Design and Optimization of Heat Exchanger

8176

September 27, 2024

Abstract

This report presents the design and optimization of a heat exchanger aimed at maximizing heat flux. A thermal model based on the steady state heat equation is used to model the performance of the heat exchanger, using finite volume discretization to analyze heat transfer across the system. Mesh convergence is evaluated to ensure accuracy in the numerical solution. The optimization process is carried out using MATLAB's fmincon tool, with the objective of maximizing heat flux under given constraints. Throughout the optimization, first-order optimality and feasibility metrics are tracked to assess convergence and the viability of solutions. The results demonstrate an effective design optimization that maximizes the heat flux to value greater than the quadruple of the nominal, flat plate design, flux value.

1 Introduction

Heat exchangers are critical components in a wide range of industrial applications, from power generation to chemical processing and HVAC systems. Their primary function is to transfer heat between two or more fluids, ensuring efficient thermal management. This heat could be transferred from liquid to liquid, liquid to gas, etc. The design and optimization of heat exchangers are crucial for enhancing performance, reducing energy consumption, and minimizing operational costs.

This project focuses on the design and optimization of heat exchanger than transfers heat from hot water, through the steel alloy heat exchanger, to air at colder temperature – similar to the heat exchange system used to heat up houses, offices, and other spaces or rooms. The design employs a sinusoidal parameterization with the objective to maximize the heat flux per unit length from the water to air. The design problem is illustrated pictorially in Fig. 1.

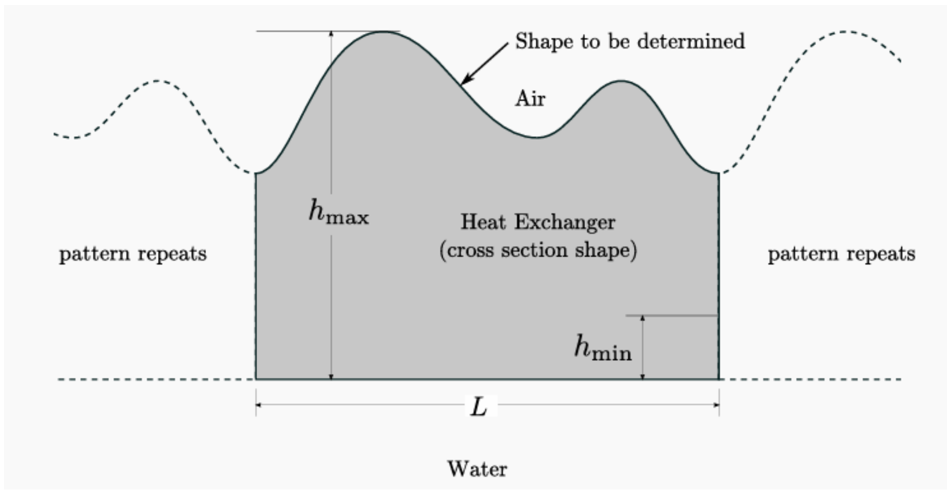

Figure 1: Heat exchanger design problem

2 Methodology

In the section, the method used to solve the problem and how the optimization problem is setup is discussed. This includes the geometric parameterization, thermal model used to analyze the heat flux, mesh independdence study to ensure that the mesh doesn't influence the result, and the optimization problem statement that is solved.

2.1 2D Geometric Parameterization

The 2D geometry is parameterized using the sinusoidal wave form expressed in Eq. 1 below:

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

where h is the height in meters (as shown in Fig. 1), a is the amplitude of oscillation, L is the length (or size) of the domain in meters, also the width of each heat exchanger section, x is the descretized locations along the x-axis, in the domain, and n represents the number of design variables which is discussed in section 2.3. Generally, the higher the number of design variables used, the more accurate the calculated heat flux and the more complex the design is. Hence there has to a trade-off among these metrics (calculated heat flux accuracy, complexity of design, and the number of design variables).

