ME-450 Computer Project Report

2D Numerical Heat Transfer Analysis

March 21, 2021

Sean McGee

Instructor: Dr. Carson Pete

NORTHERN ARIZONA UNIVERSITY

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

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:

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

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 \ 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:

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

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 note 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

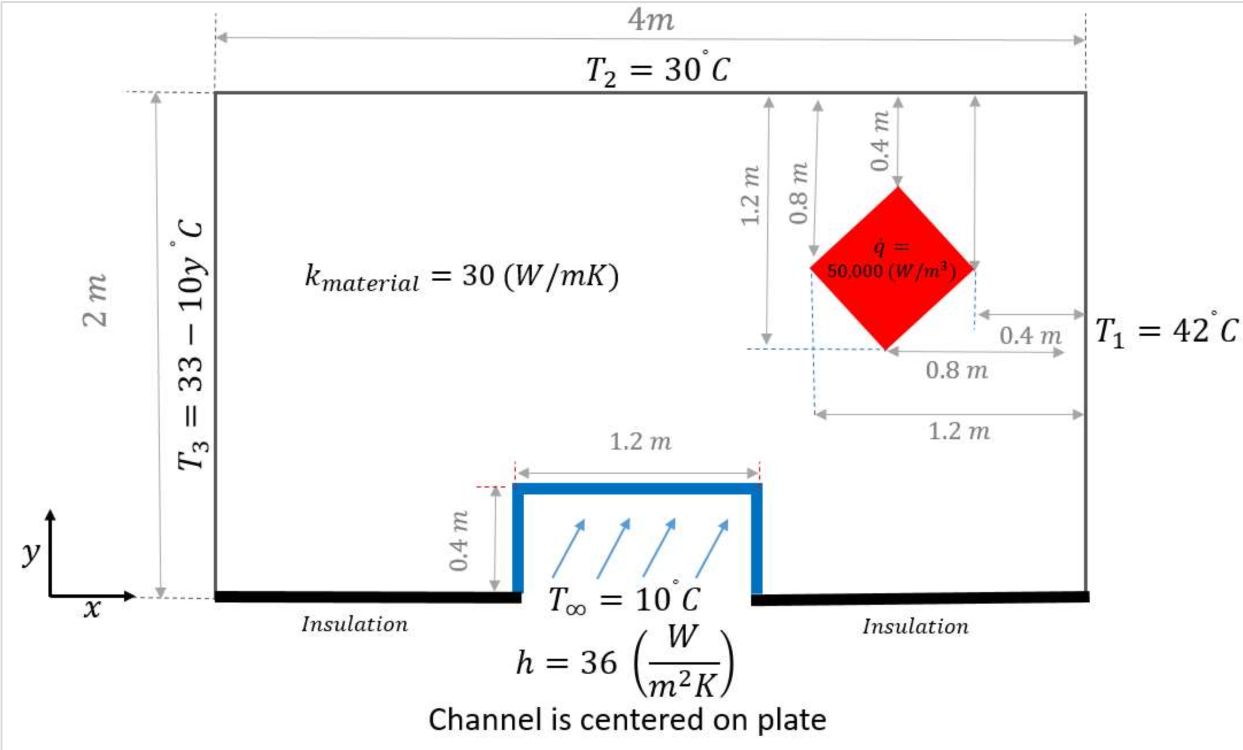

Figure 1: Provided problem geometry [1]

(Note: Convective section is centered along bottom edge)

