Two-Dimensional Heat Transfer Calculation Software

Owen Bensel

Luke McLaughlin

David Tobin

Brendan Taedter

Submitted on 5/10/2017

ME 3360: Transport Phenomena

Dr. Michael Stoellinger

Table of Contents

Problem Statement……...1

Existing Technology……1

Contemporary Issues……2

Assumptions and Calculation Methodology……3

Results and Conclusion……4

Attachments

Attachment 1: 3-d Rendering of Wall……..5

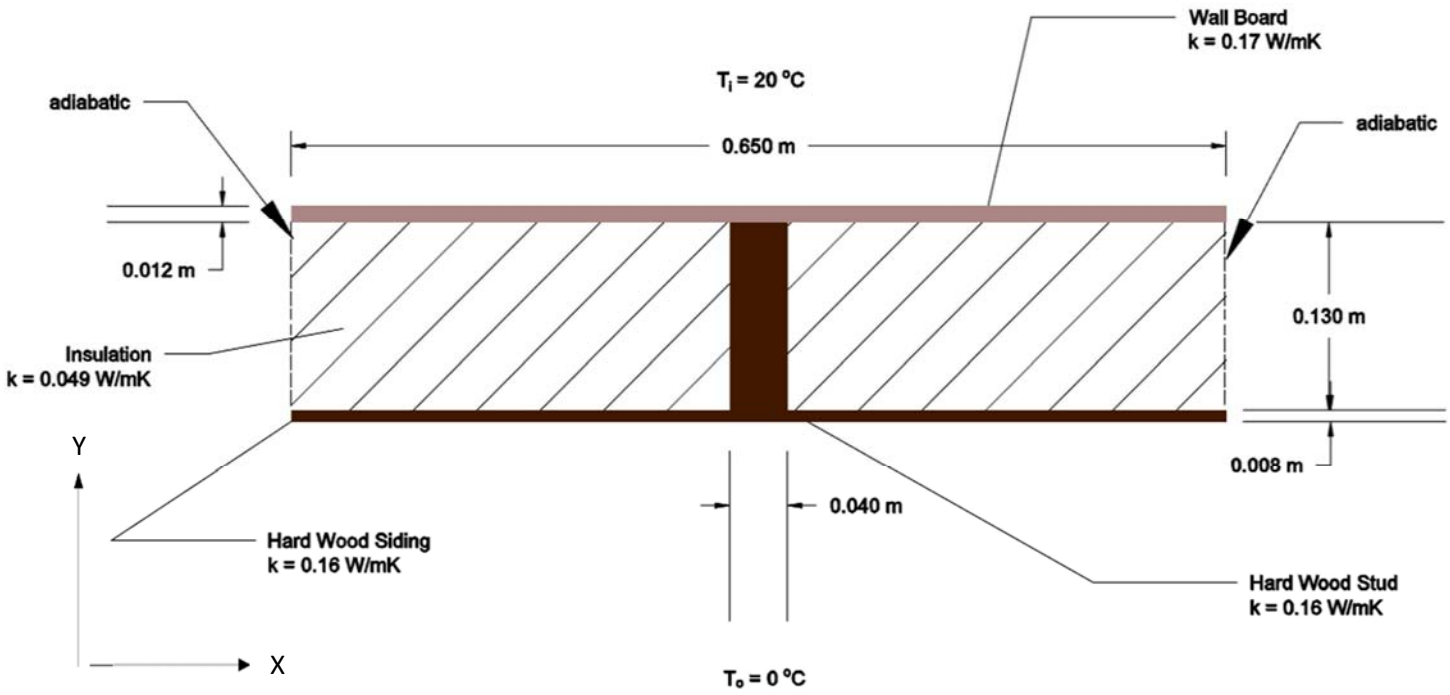
Attachment 2: Finite Volume Method Matlab Script……..6

Attachment 3: Cost Analysis Matlab Script……...20

Problem Statement

A house in Laramie, WY has a 6.5m long insulated wall that stands 2.5m tall. The exterior of the wall is made of hardwood siding with a thermal conductivity value of k = 0.16 W/mK. The wall is filled with 10 studs which are spaced evenly which are also made of hardwood with k = 0.16 W/mK. The insulation within the wall and between the studs is a fiberglass insulation initially having a density of 16 kg/m^3 with a thermal conductivity of k = 0.046 W/mK. Finally, the interior of the wall is a gypsum wallboard that has thermal conductivity of k = 0.17 W/mK. Assume that the exterior wall temperature is maintained at a constant 0^{o}C and the interior temperature of the wall is maintained at a constant 20 ^{o}C. First determine the heat transfer through the wall with the given insulation (16 kg/m3) being used. Then, re-calculate the heat transfer rate through the wall using a more expensive fiberglass insulation with a density of 40kg/m^3 and thermal conductivity k = 0.035 W/mK to see how much heat would be saved in a typical year. Perform a cost analysis to determine if the more expensive insulation would be a wise investment.

Figure 1: Top View of Wall Segment



Existing Technology

The Lawrence Berkeley National Laboratory developed a computer program very similar in theory to the one we created for this project. Their program, called THERM, models two-dimensional heat-transfer effects in building components such as windows, walls, foundations,

roofs, and doors; appliances; and other products where thermal bridges are of concern. Similar to our Matlab program, THERM's two-dimensional conduction heat-transfer analysis is based on the finite-element method, and allows the user to evaluate a product's energy efficiency and local temperature patterns.

Another two-dimensional heat transfer software is HEAT2, which differs from THERM in that it appears to better handle optimization problems like insulation fitting, floor heating systems, and heat losses from a building to the ground. HEAT2 also uses the finite-element method, and features automatic mesh generation. This program can handle up to 6,250,000 finite elements, or "nodes", as the program's developers call them. In contrast, Matlab is limited to less than 40,000 finite elements if a reasonable runtime is desired.

