heat rate through edges for various numbers of elements

	10x20 elements (10 ³ W/m)	20x40 elements (10 ³ W/m)	40x80 elements (10 ³ W/m)
Top edge:	-6.7459	-6.7118	-6.6626
Left edge:	-0.9945	-1.2278	-1.4582
Right edge:	-5.6432	-5.5073	-5.3604
Left convection edge:	-0.2209	-0.2201	-0.2193
Top convection edge:	-1.5431	-1.5060	-1.4859
Right convection edge:	-0.8524	-0.8271	-0.8136

Table 3: Summed heat rates through edges for various numbers of elements

	10x20 elements (10 ³ W/m)	20x40 elements (10 ³ W/m)	40x80 elements (10 ³ W/m)
Top edge:	-6.7459	-6.7118	-6.6626
Left edge:	-0.9945	-1.2278	-1.4582
Right edge:	-5.6432	-5.5073	-5.3604
Left convection edge:	-0.2209	-0.2201	-0.2193
Top convection edge:	-1.5431	-1.5060	-1.4859
Right convection edge:	-0.8524	-0.8271	-0.8136
Sum – 16 kW:	-7.276E-12 W	-4.91E-10 W	1.16E-9 W

Table 4: Average and maximum errors of individual node net heat flow

	10x20 elements (W)	20x40 elements (W)	40x80 elements (W)
Average error	-3.69E-14	6.15E-13	3.63E-13
Maximum error	3.41E-11	5.12E-11	5.80E-11

Table 5: Differences in calculated heat flow through boundaries by number of elements

	10x20 elements (kW)	20x40 elements (kW)	40x80 elements (kW)
Calc. Net Heat Flow through Boundaries	-16.0000000000000	-15.9999999999995	-15.9999999999988
Difference from Previous	N/A	(3.13%)E-12	(4.377%)E-12

8.4 Appendix IV: Full Code, Main Script

[Begin “smm573_ME450_ComputerProject_MainCode.m” code]

```
1 %% Physical size
2 widthx = 4; % Horizontal length in meters
3 heighty = 2; % Vertical length in meters
4
5 %% Size of matrix
6 numRows = 20; % Number of rows in matrix (positive integer multiple of 10)
7
8 numColumns = round(widthx/heighty) * numRows; % Number of columns in matrix (twice numRows)
9
10 if ((mod(numRows,5) ~=0) || (mod(numColumns,10) ~= 0))
11     error('Value of numRows should be a multiple of 10');
12 end
13
14 %% Initialize matrix variables
15 dx = widthx / numColumns; % x-distance between matrix columns
16 dy = heighty / numRows; % y-distance between matrix rows
17
18 conv_xMin = 1.4; % Minimum x-value of convection block
19 conv_xMax = 2.6; % Maximum x-value of convection block
20 conv_yMin = 0.0; % Minimum y-value of convection block
21 conv_yMax = 0.4; % Maximum y-value of convection block
22
23 gen_xMin = 2.8; % Location of left-most gen block corner
24 gen_xMax = 3.6; % Location of right-most gen block corner
25 gen_yMin = 0.8; % Location of top-most gen block corner
26 gen_yMax = 1.6; % Location of bottom-most gen block corner
27
28
29 %% Find limits of cutout block in m,n
30 conv_mMin = conv_yMin / dy + 0.5;
31 conv_mMax = conv_yMax / dy + 0.5;
32 conv_nMin = conv_xMin / dx + 0.5;
33 conv_nMax = conv_xMax / dx + 0.5;
34
35
36 %% Find limits of generation block in m,n
37 gen_mMin = gen_yMin / dy + 0.5;
38 gen_mMax = gen_yMax / dy + 0.5;
39 gen_nMin = gen_xMin / dx + 0.5;
40 gen_nMax = gen_xMax / dx + 0.5;
41
42
43 %% Various BC values
44 T_1C = 42;
```

```

45 T_2C = 30;
46 T_CC = 10;
47 T_1K = T_1C + 273.15;
48 T_2K = T_2C + 273.15;
49 T_CK = T_CC + 273.15;
50 h = 36;
51 g = 50000;
52 k = 30;
53
54
55 %% Input corresponding values into matrix
56 NT_in = 0;
57 NT_tl = 1;
58 NT_t = 2;
59 NT_tr = 3;
60 NT_r = 4;
61 NT_br = 5;
62 NT_b = 6;
63 NT_bl = 7;
64 NT_l = 8;
65 NT_gtl = 9;
66 NT_gtr = 10;
67 NT_gbr = 11;
68 NT_gbl = 12;
69 NT_gin = 13;
70 NT_cbl = 14;
71 NT_cl = 15;
72 NT_ct = 16;
73 NT_cr = 17;
74 NT_cbr = 18;
75
76 % Initialize matrices
77 labelMatrix = zeros(numRows, numColumns);
78 tempMatrixK = zeros(numRows, numColumns);
79
80 %% Fill label matrix
81 % Four corner elements
82 labelMatrix(1, 1) = NT_bl;
83 labelMatrix(1, numColumns) = NT_br;
84 labelMatrix(numRows, 1) = NT_tl;
85 labelMatrix(numRows, numColumns) = NT_tr;
86
87 % Top, bottom edge
88 for n = 2:(numColumns - 1)
89     labelMatrix(1, n) = NT_b;
90     labelMatrix(numRows, n) = NT_t;
91 end
92
93 % Left edge
94 for m = 2:(numRows - 1)
95     labelMatrix(m, 1) = NT_l;
96 end
97
98 % Right edge

