MANE 4280: Numerical Design Optimization

Project #1: Heat Exchanger

ID #1549

I. Executive Summary

The following report outlines the processes used to calculate and analyse the heat flux across a heat exchanger, in pursuit of the optimal shape for heating air with hot water. The optimization used to create this shape assumes an ambient temperature of 20 ℃ and a water temperature of 90 ℃. In addition, the exchanger is made of a steel alloy with a thermal conductivity, K, of 20 W/mK. The heat exchanger was modeled by its two-dimensional cross section, such that the shape of the top edge of the cross section would be optimized for the best possible heat flux. Using optimization methods described in later sections, the final shape of the heat exchanger was found such that the heat flux was maximized through the section. This result was also analysed for its validity, to confirm that the best possible design was achieved.

II. Analysis Method

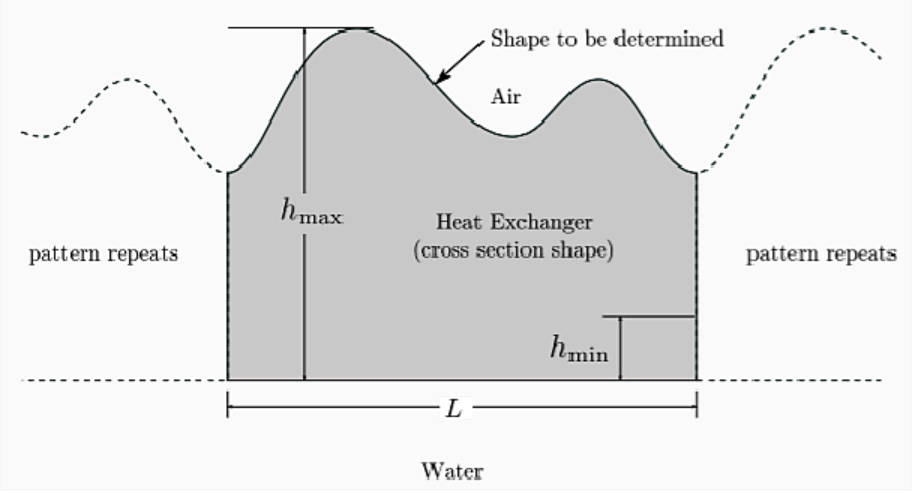

Figure1. Visualization of a heat exchanger.

The cross section of the heat exchanger is selected for one unit length of the total, as seen in Figure 1, above. The heat flux through the cross section was modeled using the steady heat equation, which was then applied to the section using Finite-Volume Discretization.

Put more plainly, the cross section was separated into "volumes" — which are ultimately areas because it is a 2-D cross section — and then the heat flux was calculated for each one. The areas were created by subdividing the height and length of the cross section, such that the length was split into equal intervals, and the height was split based on the following equation.

$$h(x) = a_1 + \sum_{k=2}^{n} a_k \sin\left(\frac{2\pi(k-1)x}{L}\right)$$

This equation helps to create the shape across the top of the cross section. In this case, x is the value across the length for which h is being calculated, and L is the unit length. The coefficients a are the design variables to be adjusted in pursuit of the shape that maximizes the heat flux. Note that the frequency of the sine wave — $2\pi(k-1)/L$ — is calculated such that the wave will always have a frequency proportional to the length. This is important for the model, because the pattern needs to be repeated across the entire heat exchanger, meaning that the end of one section must lead into the beginning of the next.

III. Optimization

Generating Results from an Initial Guess

The heat flux can be maximized through the shape across the top of the cross section. It can have any shape, provided that it remains between a maximum height of 5 centimeters, and a minimum height of 1 centimeter, as stated in the problem description. To explain it in a broad sense, the result was computed in MATLAB using *fmincon*, a function designed to find the minimum value of a design variable for a given function. In this case, *fmincon* needed to optimize the design variables — given by the a vector — to find the largest possible flux that still satisfied the constraints of the problem. Since this function automatically searches for the minimizer, the maximum flux was found by instead searching for the smallest value of 1/flux.