Contemporary Issues

Temporary Housing

Because our program is not nearly as sophisticated as THERM or HEAT2, we envision it being used for applications in which ease of use is more important than the precision of the results. In addition, our program is limited to analyzing a simple wall or similar structure. Therefore, we could see our program used for designing temporary homes or shelters for people living in underdeveloped, war-torn, or impoverished areas. These structures would not have to be built with extreme durability, efficiency, or other qualities of more permanent housing, but it would still be important to ensure they maintain a comfortable indoor temperature in their respective climates.

Reducing Thermal Bridging

We noticed in our analysis of the wall that the studs can greatly enhance heat transfer across the wall due to their heat transfer coefficient being significantly higher than that of the insulation. This is known as thermal bridging, and it can be seen pretty clearly in poorly built houses during a frost. While a wall is covered in frost, you may notice vertical lines where there is no frost at all. These lines correspond to studs in the wall transferring heat to the outside and melting the frost.

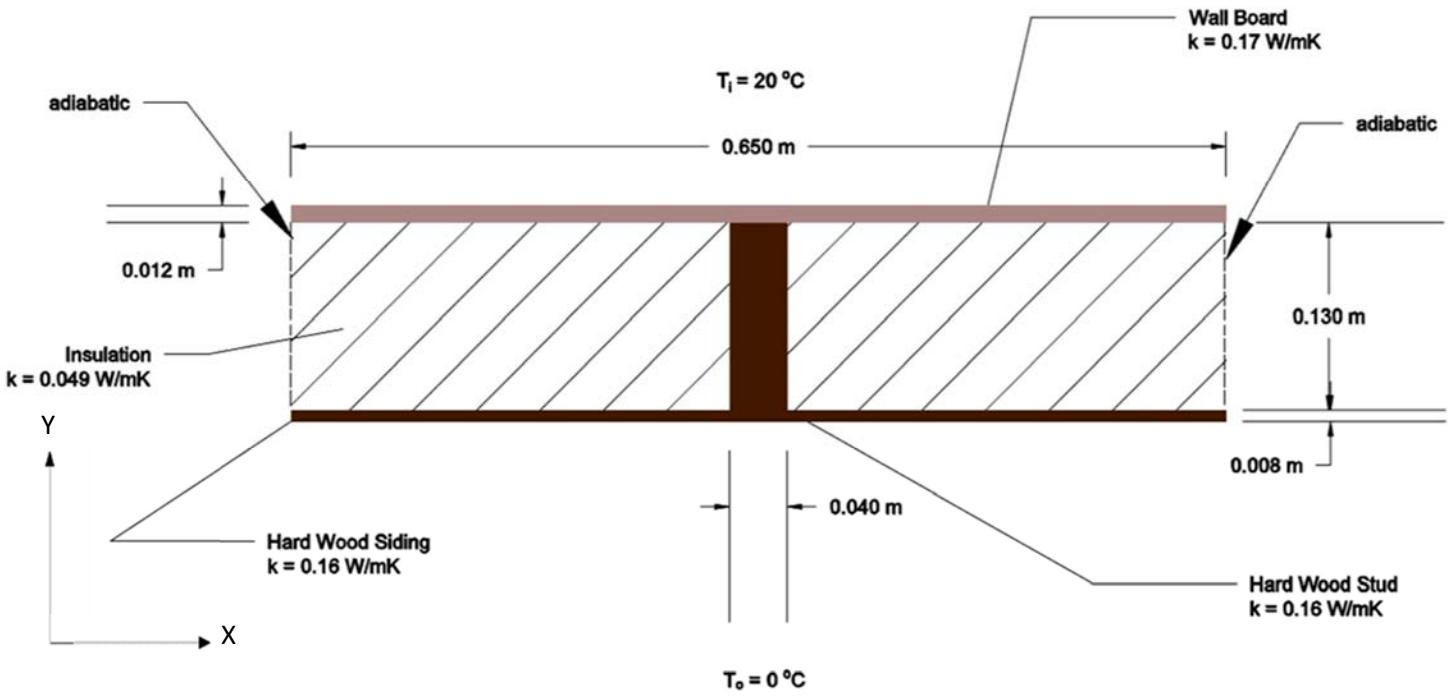
Considering that studs make up a significant portion of framed walls, this can lead to a great deal of energy losses from thermal bridging. One building method to prevent this issue is the use of structural insulated panels (SIPs). SIPs are load-bearing panels made by sandwiching dense Styrofoam insulation between two sheets of plywood and are used as a substitute for traditional framed walls. The panels are very wide and although they do still require studs to assemble, they increase the spacing of studs significantly. Traditionally studs are spaced 16 inches apart, but with SIPs, stud spacing can be increased to several feet. This greatly reduces the portion of a wall's area made up of studs, and therefore reduces thermal bridging and saves the homeowner money.

Assumptions and Calculation Methodology

Simplifying the heat transfer process down to a 2-d conduction approach, our team created a Matlab script that plots the temperature distribution throughout the insulated wall and calculates the total heat transferred through the wall. Because our analysis dealt with only one symmetrical segment of the entire length of wall, the left and right boundaries of the segment were assumed adiabatic. That is, we said the temperatures just inside the boundary are equal to the corresponding temperatures directly across and outside the boundary. Additionally, the interior and exterior wall surfaces were assumed to be at fixed temperatures of 20°C and 0°C, respectively. For the cost analysis, it is assumed that the indoor temperature is constant throughout the year, and the outside temperature is an average value over the year.

