Executive Summary

The purpose of this project was to design the shape of the top side of a heat exchanger so that it would produce the maximum heat flux per unit length. The bottom side is flat, the minimum and maximum thickness are 1cm and 5cm.

To determine the design of the top side of the heat exchanger, a parameterization of the height was made. It follows a sine wave with variables of initial height, amplitude, and frequency. Then, with the height function, heat flux calculations were made through Matlab.

The function fmincon() was utilized to find the height function at which the reciprocal of the produced heat flux is at a minimum. It was determined that the optimal height function would be h=3+2*sin(3pi*x/5) producing a heat flux of 5.3451*10^3 W/m^2.

Analysis Method

The analysis is achieved by first modeling the flow of energy, then approximating the heat equation through finite volume discretion. This would produce linear system of equations which can be solved to find the temperature at the center of each volume, which then allows for the calculation of the heat flux. Finally, with the height equation, the optimization can be run to find the maximum heat flux per unit length.

++

Assumptions and limitations

- The section to be designed is two-dimensional and repeats horizontally.
- The air-side shape is a function of horizontal distance.
- The thickness of the heat exchanger is at least 1 cm and at most 5 cm.

Parameterization

The system is parameterized by the variables of the height function. Since the top side of the system is likely symmetric and its derivative is likely continuous. It is assumed that the height function can be approximated by a sine function h=a1+a2*sin(a3*x).

The parameters a1, a2, and a3 are initial height, amplitude, and frequency respectively.

Optimization

The objective of the optimization is to maximize the heat flux of the heat exchanger, and its constraint is the thickness of it: The top side of the heat exchanger must be at least 1 cm and at most 5 cm from the bottom. The objective function of the problem is the reciprocal of the heat flux, which can be represented with the parameters of the height function.

The function fmincon() in MATLAB was utilized. Given the parameter inequalities, this function finds the minimum value of the objective function and its parameters through many iterations. For this problem, the used function is shown below:

```
[a,fval]= fmincon(fun,a0,Aineq,bineq,[],[],[1,0,0],[5,2,],[],options)
```

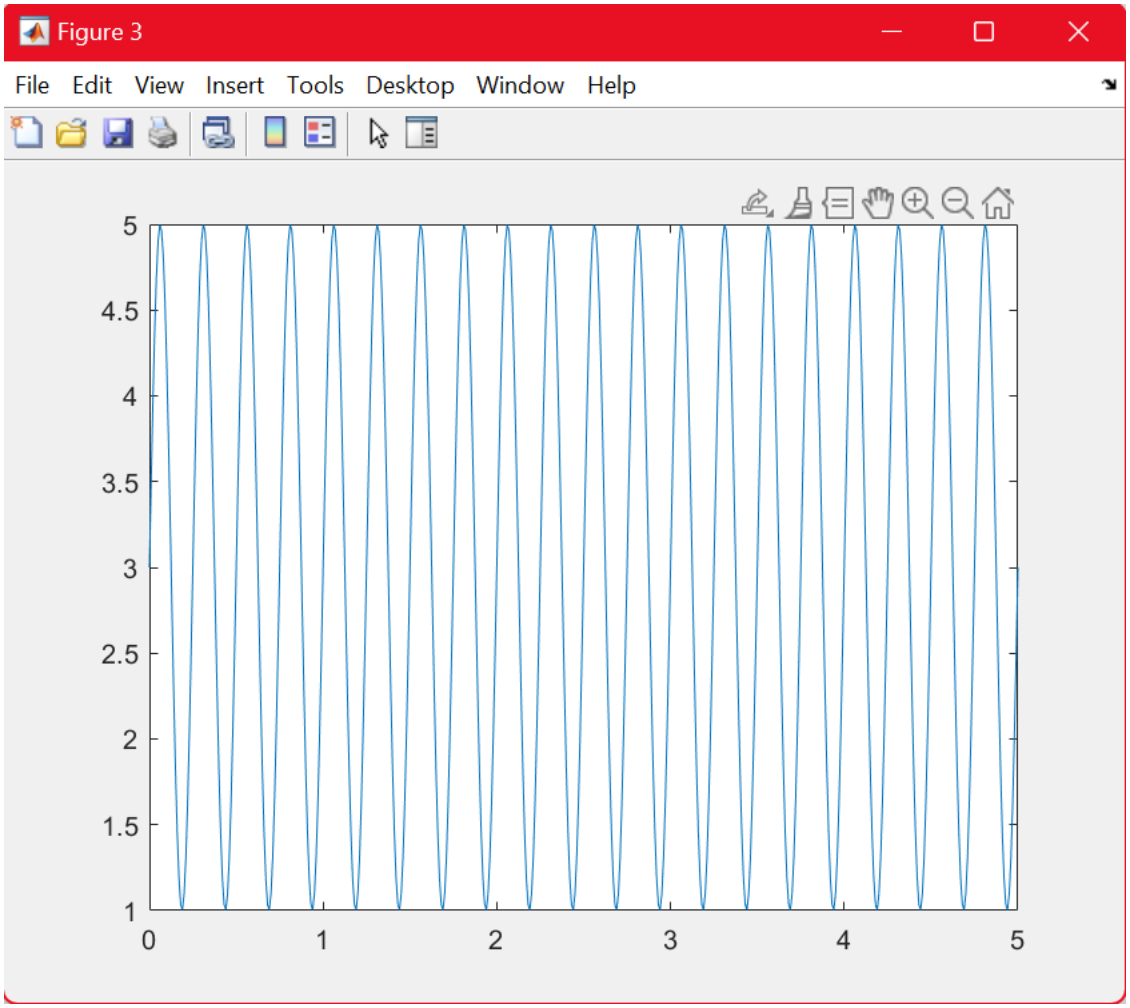
This takes in the objective function (1/flux), the initial parameters (an array of [$a1_0$, $a2_0$, $a3_0$]=[2,1,50]), the inequality matrix Aineq=$\begin{matrix}-1 & 1 & 0\\ 1 & 1 & 0\end{matrix}$, the vector matrix bineq=$\begin{matrix}-1\\ 5\end{matrix}$, and the upper and lower bounds of the parameters.

