Saahas Yechuri

Donovan Hamby

Christian Bennett

Numerical Project

1. Abstract:

This project seeks to accurately visualize and model the temperature distribution of a given section. Numerical analysis was performed, and results were compared for different heat transfer coefficients. Our methodology and equations modelling the 2D conduction equation along with our boundary conditions numerically are outlined in section 2. In section 3, we perform a grid independence test, find temperature distributions corresponding to our equations and parameters, and plot them. Interesting observations are also noted, such as the paraboloid shape of our final temperature distributions, and the changes to the "steepness" or eccentricity of these parabolic cross sections based on the changing values of the convection coefficient in the y direction (h_y).

2. Introduction:

A generalized code was written to solve for any given m and n. The x-direction is defined by n (columns) and the y-direction by m (rows). Square grids were used to simplify the equations. Due to the lines of symmetry, the upper right quadrant was thoroughly analyzed and then later mirrored over into the respective three other quadrants.

Given information was assessed and categorized in the x and y directions. The thermal conductivity and convection coefficients were different for the x and y directions, so this was considered when setting up the equations. Another implemented factor was the lines of symmetry being adiabatic. This reduced the number of unknowns to find. A symbolic matrix of all unknown temperature points was created.

Boundary conditions were accounted for in the corners and sides. These equations were based on Table 4.2 in the class textbook (*Wiley's Fundamentals of Heat and Mass Transfer*, Page 226). For each of the sides and corners, we had to be mindful of what terms to include or exclude in the equations. For example, we removed the corresponding heat flows for the adiabatic sides (left and bottom), and we included for the convective sides (top and left) a convection term for those corresponding heat flows. The rest of the heat flows were conductive and were modelled as such. Furthermore, for a given grid point, it was also important to consider the area across which heat transfer processes were occurring, and volume corresponding to that grid point. All the boundary condition grid points were either half a standard grid's volume (sides) or a quarter of a grid (corners). This had to be accounted for in the heat generation terms. Furthermore, the grid points had certain areas that were half of a standard grid size – the left and right sides had top and bottom areas that were half, the top and bottom had left and right sides that were half, and the corners had all sides being a half of the standard grid size. This had to be accounted for in the conduction and convection terms corresponding to those surfaces. Taking these into consideration, Boundary conditions equations were formed in terms of m and n to maintain the general structure. An example of a boundary condition equation is shown below for the top right corner. Refer to the Appendix with the documented MATLAB code for a complete list of the equations.

$$0.5 * k_y * \big(T(2,n) - T(1,n)\big) + 0.5 * k_x * \big(T(1,n-1) - T(1,n)\big) + h_x * 0.5 * dx * \big(T_\infty - T(1,n)\big) + h_y * 0.5 * dx * \big(T_\infty - T(1,n)\big) + 0.25 * q''' * dx^2 = 0$$

After the boundary conditions were set up, the interior node equation was applied to the rest of the points – modelling all heat flows from all sides simply as conduction. Unknown symbolic temperatures were sent to the MATLAB solve function to find the numeric values. Temperature distribution matrix data was labeled and sent to Excel to access later. This saved time and resources by not having to run multiple times for the same information. The equation for the interior nodes is below. Refer to the Appendix for its implementation in code.

$$k_x * \big(T(i,j-1) - T(i,j)\big) + k_x * \big(T(i,j+1) - T(i,j)\big) + k_y * \big(T(i+1,j) - T(i,j)\big) + k_y * \big(T(i-1,j) - T(i,j)\big) + q''' * dx^2 = 0$$

Each h_y and grid size increment was run separately to save time and check if the code and results were justifiable. After the code and equations were verified, grid independence was performed. Refer to Section 3a for how the grid independence test was performed.

After a grid size was determined, temperature distribution matrices were found for it. Then, this data was plotted. The plots and results are displayed and discussed in Section 3b and 3c.