Hand calculations were initially performed using approximate circuit-resistance analysis to find the expected heat transferred through the wall. The heat transferred through the wall was calculated by hand, first for surfaces normal to the y-direction being isothermal, and then for surfaces parallel to the y-direction being adiabatic. These preliminary calculations provided us with "reasonable" answers so we were better able to judge the correctness of our Matlab program's results. The heat transfer rate through the insulated wall was then found using mesh analysis in Matlab (see **Attachment 2** for the full Matlab script). The basis of our Matlab script came from the Homework 5 script used for simpler 2-d conduction. By redefining boundary conditions, generating a new mesh, and matching each material's k value with its corresponding positions in the wall segment, the Matlab script calculates the total heat transferred through the 2-d insulated wall and plots the temperature distribution. The mesh calculates the heat transferred through an individual segment, shown in **Figure 1**. Because the entire wall is composed of ten identical segments, the total heat transferred through the wall is ten times the heat transferred through one segment.

Figure 1: Top View of Wall Segment



Results and Conclusion

The fiberglass insulation having a density of 16kg/m^3 with a thermal conductivity of k = 0.046 W/mK was used for all initial calculations of heat transfer and temperature distribution. See **Table 1** for results of initial hand calculations. Initially, our Matlab script returned an unreasonably large heat transfer value. We were able to determine that the inaccurate results were due to an inconsistency in the way the axis were defined throughout the script. After performing initial hand calculations and correcting the functionality of the Matlab script, the heat transfer through the wall was found to be 138 W. This value is deemed reasonable due to its similarity to the results of the hand calculations. The differences in the three heat transfer values are due to differences in each method. Theoretically, the most accurate result is provided by Matlab's finite volume method. The heat transfer through the wall with the better insulation was calculated simply by updating the k value of the insulation. See **Table 2** for results of Matlab calculations, and **Figure 2** for a plot of the temperature distribution in one wall segment. From **Table 2**, results show that the more expensive insulation decreases the heat transfer through the wall by 26 W, or about 19%.

To complete the cost analysis, we first determined that it costs nearly $38 more to fill this wall with the better insulation. Then, using the heat transfer values calculated with Matlab and using a rate of $0.20 per kilowatt-hour from Rocky Mountain Power, it was determined that with the better insulation, the homeowner would save $44.55 each year. Therefore, it would only take about ten months to break even, meaning the better insulation would be a wise purchase. See **Attachment 3** for the cost analysis Matlab script, which contains details of insulation prices, quantity needed, and other relevant information.

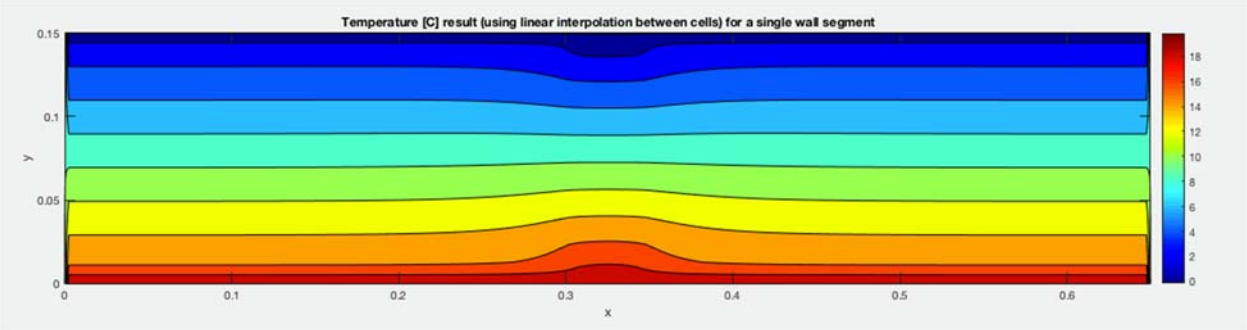
Table 1: Results of Hand Calculations

	Insulation 1 (16 kg/m^3)	Insulation 2 (40 kg/m^3)
Heat transferred through wall, (W) (Surfaces normal to Y isothermal)	126	102.7
Heat transferred through wall, (W) (Surfaces parallel to Y adiabatic)	130.1	105.3