2.2 Thermal Model

The flow of thermal energy from water to air is modeled using the 2-dimentional steady state heat equation, also known as the Fourier's Law:

$$\nabla \cdot (k\nabla T) = 0, \quad \forall x \in \Omega \tag{2}$$

where T is the temperature in Kelvin and k is the thermal conductivity in W/(mK), Ω represents the domain of the heat exchanger being modeled. The boundary condition along the x axis is periodic i.e.

$$T(x=0,y) = T(x=L,y),$$

while the boundary condition along the y axis is prescribed as:

$$T(x,y=0) = T_{water},$$
$$T(x,y(x)) = T_{air}.$$

At any point in the domain, the temperature is defined as:

$$T(x,y) = T(x+L,y)$$

For this design problem, the thermal parameters are summarized in the Table 1 below.

Table 1: Thermal properties

Parameter	Value
Thermal Conductivity (k)	20W/(mK)
Water Temperature (T_{water})	90°C
Air Temperature (T_{air})	20°C
Domain size (L)	5cm

2.3 Mesh Independence Study

To perform a mesh convergence/independence study, a fixed design is considered with ten (10) design variables (i.e. n = 10 in Eq. 1) which implies that n = 10 is the most complex design this mesh convergence study would be valid for. Specifically, the values of a used is defined as

$$a = [0.0300, -0.0003, 0.0, 0.0, 0.0, 0.0, 0.0, -0.0001, 0.0, 0.0203]^T$$

and the resulting geometry is shown in Fig. 2. Notably, the geometry doesn't fully stay within the constraint which would be discussed in subsection 2.4.

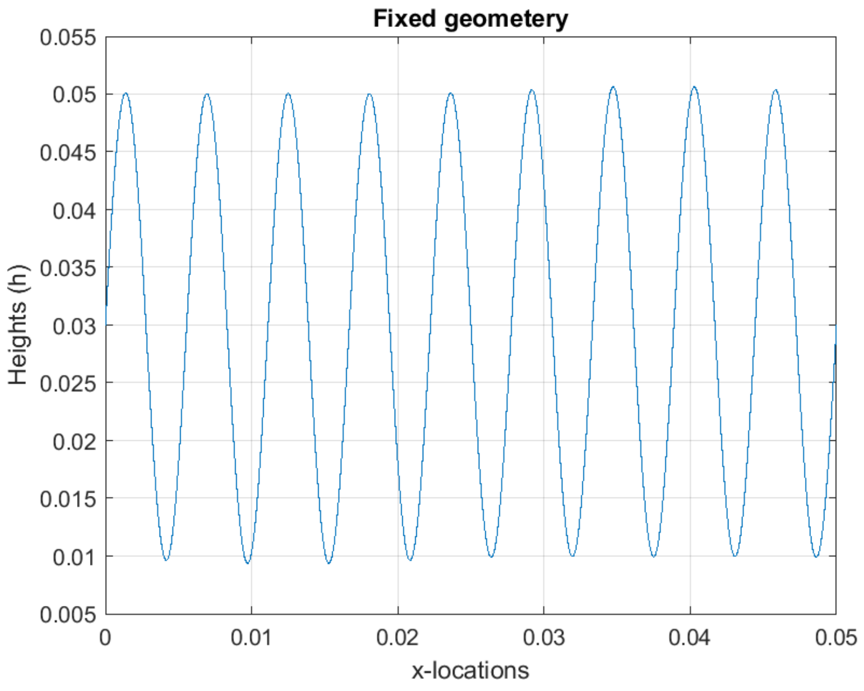

Figure 2: Fixed geometry for mesh convergence study

The governing heat equation is approximated using a finite volume discretization that comprise subdivision of the domain into finite number of quadrilaterals. The heat equation is solved for each of these quadrilaterals (finite volumes) and integrated over the whole domain (volume). For some mesh size, the flux value is dependent on the size of the mesh (or number of quadrilaterals). Specifically, the higher the number of quadrilateral, the higher the flux, and the more accurate the result is. This trend continues until the flux value no longer change with increasing mesh size. At this point, the mesh is said to have converged and heat flux value is no longer dependent on the mesh size. This convergence study is performed and plotted in Fig. 3, with number of cells in both x and y axes (Nx and Ny) starting from 50 up to 1000 and with a step size or increment of 50. Notably, the mesh convergence is observed around 900 - 1000 Nx values. Interestingly, as the number of cells increase, it become more computationally expensive to analyze and optimize. Therefore, number of cells (Nx and Ny) are selected to be 400 and error is calculated as below:

$$\begin{aligned} Error &= \frac{\text{Flux at Nx}_{1000} - \text{Flux at Nx}_{400}}{\text{Flux at Nx}_{1000}} \times 100\% \\ &= \frac{29439.4 - 29176.7}{29439.4} \times 1001\% = 0.8923\% \end{aligned}$$

Since Nx = Ny = 400 produces an allowable error of 0.8923%, this mesh size is considered for the optimization.