```

```

% Cutout BC: Convective, Tinf = 10 degC
% Convert BC temps to Kelvin

% Convective constant = 36 W/m^2K
% Heat generation is 50kW/m^2 = 50,000 W/m^2
% Thermal conductivity = 30 W/mK

% Type 0: Inner elements with no BCs or generation
% Type 1: Top-left corner
% Type 2: Top edge
% Type 3: Top-right corner
% Type 4: Right edge
% Type 5: Bottom-right corner
% Type 6: Bottom edge
% Type 7: Bottom-left corner
% Type 8: Left edge
% Type 9: Generation, top-left edge
% Type 10: Generation, top-right edge
% Type 11: Generation, bottom-right edge
% Type 12: Generation, bottom-left edge
% Type 13: Generation, internal
% Type 14: Convection, bottom-left corner
% Type 15: Convection, left edge
% Type 16: Convection, top edge
% Type 17: Convection, right edge
% Type 18: Convection, bottom-right edge

% Create matrix for node labels
% Create matrix for solving temperature profile

% Top-left
% Top-right
% Bottom-left
% Bottom-right

```

```

99   for m = 2:(numRows - 1)
100       labelMatrix(m, numColumns) = NT_r;
101   end
102
103   % Generation block
104   %   Boundaries of gen block are  $y = |x - 3.2| + 0.8$  ;  $y = -|x - 3.2| + 1.6$ 
105   gen_nCenter = mean([gen_nMin gen_nMax]);
106   gen_mCenter = mean([gen_mMin gen_mMax]);
107   for n = ceil(gen_nMin):floor(gen_nMax)
108       mGT = abs(n-gen_nCenter)+gen_mMin;
109       mLT = -abs(n-gen_nCenter)+gen_mMax;
110       for m = ceil(gen_mMin):floor(gen_mMax)
111           if (m == mGT)
112               if (n < gen_nCenter)
113                   labelMatrix(m,n) = NT_gbl;
114               else
115                   labelMatrix(m,n) = NT_gbr;
116               end
117           elseif (m == mLT)
118               if (n < gen_nCenter)
119                   labelMatrix(m,n) = NT_gtl;
120               else
121                   labelMatrix(m,n) = NT_gtr;
122               end
123           elseif ((m > mGT) && (m < mLT))
124               labelMatrix(m,n) = NT_gin;
125           end
126       end
127   end
128
129   % Cutout block
130   for m = ceil(conv_mMin):floor(conv_mMax)
131       for n = ceil(conv_nMin):floor(conv_nMax)
132           labelMatrix(m,n) = NaN;
133       end
134   end
135   labelMatrix(ceil(conv_mMax), ceil(conv_nMin):floor(conv_nMax)) = NT_ct;
136   labelMatrix(ceil(conv_mMin)+1:floor(conv_mMax), floor(conv_nMin)) = NT_cl;
137   labelMatrix(ceil(conv_mMin)+1:floor(conv_mMax), ceil(conv_nMax)) = NT_cr;
138   labelMatrix(ceil(conv_mMin), floor(conv_nMin)) = NT_cbl;
139   labelMatrix(ceil(conv_mMin), ceil(conv_nMax)) = NT_cbr;
140
141
142   %% Create coefficient matrix
143   coeffMatrix = zeros(numRows * numColumns);
144   b = coeffMatrix(:,1);
145   convConst = h * dx / k;
146   genConst = g * dx * dx / k;
147
148   for m = 1:numRows
149       T_3K = 273.15 + 33 - 10 * ((m-1) * dy + dy/2);
150       for n = 1:numColumns
151           T_pos = (m - 1) * numColumns + n;
152           switch labelMatrix(m,n)

```

```

% "In. gen block" RHS only depend on n, so calculate them
% only in "n" loop to save time

% If m is on the bottom boundary...
%   ...and n is before the center...
%   ...then this is the bottom-left corner
%   Otherwise...
%   ...this is the bottom-right corner

% If m is on the top boundary...
%   ...and n is before the center...
%   ...then this is the top-left corner
%   Otherwise...
%   ...this is the top-right corner

% Otherwise, if not on either boundary but IS in the range...
%   ...The element is fully within gen block