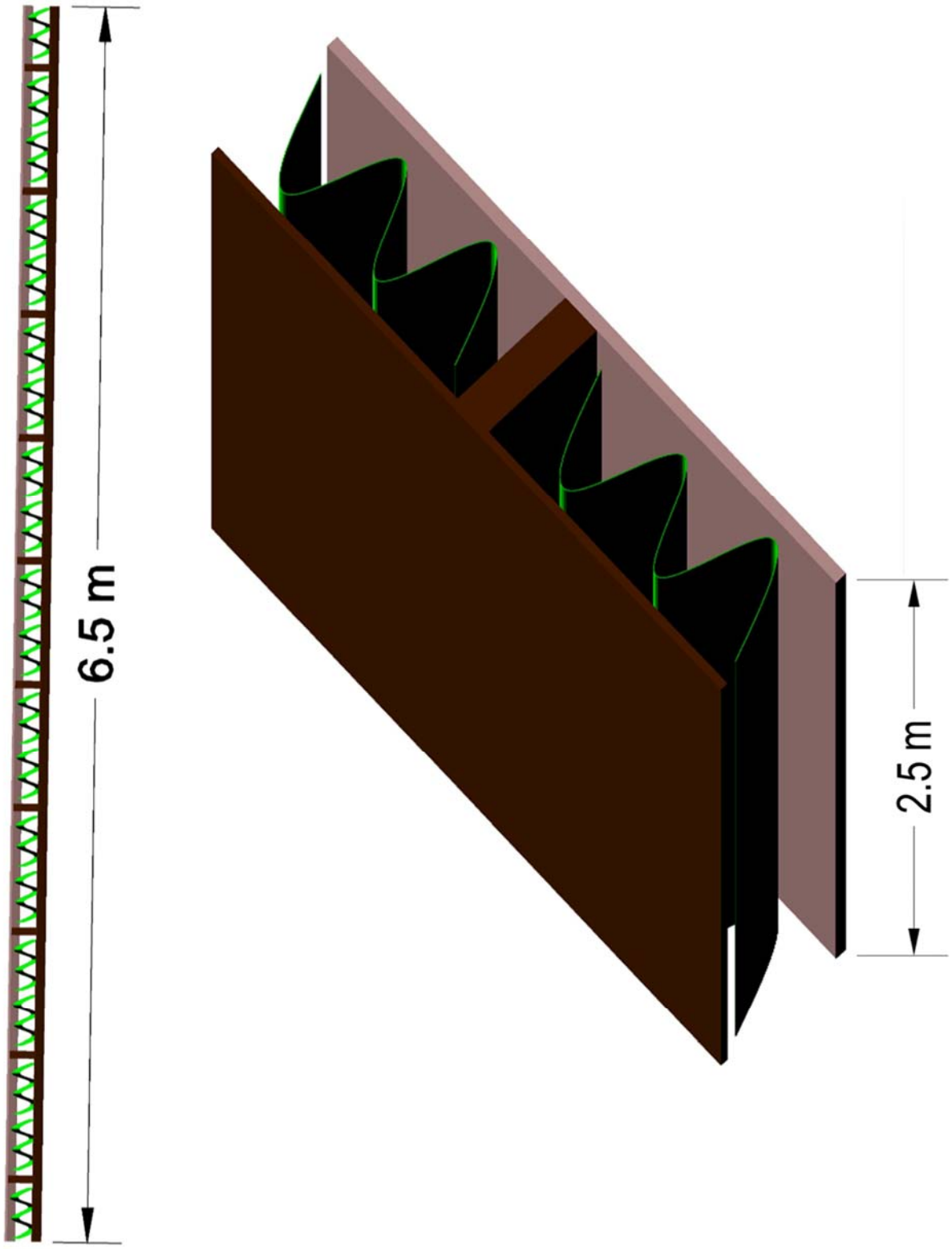
Table 2: Results of Matlab Calculations

	Insulation 1 (16 kg/m^3)	Insulation 2 (40 kg/m^3)
Heat transferred through wall, (W)	138	112

Figure 2: Matlab Plot of Temperature Distribution in Wall Segment



Attachment 1: 3-d Renderings of Wall



Attachment 2: Finite Volume Method Matlab Script

Table of Contents

TRANSPORT PHENOMENA DESIGN PROJECT 1
Input 1
Creating K Mesh's 2
Generate a computational grid using NI,NJ grid lines 4
Set up linear system 5
Boundary conditions 5
Loop over internal cells 5
Corners 6
Boundaries 7
Solve linear system 9
Find Total Q through wall 9
Postprocessing 10

TRANSPORT PHENOMENA DESIGN PROJECT

```
%2-D Heat Transfer Software
%ME3360-01
%David Tobin
%Owen Bensel
%Luke McLaughlin
%Brendan Taedter

%Script outline provied by M. Stoellinger

%Note: All units in base SI system (m, W, K, etc.) and their
%derivitives (eg. W/(m*k)) unless otherwise stated.
```

Input

```
clear
clc

% Geometry: Assume the coordinate system sits in the south-west corner
L = 0.65;
W = 0.15;

% BC's
Tb1 = 20 + 273;
Tb2 = 0 + 273;

% Heat generation term
qdot = 0;

% Number of grid lines in y-direction
Nj = 76;
yg=0:W/(Nj-1):W;

% Number of grid lines in x-direction
```

