Contents


```
clc
clear all
```

Setting up Initial Starting point

using a triangluar parameterization

```
%---------------------------------------------------------------------------
a0 = [3 4 10]'; %SP1
%a0 = [3 2 16]'; %SP2
hmin = 1; % min height of radiator in cm
hmax = 5; % max height of radiator in cm
L = 5; % Length of Radiator in cm
n = length(a0); % length of Design var
x = (0:0.02:5)'; % descretization of x-axis
```

optimization algorithm used - fmincon

```
A = [];
b = [];
Aeq = [];
beq = [];
lb = [];
ub = [];
nonlcon = @nonlincon; % function call to calculate nonlinear constraints
options = optimoptions(@fmincon,'Display','iter');
[X,fvalue,exitflag,output] = fmincon(@CalcFlux_obj,a0,A,b,Aeq,beq,lb,ub,...
    nonlcon,options);
```

Plotting

```
[flux_final,T_final,dTdX,XY] = plot_profile(X,a0);
```

Setting up Objective function

It calls the functions - Calc_h to generate the profile which is a funciton of the design variable 'a' It calls upon the function CalcFlux.m to calculate the flux CalcFlux_obj returns the -Flux calculated by CalcFlux.m Objective function only requires input - the design variable 'a'

```
%---------------------------------------------------------------------------

function Flux = CalcFlux_obj(a)
    L = 5; % cm
    Kappa = 20; % W/(m.K)
    T_top = 20; % deg cel
    T_btm = 90; % deg cel
    x = (0:0.02:5)'; % cm

    % calculate h
    h = Calc_h(x,a,L);

    % set Nx and Ny
    nx = length(h)-1;
    ny = 150;

    % Calculate Flux
    [Flux,~,~,~] = CalcFlux(L,h,nx,ny,Kappa,T_top,T_btm);

    % Negate Flux for maximization problem
    Flux = -1*Flux;
end
```

Setting up nonlinear Constraints

This function is set up to calculate the nonlinear constraints c(a)<= s; Here c(a) - vector of functions taking the same input 'a' and 's' is the containing the constraint values Input to function : a - Design Variable

```
%---------------------------------------------------------------------------

function [c,ceq] = nonlincon(a)
x = (0:0.02:5)';
L = 5; %cm
hmin = 1; % cm
hmax = 5; % cm
for i=1:length(x)
    S = 0;
    for j=1:50
        S = S + (a(2)/pi)*(-1^j)*sin(2*pi*a(3)*j*x(i)/L)/j;
    end
    c1(i,1) = a(1) - S - hmax;
end
for i=1:length(x)
    S = 0;
    for j=1:50
        S = S + (a(2)/pi)*(-1^j)*sin(2*pi*a(3)*j*x(i)/L)/j;
    end
    c2(i,1) = hmin-(a(1) - S);
end
ceq = [];
c = [c1;c2];
end
```

plot final profile

This function generates a plot comparing the Optimized profile with the initial profile and also another plot containing the zoomed optimal profile This function takes inputs: X- Optimized design, a0 - initial design.

```
%--------------------------------------------------------------------------

function [flux_final,T_final,dTdX,XY] =  plot_profile(X,a0)
x = (0:0.02:5)';
L = 5;
T_top = 20;
T_btm = 90;
kappa = 20;
h = Calc_h(x,X,L);
nx = length(h)-1;
ny = 150;
[flux_final,T_final,dTdX,XY] = CalcFlux(L,h,nx,ny,kappa,T_top,T_btm);
h0 = Calc_h(x,a0,L);
figure(1);
plot(x,h,'k');
hold on;
plot(x,h0);
plot([0,0],[0,h(1)],'k');
plot([5,5],[0,h(end)],'k');
plot([0,0],[0,0],'k');
legend('Optimized Profile','Starting Profile','Left Edge',...
    'Right Edge','Width');
axis([-2 7 0 6]);
title('Optimized v Starting Profile');
xlabel('x');
ylabel('h');

figure(2)
plot(x,h,'k');
title('Optimized Profile - Zoomed');
xlabel('x');
ylabel('h');
end
```

Calculate h - Profile height

Function returns the profile height h - h(x;a) Function takes inputs - x: discretization about X-axis, a: the design variable, L: Length of the heat exchanger

```
%--------------------------------------------------------------------------

function h = Calc_h(x,a,L)
% Uses triangular parameterization to create the profile
% h(x) = a1 + sigma ((a(2)/pi)*(-1^j)*sin(2*pi*a(3)*j*x(i)/L)/j) j:1(1)N

for i=1:length(x)
    S = 0;
    for j=1:50
        S = S + (a(2)/pi)*(-1^j)*sin(2*pi*a(3)*j*x(i)/L)/j;
    end
    h(i,1) = a(1) - S;
end
end
```