The inequalities ensure that a1+a2<5 and a1-a2>1, the upper and lower bounds ensure that 1<=a1<=5 and the maximum amplitude doesn't exceed 2. These combined satisfy the parameter constraints.

This function returns the optimal a1 and a2 values while not changing a3 much from its initial parameterization $a3_0$. Therefore, a second optimization was needed.

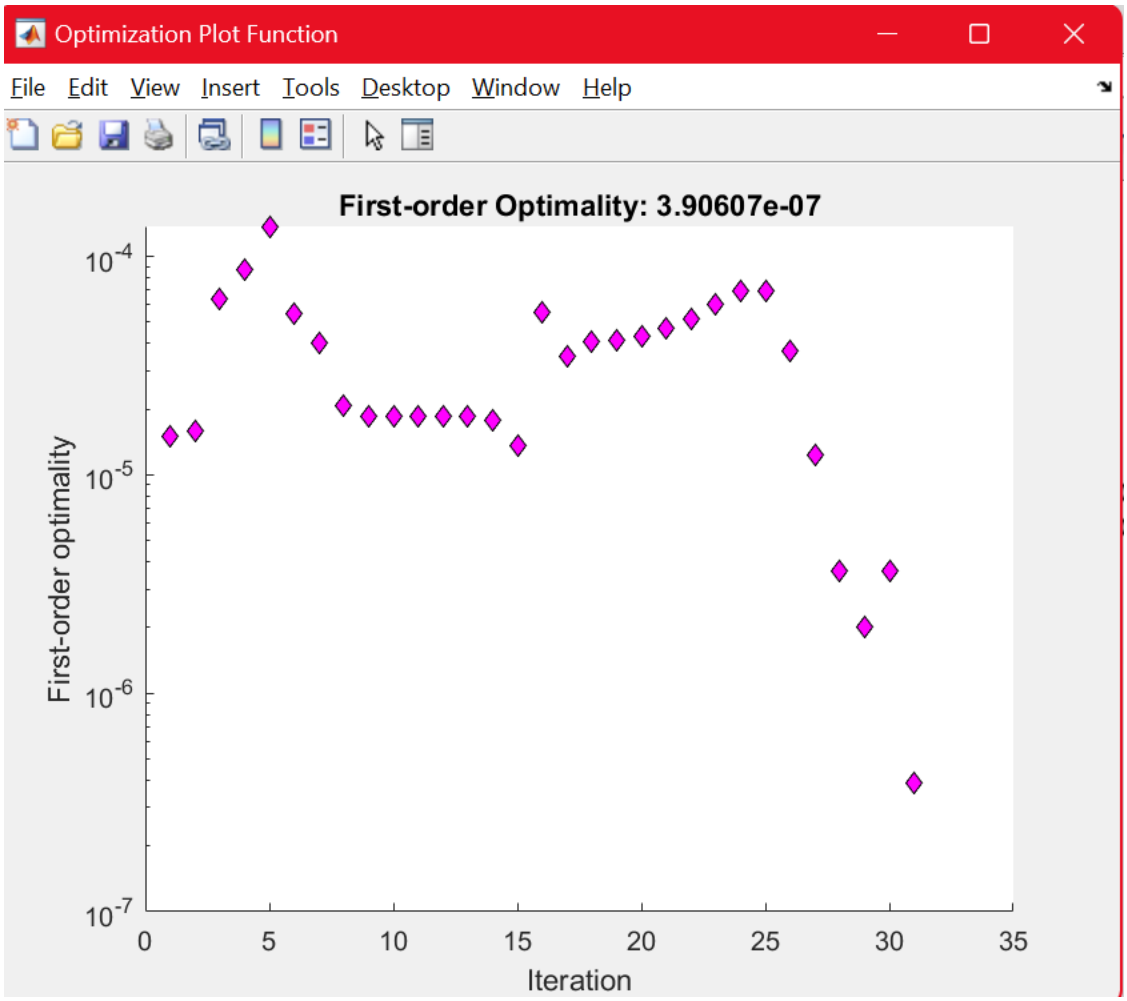
To find the optimal frequency a3, a simple comparison code was implemented. Using the found a1 and a2 values, the heat flux is expected to have a local maximizer for a3>0. Therefore by comparing a3 values, the maximum heat flux can be found.