```

```

% Instantiate coefficient matrix
% Instantiate b vector
% Calculate convective term so it's not recalculated every loop
% Calculate generation term so it's not recalculated every loop

% Calculate T_3 for each new row

% Calculate row of coefficient matrix, b vector

```

```

153
154 case NT_in % 0
155     % Self:
156     coeffMatrix(T_pos, T_pos) = -4;
157     % Top:
158     coeffMatrix(T_pos, T_pos + numColumns) = 1;
159     % Bottom:
160     coeffMatrix(T_pos, T_pos - numColumns) = 1;
161     % Left:
162     coeffMatrix(T_pos, T_pos - 1) = 1;
163     % Right:
164     coeffMatrix(T_pos, T_pos + 1) = 1;
165     % Constant: NONE
166     b(T_pos) = 0;
167
168 case NT_tl % 1
169     % Self:
170     coeffMatrix(T_pos, T_pos) = -6;
171     % Top: T_2 BC
172     % Bottom:
173     coeffMatrix(T_pos, T_pos - numColumns) = 1;
174     % Left: T_3 BC
175     % Right:
176     coeffMatrix(T_pos, T_pos + 1) = 1;
177     % Constant:
178     b(T_pos,1) = -2*(T_2K + T_3K);
179
180 case NT_t % 2
181     % Self:
182     coeffMatrix(T_pos, T_pos) = -5;
183     % Top: T_2 BC
184     % Bottom:
185     coeffMatrix(T_pos, T_pos - numColumns) = 1;
186     % Left:
187     coeffMatrix(T_pos, T_pos - 1) = 1;
188     % Right:
189     coeffMatrix(T_pos, T_pos + 1) = 1;
190     % Constant:
191     b(T_pos,1) = -2*T_2K;
192
193 case NT_tr % 3
194     % Self:
195     coeffMatrix(T_pos, T_pos) = -6;
196     % Top: T_2 BC
197     % Bottom:
198     coeffMatrix(T_pos, T_pos - numColumns) = 1;
199     % Left:
200     coeffMatrix(T_pos, T_pos - 1) = 1;
201     % Right: T_1 BC
202     % Constant:
203     b(T_pos,1) = -2*(T_1K + T_2K);
204
205 case NT_r % 4
206     % Self:

```

```

207         coeffMatrix(T_pos, T_pos) = -5;
208         % Top:
209         coeffMatrix(T_pos, T_pos + numColumns) = 1;
210         % Bottom:
211         coeffMatrix(T_pos, T_pos - numColumns) = 1;
212         % Left:
213         coeffMatrix(T_pos, T_pos - 1) = 1;
214         % Right: T_1 BC
215         % Constant:
216         b(T_pos,1) = -2*T_1K;
217
218     case NT_br % 5
219         % Self:
220         coeffMatrix(T_pos, T_pos) = -4;
221         % Top:
222         coeffMatrix(T_pos, T_pos + numColumns) = 1;
223         % Bottom: INS. BC
224         % Left:
225         coeffMatrix(T_pos, T_pos - 1) = 1;
226         % Right: T_1 BC
227         % Constant:
228         b(T_pos,1) = -2*T_1K;
229
230     case NT_b % 6
231         % Self:
232         coeffMatrix(T_pos, T_pos) = -3;
233         % Top:
234         coeffMatrix(T_pos, T_pos + numColumns) = 1;
235         % Bottom: INS. BC
236         % Left:
237         coeffMatrix(T_pos, T_pos - 1) = 1;
238         % Right: T_1 BC
239         coeffMatrix(T_pos, T_pos + 1) = 1;
240         % Constant: NONE
241         b(T_pos,1) = 0;
242
243     case NT_bl % 7
244         % Self:
245         coeffMatrix(T_pos, T_pos) = -4;
246         % Top:
247         coeffMatrix(T_pos, T_pos + numColumns) = 1;
248         % Bottom: INS. BC
249         % Left: T_3 BC
250         % Right: T_1 BC
251         coeffMatrix(T_pos, T_pos + 1) = 1;
252         % Constant:
253         b(T_pos,1) = -2*T_3K;
254
255     case NT_l % 8
256         % Self:
257         coeffMatrix(T_pos, T_pos) = -5;
258         % Top:
259         coeffMatrix(T_pos, T_pos + numColumns) = 1;
260         % Bottom:

```

```

261         coeffMatrix(T_pos, T_pos - numColumns) = 1;
262         % Left: T_3 BC
263         % Right:
264         coeffMatrix(T_pos, T_pos + 1) = 1;
265         % Constant:
266         b(T_pos,1) = -2*T_3K;
267
268     case NT_gtl % 9
269         % Self:
270         coeffMatrix(T_pos, T_pos) = -4;
271         % Top:
272         coeffMatrix(T_pos, T_pos + numColumns) = 1;
273         % Bottom:
274         coeffMatrix(T_pos, T_pos - numColumns) = 1;
275         % Left:
276         coeffMatrix(T_pos, T_pos - 1) = 1;
277         % Right:
278         coeffMatrix(T_pos, T_pos + 1) = 1;
279         % Constant:
280         b(T_pos,1) = -genConst/2;
281
282     case NT_gtr % 10
283         % Self:
284         coeffMatrix(T_pos, T_pos) = -4;
285         % Top:
286         coeffMatrix(T_pos, T_pos + numColumns) = 1;
287         % Bottom:
288         coeffMatrix(T_pos, T_pos - numColumns) = 1;
289         % Left:
290         coeffMatrix(T_pos, T_pos - 1) = 1;
291         % Right:
292         coeffMatrix(T_pos, T_pos + 1) = 1;
293         % Constant:
294         b(T_pos,1) = -genConst/2;
295
296     case NT_gbr % 11
297         % Self:
298         coeffMatrix(T_pos, T_pos) = -4;
299         % Top:
300         coeffMatrix(T_pos, T_pos + numColumns) = 1;
301         % Bottom:
302         coeffMatrix(T_pos, T_pos - numColumns) = 1;
303         % Left:
304         coeffMatrix(T_pos, T_pos - 1) = 1;
305         % Right:
306         coeffMatrix(T_pos, T_pos + 1) = 1;
307         % Constant:
308         b(T_pos,1) = -genConst/2;
309
310     case NT_gbl % 12
311         % Self:
312         coeffMatrix(T_pos, T_pos) = -4;
313         % Top:
314         coeffMatrix(T_pos, T_pos + numColumns) = 1;

```



```

315         % Bottom:
316         coeffMatrix(T_pos, T_pos - numColumns) = 1;
317         % Left:
318         coeffMatrix(T_pos, T_pos - 1) = 1;
319         % Right:
320         coeffMatrix(T_pos, T_pos + 1) = 1;
321         % Constant:
322         b(T_pos,1) = -genConst/2;
323
324     case NT_gin % 13
325         % Self:
326         coeffMatrix(T_pos, T_pos) = -4;
327         % Top:
328         coeffMatrix(T_pos, T_pos + numColumns) = 1;
329         % Bottom:
330         coeffMatrix(T_pos, T_pos - numColumns) = 1;
331         % Left:
332         coeffMatrix(T_pos, T_pos - 1) = 1;
333         % Right:
334         coeffMatrix(T_pos, T_pos + 1) = 1;
335         % Constant:
336         b(T_pos,1) = -genConst;
337
338     case NT_cbl % 14
339         % Self:
340         coeffMatrix(T_pos, T_pos) = -(2+convConst);
341         % Top:
342         coeffMatrix(T_pos, T_pos + numColumns) = 1;
343         % Bottom: INS. BC
344         % Left:
345         coeffMatrix(T_pos, T_pos - 1) = 1;
346         % Right: CONV. BC
347         % Constant:
348         b(T_pos,1) = -convConst * T_CK;
349
350     case NT_cl % 15
351         % Self:
352         coeffMatrix(T_pos, T_pos) = -(3+convConst);
353         % Top:
354         coeffMatrix(T_pos, T_pos + numColumns) = 1;
355         % Bottom:
356         coeffMatrix(T_pos, T_pos - numColumns) = 1;
357         % Left:
358         coeffMatrix(T_pos, T_pos - 1) = 1;
359         % Right: CONV. BC
360         % Constant:
361         b(T_pos,1) = -convConst * T_CK;
362
363     case NT_ct % 16
364         % Self:
365         coeffMatrix(T_pos, T_pos) = -(3+convConst);
366         % Top:
367         coeffMatrix(T_pos, T_pos + numColumns) = 1;
368         % Bottom: CONV. BC

```

```

369         % Left:
370         coeffMatrix(T_pos, T_pos - 1) = 1;
371         % Right:
372         coeffMatrix(T_pos, T_pos + 1) = 1;
373         % Constant:
374         b(T_pos,1) = -convConst * T_CK;
375
376     case NT_cr % 17
377         % Self:
378         coeffMatrix(T_pos, T_pos) = -(3+convConst);
379         % Top:
380         coeffMatrix(T_pos, T_pos + numColumns) = 1;
381         % Bottom:
382         coeffMatrix(T_pos, T_pos - numColumns) = 1;
383         % Left: CONV. BC
384         % Right:
385         coeffMatrix(T_pos, T_pos + 1) = 1;
386         % Constant:
387         b(T_pos,1) = -convConst * T_CK;
388
389     case NT_cbr % 18
390         % Self:
391         coeffMatrix(T_pos, T_pos) = -(2+convConst);
392         % Top:
393         coeffMatrix(T_pos, T_pos + numColumns) = 1;
394         % Bottom: INS. BC
395         % Left: CONV. BC
396         % Right:
397         coeffMatrix(T_pos, T_pos + 1) = 1;
398         % Constant:
399         b(T_pos,1) = -convConst * T_CK;
400
401     otherwise
402         % Within convection block OR some error
403         if (isnan(labelMatrix(m,n)))
404             coeffMatrix(T_pos, T_pos) = 1;
405             b(T_pos,1) = 0;
406         else
407             msg1 = 'Label at labelMatrix(';
408             msg2 = ') not recognized, something is very wrong...';
409             error("%s%d,%d%s",msg1,m,n,msg2);
410         end
411     end
412 end
413 end
414
415 %% Solve system of equations
416
417 tempVector = coeffMatrix \ b;
418
419
420 %% Fill tempMatrixK with solved values
421
422 m = 1;