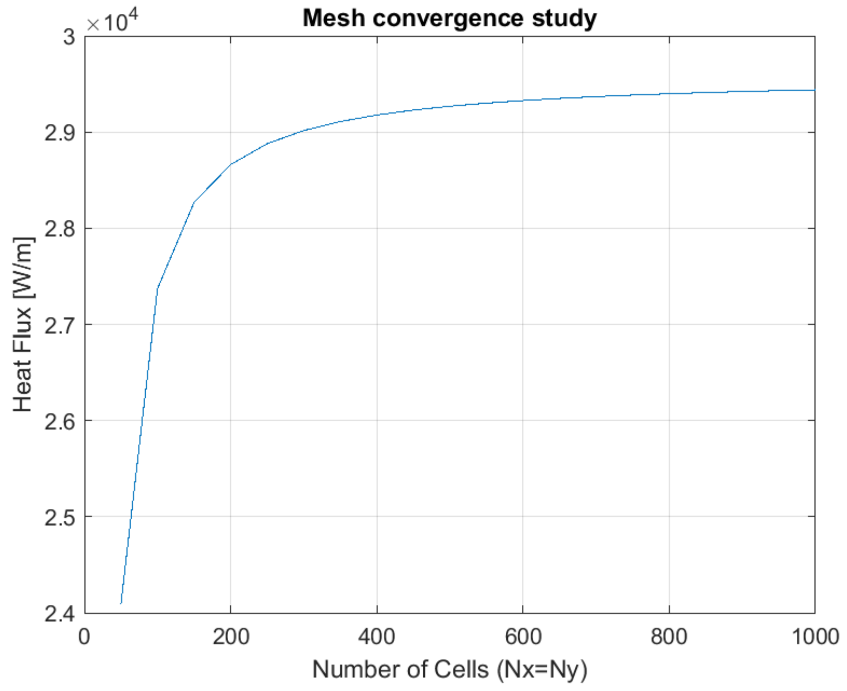

Figure 3: Mesh convergence study

2.4 Optimization Problem Statement

The optimization problem statement is summarized in Table 2 where the objective is to maximize the heat flux. The design variables are elements of vector a, the amplitudes of oscillation as discussed in section 2.1, and the constraint is on the height. This optimization problem is solved on Matlab using Fmincon.

Table 2: Optimization problem statement (see Eq. 1 for h and a relation)

Maximize	Heat Flux	
With respect to	Amplitudes, a	10 design variables
Subject to	Heights, h	1cm $\leq$ h $\leq$ 5cm

3 Results

In this section, the result from running the optimization is discussed and the resulting maximized heat flux from the optimized heat exchanger design is analyzed.

3.1 Optimization Result

Following the problem setup discussed in section 2 and optimization problem statement discussed in subsection 2.4 (using 10 design variables), the design is optimized and heat flux is maximized using fmincon, an optimization tool in MATLAB (the code is listed in Appendix D). The resulting optimal design is a well structure sinusoidal form geometry which satisfies the constraints (see Table 2) placed on the heights (h) as shown in Fig. 4. Consquently, the resulting maximized heat flux corresponding to this optimal geometry is **28610.57 W/m**. To validate this result, based on the chosen mesh (Nx = 400, see subsection 2.3), the maximal heat flux is expected to quadruple the nominal heat flux of approximately 6930.693 W/m, based off a flat plate geometry. Since 28610.57 > 27722.77 (i.e. the

