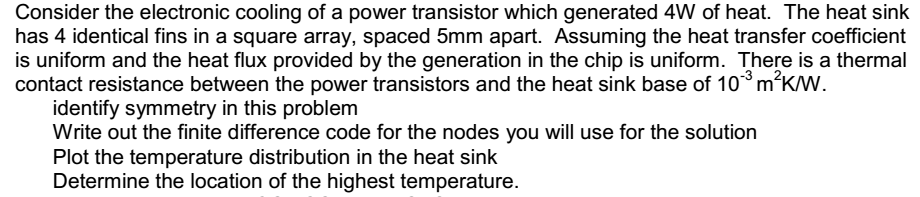
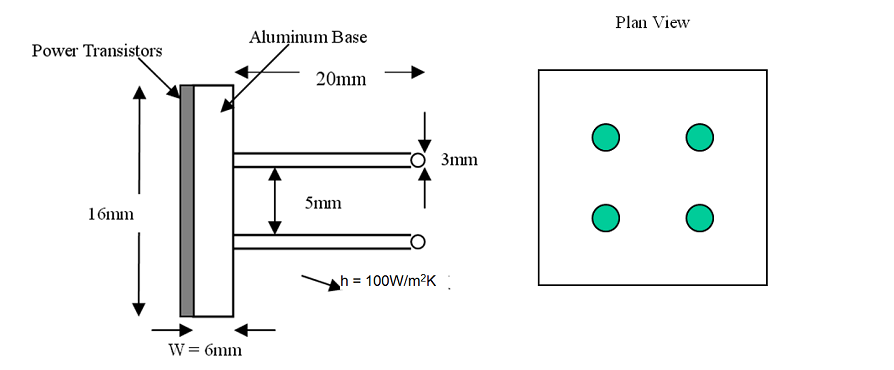
ME3345 Term Project

Tiancheng Gong (tgong7)

4/22/2015

**TOPIC:**

Heat Sink Problem (Topic NO.2)

**PROBLEM DESCRIPTION:**

**PURPOSE:**

Find and plot the temperature distribution of the heat sink, find the position with the highest temperature using numerical method.

**ASSUMPTIONS:**

- The heat transfer coefficient is uniform.

- The heat flux provided by the generation in the chip is uniform.

- Steady state.

- 3D conduction.

- No radiation happens.

- The temperature within in each block unit is uniform.

- The temperature of the heat sink is around 300K.

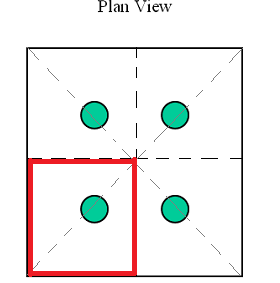
- The temperature of the infinite distance is around 300K. It is the difference between the environment and the chip instead of the temperature of environment itself that matters in this problem.

**ANALYSIS AND METHOD:**

Since the shape of the heat sink is symmetric, the model can be simplified by dividing the heat sink into 4 identical parts, 4 square bases and one rod in the middle of each base. All the analysis in this report are based on one of the identical parts. *(It can be divided into at most 8 identical triangle parts, but it is not easy to build this model using cube blocks.) See Figure 1.1 below.*

The tool used here is the finite difference equations. For each cube block unit (or may be triangle block), we can derive a finite difference equation based on the temperature of its neighbors and the property of the material. We will have *n* cubes as n unknowns as well as *n* equations, thus the equations are solvable. *See Figure 1.2 below for block unit detail.*

Using the solution derived, we can thus plot the temperature distribution and find the position of the highest temperature.



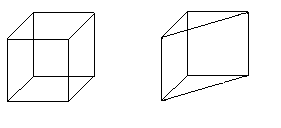


Figure 1.2

The cube block on the left side is used to represent the temperature of the point in the square base and the rod. The triangle block is just half of the cube block, used to represent the edge of the rod. The side length of the cube is 0.5 mm.

Figure 1.1

The heat sink is symmetric to the dashed lines. The region marked by red is the one of the identical part to be analyzed.

**BLOCK UNIT DETAILS:**

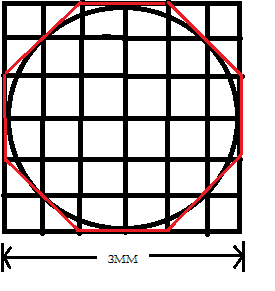
 To make a precise simulation, I decide to represent the temperature of the rod combining two kinds of the blocks shown above. **The side length of the cube is 0.5mm.** *The details are shown in Figure 1.3 below.* To explain it in a mathematical way, I do the calculation as following. The actual circumference of the rod cross-section is. The circumference I used here would be. The error is below 3%, so it would be a satisfying approximation.