```

% NaN indicates convection block

```

423 n = 1;
424 for i = 1:length(tempVector)
425     tVi = tempVector(i);
426     if (tVi == 0)
427         tempMatrixK(m,n) = NaN;
428     else
429         tempMatrixK(m,n) = tempVector(i,1);
430     end
431     n = n + 1;
432     if (n > numColumns)
433         n = 1;
434         m = m + 1;
435     end
436 end
437
438
439 %% Convert temps from K to C
440 tempMatrixC = tempMatrixK - 273.15;
441
442
443 %% Plot temp profile
444 create3D = true;
445 createContour = false;
446 createTiles = true;
447 [x,y] = meshgrid((dx/2):dx:(widthx - dx/2), (dy/2):dy:(heighty - dy/2));
448
449 if create3D
450     smm573_create_3Dplot;
451 end
452
453 if createContour
454     smm573_create_Contourplot;
455 end
456
457 if createTiles
458     smm573_create_Tileplot;
459 end
460
461
462 %% Check errors
463 sumElemEdges = labelMatrix * 0;
464 q_N = 0;
465 q_S = 0;
466 q_E = 0;
467 q_W = 0;
468 sumOuterEdgesN = 0;
469 sumOuterEdgesE = 0;
470 sumOuterEdgesW = 0;
471 sumOuterEdgesS = 0;
472
473 for m = 1:numRows
474     T_3K = 273.15 + 33 - 10 * ((m-1) * dy + dy/2);
475     for n = 1:numColumns
476         %T_pos = (m - 1) * numColumns + n;

```

```

477 %indicesMatrix(m,n) = T_pos;
478 switch labelMatrix(m,n)
479     case NT_in % 0
480         % North:
481         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
482         % South:
483         q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
484         % East:
485         q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
486         % West:
487         q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
488         % Net heat flow:
489         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
490
491     case NT_tl % 1
492         % North:
493         q_N = k * 2*(T_2K - tempMatrixK(m, n));
494         % South:
495         q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
496         % East:
497         q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
498         % West:
499         q_W = k * 2*(T_3K - tempMatrixK(m, n));
500         % Net heat flow:
501         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
502         sumOuterEdgesN = sumOuterEdgesN + q_N;
503         sumOuterEdgesW = sumOuterEdgesW + q_W;
504
505     case NT_t % 2
506         % North:
507         q_N = k * 2*(T_2K - tempMatrixK(m, n));
508         % South:
509         q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
510         % East:
511         q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
512         % West:
513         q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
514         % Net heat flow:
515         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
516         sumOuterEdgesN = sumOuterEdgesN + q_N;
517
518     case NT_tr % 3
519         % North:
520         q_N = k * 2*(T_2K - tempMatrixK(m, n));
521         % South:
522         q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
523         % East:
524         q_E = k * 2*(T_1K - tempMatrixK(m, n));
525         % West:
526         q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
527         % Net heat flow:
528         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
529         sumOuterEdgesN = sumOuterEdgesN + q_N;
530

```

```

531         sumOuterEdgesE = sumOuterEdgesE + q_E;
532
533     case NT_r % 4
534         % North:
535         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
536         % South:
537         q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
538         % East:
539         q_E = k * 2*(T_1K - tempMatrixK(m, n));
540         % West:
541         q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
542         % Net heat flow:
543         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
544         sumOuterEdgesE = sumOuterEdgesE + q_E;
545
546     case NT_br % 5
547         % North:
548         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
549         % South:
550         q_S = 0;
551         % East:
552         q_E = k * 2*(T_1K - tempMatrixK(m, n));
553         % West:
554         q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
555         % Net heat flow:
556         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
557         sumOuterEdgesE = sumOuterEdgesE + q_E;
558         sumOuterEdgesS = sumOuterEdgesS + q_S;
559
560     case NT_b % 6
561         % North:
562         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
563         % South:
564         q_S = 0;
565         % East:
566         q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
567         % West:
568         q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
569         % Net heat flow:
570         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
571         sumOuterEdgesS = sumOuterEdgesS + q_S;
572
573     case NT_bl % 7
574         % North:
575         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
576         % South:
577         q_S = 0;
578         % East:
579         q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
580         % West:
581         q_W = k * 2*(T_3K - tempMatrixK(m, n));
582         % Net heat flow:
583         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
584         sumOuterEdgesW = sumOuterEdgesW + q_W;