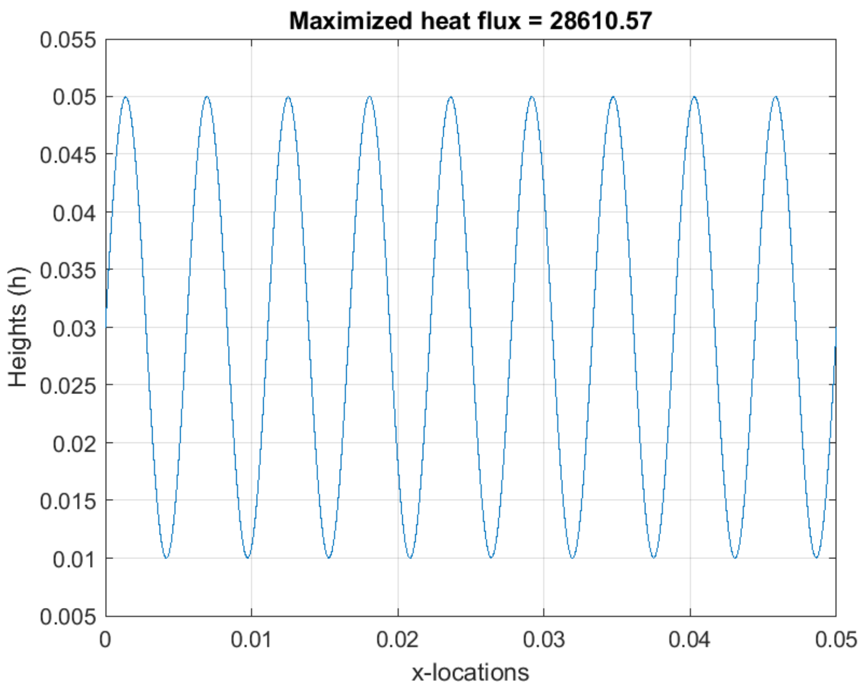

Figure 4: Heat exchanger optimal geometry design

maximized heat flux is greater than quadruple of the nominal heat flux), the optimization result is validated.

3.2 Feasibility and Optimality

First order optimality is an indication of how close a design is to the optimal point. Essentially, the condition is defined as:

$$\nabla f(x^*) = 0 \tag{3}$$

This implies that at the optimal design, the gradient of the function is zero, i.e. there is no direction to move towards in order to improve the objective. Generally, first order optimality could be initially high, particularly at the start of the optimization process, as could be seen in Fig. 5. As the optimization progresses, reduction in this optimality value shows that the design is approaching optimal design, an very low value typically imply reaching the optimal design. This trend is found in our optimality plot in Fig. 5.

A feasibility plot against iteration number in optimization typically shows how well the current solution satisfies the problem's constraints as the optimization algorithm progresses. Because the initial guess may not satisfy the constraint, the feasibility may start of being high but tend to zero as the algorithm approaches the optimal solution. In Fig. 5, this trend could be seen briefly between iteration 1 and 3. Overall, the feasibility plot is akin to a flat line but notably, the feasibility values are generally very small which implies that solutions for these iterations obey the constraints. Notably, the feasibility at iteration 0 isn't plotted because the feasibility value at that iteration is zero and semilogy function used to plot the y-axis on a logarithmic scale couldn't evaluate Log(0) (see Appendix G).

4 Conclusion

This design and optimization of a heat exchanger with sinusoidal wave form geometry is performed and the maximized heat flux that satisfy the constraints is 28610.57 W/m. The result was validated through comparison with the quadruple of the nominal value based of a flat plate geometry and the optimization result met the expectation. In addition, the feasibility and first order optimality bolsters the validity of the optimization result.