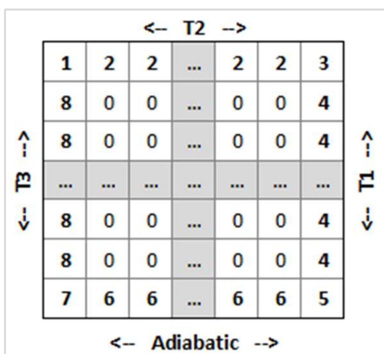

Figure 2: Edge of geometry node types

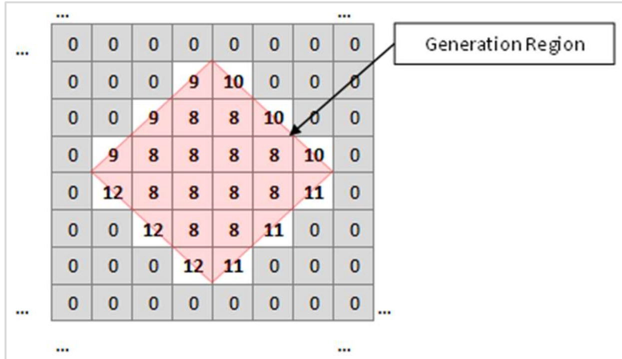

Figure 3: Generation region node types

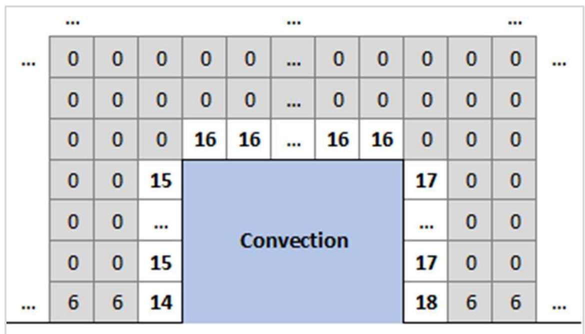

Figure 4: Convection region node types

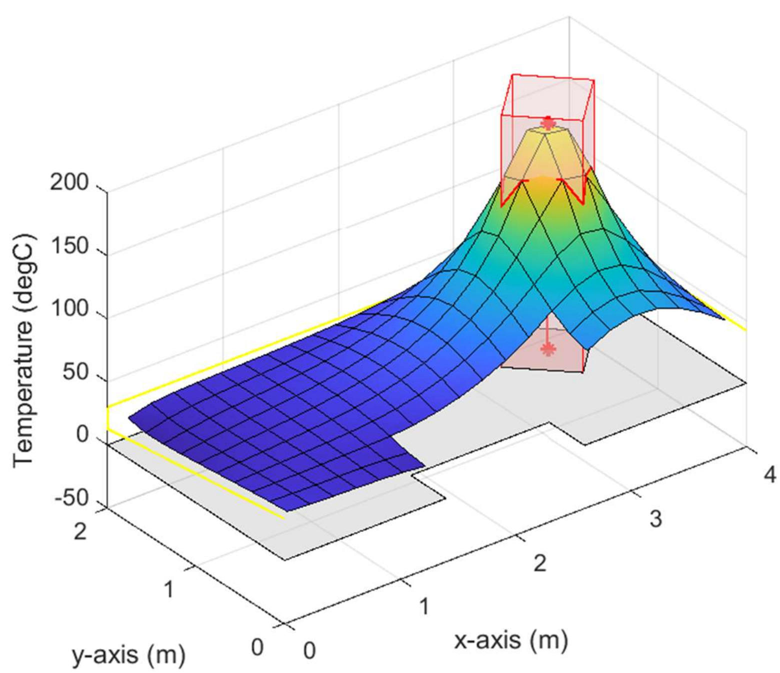

Figure 5: Temperature surface plot for 10x20 elements

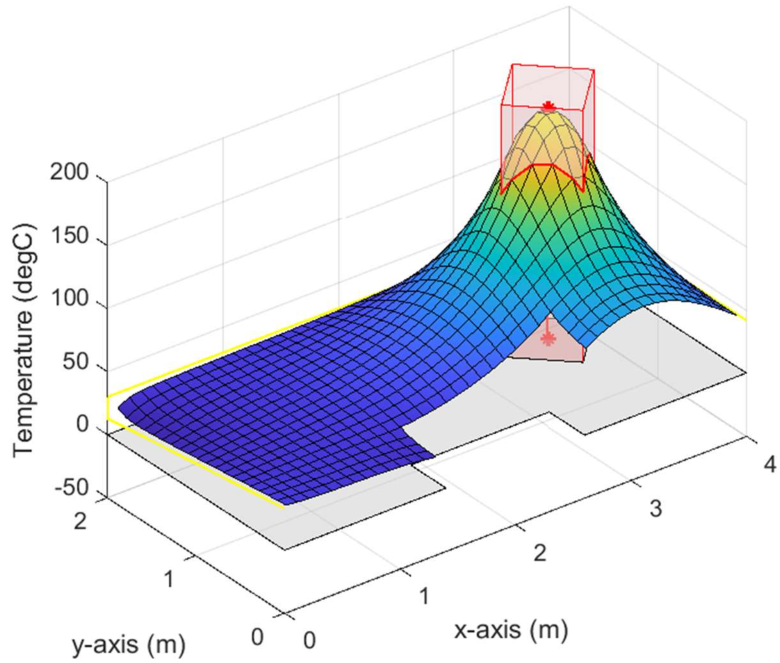

Figure 6: Temperature surface plot for 20x40 elements

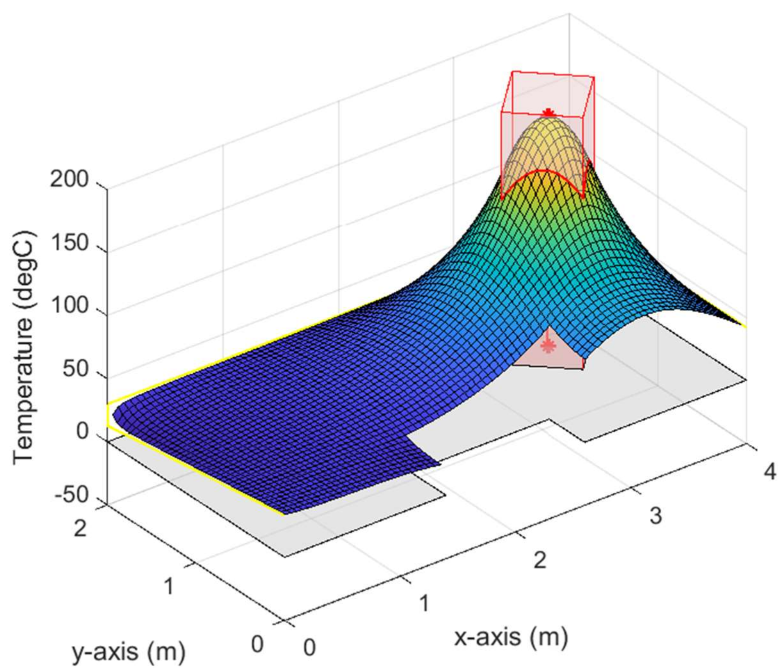

Figure 7: Temperature surface plot for 40x80 elements

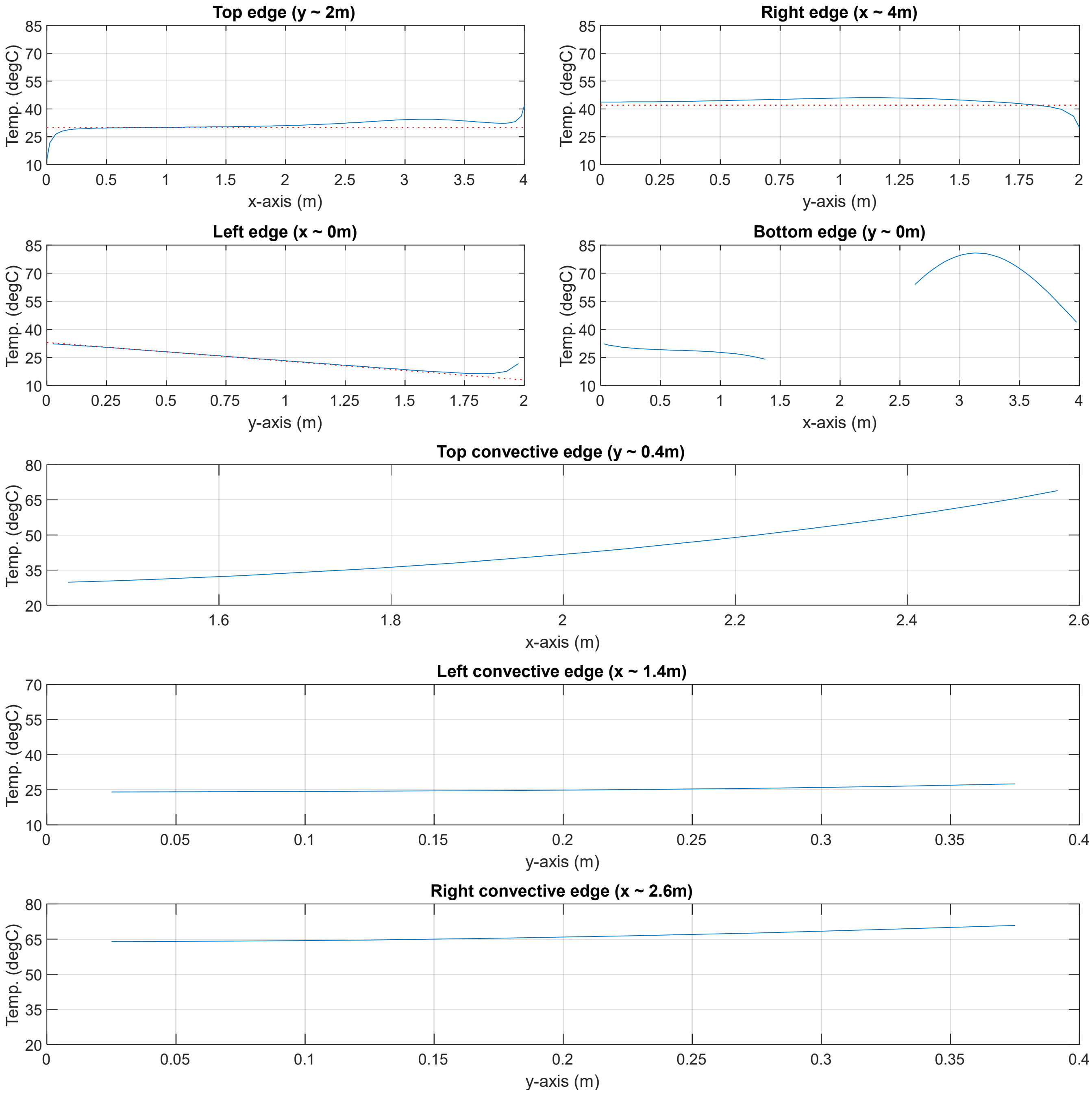

Figure 8: Temperature profile along outer boundaries, 40x80 elements

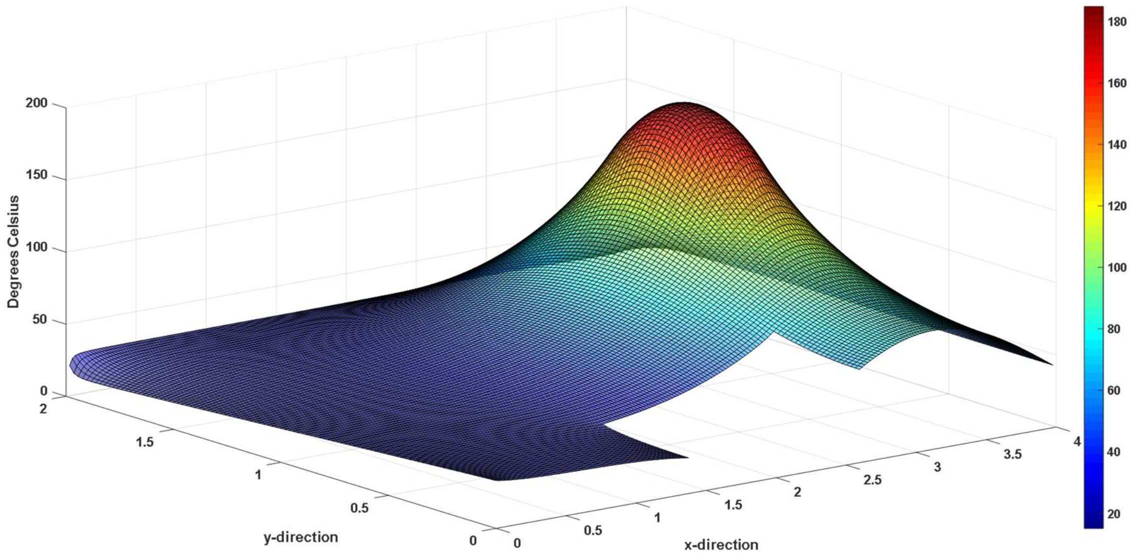

Figure 9: Temperature surface plot provided for validation [2]