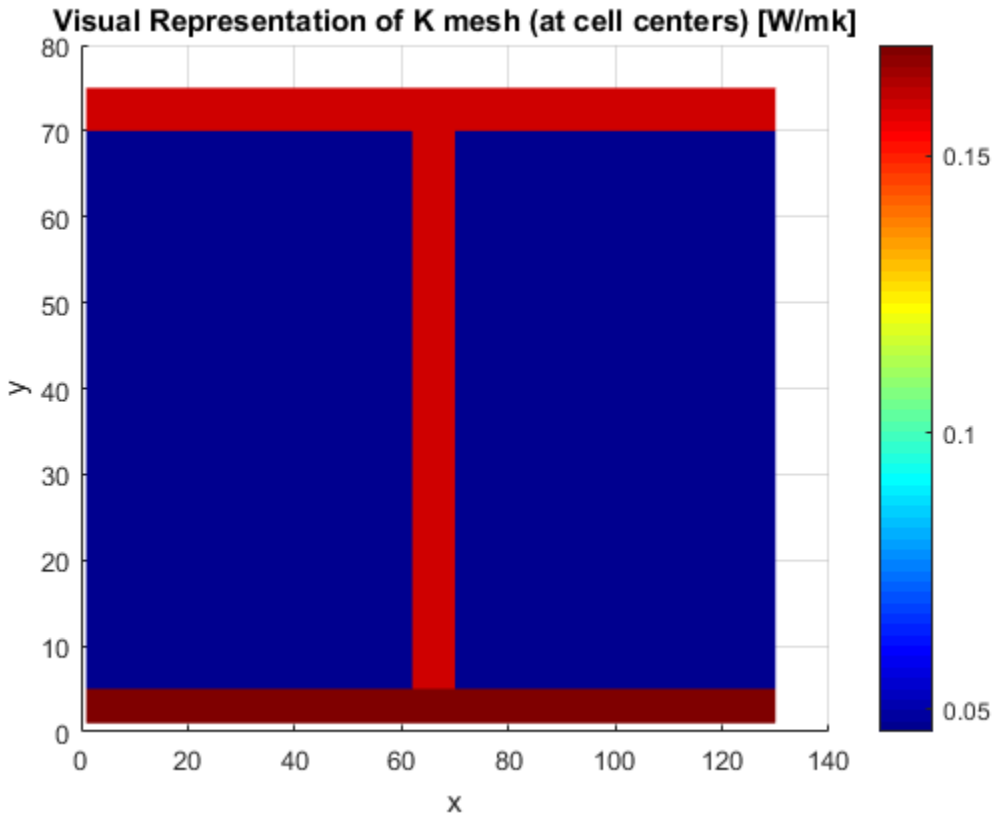
```
Ni = 131;
xg= 0:L/(Ni-1):L;
```

Creating K Mesh's

```
k_w = .16; %Wood
k_i = .046; %Insulaiton
k_g = .17; %Wallboard

%Define mesh of k values at each cell center for the section of wall.
%Each cell in the overall mesh contains only one k value.

K=zeros(Nj-1,Ni-1);

K(:,:)=k_i;
for i=7:71
    for j=62:69
        K(i,j)=k_w;
    end
end
for i=1:6
K(i,:)=k_w;
end
for i=72:75
K(i,:)=k_g;
end

%Visual Representation of K mesh
figure (1)
Z = flipud(K);
surf(Z,'EdgeColor','None');
colorbar;
xlabel('x')
ylabel('y')
title('Visual Representation of K mesh (at cell centers) [W/mk]');
colormap(jet)
view(2)

%Define 4 more meshes of k values, representing the k values at the
 north,
%south, east, and west faces, respectively.
 KNave=zeros(Nj-1,Ni-1);
 KSave=zeros(Nj-1,Ni-1);
 KEave=zeros(Nj-1,Ni-1);
 KWave=zeros(Nj-1,Ni-1);

%Internal Cells
for j = 2:Nj-2
    for i = 2:Ni-2
        %Define K values for cell and bordering cells, interpolate for
 k
        %value at cell interfaces (just take the mean since distances
        %between each cell center are the same)
```

```
        KP=K(j,i); %K of current cell
        KN=K(j-1,i); %K of North cell center
        KS=K(j+1,i); %K of south cell center
        KE=K(j,i+1); %K of east cell center
        KW=K(j,i-1); %k of west cell center

        %Finding k value at P cell boundaries
        KNave(j,i)=(KN+KP)/2;
        KSave(j,i)=(KS+KP)/2;
        KEave(j,i)=(KE+KP)/2;
        KWave(j,i)=(KW+KP)/2;
    end
end

%North Boundary
j=1;
for i = 2:Ni-2
     KNave(j,i)=K(j,i);
     KSave(j,i)=K(j,i);
     KEave(j,i)=K(j,i);
     KWave(j,i)=K(j,i);
end

%South Boundary
j=Nj-1;
for i = 2:Ni-2
     KNave(j,i)=K(j,i);
     KSave(j,i)=K(j,i);
     KEave(j,i)=K(j,i);
     KWave(j,i)=K(j,i);
end

%East Boundary
i=Ni-1;
for j = 2:74
     KNave(j,i)= (K(j,i)+K(j-1,i))/2;
     KSave(j,i)= (K(j,i)+K(j+1,i))/2;
     KEave(j,i)= K(j,i);
     KWave(j,i)= K(j,i);
end

%West Boundary
i=1;
for j = 2:74
     KNave(j,i)= (K(j,i)+K(j-1,i))/2;
     KSave(j,i)= (K(j,i)+K(j+1,i))/2;
     KEave(j,i)= K(j,i);
     KWave(j,i)= K(j,i);
end

%NW corner
     KNave(1,1)=K(1,1);
     KSave(1,1)=K(1,1);
     KEave(1,1)=K(1,1);
```

```
      KWave(1,1)=K(1,1);

%NE corner
      KNave(1,130)=K(1,130);
      KSave(1,130)=K(1,130);
      KEave(1,130)=K(1,130);
      KWave(1,130)=K(1,130);

%SW corner
      KNave(75,1)=K(75,1);
      KSave(75,1)=K(75,1);
      KEave(75,1)=K(75,1);
      KWave(75,1)=K(75,1);

%SE corner
      KNave(75,130)=K(75,130);
      KSave(75,130)=K(75,130);
      KEave(75,130)=K(75,130);
      KWave(75,130)=K(75,130);
```



Generate a computational grid using NI,NJ grid lines

```
% Define grid coordinate vectors
x = (0:L/(Ni-1):L)';
```

```
y = (0:W/(Nj-1):W)';

% The mesh defines a total Nc finite volume cells == total of Nc
 unknowns
Nc = (Ni-1)*(Nj-1);
```

Set up linear system

```
% initialize coeffcient matrix A and rhs vector Q
A = zeros(Nc,Nc);
Q = zeros(Nc,1);
```

Boundary conditions

```
% Set north boundary temperature to Tb2
TN = Tb2;
% Set south boundary temperature to Tb1
TS = Tb1;
% Set east boundary temperature to (Tb1+Tb2)/2 (only for temperature
 plot!)
TE = (Tb1+Tb2)/2;
% Set west boundary temperature to (Tb1+Tb2)/2 (only for temperature
 plot!)
TW = (Tb1+Tb2)/2;
```

Loop over internal cells

```
for j = 2:Nj-2
    for i = 2:Ni-2
        % get index of cell p
        P = (j-1) * (Ni-1)+i;
        % get cell size
        dy = xg(i+1)-xg(i);
        dx = yg(j+1)-yg(j);
        % East index and coefficient
        E = P+1;
        AE = -KEave(j,i)*dy/(dx);
        A (P,E) = AE;
        % West index and coefficient
        W = P-1;
        AW = -KWave(j,i)*dy/(dx);
        A (P,W) = AW;
        % North index and coefficient
        N = P+(Ni-1);
        AN = -KNave(j,i)*dx/(dy);
        A (P,N) = AN;
        % South index and coefficient
        S = P-(Ni-1);
        AS = -KSave(j,i)*dx/(dy);
        A (P,S) = AS;
        % Cell coefficient
        AP = -(AS+AN+AW+AE);
```

```
            A(P,P) = AP;
            %rhs vector
            Q(P) = qdot*dx*dy;
        end %i
    end %j
```