Results and discussion



Graph 1. The shape of the air side

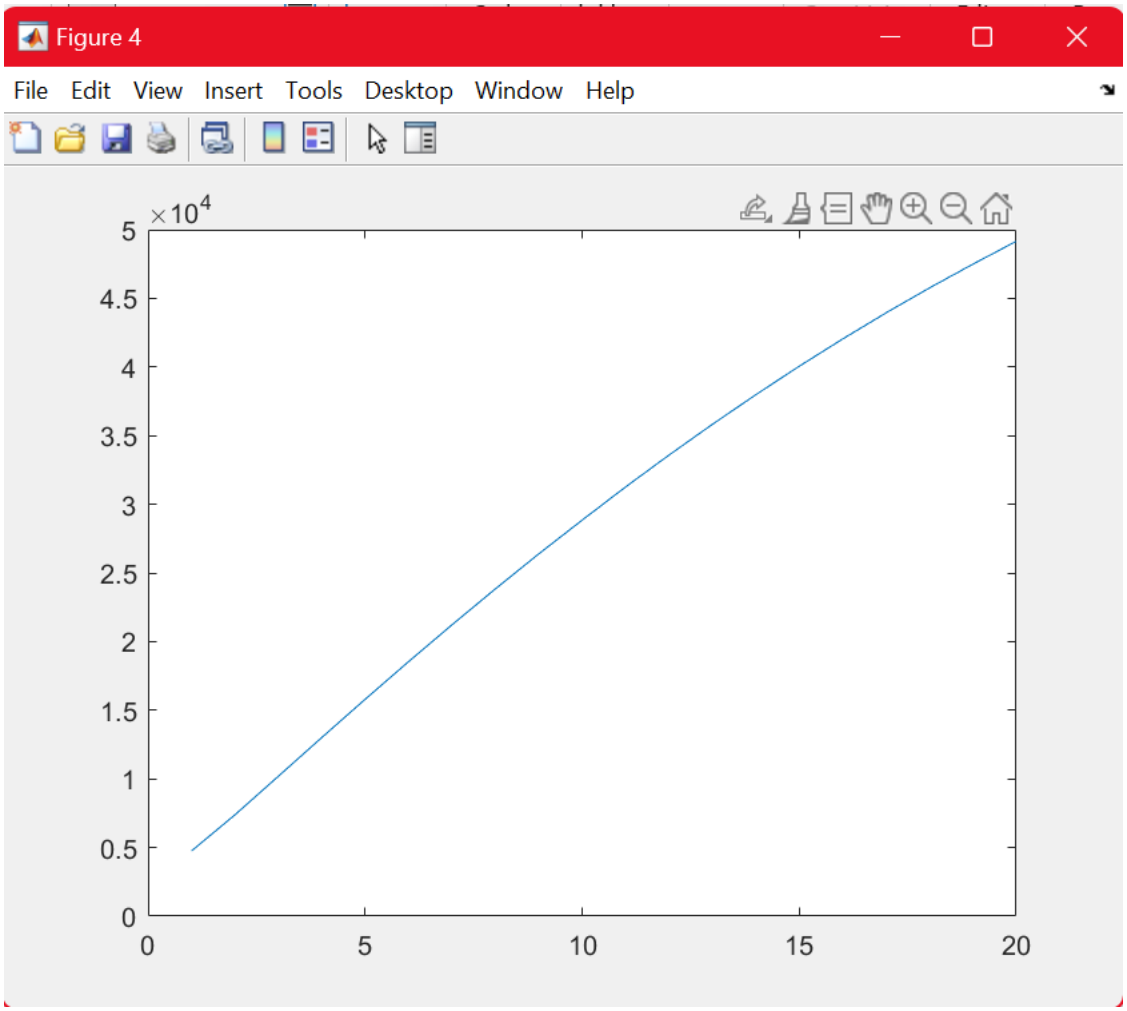
At Nx = Ny = 100, the resulting array a from the fmincon() function was (2.9463 1.9356 4.8987). The resulting frequency value was 8 pi. Therefore the resulting array of parameters is (2.9463, 1.9356, 8pi). The maximum heat flux found is 4.9158*10^4 W/m.