```

```

585         sumOuterEdgesS = sumOuterEdgesS + q_S;
586
587     case NT_l % 8
588         % North:
589         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
590         % South:
591         q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
592         % East:
593         q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
594         % West:
595         q_W = k * 2*(T_3K - tempMatrixK(m, n));
596         % Net heat flow:
597         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
598         sumOuterEdgesW = sumOuterEdgesW + q_W;
599
600     case NT_gtl % 9
601         % North:
602         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
603         % South:
604         q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
605         % East:
606         q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
607         % West:
608         q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
609         % Net heat flow:
610         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W) + g * dx^2 / 2;
611
612     case NT_gtr % 10
613         % North:
614         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
615         % South:
616         q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
617         % East:
618         q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
619         % West:
620         q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
621         % Net heat flow:
622         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W) + g * dx^2 / 2;
623
624     case NT_gbr % 11
625         % North:
626         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
627         % South:
628         q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
629         % East:
630         q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
631         % West:
632         q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
633         % Net heat flow:
634         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W) + g * dx^2 / 2;
635
636     case NT_gbl % 12
637         % North:
638         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));

```

```

639         % South:
640         q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
641         % East:
642         q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
643         % West:
644         q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
645         % Net heat flow:
646         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W) + g * dx^2 / 2;
647
648     case NT_gin % 13
649         % North:
650         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
651         % South:
652         q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
653         % East:
654         q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
655         % West:
656         q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
657         % Net heat flow:
658         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W) + g * dx^2;
659
660     case NT_cbl % 14
661         % North:
662         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
663         % South:
664         q_S = 0;
665         % East:
666         q_E = h * dx * (T_CK - tempMatrixK(m, n));
667         % West:
668         q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
669         % Net heat flow:
670         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
671         sumOuterEdgesE = sumOuterEdgesE + q_E;
672
673     case NT_cl % 15
674         % North:
675         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
676         % South:
677         q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
678         % East:
679         q_E = h * dx * (T_CK - tempMatrixK(m, n));
680         % West:
681         q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
682         % Net heat flow:
683         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
684         sumOuterEdgesE = sumOuterEdgesE + q_E;
685
686     case NT_ct % 16
687         % North:
688         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
689         % South:
690         q_S = h * dx * (T_CK - tempMatrixK(m, n));
691         % East:
692         q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));

```

```

693         % West:
694         q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
695         % Net heat flow:
696         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
697         sumOuterEdgesS = sumOuterEdgesS + q_S;
698
699     case NT_cr % 17
700         % North:
701         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
702         % South:
703         q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
704         % East:
705         q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
706         % West:
707         q_W = h * dx * (T_CK - tempMatrixK(m, n));
708         % Net heat flow:
709         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
710         sumOuterEdgesW = sumOuterEdgesW + q_W;
711
712     case NT_cbr % 18
713         % North:
714         q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
715         % South:
716         q_S = 0;
717         % East:
718         q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
719         % West:
720         q_W = h * dx * (T_CK - tempMatrixK(m, n));
721         % Net heat flow:
722         sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
723         sumOuterEdgesW = sumOuterEdgesW + q_W;
724     otherwise
725
726         sumElemEdges(m,n) = 0;
727     end
728     %}
729
730 end
731 end
732
733 mean(mean(sumElemEdges))
734 max(max(abs(sumElemEdges)))
735
736 sumOuterEdges = vpa((sumOuterEdgesN + sumOuterEdgesE + sumOuterEdgesW + sumOuterEdgesS) + (sumConvLeft + sumConvTop + sumConvRight)) % + g * 2
737 * 0.4^2
738
739 figure
740 histogram(sort(sumElemEdges), [-5.75e-11:0.5e-11:5.75e-11])
741 title('Errors in Element Net Heat Flow (40x80 elements)')
742 xlabel('Error (W)')
743 ylabel('Number of Elements')
744 grid on
745

```



```

746 sumElemEdges(ceil(conv_mMin):ceil(conv_mMax), ceil(conv_nMin):floor(conv_nMax)) = NaN;
747 figure
748 %surf(sumElemEdges, 'EdgeColor', 'None');
749 surf(x, y, sumElemEdges, 'EdgeColor', 'None');
750 view(2)
751 colormap jet
752 title('Net Element Heat Flow Error')
753 xlabel('x-axis (m)')
754 ylabel('y-axis (m)')
755 errorColorBar = colorbar;
756 errorColorBar.Label.String = 'Net heat flow error (W)';

```

[End “smm573_ME450_ComputerProject_MainCode.m” code]