Corners

```
    % South-West corner
        %get index
        P=1;
        %West index and coefficent
           AW=0;%-k*2*dy/dx;          %BECAUSE FLUX=0!
        %North index and coefficent
           N= P+(Ni-1);
           AN=-KNave(75,1)*dx/(dy);
           A(P,N)=AN;
        %South index and coefficent
           AS=-KSave(75,1)*2*dx/dy;
        %East index and coefficient
           E=P+1;
           AE=-KEave(75,1)*dy/dx;
           A(P,E)=AE;
        %Cell Coefficient
           AP=-(AS+AN+AW+AE);
           A(P,P)=AP;
        %r.h.s.
           Q(P)=qdot*dx*dy -((Tb1*AW)+(Tb1*AS));

    % North-West corner
        %get index
        P=((Ni-1)*(Nj-1))-(Ni-2);
        %West index and coefficent
           AW=0;%-k*2*dy/dx;           %BECAUSE FLUX=0!
        %North index and coefficent
           AN=-KNave(1,1)*2*dx/dy;
        %South index and coefficent
           S= P-(Ni-1);
           AS=-KSave(1,1)*dx/(dy);
           A(P,S)=AS;
        %East index and coefficient
           E=P+1;
           AE=-KEave(1,1)*dy/dx;
           A(P,E)=AE;
        %Cell Coefficient
           AP=-(AS+AN+AW+AE);
           A(P,P)=AP;
        %r.h.s.
           Q(P)=qdot*dx*dy - ((Tb1*AW)+(Tb2*AN));

    % South-East corner
        %get index
        P=(Ni-1);
```

```
    %West index and coefficent
       W=P-1;
       AW=-KWave(75,130)*dy/dx;
       A(P,W)=AW;
    %North index and coefficent
       N= P+(Ni-1);
        AN=-KNave(75,130)*dx/(dy);
        A(P,N)=AN;
    %South index and coefficent
        AS=-KSave(75,130)*2*dx/dy;
    %East index and coefficient
       AE=0;%-k*2*dy/dx;           %BECAUSE FLUX=0!
    %Cell Coefficient
       AP=-(AS+AN+AW+AE);
       A(P,P)=AP;
    %r.h.s.
       Q(P)=qdot*dx*dy - ((Tb1*AE)+Tb1*AS);

% North-East corner
    %get index
    P= (Ni-1)*(Nj-1);
    %West index and coefficent
       W=P-1;
        AW=-KWave(1,130)*dy/dx;
        A(P,W)=AW ;
    %North index and coefficent
        AN=-KNave(1,130)*2*dx/dy;
    %South index and coefficent
        S= P-(Ni-1);
        AS=-KSave(1,130)*dx/(dy);
        A(P,S)=AS;
    %East index and coefficient
      AE=0;%-k*2*dy/dx;            %BECAUSE FLUX=0!
    %Cell Coefficient
       AP=-(AS+AN+AW+AE);
       A(P,P)=AP;
    %r.h.s.
       Q(P)=qdot*dx*dy - ((Tb1*AE)+Tb2*AN);
```

Boundaries

```
%North boundary
j=1;
for i = 2:Ni-2
    P= ((Ni-1)*(Nj-2))+i;
        %East index and coefficent
        E= P+1;
        AE= -KEave(j,i)*dy/(dx);
        A(P,E)=AE;
        %West index and coefficent
        W= P-1;
        AW=-KWave(j,i)*dy/(dx);
        A(P,W)=AW;
```

```
        %South index and coefficent
        S= P-(Ni-1);
        AS=-KSave(j,i)*dx/(dy);
        A(P,S)=AS;
        %Define AN
        AN= -KNave(j,i)*2*dx/(dy);
        % to get cell Coefficient
        AP=-(AS+AN+AW+AE);
        A(P,P)=AP;
        %r.h.s. vector
        Q(P)=qdot*dx*dy-(Tb2*AN);
end

%South boundary
j=Nj-1;
for i = 2:Ni-2
    P= i;
        %East index and coefficent
        E= P+1;
        AE=-KEave(j,i)*dy/(dx);
        A(P,E)=AE;
        %West index and coefficent
        W= P-1;
        AW=-KWave(j,i)*dy/(dx);
        A(P,W)=AW;
        %North index and coefficent
        N=P+(Ni-1);
        AN=-KNave(j,i)*dx/(dy);
        A(P,N)=AN;
        %Define AS
        AS = -KSave(j,i)*2*dx/(dy);
        % to get cell Coefficient
        AP=-(AS+AN+AW+AE);
        A(P,P)=AP;
    %r.h.s.
    Q(P)=qdot*dx*dy - (Tb1*AS);
end

% West boundary
i=1;
for j = 2:Nj-2
    P= (j-1)*(Ni-1)+i;
        %East index and coefficent
        E= P+1;
        AE=-KEave(j,i)*dy/(dx);
        A(P,E)=AE;
        %North index and coefficent
        N= P+ (Ni-1);
        AN=-KNave(j,i)*dx/(dy);
        A(P,N)=AN;
        %South index and coefficent
        S=P-(Ni-1);
        AS=-KSave(j,i)*dx/(dy);
        A(P,S)=AS;
```

```
        %Define AW
        AW= 0;%-k*2*dy/(dx);          %BECAUSE FLUX=0!
        % to get cell Coefficient
        AP=-(AS+AN+AW+AE);
        A(P,P)=AP;
    %r.h.s.
    Q(P)=qdot*dx*dy - (Tb1*AW);
end

% East boundary
i=Ni-1;
for j = 2:Nj-2
    P= (j)*(Ni-1);
        %West index and coefficent
        W= P-1;
        AW=-KWave(j,i)*dy/(dx);
        A(P,W)=AW;
        %North index and coefficent
        N= P+ (Ni-1);
        AN=-KNave(j,i)*dx/(dy);
        A(P,N)=AN;
        %South index and coefficent
        S=P-(Ni-1);
        AS=-KSave(j,i)*dx/(dy);
        A(P,S)=AS;
       %Define AE
       AE=0;% -k*2*dy/(dx);         %BECAUSE FLUX=0!
       % to get cell Coefficient
       AP=-(AS+AN+AW+AE);
       A(P,P)=AP;
    %r.h.s.
    Q(P)=qdot*dx*dy - (Tb1*AE);
end
```