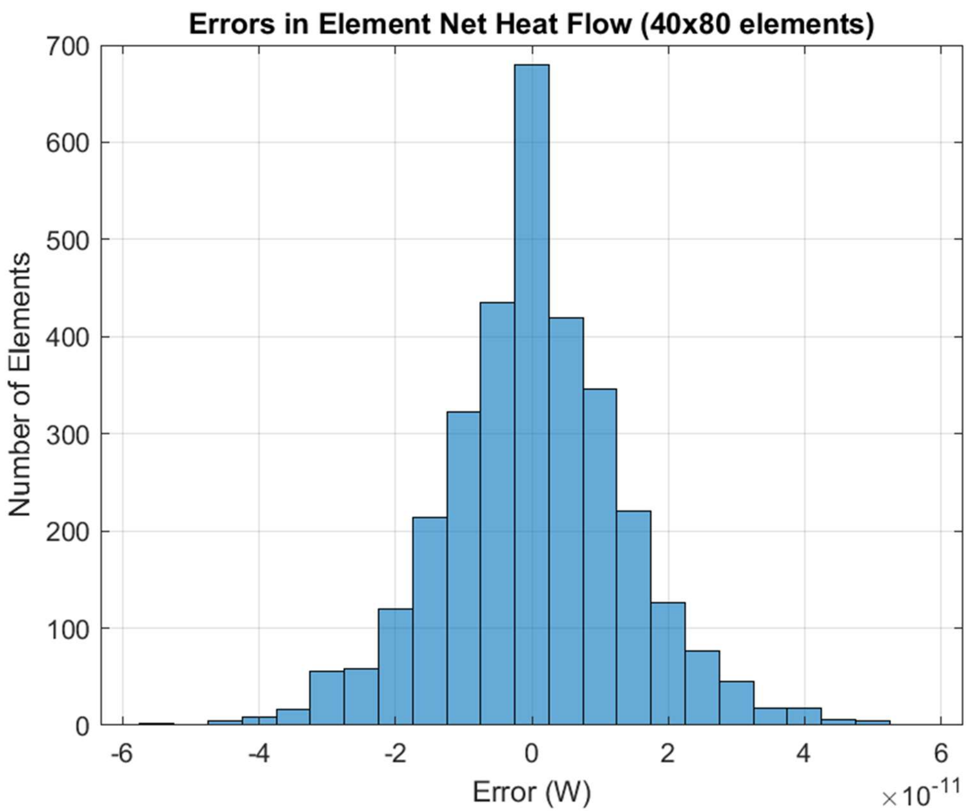

Figure 10: Distribution of nodal boundary errors, 40x80 elements

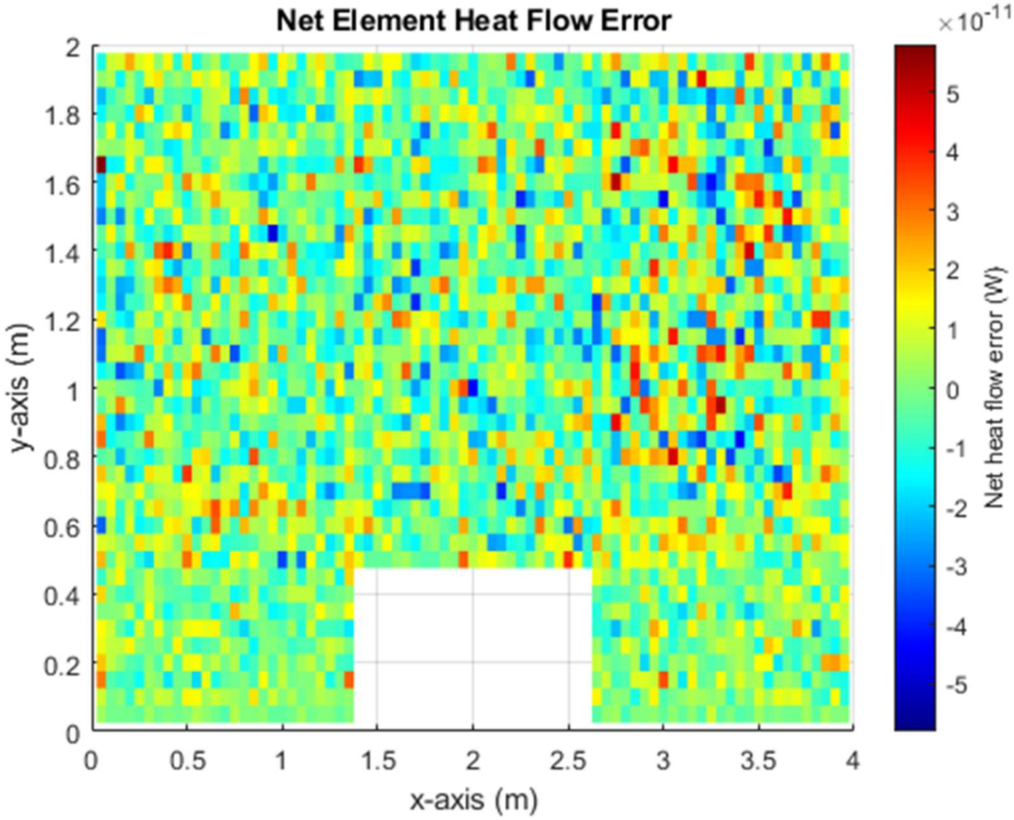

Figure 11: Error in net heat flow through element boundaries

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:	$T_N + T_W + T_E + T_S - 4T = -\mathrm{g}\frac{(\Delta x)^2}{2k}$
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:	$T_N + T_W + T_E + T_S - 4T = -\mathrm{g}\frac{(\Delta x)^2}{2k}$
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:	$T_N + T_W + T_E + T_S - 4T = -\mathrm{g}\frac{(\Delta x)^2}{2k}$
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:	$T_N + T_W + T_E + T_S - 4T = -\mathrm{g}\frac{(\Delta x)^2}{2k}$
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:	$T_N + T_W + T_E + T_S - 4T = -\mathrm{g}\frac{(\Delta x)^2}{k}$