Figure 1.3

This is the top view of the rod. The area circled by the red line is used to approximate the rod cross-section. For each level, 8 triangle block units and 24 cube block units are used.

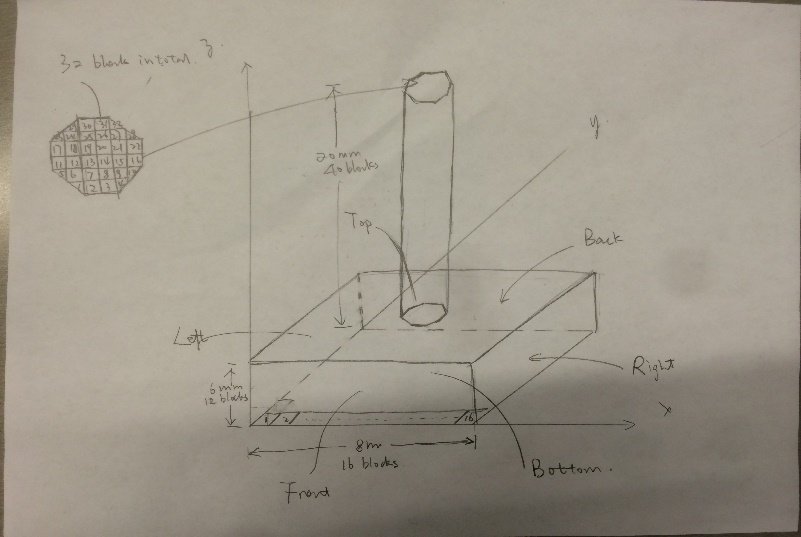
 *Figure1.4* below(hand copy attached at the end) shows the way I represent the identical part (one-fourth of the whole heat sink) using the block units. Each block has an index based on the z-y-x order, from 1 to 4352 (). Each block units has neighbors up to 6: left\_neighbor, right\_neighbor, front\_neighbor, back\_neighbor, top\_neighbor, bot\_neighbor.

Figure 1.4

This is how I arrange and index the block units to representing the identical part of the heat sink.

**DERIVING EQUATIONS:**

For each block unit, it has at most ***f*** (5 or 6) face directions depending on the shape. For each face direction, it can be one of the situations below.

1. It has a neighbor at this direction. There is conduction between them:
2. It is exposed to the environment. Then convection occurs:
3. It is at the connection part between the triangle unit and the cube unit. Then the heat flow between the cube and the triangle unit is the combination of (1) and (2):
4. It touches the chip. Then there will be heat flow from the chip:
5. It is on the border line and face to the identical parts. Then no heat flow in this direction:

According to the finite difference theorem and the conservation rule, we can wrap the equations for all face directions together.

Important values:

Room temperature: 300K (Assumption)

k = 237 W/m\*K;

h = 100 W/m^2\*K;

Total power = 4W;

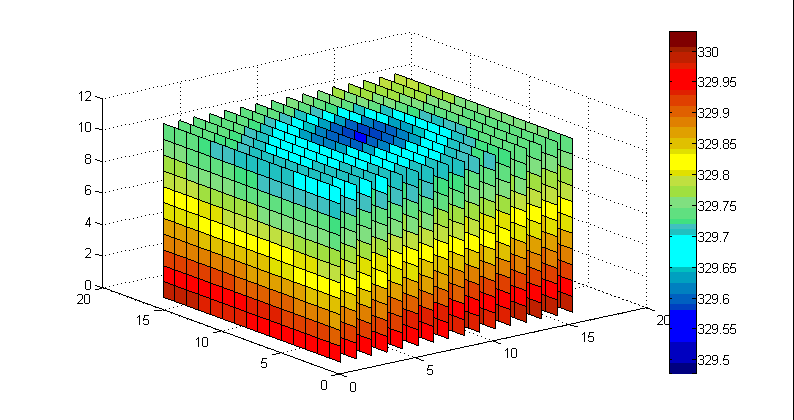
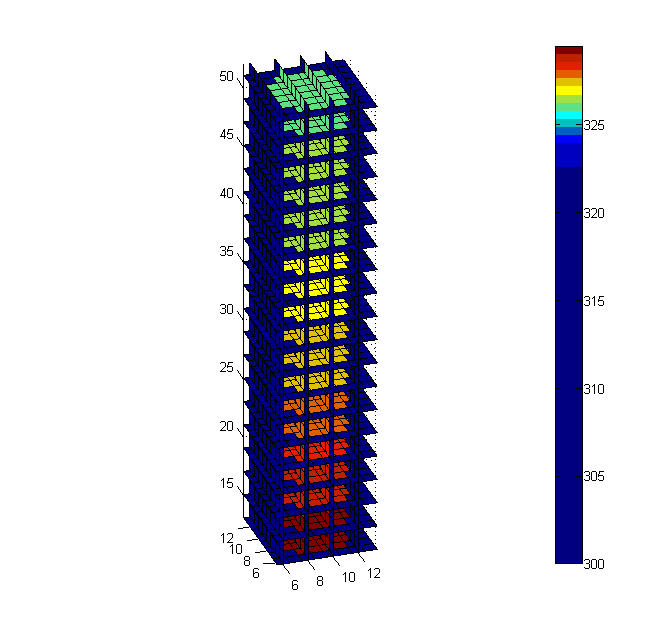
Total base area = 256E-6 m^2