Solve linear system

```
T = linsolve(A,Q);
```

Find Total Q through wall

```
% Process/explanation:
% 1. Find the heat flux through each individual cell on the bottom
 row of the mesh (inner edge of wall) with q" = -k_g (T.cell - Tb1)/
(dy/2).
% 2. Find the heat flow through each individual cell on the bottom row
 of the the mesh (inner edge of the wall) with q = q" * dx * (height
 of wall).
% 3. Find the total heat flow through the entire bottom edge of the
 wall with q.total = sum(q) * 10 (all bottom cells on all 10 wall
 segments.)
% 4. Repeat 1-3 for the top row of the mesh (outer edge of the wall)
 to check that the heat flow through the top edge of the wall is
 (approximately) equal to the heat flow through the bottomw edge
```

```
 of the wall (because heat flux through the horizontal (east/west)
 boundaries = 0!).

h_wall = 2.5; %Height of wall

Q_bottomrow = zeros(1,Ni-1);
for i=1:Ni-1
    Q_bottomrow (i) = -k_g*(2*(T(i)-Tb1)/dy)*dx*h_wall;
end

Q_TOTAL_bottom = (sum(Q_bottomrow))*10;

Q_toprow = zeros(1,Ni-1);
for i=1:Ni-1
    Q_toprow (i) = k_w*(2*(T((Ni-1)*(Nj-2)+i)-Tb2)/dy)*dx*h_wall;
end

Q_TOTAL_top = (sum(Q_toprow))*10;

%Rounds Final Results to nearest integer
Q_TOTAL_bottom_rounded = int16(round (Q_TOTAL_bottom));
Q_TOTAL_top_rounded = int16(round (Q_TOTAL_top));

%Display Final Results
fprintf('The total heat flow through the inner surface of the wall is
 %d Watts.\n',Q_TOTAL_bottom_rounded)
fprintf('The total heat flow through the outer surface of the wall is
 %d Watts.\n',Q_TOTAL_top_rounded)
```

```
The total heat flow through the inner surface of the wall is 138
 Watts.
The total heat flow through the outer surface of the wall is 138
 Watts.
```