3. Results and Discussion:

a) Grid Independence:

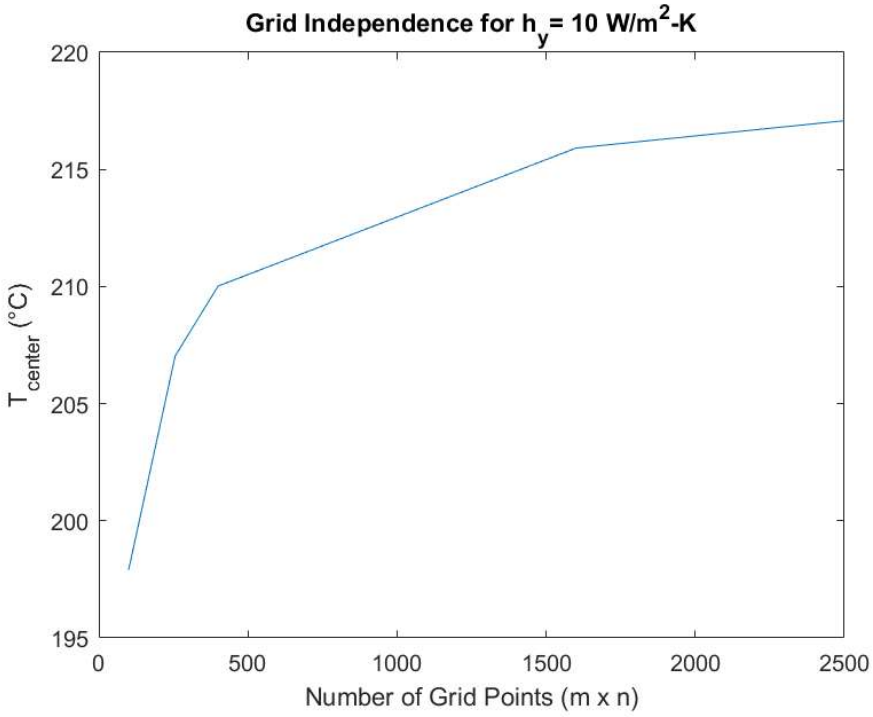

Figure 1: Temperature at the center with increasing number of grids

Grid Independence Verification						
h_y (W/m^{2}K)	T_1 (°C)	T_2 (°C)	Percent Difference (%)	x_1	x_2	$1.5x_1$
10	215.911259	217.0774	0.54011	1600	2500	2400
30	139.6198257	140.5845	0.690921146	1600	2500	2400
50	108.6728184	109.476	0.739061995	1600	2500	2400
70	91.92340659	92.61968	0.757446766	1600	2500	2400
100	77.53391229	78.12723	0.765240193	1600	2500	2400

Table 1: Grid Independence Verification that demonstrates by increasing the number of grids by 50%, the temperature changes by less than 2%

For an accurate result, we ran a grid independence test. The tests were completed under the conditions of increasing the number of grids by 50% and noting that the resulting temperature changes by less than 2%. This is represented by the equation below.

$$x_2 = 1.5x_1 \implies T_2 = T_1 \pm 0.02T_1$$

When x_1 and x_2 are the total number of grid points at different grid size increments.

These two grid independence conditions were met by two points for all the h_y values. It was determined a grid size of 100 x 25 (m x n) would be more than sufficient with percentage changes less than even 1 percent.

b) Line Plots:

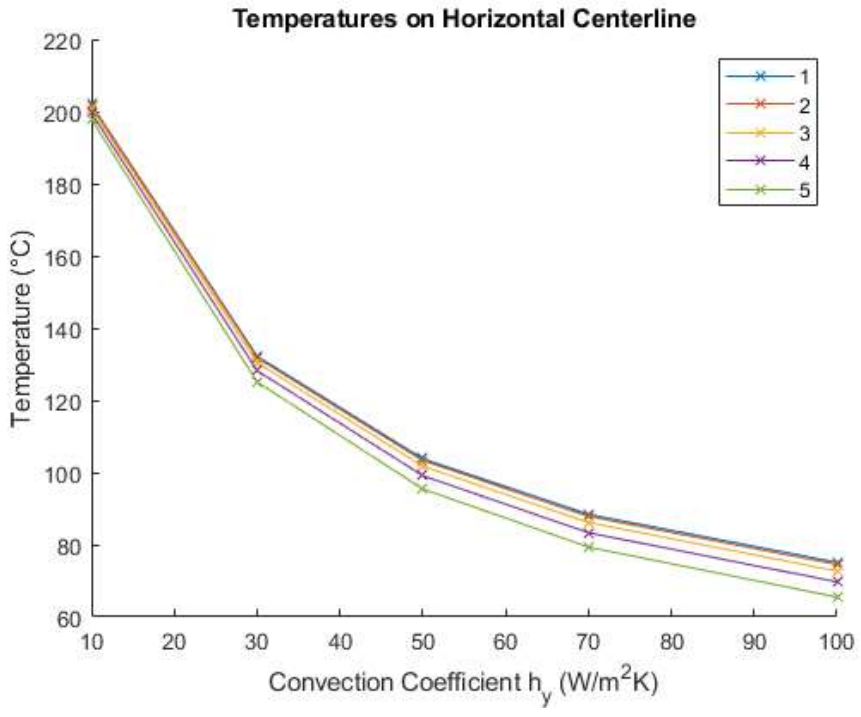

Figure 2: Temperatures on horizontal centerline at various h_y

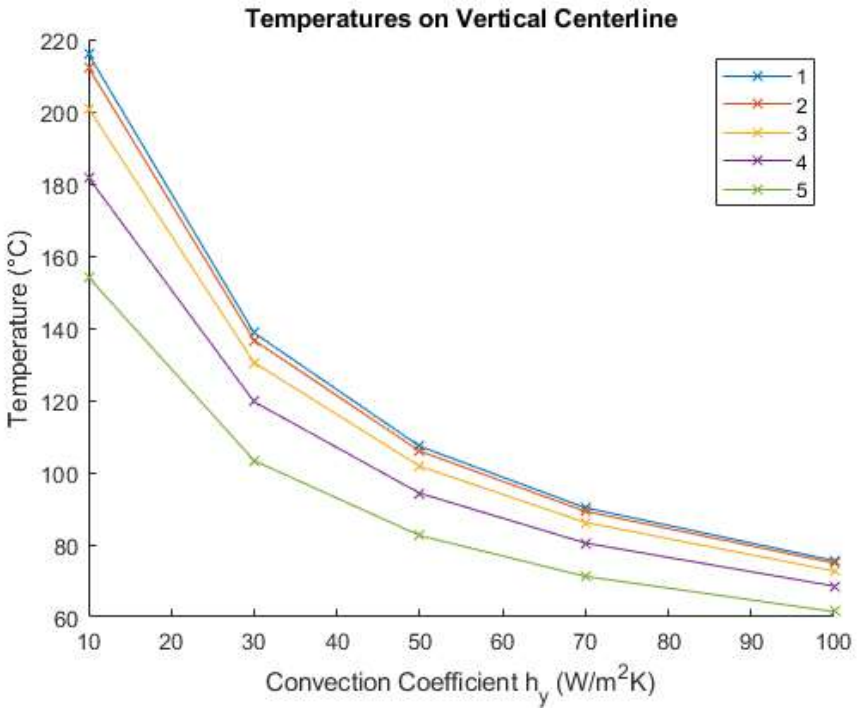

Figure 3: Temperatures on vertical centerline at various h_y

Examining these line plots, we notice a few key observations. For one, the temperature appears to be higher for the points closer to the center of the container, and lower for the points closer to the edges of the container. This matches our intuition as we expect the surfaces of the container to cool down more as compared to its insides when exposed to cooling fluid. With regards to the convection coefficient in the y direction, we notice a clear correlation between higher convection coefficient resulting in lower temperatures at all points. This also makes sense, as a higher convection coefficient will result in more heat transferred for the same difference in temperature – and therefore increased cooling of the container. Finally, between the vertical and horizontal centerlines, it was observed that the vertical centerlines have higher temperature between points at the same h_y. This is partially a result of the geometry of the container – there are the same amount of sample points along the x and y centerlines, but the container is longer in the y direction, so we expect that there is a greater temperature difference between the sample points. However, we can also notice that for the horizontal centerline, for low values of h_y, all 5 points are tightly bunched up – with an almost constant temperature along the horizontal axis – and they get further apart as h_y increases. This makes sense, as with low values of the convection coefficient in the y direction, most of the heat transfer happens at the top and bottom of the container even though it has smaller area – simply because h_x is higher. However, as h_y gets larger, we notice that the gaps between the points also gets larger – a result of greater heat transfer on the right side.

c) Contour/Surface Plots:

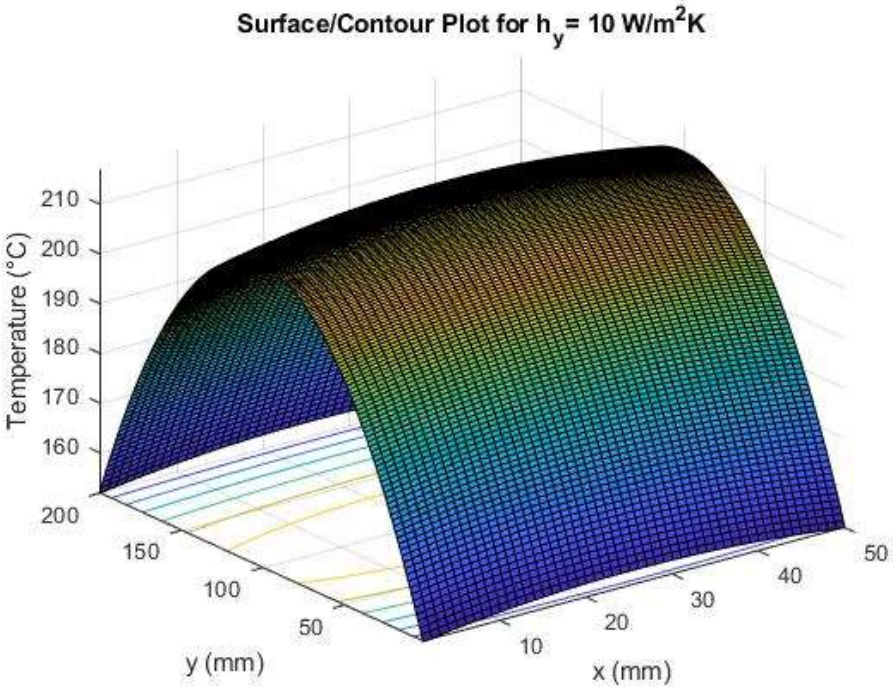

Figure 4: Temperature distribution plotted in 3D for h_y=10 W/m^{2}K

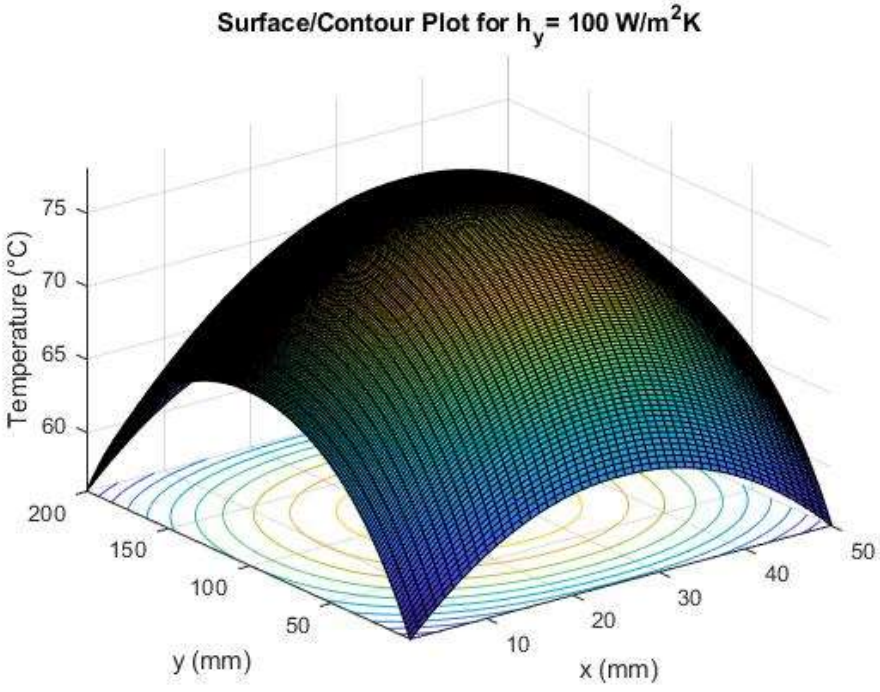

Figure 5: Temperature distribution plotted in 3D for h_y=100 W/m^{2}K

Examining these surface plots for convection coefficients of 10 and 100 respectively, we notice their clear paraboloid shape that we would expect for a 2D block with heat generation. This is because for a 1D block with heat generation, we expect a parabolic temperature distribution, so it makes sense when this is carried into 2D. Furthermore, the maximum temperature for both blocks is also in the middle, which is also expected due to the container's symmetry. The two main differences between these plots that we see are the magnitudes of their temperatures, as well as their shape – specifically the eccentricities of the parabolas that form this paraboloid. With regards to the temperatures of both plots, we can clearly see that the plot for $h_y = 10$ has higher temperatures than $h_y = 100$ across the board. This is due to the greater amount of total heat transfer away from the container in the latter case due to the additional convective heat loss, resulting in lower temperatures overall. The shape of the first paraboloid is interesting as it has a significantly higher eccentricity along the y axis, and a significantly lower eccentricity along the x axis. This is due to the significantly lower amount of convective heat transfer along the right and left sides of the container in the former case ($h_y = 10$) and higher convective heat transfer of the top and bottom. On the other hand, the latter case has equal convection coefficients in both directions, resulting in much closer eccentricities.

4. Conclusion:

To summarize our work on this project: we used numerical methods to solve a heat transfer problem about a 2D container with heat transfer convective heat transfer across all 4 sides of it. This problem was made complicated by having different convection coefficients in the x and y directions (especially considering that we had to solve for several values of h_y) and that we had to determine the best grid size to get an accurate temperature distribution independent of grid size. We discussed our problem-solving approach to solving for all the grid points in section 2 – explaining our approach to the boundary nodes, as well as the inside nodes. We explored grid size values in section 3a and determined the best grid size for our purposes to be dx = 1mm, which resulted in less than 1% changes in T for all values of h for a 50% increase in grid points from the previous value (dx = 1.25mm). Finally, in sections 3b and 3c, we discussed some of our observations.