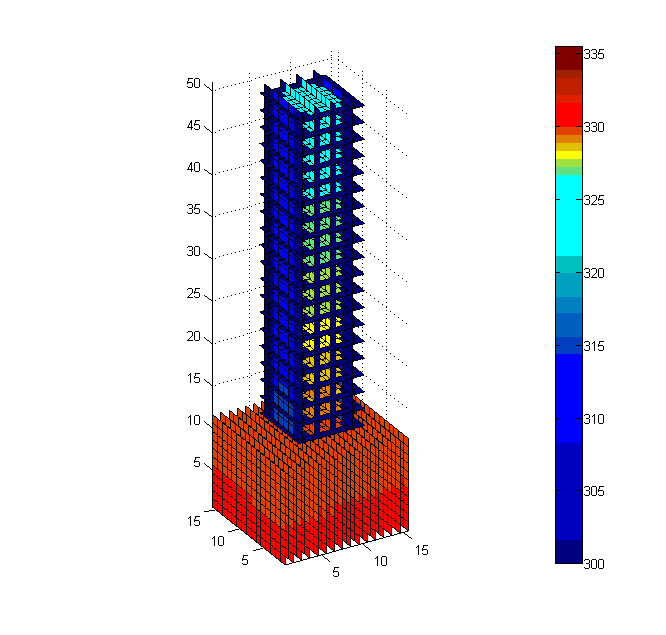
The face area depends on the unit type and face direction.

**PROCEDURE:**

1. Create the array for the coefficients of the equations.
2. Set the value of coefficients based on finite difference theorem and conservation rule.
3. Find the solution using matrix:
4. Plot and interpret the solution.

**WRAPPING UP AND CONCLUSIONS:**

1. The heat sink is a symmetric shape. It can be divided into 8 identical parts. But for this problem, I choose to divide it into four identical parts for the purpose of simplifying the analysis and modelling.
2. Code details is attached at the end.
3. Here are the plotting of the temperature distribution of the square base, the rod and both respectively (for one identical part). ***The unit length in the plotting is 0.5mm!***



1. The highest temperature:

[T\_max, max\_loc] = max(T);

T\_max = 330.03

max\_loc = 256

The highest temperature is 330.03K based on the assumption that the room temperature is 300K. The highest temperature is the middle point of the bottom plane of the heat sink. The conclusion make sense because that point is closest to the chip and farthest to the environment.

1. The temperature difference between the chip and the environment is around 45.66K.

**CODE DETAILS:**

%% ME 3345

%% Spring 2015

%% Term project -- Topic 2 (Heat sink problem)

%% AUTHOR: Tiancheng Gong (tgong7)

%% PURPOSE:

% Find and plot the temperature distribution of the heat sink, find

% the position with the highest temperature using numerical method.

%% ASSUMPTIONS:

% - The heat transfer coefficient is uniform.

% - The heat flux provided by the generation in the chip is uniform.

% - Steady state.

% - 3D conduction.

% - No radiation happens.

% - The size of the block unit used is .5mm\*.5mm\*.5mm. The temperature

% within in each block is uniform.

% - The temperature of the heat sink is around 300K.

% - The temperature of the infinite distance is around 300K. It is the

% difference between the environment and the chip instead of the

% temperature of environment itself that matters in this problem.

%% IMPORTANT VALUES:

% k = 237 W/m\*K;

% h = 100 W/m^2\*K;

% size = 0.5 mm = 5e(-4) m;

% Total power = 4W;

%% ANALYSIS AND METHOD:

% - Since the shape of the heat sink is symmetric, the model can be

% simplified by dividing the heat sink into 4 identical parts, 4

% squares and one rod in the middle of each square. (It can be

% divided into at most 8 identical triangle parts, but it is not

% easy to build this model using cube blocks.)