To start the optimization, *fmincon* was provided with an initial design, a vector of ten coefficients randomly generated such that the constraints were satisfied. Since the flux is calculated based on the discretization of the cross section, more accurate results will be gained from a number of nodes that approaches infinity. In other words, smaller areas means more accuracy. To that end, the initial guess for the *a* vector was chosen such that the length and height were divided into 400 sections each. The design variables and flux value generated for this guess were then used as a benchmark to search for the best number of nodes.

Searching for Nx and Ny

Using fixed values for the design variables, the number of nodes was increased at regular intervals, then plotted against a corresponding value for the flux. It can be seen from Figure 2, that as the nodes approach infinity, the flux approaches a value of 28,699 W/m, within a +/-1% error.

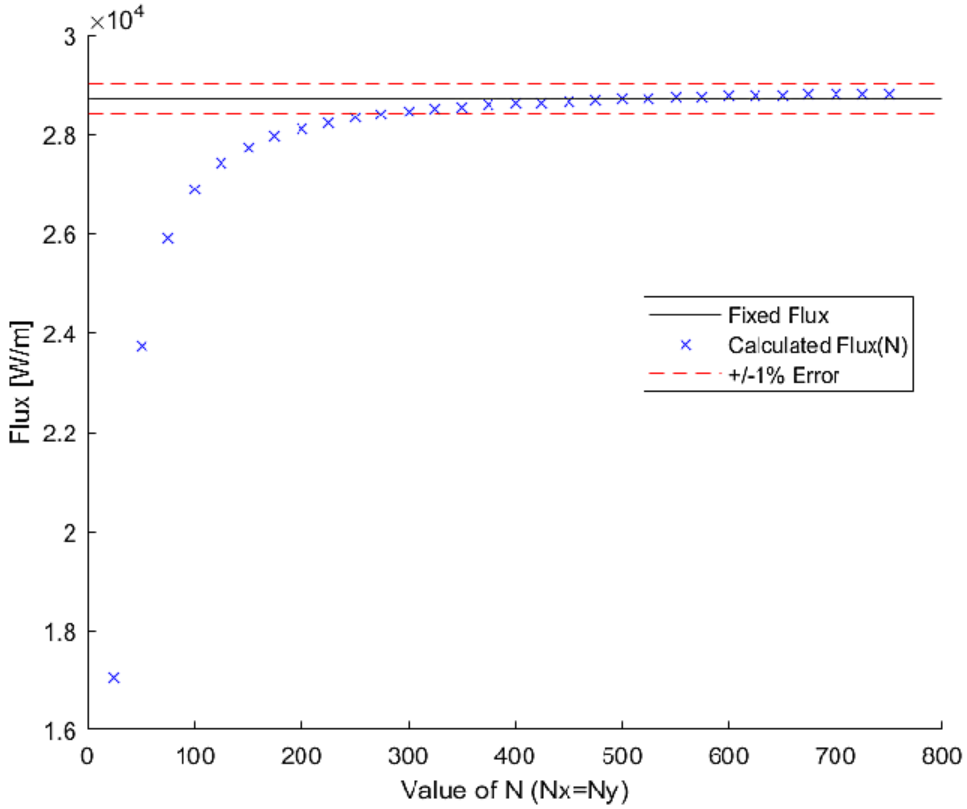

Figure 3. Accuracy of flux based on node value.

In addition to accuracy, computational cost was also considered in choosing the number of nodes. More nodes requires more computational power. Since values for Nx and Ny approach one flux value, the lowest accurate number of nodes was chosen to compute the final answer, which can be viewed in the Results section of this report.

IV. Results

Using the optimization methods outlined in the previous section, an ideal shape was generated. The final geometry is as shown in Figure 3, and yields a heat flux value of 28,699 W/m.