5. Appendix:

Project.mlx:

```
kx = 5; % W/mK
ky = 10; % W/mK
qdot = 200000; % W/m^3

Tinf = 25; %deg C

hy_vec = [10 30 50 70 100]; % W/m^2K
hx = 100; % W/m^2K


x = 0.025; %m
y = 0.100; %m

for hy = hy_vec
    % did grid independence, found that dx = 1 mm is suitable
    dx = 1/1000 %m
    n = x/dx;
    m = y/dx;
    T = sym('T',[m n]); % m rows, n columns
    eqns = sym('eqns',[m n]);


    % Boundary Conditions
    % corners:
    % bottom left
    eqns(m,1) = 0.5*ky*(T(m-1,1) - T(m,1)) + 0.5*kx*(T(m,2) - T(m,1)) + 0.25*qdot*dx^2 == 0;
    % top left
    eqns(1,1) = 0.5*ky*(T(2,1) - T(1,1)) + 0.5*kx*(T(1,2) - T(1,1)) + hx*0.5*dx*(Tinf - T(1,1)) +
0.25*qdot*dx^2 == 0;
    % bottom right
    eqns(m,n) = 0.5*ky*(T(m-1,n) - T(m,n)) + 0.5*kx*(T(m,n-1) - T(m,n)) + hy*0.5*dx*(Tinf -
T(m,n)) + 0.25*qdot*dx^2 == 0;
    % top right
    eqns(1,n) = 0.5*ky*(T(2,n) - T(1,n)) + 0.5*kx*(T(1,n-1) - T(1,n)) + hx*0.5*dx*(Tinf - T(1,n))
+ hy*0.5*dx*(Tinf - T(1,n)) + 0.25*qdot*dx^2 == 0;
    % sides:
    % left
    for i = 2:m-1
        eqns(i,1) = 0.5*ky*(T(i-1,1) - T(i,1)) + 0.5*ky*(T(i+1,1) - T(i,1)) + kx*(T(i,2) - T(i,1)) +
0.5*qdot*dx^2 == 0;
    end
    % right
    for i = 2:m-1
```

```
    eqns(i,n) = 0.5*ky*(T(i-1,n) - T(i,n)) + 0.5*ky*(T(i+1,n) - T(i,n)) + kx*(T(i,n-1) - T(i,n)) +
hy*dx*(Tinf - T(i,n)) + 0.5*qdot*dx^2 == 0;
  end
  % top
  for i = 2:n-1
    eqns(1,i) = 0.5*kx*(T(1,i-1) - T(1,i)) + 0.5*kx*(T(1,i+1) - T(1,i)) + ky*(T(2,i) - T(1,i)) +
hx*dx*(Tinf - T(1,i)) + 0.5*qdot*dx^2 == 0;
  end
  % bottom
  for i = 2:n-1
    eqns(m,i) = 0.5*kx*(T(m,i-1) - T(m,i)) + 0.5*kx*(T(m,i+1) - T(m,i)) + ky*(T(m-1,i) -
T(m,i)) + 0.5*qdot*dx^2 == 0;
  end

  % interior nodes
  for i = 2:m-1
    for j = 2:n-1
      eqns(i,j) = kx*(T(i,j-1) - T(i,j)) + kx*(T(i,j+1) - T(i,j)) + ky*(T(i+1,j) - T(i,j)) + ky*(T(i-
1,j) - T(i,j)) + qdot*dx^2 == 0;
    end
  end

  % solving system of equations and substituting back into original T
  % matrix
  Temps = solve(eqns(:), T(:));
  T = subs(T, Temps);
  T_final= double(T)

  C= num2cell(T_final);
  filename = sprintf("hy=%d.xls", hy)
  writecell(C, filename,'WriteMode','append');
end

beep on
```