% - The tool used here is the finite difference equations. For each

% cube block unit, we can derive a finite difference equation based

% on the temperatrue of its neighbors and the property of the

% material. We will have n cubes as n unknowns as well as n

% equations, thus the equations are solvable.

% - Using the solution derived, we can thus plot the temperature

% distribution and find the position of the heighest temperature.

%% PART 1: Create the array for the coefficients of the equations.

% Linear Equations: At = B; B is all zeros.

size = .0005;

a = .008 / size;

b = .008 / size;

c = .006 / size;

l = .02 / size;

blocks\_in\_base = a \* b \* c;

blocks\_in\_rod = 32 \* l;

total\_blocks = blocks\_in\_base + blocks\_in\_rod;

A = zeros(total\_blocks, total\_blocks); % For all blocks

B = zeros(total\_blocks, 1);

% Set the value of coefficients based on finite difference theorem.

% Consider the border cubes carefully.

area = size ^ 2;

k = 237;

h = 100;

total\_power = 4;

rm\_temp = 300;

heat\_from\_chip = total\_power / (4 \*(a \* b));

% the top plane for the base 0: nothing above it; 0.5: triangle rod block

% is above it; 1: cube rod block is above it

top\_plane = zeros(a, b);

top\_plane(6:11, 7:10) = .5;

top\_plane(7:10, 6:11) = .5;

top\_plane(6:11, 8:9) = 1;

top\_plane(7:10, 7:10) = 1;

top\_plane(8:9, 6:11) = 1;

% collect all the finite difference equations of the bese part.

count = 0; % count the top blocks connecting to the rod blocks

for z = 0 : (c - 1)

for y = 0 : (b - 1)

for x = 0 : (a - 1)

cur\_index = z \* (a \* b) + y \* a + x + 1;

% left neighbor

if (x ~= 0)

left\_index = cur\_index - 1;

coe = k / size \* area;

A(cur\_index, left\_index) = - coe;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

else

coe = h \* area;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

B(cur\_index, 1) = B(cur\_index, 1) + coe \* rm\_temp;

end

% right neighbor

if (x ~= (a - 1))

right\_index = cur\_index + 1;

coe = k / size \* area;

A(cur\_index, right\_index) = - coe;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

end

% front neighbor

if (y ~= 0)

front\_index = cur\_index - a;

coe = k / size \* area;

A(cur\_index, front\_index) = - coe;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

else

coe = h \* area;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

B(cur\_index, 1) = B(cur\_index, 1) + coe \* rm\_temp;

end

% back neighbor

if (y ~= (b - 1))

back\_index = cur\_index + a;

coe = k / size \* area;

A(cur\_index, back\_index) = - coe;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

end

% bot neighbor

if (z ~= 0)

bot\_index = cur\_index - a \* b;

coe = k / size \* area;

A(cur\_index, bot\_index) = - coe;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

else

B(cur\_index, 1) = B(cur\_index, 1) + heat\_from\_chip;

end

% top neighbor

if (z ~= (c - 1))

top\_index = cur\_index + a \* b;

coe = k / size \* area;

A(cur\_index, top\_index) = - coe;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

else

% check whether the block is connected to the rod

% top neighbor is the environment

if (top\_plane(x+1, y+1) == 0)

coe = h \* area;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

B(cur\_index, 1) = B(cur\_index, 1) + coe \* rm\_temp;

% top neighbor is the cube rod block

elseif (top\_plane(x+1, y+1) == 1)

count = count + 1;

top\_index = blocks\_in\_base + count;

coe = k / size \* area;

A(cur\_index, top\_index) = - coe;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

% top neighbor is the triangle rod block

else

count = count + 1;

top\_index = blocks\_in\_base + count;

coe\_k = k / size \* area / 2;

coe\_h = h \* area / 2;

A(cur\_index, top\_index) = - coe\_k;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe\_k + coe\_h;

B(cur\_index, 1) = B(cur\_index, 1) + coe\_h \* rm\_temp;

end

end

end

end

end

count = 0;

index\_matrix = zeros(6, 6);

condition = zeros(6, 6);

for j = 1 : 6

for i = 1: 6

if (top\_plane(i + 5, j + 5) ~= 0)

condition(i, j) = top\_plane(i + 5, j + 5);

count = count + 1;

index\_matrix(i, j) = count;

end

end

end

side1 = [2, 3, 11, 16, 17, 22, 30, 31];

side2 = [1, 4, 5, 10, 23, 28, 29, 32];

for z = 1 : l

for y = 1 : 6

for x = 1 : 6

if (condition(x, y) ~= 0)

cur\_index = blocks\_in\_base + (z - 1) \* 32 + index\_matrix(x, y);