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]

```
%% Physical size
 widthx = 4;                                                              % Horizontal length in meters
heighty = 2;                                                              % Vertical length in meters

%% Size of matrix
numRows = 20;                                                             % Number of rows in matrix (positive integer multiple of 10)

numColumns = round(widthx/heighty) * numRows;                             % Number of columns in matrix (twice numRows)

if ((mod(numRows,5) ~=0) || (mod(numColumns,10) ~= 0))
    error('Value of numRows should be a multiple of 10');
end

%% Initialize matrix variables
dx = widthx / numColumns;                                                 % x-distance between matrix columns
dy = heighty / numRows;                                                   % y-distance between matrix rows

conv_xMin = 1.4;                                                          % Minimum x-value of convection block
conv_xMax = 2.6;                                                          % Maximum x-value of convection block
conv_yMin = 0.0;                                                          % Minimum y-value of convection block
conv_yMax = 0.4;                                                          % Maximum y-value of convection block

gen_xMin = 2.8;                                                           % Location of left-most gen block corner
gen_xMax = 3.6;                                                           % Location of right-most gen block corner
gen_yMin = 0.8;                                                           % Location of top-most gen block corner
gen_yMax = 1.6;                                                           % Location of bottom-most gen block corner


%% Find limits of cutout block in m,n
conv_mMin = conv_yMin / dy + 0.5;
conv_mMax = conv_yMax / dy + 0.5;
conv_nMin = conv_xMin / dx + 0.5;
conv_nMax = conv_xMax / dx + 0.5;


%% Find limits of generation block in m,n
gen_mMin = gen_yMin / dy + 0.5;
gen_mMax = gen_yMax / dy + 0.5;
gen_nMin = gen_xMin / dx + 0.5;
gen_nMax = gen_xMax / dx + 0.5;


%% Various BC values
T_1C = 42;
```

```
T_2C = 30;
T_CC = 10;                                   % Cutout BC: Convective, Tinf = 10 degC
T_1K = T_1C + 273.15;                        % Convert BC temps to Kelvin
T_2K = T_2C + 273.15;
T_CK = T_CC + 273.15;
h = 36;                                      % Convective constant = 36 W/m^2K
g = 50000;                                   % Heat generation is 50kW/m^2 = 50,000 W/m^2
k = 30;                                      % Thermal conductivity = 30 W/mK


%% Input corresponding values into matrix
 NT_in = 0;                                  % Type 0: Inner elements with no BCs or generation
 NT_tl = 1;                                  % Type 1: Top-left corner
  NT_t = 2;                                  % Type 2: Top edge
 NT_tr = 3;                                  % Type 3: Top-right corner
  NT_r = 4;                                  % Type 4: Right edge
 NT_br = 5;                                  % Type 5: Bottom-right corner
  NT_b = 6;                                  % Type 6: Bottom edge
 NT_bl = 7;                                  % Type 7: Bottom-left corner
  NT_l = 8;                                  % Type 8: Left edge
NT_gtl = 9;                                  % Type 9: Generation, top-left edge
NT_gtr = 10;                                 % Type 10: Generation, top-right edge
NT_gbr = 11;                                 % Type 11: Generation, bottom-right edge
NT_gbl = 12;                                 % Type 12: Generation, bottom-left edge
NT_gin = 13;                                 % Type 13: Generation, internal
NT_cbl = 14;                                 % Type 14: Convection, bottom-left corner
 NT_cl = 15;                                 % Type 15: Convection, left edge
 NT_ct = 16;                                 % Type 16: Convection, top edge
 NT_cr = 17;                                 % Type 17: Convection, right edge
NT_cbr = 18;                                 % Type 18: Convection, bottom-right edge

% Initialize matrices
labelMatrix = zeros(numRows, numColumns);    % Create matrix for node labels
tempMatrixK = zeros(numRows, numColumns);    % Create matrix for solving temperature profile

%% Fill label matrix
% Four corner elements
labelMatrix(1, 1) = NT_bl;                   % Top-left
labelMatrix(1, numColumns) = NT_br;          % Top-right
labelMatrix(numRows, 1) = NT_tl;             % Bottom-left
labelMatrix(numRows, numColumns) = NT_tr;    % Bottom-right

% Top, bottom edge
for n = 2:(numColumns - 1)
    labelMatrix(1, n) = NT_b;
    labelMatrix(numRows, n) = NT_t;
end

% Left edge
for m = 2:(numRows - 1)
    labelMatrix(m, 1) = NT_l;
end

% Right edge
```

```
for m = 2:(numRows - 1)
    labelMatrix(m, numColumns) = NT_r;
end

% Generation block
%   Boundaries of gen block are y = |x - 3.2| + 0.8 ; y = -|x - 3.2| + 1.6
gen_nCenter = mean([gen_nMin gen_nMax]);
gen_mCenter = mean([gen_mMin gen_mMax]);
for n = ceil(gen_nMin):floor(gen_nMax)
    mGT = abs(n-gen_nCenter)+gen_mMin;                                      % "In. gen block" RHS only depend on n, so calculate them
    mLT = -abs(n-gen_nCenter)+gen_mMax;                                     % only in "n" loop to save time
    for m = ceil(gen_mMin):floor(gen_mMax)
        if (m == mGT)                                                       % If m is on the bottom boundary...
            if (n < gen_nCenter)                                            %    ...and n is before the center...
                labelMatrix(m,n) = NT_gbl;                                  %        ...then this is the bottom-left corner
            else                                                            %    Otherwise...
                labelMatrix(m,n) = NT_gbr;                                  %        ...this is the bottom-right corner
            end
        elseif (m == mLT)                                                   % If m is on the top boundary...
            if (n < gen_nCenter)                                            %    ...and n is before the center...
                labelMatrix(m,n) = NT_gtl;                                  %        ...then this is the top-left corner
            else                                                            %    Otherwise...
                labelMatrix(m,n) = NT_gtr;                                  %        ...this is the top-right corner
            end
        elseif ((m > mGT) && (m < mLT))                                     % Otherwise, if not on either boundary but IS in the range...
            labelMatrix(m,n) = NT_gin;                                      %    ...The element is fully within gen block
        end
    end
end

% Cutout block
for m = ceil(conv_mMin):floor(conv_mMax)
    for n = ceil(conv_nMin):floor(conv_nMax)
        labelMatrix(m,n) = NaN;
    end
end
labelMatrix(ceil(conv_mMax), ceil(conv_nMin):floor(conv_nMax)) = NT_ct;
labelMatrix(ceil(conv_mMin)+1:floor(conv_mMax), floor(conv_nMin)) = NT_cl;
labelMatrix(ceil(conv_mMin)+1:floor(conv_mMax), ceil(conv_nMax)) = NT_cr;
labelMatrix(ceil(conv_mMin), floor(conv_nMin)) = NT_cbl;
labelMatrix(ceil(conv_mMin), ceil(conv_nMax)) = NT_cbr;


%% Create coefficient matrix
coeffMatrix = zeros(numRows * numColumns);                                  % Instantiate coefficient matrix
b = coeffMatrix(:,1);                                                       % Instantiate b vector
convConst = h * dx / k;                                                     % Calculate convective term so it's not recalculated every loop
genConst = g * dx * dx / k;                                                 % Calculate generation term so it's not recalculated every loop

for m = 1:numRows
    T_3K = 273.15 + 33 - 10 * ((m-1) * dy + dy/2);                          % Calculate T_3 for each new row
    for n = 1:numColumns
        T_pos = (m - 1) * numColumns + n;                                   % Calculate row of coefficient matrix, b vector
        switch labelMatrix(m,n)
```

```
            case NT_in % 0
                % Self:
                coeffMatrix(T_pos, T_pos) = -4;
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
                % Bottom:
                coeffMatrix(T_pos, T_pos - numColumns) = 1;
                % Left:
                coeffMatrix(T_pos, T_pos - 1) = 1;
                % Right:
                coeffMatrix(T_pos, T_pos + 1) = 1;
                % Constant: NONE
                b(T_pos) = 0;

            case NT_tl % 1
                % Self:
                coeffMatrix(T_pos, T_pos) = -6;
                % Top: T_2 BC
                % Bottom:
                coeffMatrix(T_pos, T_pos - numColumns) = 1;
                % Left: T_3 BC
                % Right:
                coeffMatrix(T_pos, T_pos + 1) = 1;
                % Constant:
                b(T_pos,1) = -2*(T_2K + T_3K);

            case NT_t % 2
                % Self:
                coeffMatrix(T_pos, T_pos) = -5;
                % Top: T_2 BC
                % Bottom:
                coeffMatrix(T_pos, T_pos - numColumns) = 1;
                % Left:
                coeffMatrix(T_pos, T_pos - 1) = 1;
                % Right:
                coeffMatrix(T_pos, T_pos + 1) = 1;
                % Constant:
                b(T_pos,1) = -2*T_2K;

            case NT_tr % 3
                % Self:
                coeffMatrix(T_pos, T_pos) = -6;
                % Top: T_2 BC
                % Bottom:
                coeffMatrix(T_pos, T_pos - numColumns) = 1;
                % Left:
                coeffMatrix(T_pos, T_pos - 1) = 1;
                % Right: T_1 BC
                % Constant:
                b(T_pos,1) = -2*(T_1K + T_2K);

            case NT_r % 4
                % Self:
```

```
                coeffMatrix(T_pos, T_pos) = -5;
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
                % Bottom:
                coeffMatrix(T_pos, T_pos - numColumns) = 1;
                % Left:
                coeffMatrix(T_pos, T_pos - 1) = 1;
                % Right: T_1 BC
                % Constant:
                b(T_pos,1) = -2*T_1K;

            case NT_br % 5
                % Self:
                coeffMatrix(T_pos, T_pos) = -4;
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
                % Bottom: INS. BC
                % Left:
                coeffMatrix(T_pos, T_pos - 1) = 1;
                % Right: T_1 BC
                % Constant:
                b(T_pos,1) = -2*T_1K;

            case NT_b % 6
                % Self:
                coeffMatrix(T_pos, T_pos) = -3;
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
                % Bottom: INS. BC
                % Left:
                coeffMatrix(T_pos, T_pos - 1) = 1;
                % Right: T_1 BC
                coeffMatrix(T_pos, T_pos + 1) = 1;
                % Constant: NONE
                b(T_pos,1) = 0;

            case NT_bl % 7
                % Self:
                coeffMatrix(T_pos, T_pos) = -4;
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
                % Bottom: INS. BC
                % Left: T_3 BC
                % Right: T_1 BC
                coeffMatrix(T_pos, T_pos + 1) = 1;
                % Constant:
                b(T_pos,1) = -2*T_3K;

            case NT_l % 8
                % Self:
                coeffMatrix(T_pos, T_pos) = -5;
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
                % Bottom:
```

```
                coeffMatrix(T_pos, T_pos - numColumns) = 1;
                % Left: T_3 BC
                % Right:
                coeffMatrix(T_pos, T_pos + 1) = 1;
                % Constant:
                b(T_pos,1) = -2*T_3K;

            case NT_gtl % 9
                % Self:
                coeffMatrix(T_pos, T_pos) = -4;
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
                % Bottom:
                coeffMatrix(T_pos, T_pos - numColumns) = 1;
                % Left:
                coeffMatrix(T_pos, T_pos - 1) = 1;
                % Right:
                coeffMatrix(T_pos, T_pos + 1) = 1;
                % Constant:
                b(T_pos,1) = -genConst/2;

            case NT_gtr % 10
                % Self:
                coeffMatrix(T_pos, T_pos) = -4;
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
                % Bottom:
                coeffMatrix(T_pos, T_pos - numColumns) = 1;
                % Left:
                coeffMatrix(T_pos, T_pos - 1) = 1;
                % Right:
                coeffMatrix(T_pos, T_pos + 1) = 1;
                % Constant:
                b(T_pos,1) = -genConst/2;

            case NT_gbr % 11
                % Self:
                coeffMatrix(T_pos, T_pos) = -4;
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
                % Bottom:
                coeffMatrix(T_pos, T_pos - numColumns) = 1;
                % Left:
                coeffMatrix(T_pos, T_pos - 1) = 1;
                % Right:
                coeffMatrix(T_pos, T_pos + 1) = 1;
                % Constant:
                b(T_pos,1) = -genConst/2;

            case NT_gbl % 12
                % Self:
                coeffMatrix(T_pos, T_pos) = -4;
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
```

```
                % Bottom:
                coeffMatrix(T_pos, T_pos - numColumns) = 1;
                % Left:
                coeffMatrix(T_pos, T_pos - 1) = 1;
                % Right:
                coeffMatrix(T_pos, T_pos + 1) = 1;
                % Constant:
                b(T_pos,1) = -genConst/2;

            case NT_gin % 13
                % Self:
                coeffMatrix(T_pos, T_pos) = -4;
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
                % Bottom:
                coeffMatrix(T_pos, T_pos - numColumns) = 1;
                % Left:
                coeffMatrix(T_pos, T_pos - 1) = 1;
                % Right:
                coeffMatrix(T_pos, T_pos + 1) = 1;
                % Constant:
                b(T_pos,1) = -genConst;

            case NT_cbl % 14
                % Self:
                coeffMatrix(T_pos, T_pos) = -(2+convConst);
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
                % Bottom: INS. BC
                % Left:
                coeffMatrix(T_pos, T_pos - 1) = 1;
                % Right: CONV. BC
                % Constant:
                b(T_pos,1) = -convConst * T_CK;

            case NT_cl % 15
                % Self:
                coeffMatrix(T_pos, T_pos) = -(3+convConst);
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
                % Bottom:
                coeffMatrix(T_pos, T_pos - numColumns) = 1;
                % Left:
                coeffMatrix(T_pos, T_pos - 1) = 1;
                % Right: CONV. BC
                % Constant:
                b(T_pos,1) = -convConst * T_CK;

            case NT_ct % 16
                % Self:
                coeffMatrix(T_pos, T_pos) = -(3+convConst);
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
                % Bottom: CONV. BC
```

```
                % Left:
                coeffMatrix(T_pos, T_pos - 1) = 1;
                % Right:
                coeffMatrix(T_pos, T_pos + 1) = 1;
                % Constant:
                b(T_pos,1) = -convConst * T_CK;

            case NT_cr % 17
                % Self:
                coeffMatrix(T_pos, T_pos) = -(3+convConst);
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
                % Bottom:
                coeffMatrix(T_pos, T_pos - numColumns) = 1;
                % Left: CONV. BC
                % Right:
                coeffMatrix(T_pos, T_pos + 1) = 1;
                % Constant:
                b(T_pos,1) = -convConst * T_CK;

            case NT_cbr % 18
                % Self:
                coeffMatrix(T_pos, T_pos) = -(2+convConst);
                % Top:
                coeffMatrix(T_pos, T_pos + numColumns) = 1;
                % Bottom: INS. BC
                % Left: CONV. BC
                % Right:
                coeffMatrix(T_pos, T_pos + 1) = 1;
                % Constant:
                b(T_pos,1) = -convConst * T_CK;

            otherwise
                % Within convection block OR some error
                if (isnan(labelMatrix(m,n)))
                    coeffMatrix(T_pos, T_pos) = 1;                          % NaN indicates convection block
                    b(T_pos,1) = 0;
                else
                    msg1 = 'Label at labelMatrix(';
                    msg2 = ') not recognized, something is very wrong...';
                    error("%s%d,%d%s",msg1,m,n,msg2);
                end
        end
    end
end

%% Solve system of equations

tempVector = coeffMatrix \ b;


%% Fill tempMatrixK with solved values

m = 1;
```

```
n = 1;
for i = 1:length(tempVector)
    tVi = tempVector(i);
    if (tVi == 0)
        tempMatrixK(m,n) = NaN;
    else
        tempMatrixK(m,n) = tempVector(i,1);
    end
    n = n + 1;
    if (n > numColumns)
        n = 1;
        m = m + 1;
    end
end


%% Convert temps from K to C
tempMatrixC = tempMatrixK - 273.15;


%% Plot temp profile
create3D = true;
createContour = false;
createTiles = true;
[x,y] = meshgrid((dx/2):dx:(widthx - dx/2),(dy/2):dy:(heighty - dy/2));

if create3D
    smm573_create_3Dplot;
end

if createContour
    smm573_create_Contourplot;
end

if createTiles
    smm573_create_Tileplot;
end


%% Check errors
sumElemEdges = labelMatrix * 0;
q_N = 0;
q_S = 0;
q_E = 0;
q_W = 0;
sumOuterEdgesN = 0;
sumOuterEdgesE = 0;
sumOuterEdgesW = 0;
sumOuterEdgesS = 0;

for m = 1:numRows
    T_3K = 273.15 + 33 - 10 * ((m-1) * dy + dy/2);
    for n = 1:numColumns
        %T_pos = (m - 1) * numColumns + n;
```

```
        %indicesMatrix(m,n) = T_pos;
        switch labelMatrix(m,n)
            case NT_in % 0
                % North:
                q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
                % South:
                q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
                % East:
                q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
                % West:
                q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
                % Net heat flow:
                sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);


            case NT_tl % 1
                % North:
                q_N = k * 2*(T_2K - tempMatrixK(m, n));
                % South:
                q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
                % East:
                q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
                % West:
                q_W = k * 2*(T_3K - tempMatrixK(m, n));
                % Net heat flow:
                sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
                sumOuterEdgesN = sumOuterEdgesN + q_N;
                sumOuterEdgesW = sumOuterEdgesW + q_W;

            case NT_t % 2
                % North:
                q_N = k * 2*(T_2K - tempMatrixK(m, n));
                % South:
                q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
                % East:
                q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
                % West:
                q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
                % Net heat flow:
                sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
                sumOuterEdgesN = sumOuterEdgesN + q_N;

            case NT_tr % 3
                % North:
                q_N = k * 2*(T_2K - tempMatrixK(m, n));
                % South:
                q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
                % East:
                q_E = k * 2*(T_1K - tempMatrixK(m, n));
                % West:
                q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
                % Net heat flow:
                sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
                sumOuterEdgesN = sumOuterEdgesN + q_N;
```

```
                    sumOuterEdgesE = sumOuterEdgesE + q_E;

                case NT_r % 4
                    % North:
                    q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
                    % South:
                    q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
                    % East:
                    q_E = k * 2*(T_1K - tempMatrixK(m, n));
                    % West:
                    q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
                    % Net heat flow:
                    sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
                    sumOuterEdgesE = sumOuterEdgesE + q_E;

                case NT_br % 5
                    % North:
                    q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
                    % South:
                    q_S = 0;
                    % East:
                    q_E = k * 2*(T_1K - tempMatrixK(m, n));
                    % West:
                    q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
                    % Net heat flow:
                    sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
                    sumOuterEdgesE = sumOuterEdgesE + q_E;
                    sumOuterEdgesS = sumOuterEdgesS + q_S;

                case NT_b % 6
                    % North:
                    q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
                    % South:
                    q_S = 0;
                    % East:
                    q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
                    % West:
                    q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
                    % Net heat flow:
                    sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
                    sumOuterEdgesS = sumOuterEdgesS + q_S;

                case NT_bl % 7
                    % North:
                    q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
                    % South:
                    q_S = 0;
                    % East:
                    q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
                    % West:
                    q_W = k * 2*(T_3K - tempMatrixK(m, n));
                    % Net heat flow:
                    sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
                    sumOuterEdgesW = sumOuterEdgesW + q_W;
```

```
                sumOuterEdgesS = sumOuterEdgesS + q_S;

            case NT_l % 8
                % North:
                q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
                % South:
                q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
                % East:
                q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
                % West:
                q_W = k * 2*(T_3K - tempMatrixK(m, n));
                % Net heat flow:
                sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
                sumOuterEdgesW = sumOuterEdgesW + q_W;

            case NT_gtl % 9
                % North:
                q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
                % South:
                q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
                % East:
                q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
                % West:
                q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
                % Net heat flow:
                sumElemEdges(m,n) = (q_N + q_S + q_E + q_W) + g * dx^2 / 2;

            case NT_gtr % 10
                % North:
                q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
                % South:
                q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
                % East:
                q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
                % West:
                q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
                % Net heat flow:
                sumElemEdges(m,n) = (q_N + q_S + q_E + q_W) + g * dx^2 / 2;

            case NT_gbr % 11
                % North:
                q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
                % South:
                q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
                % East:
                q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
                % West:
                q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
                % Net heat flow:
                sumElemEdges(m,n) = (q_N + q_S + q_E + q_W) + g * dx^2 / 2;

            case NT_gbl % 12
                % North:
                q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
```

```
                % South:
                q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
                % East:
                q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
                % West:
                q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
                % Net heat flow:
                sumElemEdges(m,n) = (q_N + q_S + q_E + q_W) + g * dx^2 / 2;

            case NT_gin % 13
                % North:
                q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
                % South:
                q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
                % East:
                q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
                % West:
                q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
                % Net heat flow:
                sumElemEdges(m,n) = (q_N + q_S + q_E + q_W) + g * dx^2;

            case NT_cbl % 14
                % North:
                q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
                % South:
                q_S = 0;
                % East:
                q_E = h * dx * (T_CK - tempMatrixK(m, n));
                % West:
                q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
                % Net heat flow:
                sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
                sumOuterEdgesE = sumOuterEdgesE + q_E;

            case NT_cl % 15
                % North:
                q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
                % South:
                q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
                % East:
                q_E = h * dx * (T_CK - tempMatrixK(m, n));
                % West:
                q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
                % Net heat flow:
                sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
                sumOuterEdgesE = sumOuterEdgesE + q_E;

            case NT_ct % 16
                % North:
                q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
                % South:
                q_S = h * dx * (T_CK - tempMatrixK(m, n));
                % East:
                q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
```

```
                % West:
                q_W = k * (tempMatrixK(m, n-1) - tempMatrixK(m, n));
                % Net heat flow:
                sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
                sumOuterEdgesS = sumOuterEdgesS + q_S;

            case NT_cr % 17
                % North:
                q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
                % South:
                q_S = k * (tempMatrixK(m-1, n) - tempMatrixK(m, n));
                % East:
                q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
                % West:
                q_W = h * dx * (T_CK - tempMatrixK(m, n));
                % Net heat flow:
                sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
                sumOuterEdgesW = sumOuterEdgesW + q_W;

            case NT_cbr % 18
                % North:
                q_N = k * (tempMatrixK(m+1, n) - tempMatrixK(m, n));
                % South:
                q_S = 0;
                % East:
                q_E = k * (tempMatrixK(m, n+1) - tempMatrixK(m, n));
                % West:
                q_W = h * dx * (T_CK - tempMatrixK(m, n));
                % Net heat flow:
                sumElemEdges(m,n) = (q_N + q_S + q_E + q_W);
                sumOuterEdgesW = sumOuterEdgesW + q_W;
            otherwise

                sumElemEdges(m,n) = 0;
        end
        %}

    end
end

mean(mean(sumElemEdges))
max(max(abs(sumElemEdges)))

sumOuterEdges = vpa((sumOuterEdgesN + sumOuterEdgesE + sumOuterEdgesW + sumOuterEdgesS) + (sumConvLeft + sumConvTop + sumConvRight)) %+ g * 2
* 0.4^2

figure
histogram(sort(sumElemEdges), [-5.75e-11:0.5e-11:5.75e-11])
title('Errors in Element Net Heat Flow (40x80 elements)')
xlabel('Error (W)')
ylabel('Number of Elements')
grid on