8.5 Appendix V: Full Code, Surface Plot

[Begin “*smm573_create_3Dplot.m*” code]

```
1 create3D = true;
2
3 if create3D
4     figure('Name','Isometric View');
5
6     surf(x,y,tempMatrixC, 'FaceColor', 'interp'); % Plot temperature profile
7
8     hold on
9     grid on
10
11     gen_yAvg = (gen_yMin + gen_yMax) / 2;
12     gen_xAvg = (gen_xMin + gen_xMax) / 2; % Find center of generation
13
14
15     fill3([gen_xMin gen_xAvg gen_xAvg gen_xMin gen_xMin], ...
16           [gen_yAvg gen_yMin gen_yMin gen_yAvg gen_yAvg], ...
17           [0 0 200 200 0],[0.9 0.8 0.8], 'FaceAlpha', 0.5, 'EdgeColor','r');
18     fill3([gen_xMax gen_xAvg gen_xAvg gen_xMax gen_xMax], ...
19           [gen_yAvg gen_yMin gen_yMin gen_yAvg gen_yAvg], ...
20           [0 0 200 200 0],[0.9 0.8 0.8], 'FaceAlpha', 0.5, 'EdgeColor','r');
21     fill3([gen_xMin gen_xAvg gen_xAvg gen_xMin gen_xMin], ...
22           [gen_yAvg gen_yMax gen_yMax gen_yAvg gen_yAvg], ...
23           [0 0 200 200 0],[0.9 0.8 0.8], 'FaceAlpha', 0.5, 'EdgeColor','r');
24     fill3([gen_xMax gen_xAvg gen_xAvg gen_xMax gen_xMax], ...
25           [gen_yAvg gen_yMax gen_yMax gen_yAvg gen_yAvg], ...
26           [0 0 200 200 0],[0.9 0.8 0.8], 'FaceAlpha', 0.5, 'EdgeColor','r'); % Plot generation boundaries
27
28     set(gca,'DataAspectRatio',[1 1 100]) % Scale x and y equally
29
30     f1 = fill3([0 0 4 4 2.6 2.6 1.4 1.4 0], [0 2 2 0 0 .4 .4 0 0], ...
31               [-0.01 -0.01 -0.01 -0.01 -0.01 -0.01 -0.01 -0.01], ...
32               [0.8 0.8 0.8], 'FaceAlpha', 0.5);
33     f2 = fill3([2.8 3.2 3.6 3.2 2.8], [1.2 1.6 1.2 0.8 1.2], ...
34               [0 0 0 0 0], [0.7 0.4 0.4], 'FaceAlpha', 0.5); % Plot geometry boundaries
35     set(gca,'DataAspectRatio',[1 1 100])
36
37     xl = [gen_xMin gen_xMin+dx/2:dx:gen_xAvg-dx/2 gen_xAvg]-0.001;
38     xr = [gen_xMax gen_xMax-dx/2:-dx:gen_xAvg+dx/2 gen_xAvg]+0.001;
39     yb = [gen_yAvg gen_yAvg-dy/2:-dy:gen_yMin+dy/2 gen_yMin]-0.001;
40     yt = [gen_yAvg gen_yAvg+dy/2:dy:gen_yMax-dy/2 gen_yMax]+0.001;
41
```

```

42 ztl = 0 * xl;
43 ztr = 0 * xl;
44 zbl = 0 * xl;
45 zbr = 0 * xl;
46
47 ztl(1) = mean(mean(tempMatrixC(floor(gen_mCenter):ceil(gen_mCenter), ...
48     floor(gen_nMin):ceil(gen_nMin))));
49 ztr(1) = mean(mean(tempMatrixC(floor(gen_mCenter):ceil(gen_mCenter), ...
50     floor(gen_nMax):ceil(gen_nMax))));
51 zbl(1) = mean(mean(tempMatrixC(floor(gen_mCenter):ceil(gen_mCenter), ...
52     floor(gen_nMin):ceil(gen_nMin))));
53 zbr(1) = mean(mean(tempMatrixC(floor(gen_mCenter):ceil(gen_mCenter), ...
54     floor(gen_nMax):ceil(gen_nMax))));
55                                     % Calculate generation bound intersections
56
57 ztl(end) = mean(mean(tempMatrixC(floor(gen_mMax):ceil(gen_mMax), ...
58     floor(gen_nCenter):ceil(gen_nCenter))));
59 ztr(end) = mean(mean(tempMatrixC(floor(gen_mMax):ceil(gen_mMax), ...
60     floor(gen_nCenter):ceil(gen_nCenter))));
61 zbl(end) = mean(mean(tempMatrixC(floor(gen_mMin):ceil(gen_mMin), ...
62     floor(gen_nCenter):ceil(gen_nCenter))));
63 zbr(end) = mean(mean(tempMatrixC(floor(gen_mMin):ceil(gen_mMin), ...
64     floor(gen_nCenter):ceil(gen_nCenter))));
65
66 for i = 1:length(ztl)-2
67     ztl(i+1) = tempMatrixC(floor(gen_mCenter)+i, floor(gen_nMin)+i)+0.1;
68     ztr(i+1) = tempMatrixC(floor(gen_mCenter)+i, ceil(gen_nMax)-i)+0.1;
69     zbl(i+1) = tempMatrixC(ceil(gen_mCenter)-i, floor(gen_nMin)+i)+0.1;
70     zbr(i+1) = tempMatrixC(ceil(gen_mCenter)-i, ceil(gen_nMax)-i)+0.1;
71 end
72
73 plot3(xl,yt,ztl, 'r', 'LineWidth', 1);
74 plot3(xr,yt,ztr, 'r', 'LineWidth', 1);
75 plot3(xl,yb,zbl, 'r', 'LineWidth', 1);
76 plot3(xr,yb,zbr, 'r', 'LineWidth', 1);
77                                     % Plot intersections
78
79 plot3([3.2 3.2], [1.2 1.2], [0 max(max(tempMatrixC))+0.1], ...
80     '-*r','LineWidth', 1);
81 plot3([0 0 0 4 4 4], [0 2 2 2 2 0], [33 13 30 30 42 42], ...
82     '-y','LineWidth', 1);
83
84 set(gca,'DataAspectRatio',[1 1 100])
85
86 xlabel('x-axis (m)')
87 ylabel('y-axis (m)')
88 zlabel('Temperature (degC)')
89 end