Graph 2. First order optimality of each iteration

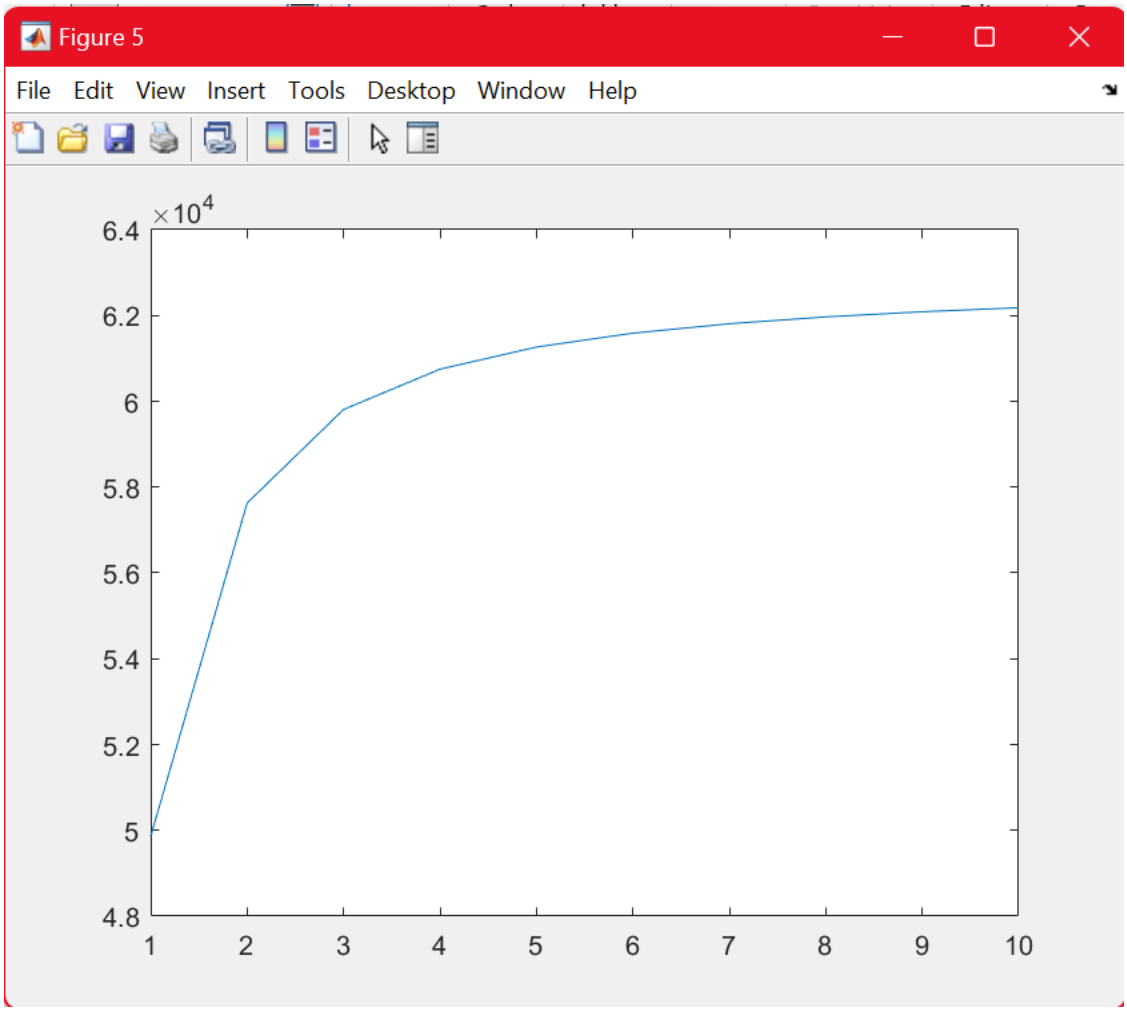
The initial array from fmincon() seemed to have a1 approaching 3 and a2 approaching 2. This makes sense as that would create the largest sweep possible, making the heat flux at maximum with these 2 parameters.