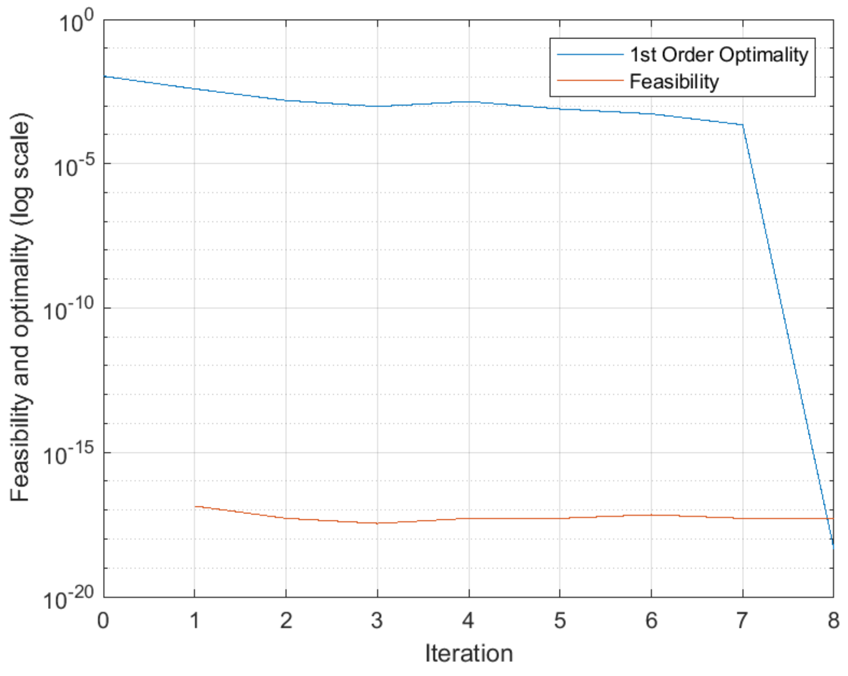

Figure 5: Feasibility and first order optimality

5 Appendix

This section details all the MATLAB codes used to solve the optimization problem and make plots.

A Geth.m

```
% This function generate two arrays for h and x. h is the height of the
% heat exchanger geometry at each x location while x is the discrete
% locations within the domain. h is a function of x, and it is computed
% using a sinusoidal correlation.
%
% Inputs:
%   a - amplitude of oscillation
%   L - length of domain in x direction
%   Nx - number of elements along the x direction
%
% Outputs:
%   h - height as a function of x; note that size(h,1) must be Nx+1
%   x - discretized locations along the x direction
%---------------------------------------------------------------------------

function [h, x] = Geth(a, L, Nx)
    dx = L/Nx;
    n = length(a); % length of a
    if n < 2
        error('Array a must have at least 2 elements');
    end
    % Array of x and h values, initialized to 0's
    x = zeros(Nx+1,1);
    h = zeros(Nx+1,1);

    for i = 1:Nx+1

        x(i) = (i-1)*dx; % Allocating values to the x array elements
        h(i) = a(1); % Initializing h to a1
        for k = 2:n
```

```
            % Adding the summation part of h(x) eqn from k=2 to k=n
            h(i) = h(i) + a(k)*sin((2*pi*(k-1)*x(i))/L);

        end
    end

end
```

B maxim.m

```
% This function calculates the objective (flux) from the CalcFlux function
%
% Inputs:
%   L - length of domain in x direction
%   a - amplitude of oscillation
%   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:
%   obj - the objective which in this case is to maximize flux and is
%   defined as the inverse of flux.
%--------------------------------------------------------------------------
function [obj] = maxim(L, a, Nx, Ny, kappa, Ttop, Tbot)
    [h,~] = Geth(a, L, Nx);
    flux = CalcFlux(L, h, Nx, Ny, kappa, Ttop, Tbot);
    obj = 1/flux;
    % obj = -1*flux;
end
```

C DefConstraint.m

```

% This function defines the linear inequalities constraints needed to be
% passed to fmincon for optimization to effectively constrain the height
% function.
%
% Inputs:
%   a - amplitude of oscillation
%   L - length of domain in x direction
%   Nx - number of elements along the x direction
%   x - discretized locations along the x direction
%
% Outputs:
%   Inequality constraints Aineq and bineq
%
%--------------------------------------------------------------------------

function [Aineq, bineq] = DefConstraint(a, L, Nx, x)
% defining constraint
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
```

D HeatExchanger.m

```
% This is the script that performs the optimization and calls all the
% needed functions to maximize the objective.
%
% Inputs:
%   L - length of domain in x direction
%   a - amplitude of oscillation
%   a0 - initial guess for a
%   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 - the maximized objective flux
%   iter (iteration) results from fmincion optimization
%   plot of x against h, showing the shape of the heat exchanger
%-------------------------------------------------------------------------

% Cleaning up the command window
clc
clear

Nx = 400; % number of cells along x-driection
Ny = 400; % number of cells along y-direction
L = 0.05; % Heat exchanger width, also length of the domain
kappa = 20; % steel thermal conductivity [W/mk]
Ttop = 293.15; % air temp in Kelvin (20 degC)
Tbot = 363.15; % water temp in Kelvin (90 degC)

% random initial guess for "a" which is scaled by 5mm
a0 = 0.005*rand(9,1); a0(1) = 0.03;

[~,x] = Geth(a0, L, Nx);

[Aineq, bineq] = DefConstraint(a0, L, Nx, x);

% anonymous function to calculate the objective
fun = @(a) maxim(L, a, Nx, Ny, kappa, Ttop, Tbot);

% Setting options to use the 'sqp' algorithm
options = optimoptions('fmincon', 'Algorithm', 'sqp', 'Display', 'iter');

[a_opt, fval] = fmincon(fun, a0, Aineq, bineq, [], [], [], [], [], options);

disp('flux');
heat_flux = 1/fval;
disp(heat_flux); % outputing the value maximized objective, flux

[h, x] = Geth(a_opt,L, Nx);
plot(x, h)
xlabel('x-locations')
ylabel('Heights (h)')
title(sprintf('Maximized heat flux = %.2f', heat_flux ));
grid on
% grid minor
saveas(gcf, 'heat_exchanger.png')
```