Grid_Independence.mlx:

```
T = [197.904 207.039 210.02 215.91 217.08 ]
dx = [5 3.125 2.5 1.25 1]/1000

n = 0.025./dx;
m = 0.100./dx;
grids = n .* m



plot(grids,T)
```

```
title('Grid Independence for h_{y}= 10 W/m^2-K')
xlabel('Number of Grid Points (m x n)')
ylabel('T_{center}  (°C)')

h10= [215.911259 217.0774173]
h30= [139.6198257 140.5844886]
h50= [108.6728184 109.4759779]
h70=[91.92340659 92.61967746]
h100=[77.53391229 78.12723295]

h_vec= [h10 h30 h50 h70 h100]

per_arr=[]
for i=1:2:length(h_vec)
   per_diff= ((h_vec(i+1)-h_vec(i))/h_vec(i))*100

   per_arr= [per_arr ; h_vec(i) h_vec(i+1) per_diff]

end
```

Line_Plots.mlx:

```
Tdata(:, :, 1) = readmatrix('hy=10.xls');
Tdata(:, :, 2) = readmatrix('hy=30.xls');
Tdata(:, :, 3) = readmatrix('hy=50.xls');
Tdata(:, :, 4) = readmatrix('hy=70.xls');
Tdata(:, :, 5) = readmatrix('hy=100.xls');

hvals = [10 30 50 70 100];

VertLines = [];
HorizLines = [];

for ind = 1:5
   h = hvals(ind);
   T = Tdata(:, :, ind);
   [m,n] = size(T);

   mid_m = (1 + m)/2;
   mid_n = (1 + n)/2;

   pointX = linspace(m,1,  5);
   pointY = ones(1,5) .* mid_n;
   VertPoints = interp2(T, pointY, pointX);
   VertLines = [VertLines VertPoints'];

   pointX = ones(1,5) .* mid_m;
```

```
    pointY = linspace(1,n, 5);
    HorizPoints = interp2(T, pointY, pointX);
    HorizLines = [HorizLines HorizPoints'];
end

f1 = figure;
hold on
for ind = 1:5
    plot(hvals, HorizLines(ind,:), 'x-')
end
legend('1', '2', '3', '4', '5');
title('Temperatures on Horizontal Centerline')
xlabel('Convection Coefficient h_{y} (W/m^2K)')
ylabel('Temperature (°C)')

f2 = figure;
hold on
for ind = 1:5
    plot(hvals, VertLines(ind,:), 'x-')
end
legend('1', '2', '3', '4', '5');
title('Temperatures on Vertical Centerline')
xlabel('Convection Coefficient h_{y} (W/m^2K)')
ylabel('Temperature (°C)')
```

Contour_Plots.mlx:

```
h10= readmatrix("hy=10.xls");
h100= readmatrix("hy=100.xls");

h10= [flip(h10, 2) h10; flip(flip(h10), 2) flip(h10)];
h100= [flip(h100, 2) h100; flip(flip(h100), 2) flip(h100)];

% contour(h10)
% % contour3(h10)
% contourf(h10)
surfc(h10)
title('Surface/Contour Plot for h_{y}= 10 W/m^2K')
xlabel('x (mm)')
ylabel('y (mm)')
zlabel('Temperature (°C)')

surfc(h100)
title('Surface/Contour Plot for h_{y}= 100 W/m^2K')
xlabel('x (mm)')
ylabel('y (mm)')
zlabel('Temperature (°C)')
```