At Nx=Ny=100, when the frequency w becomes larger than 20, the height function becomes inaccurate, therefore for the frequency was obtained from the largest flux values between w=1 and w=20. It is shown that flux scales with frequency somewhat linearly, however, real world applications would not allow for high frequencies due to both cost and feasibility. At high frequencies, it would become a layer of material with a 4cm thickness.



Graph 3. Flux with respect to frequency

With an ideal height function of h=3+2*(8*pi*x), when Nx=Ny=100, the heat flux =4.9856*10^4 W/m. However this is not the most accurate flux value as this is computed through volume sections, the more sections there are, the closer to the true value it becomes. Below is a graph showing the heat flux at different Nx and Ny values.



Graph 4. Flux with respect to Nx/Ny

Appendix

Runopt.m

```
L=5;
kappa=20;
Ttop=20;
Tbot=90;
Nx=100;
Ny=100;

%setup the inequality matrix and vectors for fmincon, determine parameters
%a1 and a2
Aineq=zeros(2,3);
bineq=zeros(2,1);
Aineq(1,1)=-1;
Aineq(2,1)=1;
Aineq(1,2)=1;
Aineq(2,2)=1;
bineq(2,1)=5;
bineq(1,1)=-1;



fun = @(a) obj(a,L,Nx,Ny, kappa, Ttop, Tbot);
a0=[2,1,5];
options=optimset('Display','iter','plotfcns','optimplotfirstorderopt');
[a,fval]= fmincon(fun,a0,Aineq,bineq,[],[],[1,-2,0],[5,2,],[],options);

set(gca, 'YScale', 'log');


h=calcheight(a,Nx,L);



%determine the parameter a3
fc=0;
farr=[];
for k=1:20

    hf=calcheight([a(1),a(2),k],Nx,L);


    fa=CalcFlux(L, hf, Nx, Ny, kappa, Ttop, Tbot);

    farr(k)=fa;

    if fa>fc
        h=hf;
        kf=k;
        fc=fa;
```

```
    end

    wf=k*2*pi/L;


end



hold on

flux= CalcFlux(L, h, Nx, Ny, kappa, Ttop, Tbot);

%shows graph of height function
dx=L/Nx;
for i=1:Nx+1
    h(i,2)=dx*(i-1);
end
figure(3);
plot(h(:,2),h(:,1));

%shows graph of flux with relation to a3
figure(4);
plot(1:20,farr);
```

Obj.m

```
%objective
function [f] = obj(a,L,Nx,Ny, kappa, Ttop, Tbot)

h=calcheight(a,Nx,L);

[flux,T,dTdx,xy] = CalcFlux(L, h, Nx, Ny, kappa, Ttop, Tbot);

f=1/flux;
end
```

calcheight.m

```
%height
function [h] = calcheight(a,Nx,L)
% a is an array of initial height, amplitude and frequency of sin function
h=[];


    dx=L/Nx;
    for i=1:Nx+1
        x=(i-1)*dx;
        h(i,1)=a(1)+a(2)*sin(2*pi*a(3)*x/L);
    end
```

cleanup.m

```
%This is for general cleanup and testing

L=5;
kappa=20;
Ttop=20;
Tbot=90;
Nx0=500;
Ny0=500;

a=[3,2,20];
h=calcheight(a, Nx0, L);

flux= CalcFlux(L, h, Nx0, Ny0, kappa, Ttop, Tbot);

fluxarr=[];
for j=1:10
   Nx=j*Nx0;
   Ny=j*Nx0;
   h=calcheight(a, Nx, L);
   fluxarr(j)=CalcFlux(L, h, Nx, Ny, kappa, Ttop, Tbot);
end

%shows plot of flux with relation to Nx/Ny
figure(5);
plot(1:10,fluxarr);

dx=L/Nx0;
for i=1:Nx0+1
    h(i,2)=dx*(i-1);
end

%shows graph of height
figure(3);
plot(h(:,2),h(:,1));
```