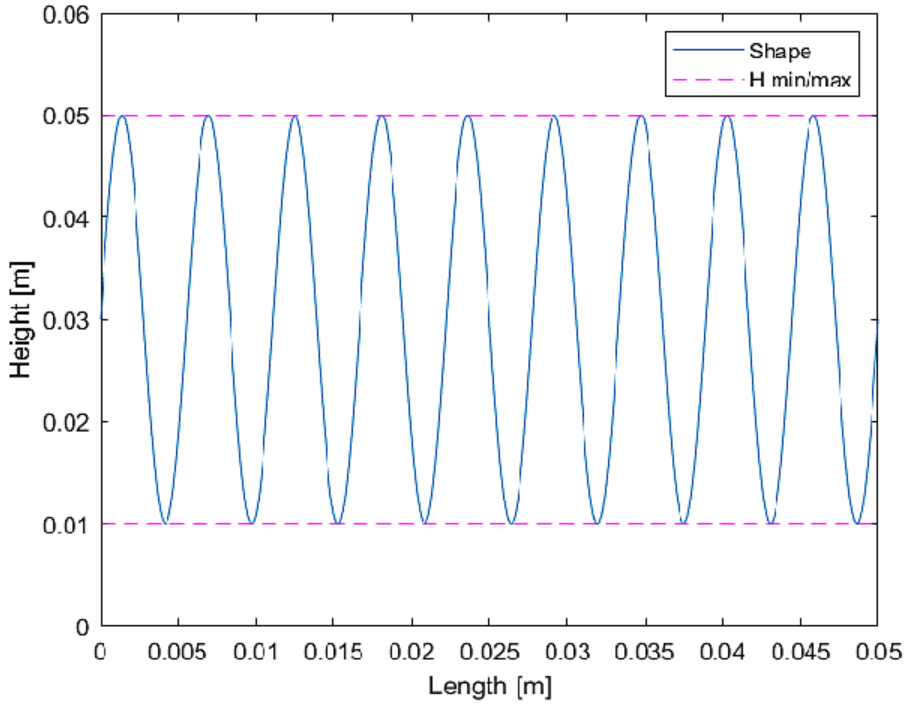

Figure 3. Optimal shape of the heat exchanger.

The shape satisfies the heat equation used to model the problem as well as the boundary constraints outlined in the problem statement. It was generated using 500 nodes in both the vertical and horizontal directions. In the following figure, the iterations of *fmincon* can be viewed, in order to gain a clearer understanding of how this result was achieved.

Figure 4. First-order optimality.

As a part of generating results, the *fmincon* function assesses the optimality of the values it generates. A given minimizer can be validated based on whether or not the optimality is decreasing over the calculations. The result for the shape above produces a flux value in ten iterations, with the final optimality approaching zero, which indicates a desirable result for the minimizer.

V. Appendix

CalcHeight.m

```
% Project 1: Parametrization of the Height
% h = a1 + sigma ( ak sin( 2pi(k-1)x / L ) )
% Function should return a vector of heights, this is the user's best
% guess for the optimized shape. It will serve as a starting point for
% the actual calculation and optimization of the flux.

function [h] = calcheight(a, nx, L)
%   a: vector of coefficients corresponding to y-values
%   nx: number of elements in the x direction
%   x: x-values used to evaluate h(x), nx+1 values
%   L: length of the cross-section, L = 0.05m
%   h: vector of heights

    x = transpose(linspace(0, L, nx+1));
    h = zeros(size(x));
    c = (2*pi)/L;           % placed in a separate variable for simplicity
    nDvar = size(a,1);

    for i = 1:(nx+1)
        % calculates h for each x value
        hx = 0;

        for k = 2:nDvar
            % adds a term for each 'a' value
            hx = hx + a(k)*sin(c*(k-1)*x(i));
        end

        h(i) = a(1)+ hx;
    end
end
```

Objective.m

```
% Project 1: Objective Function
% Combines calcfulx() and calcheight() to return the flux in a way that
% will work for fmincon() to find the minimizer.

function [f] = objective(A,Nx,Ny,L,kappa,Ttop,Tbot)
%   L: length of the cross-section, L = 0.05m
%   h: height as a function of x; note that size(h,1) must be Nx+1
%   Nx: number of elements in the x direction
%   Ny: number of elements in the y direction
%   kappa: thermal conductivity
%   Ttop: the air temperature along the top of the domain
%   Tbot: the fluid temperature along the bottom of the domain
%   f: 1/flux

    heights = calcheight(A, Nx, L);
    [flux,~,~,~] = CalcFlux(L,heights,Nx,Ny,kappa,Ttop,Tbot);
    f = 1/flux;
end