```

```
sumElemEdges(ceil(conv_mMin):ceil(conv_mMax), ceil(conv_nMin):floor(conv_nMax)) = NaN;
figure
%surf(sumElemEdges, 'EdgeColor', 'None');
surf(x, y, sumElemEdges, 'EdgeColor', 'None');
view(2)
colormap jet
title('Net Element Heat Flow Error')
xlabel('x-axis (m)')
ylabel('y-axis (m)')
errorColorBar = colorbar;
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]

```
create3D = true;

if create3D
    figure('Name','Isometric View');

    surf(x,y,tempMatrixC, 'FaceColor', 'interp');                                % Plot temperature profile

    hold on
    grid on

    gen_yAvg = (gen_yMin + gen_yMax) / 2;
    gen_xAvg = (gen_xMin + gen_xMax) / 2;                                        % Find center of generation


    fill3([gen_xMin gen_xAvg gen_xAvg gen_xMin gen_xMin], ...
        [gen_yAvg gen_yMin gen_yMin gen_yAvg gen_yAvg], ...
        [0 0 200 200 0],[0.9 0.8 0.8], 'FaceAlpha', 0.5, 'EdgeColor','r');
    fill3([gen_xMax gen_xAvg gen_xAvg gen_xMax gen_xMax], ...
        [gen_yAvg gen_yMin gen_yMin gen_yAvg gen_yAvg], ...
        [0 0 200 200 0],[0.9 0.8 0.8], 'FaceAlpha', 0.5, 'EdgeColor','r');
    fill3([gen_xMin gen_xAvg gen_xAvg gen_xMin gen_xMin], ...
        [gen_yAvg gen_yMax gen_yMax gen_yAvg gen_yAvg], ...
        [0 0 200 200 0],[0.9 0.8 0.8], 'FaceAlpha', 0.5, 'EdgeColor','r');
    fill3([gen_xMax gen_xAvg gen_xAvg gen_xMax gen_xMax], ...
        [gen_yAvg gen_yMax gen_yMax gen_yAvg gen_yAvg], ...
        [0 0 200 200 0],[0.9 0.8 0.8], 'FaceAlpha', 0.5,'EdgeColor','r');        % Plot generation boundaries

    set(gca,'DataAspectRatio',[1 1 100])                                         % Scale x and y equally

    f1 = fill3([0 0 4 4 2.6 2.6 1.4 1.4 0], [0 2 2 0 0 .4 .4 0 0], ...
        [-0.01 -0.01 -0.01 -0.01 -0.01 -0.01 -0.01 -0.01 -0.01], ...
        [0.8 0.8 0.8], 'FaceAlpha', 0.5);
    f2 = fill3([2.8 3.2 3.6 3.2 2.8], [1.2 1.6 1.2 0.8 1.2], ...
        [0 0 0 0 0], [0.7 0.4 0.4], 'FaceAlpha', 0.5);                           % Plot geometry boundaries
    set(gca,'DataAspectRatio',[1 1 100])

    xl = [gen_xMin gen_xMin+dx/2:dx:gen_xAvg-dx/2 gen_xAvg]-0.001;
    xr = [gen_xMax gen_xMax-dx/2:-dx:gen_xAvg+dx/2 gen_xAvg]+0.001;
    yb = [gen_yAvg gen_yAvg-dy/2:-dy:gen_yMin+dy/2 gen_yMin]-0.001;
    yt = [gen_yAvg gen_yAvg+dy/2:dy:gen_yMax-dy/2 gen_yMax]+0.001;

```