E FixedGeometry.m

```matlab
% This script generates a plot that represents the fixed geometry for the
% mesh convergence study
%
% Inputs:
%   a - amplitude of oscillation
%   L - length of domain in x direction
%   Nx - number of elements along the x direction
%
%
% Outputs:
%   fixed_geometry.png
%--------------------------------------------------------------------------

a = [0.0300; -0.0003; 0.0; 0.0; 0.0; 0.0; 0.0; -0.0001; 0.0; 0.0203];
Nx = 400; % number of cells along x-driection
L = 0.05; % Heat exchanger width, also length of the domain
[h, x] = Geth(a, L, Nx);

plot(x, h)
title('Fixed geometery'); % Title of the plot
xlabel('x-locations');  % Label for the x-axis
ylabel('Heights (h)');  % Label for the y-axis
grid on
saveas(gcf, 'fixed_geometry.png')
```

F MeshConvergence.m

```matlab
% This script generates the mesh convergence plot and save in a .png file.
% It is predominantly hardcoded.
%
% Inputs:
%   a - amplitude of oscillation
%   L - length of domain in x direction
%   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:
%   mesh_convergence.png
%--------------------------------------------------------------------------

a = [0.0300; -0.0003; 0.0; 0.0; 0.0; 0.0; 0.0; -0.0001; 0.0; 0.0203];
Nx = 550; % number of cells along x-driection
Ny = Nx; % number of cells along y-direction
L = 0.05; % Heat exchanger width, also length of the domain
kappa = 20; % steel thermal conductivity
Ttop = 293.15; % air temp in Kelvin (20 degC)
Tbot = 363.15; % water temp in Kelvin (90 degC)

[h, ~] = Geth(a, L, Nx);
[flux,~,~,~] = CalcFlux(L, h, Nx, Ny, kappa, Ttop, Tbot);

% disp('Heat flux')
% disp(flux)
flux_array = [];
Nx_array = [];
i = 1;

for Nx = 50:50:1000
    Ny = Nx;
    [h, ~] = Geth(a, L, Nx);
    [flux,~,~,~] = CalcFlux(L, h, Nx, Ny, kappa, Ttop, Tbot);
    flux_array(i) = flux;
    Nx_array(i) = Nx;
    i = i+1;
end

plot(Nx_array, flux_array)
title('Mesh convergence study'); % Title of the plot
xlabel('Number of Cells (Nx=Ny)');  % Label for the x-axis
ylabel('Heat Flux [W/m]');  % Label for the y-axis
grid on
saveas(gcf, 'mesh_convergence.png')
```

G FeasibilityAndOptimality.m

```
% This script plots feasibility and first order optimality plot, with hardcoded
% feasibility and optimization values gotten from ruinning the HeatExchanger.m script

Feas = [0.000e+00; 1.388e-17; 5.204e-18; 3.469e-18; 5.204e-18;
     5.204e-18; 6.939e-18; 5.204e-18; 5.204e-18];
Optimality = [1.074e-02; 3.913e-03; 1.550e-03; 9.588e-04; 1.427e-03;
    7.888e-04; 5.322e-04; 2.232e-04; 4.337e-19];
Iter = transpose(0:1:8);

semilogy(Iter, Optimality) % plotting Optimality on log scal and iter on linear
hold on
semilogy(Iter,Feas)
hold off

xlabel('Iteration')
ylabel('Feasibility and optimality (log scale)')
legend('1st Order Optimality', 'Feasibility')
grid on
saveas(gcf, 'feasibility_and_optimality.png')
```