```

CalcIneq.m

```
function [Aineq,bineq] = CalcIneq(a,Nx,L)
%   Uses the contraints, Hmin = 0.00m and Hmax = 0.01m, to create the lower
%   and upper bounds for fmincon(). This is as written by Prof. Hicken,
%   modified to fit into a function.

    x = [0:L/Nx:L].';
    nvar = size(a,1); % number of design variables
    Aineq = zeros(2*(Nx+1),nvar);
    bineq = zeros(2*(Nx+1),1);

    for i = 1:(Nx+1)
      % first, the upper bound
      Aineq(i,1) = 1.0; % this coefficient corresponds to variable a_1
      for k = 2:nvar
        Aineq(i,k) = sin(2*pi*(k-1)*x(i)/L); % this coefficient corresponds to variable a_k
      end
      bineq(i) = 0.05; % the upper bound value

      % next, the lower bound; we use ptr to shift the index in Aineq and bineq
      ptr = Nx+1;
      Aineq(ptr+i,1) = -1.0; % note the negative here!!! fmincon expects inequality in a form A*x < b
      for k = 2:nvar
        Aineq(ptr+i,k) = -sin(2*pi*(k-1)*x(i)/L); % again, a negative
      end
      bineq(ptr+i) = -0.01; % negative lower bound
    end

end

```

Choose_nxny.m

```
% Project 1: Choosing Nx and Ny
% This code uses set values for vector 'a' and flux to determine the best
% possible values for Nx and Ny. These set values were chosen from a
% previously calculated and reasonable result. Note Nx = Ny = N.

%% Define Parameters
L = 0.05;
K = 20;
top = 20;
bot = 90;

%% Generating Data

a = [0.030000000000000;4.751791989641860e-06;-1.382357769919018e-18;
    1.718134174916240e-06;-6.505213034913027e-19;-1.970341003216931e-05;
    -1.897353801849633e-18;-4.578004164114673e-07;-2.412349833780247e-18;
    0.020016211951801];

nvals = (1:30)*25;                      % creates a range of N values
fvals = zeros(size(nvals));

for i = (1:30)
    h = calcheight(a, nvals(i), L);
    [f,~,~,~] = CalcFlux(L, h, nvals(i), nvals(i), K, top, bot);
    fvals(i) = f;
end

%% Plotting and Comparing Nx/Ny
% Plots the N values and their corressponding flux values. The plot should
% asymptotically approach the optimized flux value.

hold on
yline(fvals(24))                        % compared against the fixed flux
yline(fvals(24)-0.01*fvals(24), 'r--')  % error is +/-1%
yline(fvals(24)+0.01*fvals(24), 'r--')
plot(nvals, fvals, 'bx')

```

Project1.m

```
% Project 1: Shape of a Heat Exchanger
% This code will take in the design parameters and constraints outlined
% by Project 1 in order to optimize the heat flux through a heat exchanger.

clear all;
close all;
clc;

%% Define Parameters
L = 0.05;    % m
K = 20;      % W/mK
top = 20;    % C
bot = 90;    % C

%% Define Initial Design
Nx = 500;
Ny = 500;

% Lines 20 and 21 used to get consistent results during testing
% rng('default');
% rng(1);

nDvar = 10;
a0 = rand(nDvar, 1)*0.005;
% First 2 values assigned to keep a reasonable amplitude for the shape
a0(1) = 0.03;
a0(2) = 0.02;

x = transpose(linspace(0, L, Nx+1));

%% Create Inputs for Fmincon
% Sets the contraints, the anonymous func, and fmincon options.

[A,b] = CalcIneq(a0,Nx,L);
obj = @(a) objective(a,Nx,Ny,L,K,top,bot);
optns = optimset('Display','Iter','Algorithm','sqp','PlotFcn', {@optimplotx @optimplotfirstorderopt});

%% Run Optimization
% Finds the minimizer for 1/flux, the values of the design variables at
% that flux value, and the heights associated with the optimized shape for
% the flux.

[a, flux] = fmincon(obj, a0, A, b, [],[],[],[],[],optns);
[heights] = calcheight(a, Nx, L);
opflux = 1/flux;
```