```
    ztl = 0 * xl;
    ztr = 0 * xl;
    zbl = 0 * xl;
    zbr = 0 * xl;

    ztl(1) = mean(mean(tempMatrixC(floor(gen_mCenter):ceil(gen_mCenter), ...
        floor(gen_nMin):ceil(gen_nMin))));
    ztr(1) = mean(mean(tempMatrixC(floor(gen_mCenter):ceil(gen_mCenter), ...
        floor(gen_nMax):ceil(gen_nMax))));
    zbl(1) = mean(mean(tempMatrixC(floor(gen_mCenter):ceil(gen_mCenter), ...
        floor(gen_nMin):ceil(gen_nMin))));
    zbr(1) = mean(mean(tempMatrixC(floor(gen_mCenter):ceil(gen_mCenter), ...
        floor(gen_nMax):ceil(gen_nMax))));                                  % Calculate generation bound intersections

    ztl(end) = mean(mean(tempMatrixC(floor(gen_mMax):ceil(gen_mMax), ...
        floor(gen_nCenter):ceil(gen_nCenter))));
    ztr(end) = mean(mean(tempMatrixC(floor(gen_mMax):ceil(gen_mMax), ...
        floor(gen_nCenter):ceil(gen_nCenter))));
    zbl(end) = mean(mean(tempMatrixC(floor(gen_mMin):ceil(gen_mMin), ...
        floor(gen_nCenter):ceil(gen_nCenter))));
    zbr(end) = mean(mean(tempMatrixC(floor(gen_mMin):ceil(gen_mMin), ...
        floor(gen_nCenter):ceil(gen_nCenter))));

    for i = 1:length(ztl)-2
        ztl(i+1) = tempMatrixC(floor(gen_mCenter)+i, floor(gen_nMin)+i)+0.1;
        ztr(i+1) = tempMatrixC(floor(gen_mCenter)+i, ceil(gen_nMax)-i)+0.1;
        zbl(i+1) = tempMatrixC(ceil(gen_mCenter)-i, floor(gen_nMin)+i)+0.1;
        zbr(i+1) = tempMatrixC(ceil(gen_mCenter)-i, ceil(gen_nMax)-i)+0.1;
    end

    plot3(xl,yt,ztl, 'r', 'LineWidth', 1);                                  % Plot intersections
    plot3(xr,yt,ztr, 'r', 'LineWidth', 1);
    plot3(xl,yb,zbl, 'r', 'LineWidth', 1);
    plot3(xr,yb,zbr, 'r', 'LineWidth', 1);

    plot3([3.2 3.2], [1.2 1.2], [0 max(max(tempMatrixC))+0.1], ...
        '-*r','LineWidth', 1);
    plot3([0 0 0 4 4 4], [0 2 2 2 2 0], [33 13 30 30 42 42], ...
        '-y','LineWidth', 1);

    set(gca,'DataAspectRatio',[1 1 100])

    xlabel('x-axis (m)')
    ylabel('y-axis (m)')
    zlabel('Temperature (degC)')
end
```

[End "smm573_create_3Dplot.m" code]

8.6 Full Code: Edge Profile Plots

[Begin "smm573_create_Tileplot.m" code]

```
createTile = true;

if createTile
    figure('Name','Edge profiles','Position', [750 100 1000 800]);
    tiledlayout(5,2);

    % Plot top edge
    nexttile([1 1])
    plot([0 x(end,:) 4], [(33-10*2) tempMatrixC(end,:) 42])
    hold on
    plot([0 4], [30 30], ':r')
    title('Top edge (y ~ 2m)')
    xlabel('x-axis (m)')
    ylabel('Temp. (degC)')
    ylim([10 85])
    xticks([0:0.5:4])
    yticks([10:15:85])
    grid on

    % Right edge
    nexttile([1 1])
    plot([0 y(:,end)' 2], [tempMatrixC(1,end)-(tempMatrixC(1,end)- ...
        tempMatrixC(2,end))/2 tempMatrixC(:,end)' 30])
    hold on
    plot([0 2], [42 42], ':r')
    title('Right edge (x ~ 4m)')
    xlabel('y-axis (m)')
    ylabel('Temp. (degC)')
    ylim([10 85])
    xticks([0:0.25:4])
    yticks([10:15:85])
    grid on

    % Left edge
    nexttile([1 1])
    plot(y(:,1),tempMatrixC(:,1))
    hold on
    plot([0 2], [33 13], ':r')
    title('Left edge (x ~ 0m)')
    xlabel('y-axis (m)')
    ylabel('Temp. (degC)')
    ylim([10 85])
    xticks([0:0.25:4])
    yticks([10:15:85])
    grid on

    % Bottom edge
    nexttile([1 1])
    plot(x(1,:),tempMatrixC(1,:))
```

```
    title('Bottom edge (y ~ 0m)')
    xlabel('x-axis (m)')
    ylabel('Temp. (degC)')
    ylim([10 85])
    xticks([0:0.5:4])
    yticks([10:15:85])
    grid on

    % Top conv. edge
    nexttile([1 2])
    plot(x(ceil(conv_mMax),ceil(conv_nMin):floor(conv_nMax)), ...
        tempMatrixC(ceil(conv_mMax),ceil(conv_nMin):floor(conv_nMax)))
    title('Top convective edge (y ~ 0.4m)')
    xlabel('x-axis (m)')
    ylabel('Temp. (degC)')
    ylim([20 80])
    xticks([1.4:0.2:2.6])
    yticks([20:15:80])
    grid on

    % Left conv. edge
    nexttile([1 2])
    plot(y(ceil(conv_mMin):floor(conv_mMax),floor(conv_nMin)), ...
        tempMatrixC(ceil(conv_mMin):floor(conv_mMax),floor(conv_nMin)))
    title('Left convective edge (x ~ 1.4m)')
    xlabel('y-axis (m)')
    ylabel('Temp. (degC)')
    ylim([10 70])
    xticks([0:0.05:0.4])
    yticks([10:15:70])
    grid on

    % Right conv. edge
    nexttile([1 2])
    plot(y(ceil(conv_mMin):floor(conv_mMax),ceil(conv_nMax)), ...
        tempMatrixC(ceil(conv_mMin):floor(conv_mMax),ceil(conv_nMax)))
    title('Right convective edge (x ~ 2.6m)')
    xlabel('y-axis (m)')
    ylabel('Temp. (degC)')
    ylim([20 80])
    xticks([0:0.05:0.4])
    yticks([20:15:80])
    grid on

end
```

[End “smm573_create_Tileplot.m” code]