Postprocessing

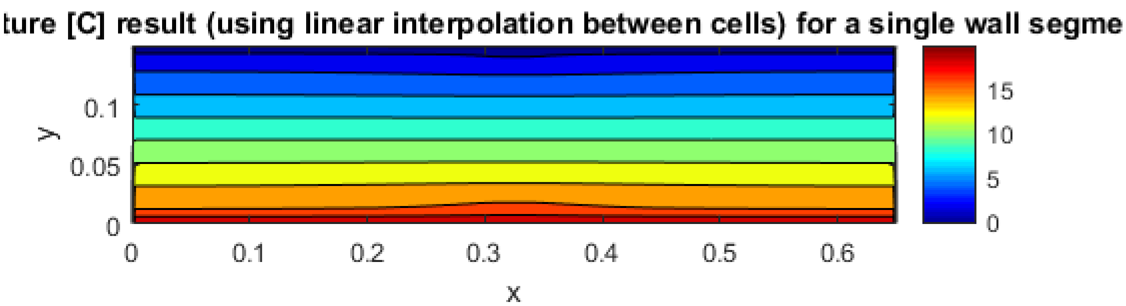
```
% Plot the computaional mesh (this works well only for small meshes NI
 < 20)
%fdrawmesh(x,y);


% % Plot the FV-solution (using the cell centered values, the boundary
 values
% % and linear interpolation in between)
PlotFVsol(x,y,T,TS,TN,TE,TW);

% % Plot the exact solution
%PlotEXACTsol(x,y,Tb1,Tb2)

%end % main function

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
```

```
%{
function fdrawmesh(x,y)
 Description:  Draws a structured 2D finite volume mesh

NI= size(x,1);
NJ= size(y,1);

[X,Y] = meshgrid(x,y);

figure('Name','Mesh');
hold on;
 plot horizontal gid lines
for j = 1:NJ
    plot(X(j,:),Y(j,:),'k-','LineWidth',1);
end
 plot vertical gid lines
for i = 1:NI
    plot(X(:,i),Y(:,i),'k-','LineWidth',1);
end

title('computational mesh')
xlabel('x')
ylabel('y')
axis equal;

 put cell centers and index in the plot
for i=1:NI-1
    for j=1:NJ-1
        l = (j-1)*(NI-1) + i;
        % cell center coordinates
        xp = (x(i+1)+x(i))/2;
        yp = (y(j+1)+y(j))/2;
        plot(xp,yp,'ko','MarkerSize',2)
        % put the label text there
        tx = xp + (x(i+1)-x(i))/20;
        ty = yp + (y(j+1)-y(j))/20;
        text(tx,ty,int2str(l),'FontSize',10);
    end
end
hold off;

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function PlotFVsol(x,y,Tc,TS,TN,TE,TW)
% Description:  Contour plot of temperature solution

NI= size(x,1);
NJ= size(y,1);
```

```
% Reconstruct cell ceter locations
xc = zeros(NI-1,1);
yc = zeros(NJ-1,1);
for i = 1:NI-1
    xc(i) = (x(i+1)+x(i))/2;
end
for j = 1:NJ-1
    yc(j) = (y(j+1)+y(j))/2;
end

% Add the boundary coordinates
xc = [x(1);xc;x(NI)];
yc = [y(1);yc;y(NJ)];


% FV-solution as 2-d array
Tsol = zeros(NI+1,NJ+1);

% Insert boundary conditions

% south
Tsol(:,1) = TS;
% north
Tsol(:,NJ+1) = TN;
% west
Tsol(1,:) = TW;
% east
Tsol(NI+1,:) = TE;

for i = 1:NI-1
    for j = 1:NJ-1
        % Corresponding cell center solution index
        l = (j-1)*(NI-1) + i;
        Tsol(i+1,j+1) = Tc(l);
    end
end

% Create matlab plotting grid based on cell centers
[X,Y] = meshgrid(xc,yc);
X = X';
Y = Y';

figure (2) %('Name','FV temperature contours');
% Plot the contour lines
contourf(X,Y,Tsol-273.15);
colormap(jet);
colorbar;
xlabel('x')
ylabel('y')
axis equal;

Tmax = max(max(Tsol));
Tmin = min(min(Tsol));
caxis([Tmin-273.15 Tmax-273.15])
```

```
title('Temperature [C] result (using linear interpolation between
 cells) for a single wall segment');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%{
function PlotEXACTsol(x,y,Tb1,Tb2)

% Number of terms used in the series expansion exact solution
Ne = 200;

NI= size(x,1);
NJ= size(y,1);

L = x(NI)-x(1);
W = y(NJ)-y(1);

% Reconstruct cell ceter locations
xc = zeros(NI-1,1);
yc = zeros(NJ-1,1);
for i = 1:NI-1
    xc(i) = (x(i+1)+x(i))/2;
end
for j = 1:NJ-1
    yc(j) = (y(j+1)+y(j))/2;
end

% Add the boundary coordinates
xc = [x(1);xc;x(NI)];
yc = [y(1);yc;y(NJ)];

% Create matlab plotting grid based on cell centers and boundaries
[X,Y] = meshgrid(xc,yc);
X = X';
Y = Y';
% Exact solution as 2-d array
theta = zeros(NI+1,NJ+1);
for i = 1:NI+1
    for j = 1:NJ+1
        % Loop to calculate the summation over the Ne-terms of the
 series
        for n = 1:Ne
            theta(i,j) = theta(i,j) + 2/pi * ((-1)^(n+1)+1)/n *
 sin(n*pi*X(i,j)/L)...
                * sinh(n*pi*Y(i,j)/L) / sinh(n*pi*W/L);
        end
    end
end


figure('Name','Exact solution on FV grid');
```

```
% Plot the contour lines
contourf(X,Y,theta*(Tb2-Tb1)+Tb1-273.15);
colormap(jet);
colorbar;
xlabel('x')
ylabel('y')
axis equal;
title('Exact Temperature [C] Distribution on FV grid');
end
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
```



Published with MATLAB® R2016b

Attachment 3: Cost Analysis Matlab Script

```
%Cost Analysis

%This script will calculate the amount of money saved in a year by using
%higher density insulation (40kg/m^3 rather than 16 kg/m^3). To determine
%insulation "R-values", it was first helpful to convert metric densities
%to English units. Then, using Home Depot's website, the densities of
%different R-values were found by dividing the listed weight of a roll of
%each R-value by its calculated volume. Doing this, it was seen that the
%given low density insulation corresponds to "R13", and the higher density
%insulation corresponds to "R15".

clc
clear
format bank
format compact

q1=138; %Total heat lost using lower density insulation, obtained
        %from main script (W)(J/s)
q2=112; %Total heat lost using higher density insulation, obtained
        %from main script (W)(J/s)

L=6.5;  %Length of wall, (m)

t=365*24*60*60; %Number of seconds in a 365-day year (s)

V=(L-(L/.65)*.04)*.13*2.5*35.315; %Total volume of insulation needed,(ft^3)

R13_cost=2.08;  %Cost of R13 insulation, dollars per cubic foot
R15_cost=2.62;   %Cost of R15 insulation, dollars per cubic foot

%http://www.homedepot.com/b/Building-Materials-Insulation-Fiberglass/Faced/
%13/15/N-5yc1vZbay7Z1z0r65uZ1z0zyamZ1z0zybk?NCNI-5

R13_total=R13_cost*V;   %Cost to fill wall with R13
R15_total=R15_cost*V;   %Cost to fill wall with R15

disp('Cost to fill this wall with R15 rather than R13 (dollars):')
Cost_diff=R15_total-R13_total;  %How much more it costs to fill wall with
                                %R15 than R13
disp(Cost_diff)


E_lost1=q1*t*2.7777778*10^-7;   %Energy lost through R13 in one year, (kWh)
E_lost2=q2*t*2.7777778*10^-7;   %Energy lost through R15 in one year, (kWh)


Esaved=E_lost1-E_lost2;   %Energy saved over one year by using R15 instead
                            %of R13, (kWh)

E_cost=.20;     %Cost of energy per kWh, Rocky Mtn Power (dollars per kWh)

disp('The money saved from energy savings, in dollars per year:')
Savings=E_cost*Esaved;    %Money saved, dollars per year
disp(Savings)
```

```
disp('Years required to break even:')
Break_even=Cost_diff/Savings;   %How long it will take to save the amount
                          %of money spent by using the better
                          %insulation,(years)
disp(Break_even)
```

```
Cost to fill this wall with R15 rather than R13 (dollars):
         37.81
The money saved from energy savings, in dollars per year:
         45.55
Years required to break even:
          0.83
```

Published with MATLAB® R2016b