```

[End “*smm573_create_3Dplot.m*” code]

8.6 Full Code: Edge Profile Plots

[Begin “*smm573_create_Tileplot.m*” code]

```
1 createTile = true;
2
3 if createTile
4     figure('Name','Edge profiles','Position', [750 100 1000 800]);
5     tiledlayout(5,2);
6
7     % Plot top edge
8     nexttile([1 1])
9     plot([0 x(end,:) 4], [(33-10*2) tempMatrixC(end,:) 42])
10    hold on
11    plot([0 4], [30 30], 'r')
12    title('Top edge (y ~ 2m)')
13    xlabel('x-axis (m)')
14    ylabel('Temp. (degC)')
15    ylim([10 85])
16    xticks([0:0.5:4])
17    yticks([10:15:85])
18    grid on
19
20    % Right edge
21    nexttile([1 1])
22    plot([0 y(:,end)' 2], [tempMatrixC(1,end)-(tempMatrixC(1,end)- ...
23        tempMatrixC(2,end))/2 tempMatrixC(:,end)' 30])
24    hold on
25    plot([0 2], [42 42], 'r')
26    title('Right edge (x ~ 4m)')
27    xlabel('y-axis (m)')
28    ylabel('Temp. (degC)')
29    ylim([10 85])
30    xticks([0:0.25:4])
31    yticks([10:15:85])
32    grid on
33
34    % Left edge
35    nexttile([1 1])
36    plot(y(:,1),tempMatrixC(:,1))
37    hold on
38    plot([0 2], [33 13], 'r')
39    title('Left edge (x ~ 0m)')
40    xlabel('y-axis (m)')
41    ylabel('Temp. (degC)')
42    ylim([10 85])
43    xticks([0:0.25:4])
44    yticks([10:15:85])
45    grid on
46
47    % Bottom edge
48    nexttile([1 1])
49    plot(x(1,:),tempMatrixC(1,:))
```

```

50 title('Bottom edge (y ~ 0m)')
51 xlabel('x-axis (m)')
52 ylabel('Temp. (degC)')
53 ylim([10 85])
54 xticks([0:0.5:4])
55 yticks([10:15:85])
56 grid on
57
58 % Top conv. edge
59 nexttile([1 2])
60 plot(x(ceil(conv_mMax),ceil(conv_nMin):floor(conv_nMax)), ...
61      tempMatrixC(ceil(conv_mMax),ceil(conv_nMin):floor(conv_nMax)))
62 title('Top convective edge (y ~ 0.4m)')
63 xlabel('x-axis (m)')
64 ylabel('Temp. (degC)')
65 ylim([20 80])
66 xticks([1.4:0.2:2.6])
67 yticks([20:15:80])
68 grid on
69
70 % Left conv. edge
71 nexttile([1 2])
72 plot(y(ceil(conv_mMin):floor(conv_mMax),floor(conv_nMin)), ...
73      tempMatrixC(ceil(conv_mMin):floor(conv_mMax),floor(conv_nMin)))
74 title('Left convective edge (x ~ 1.4m)')
75 xlabel('y-axis (m)')
76 ylabel('Temp. (degC)')
77 ylim([10 70])
78 xticks([0:0.05:0.4])
79 yticks([10:15:70])
80 grid on
81
82 % Right conv. edge
83 nexttile([1 2])
84 plot(y(ceil(conv_mMin):floor(conv_mMax),ceil(conv_nMax)), ...
85      tempMatrixC(ceil(conv_mMin):floor(conv_mMax),ceil(conv_nMax)))
86 title('Right convective edge (x ~ 2.6m)')
87 xlabel('y-axis (m)')
88 ylabel('Temp. (degC)')
89 ylim([20 80])
90 xticks([0:0.05:0.4])
91 yticks([20:15:80])
92 grid on
93
94 end

```

[End “smm573_create_Tileplot.m” code]