BuildSystem.m

```
%This is for general cleanup and testing

L=5;
kappa=20;
Ttop=20;
Tbot=90;
Nx0=500;
Ny0=500;

a=[3,2,20];
h=calcheight(a, Nx0, L);

flux= CalcFlux(L, h, Nx0, Ny0, kappa, Ttop, Tbot);
```

```
fluxarr=[];
for j=1:10
   Nx=j*Nx0;
   Ny=j*Nx0;
   h=calcheight(a, Nx, L);
   fluxarr(j)=CalcFlux(L, h, Nx, Ny, kappa, Ttop, Tbot);
end

%shows plot of flux with relation to Nx/Ny
figure(5);
plot(1:10,fluxarr);

dx=L/Nx0;
for i=1:Nx0+1
    h(i,2)=dx*(i-1);
end

%shows graph of height
figure(3);
plot(h(:,2),h(:,1));
```

BuildMesh.m

```
function [xy] = BuildMesh(L, h, Nx, Ny)
% Construct a quadrilateral mesh for a domain with uneven top
% Inputs:
%   L - length of domain in x direction
%   h - height as a function of x; note that size(h,1) must be Nx+1
%   Nx - number of elements along the x direction
%   Ny - number of elements along the y direction
% Outputs:
%   xy - mesh coordinates with indices ordered (coordinate,xindex,yindex)
%--------------------------------------------------------------------------
if (size(h,1) ~= Nx+1)
    error('size(h,1) must be equal to Nx+1');
end
if (Nx <= 0) || (Ny <= 0)
    error('Nx and Ny must be strictly positive');
end
dx = L/Nx;
for i = 1:Nx+1
    dy = h(i)/Ny;
    for j = 1:Ny+1
        xy(1,i,j) = (i-1)*dx;
        xy(2,i,j) = (j-1)*dy;
    end
end

end
```

calcflux.m

```
function [flux,T,dTdx,xy] = CalcFlux(L, h, Nx, Ny, kappa, Ttop, Tbot)
% Solves for the temperature in a simple domain, and returns the heat flux
% per unit length from the water to the air
% Inputs:
%   L - length of domain in x direction
%   h - height as a function of x; note that size(h,1) must be Nx+1
%   Nx - number of elements along the x direction
%   Ny - number of elements along the y direction
%   kappa - thermal conductivity
%   Ttop - the ambient air temperature along the top of the domain
%   Tbot - the fluid temperature along the bottom of the domain
% Outputs:
%   flux - heat flux per unit length out of top/bottom boundary
%   T - temperature over the elements; note size(T) = (Nx*Ny,1)
%   dTdx - derivative of T in x direction over the elements
%   xy - mesh nodes
%
% Notes:
%   The domain has straight sides on the left and right and bottom.  The
%   boundary conditions are periodic on the left and right.  Prescribed
%   (i.e. Dirichlet) boundary conditions are applied along the top and
%   bottom.  The flux is computed along the bottom boundary.
%--------------------------------------------------------------------------
if (size(h,1) ~= Nx+1)
    error('size(h,1) must be equal to Nx+1');
end
if (Nx <= 0) || (Ny <= 0)
    error('Nx and Ny must be strictly positive');
end
% get the mesh
xy = BuildMesh(L, h, Nx, Ny);
dx = L/Nx;

% solve for the temperature and x-derivative of temperature
[A,b] = BuildSystem(L, h, Nx, Ny, kappa, Ttop, Tbot);
u = A\b;
T = u(1:Nx*Ny);
dTdx = u(Nx*Ny+1:2*Nx*Ny);

% loop over the bottom faces and compute the flux
flux = 0.0;
for i = 1:Nx
    % get dy
    dy = 0.5*((xy(2,i+1,2)-xy(2,i+1,1)) + (xy(2,i,2)-xy(2,i,1)));
    % get dydS/dy
    nydSdy = (xy(1,i+1,1) - xy(1,i,1))/dy;
    % Note: nx*dS contribution is zero
    % get the indices of the temperature
    idxT = (i-1)*Ny + 1;
    % add flux contribution
    flux = flux - kappa*nydSdy*(T(idxT) - Tbot);
end

end
```