% left neighbor

if ((x ~= 1) && (condition(x - 1, y) ~= 0))

left\_index = cur\_index - 1;

coe = k / size \* area;

A(cur\_index, left\_index) = - coe;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

end

% right neighbor

if ((x ~= 6) && (condition(x + 1, y) ~= 0))

right\_index = cur\_index + 1;

coe = k / size \* area;

A(cur\_index, right\_index) = - coe;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

end

% front neighbor

if ((y ~= 1) && (condition(x, y - 1) ~= 0))

front\_index = blocks\_in\_base + (z - 1) \* 32 + index\_matrix(x, y - 1);

coe = k / size \* area;

A(cur\_index, front\_index) = - coe;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

end

% back neighbor

if ((y ~= 6) && (condition(x, y + 1) ~= 0))

back\_index = blocks\_in\_base + (z - 1) \* 32 + index\_matrix(x, y + 1);

coe = k / size \* area;

A(cur\_index, back\_index) = - coe;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

end

% top neighbor

if (z ~= l)

top\_index = cur\_index + 32;

coe = k / size \* area \* condition(x, y); % whether the cross section is a triangle

A(cur\_index, top\_index) = - coe;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

else

coe = h \* area \* condition(x, y);

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

B(cur\_index, 1) = B(cur\_index, 1) + coe \* rm\_temp;

end

% bot neighbor

if (z ~= 1)

bot\_index = cur\_index - 32;

else

bot\_index = blocks\_in\_base - a \* b + (y + 4) \* a + x + 5;

end

coe = k \* condition(x, y)/ size \* area;

A(cur\_index, bot\_index) = - coe;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

% consider the h

if (~isempty(find(side1 == index\_matrix(x, y), 1)))

coe = h \* area;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

B(cur\_index, 1) = B(cur\_index, 1) + coe \* rm\_temp;

elseif (~isempty(find(side2 == index\_matrix(x, y), 1)))

coe = h \* area \* 1.41421;

A(cur\_index, cur\_index) = A(cur\_index, cur\_index) + coe;

B(cur\_index, 1) = B(cur\_index, 1) + coe \* rm\_temp;

end

end

end

end

end

%% PART 2: Find the solution

T = A \ B;

%% PART 3: Plot and interpret the solution.

% put the temperatre soulutions into T\_plot matrix

% T\_plot = ones(a, b, c + l) \* rm\_temp;

T\_base\_plot = zeros(a, b, c);

T\_rod\_plot = ones(a, b, l) \* rm\_temp;

% put in the base T elements

for z = 1 : c

for y = 1 : b

for x = 1 : a

index = (z - 1) \* (a \* b) + (y - 1) \* a + x;

T\_base\_plot(x, y, z) = T(index);

end

end

end

% plug in the rod T elements

index = blocks\_in\_base;

for z = 1 : l

for y = 1 : b

for x = 1 : a

if (top\_plane(x, y) ~= 0)

index = index + 1;

T\_rod\_plot(x, y, z) = T(index);

end

end

end

end

figure(1)

[x, y, z] = meshgrid(0.5 : a - 0.5, .5 : b - .5, .5 : c - .5);

xslice = .5 : 1 : a - .5;

slice(x, y, z, T\_base\_plot, xslice, [], []);

colorbar;

figure(2)

[x, y, z] = meshgrid(5.5:13.5, 5.5:13.5, 13:l+12);

xslice = 6:2:13;

zslice = 13: 2: l+12;

slice(x, y, z, T\_rod\_plot(5:13, 5:13, :), xslice, [], zslice);

axis equal

colorbar;

figure(3)

[x, y, z] = meshgrid(0.5 : a - 0.5, .5 : b - .5, .5 : c - .5);

xslice = .5 : 1 : a - .5;

slice(x, y, z, T\_base\_plot, xslice, [], []);

hold

[x, y, z] = meshgrid(4.5:12.5, 4.5:12.5, 12:l+11);

xslice = 5:2:12;

zslice = 12: 2: l+11;

slice(x, y, z, T\_rod\_plot(5:13, 5:13, :), xslice, [], zslice);

axis equal

colorbar;

[T\_max, max\_loc] = max(T);

[T\_min, min\_loc] = min(T);

T\_chip = T\_max + .001 / (.008 \* .008);