ME6103

Assignment 3

3/17/25

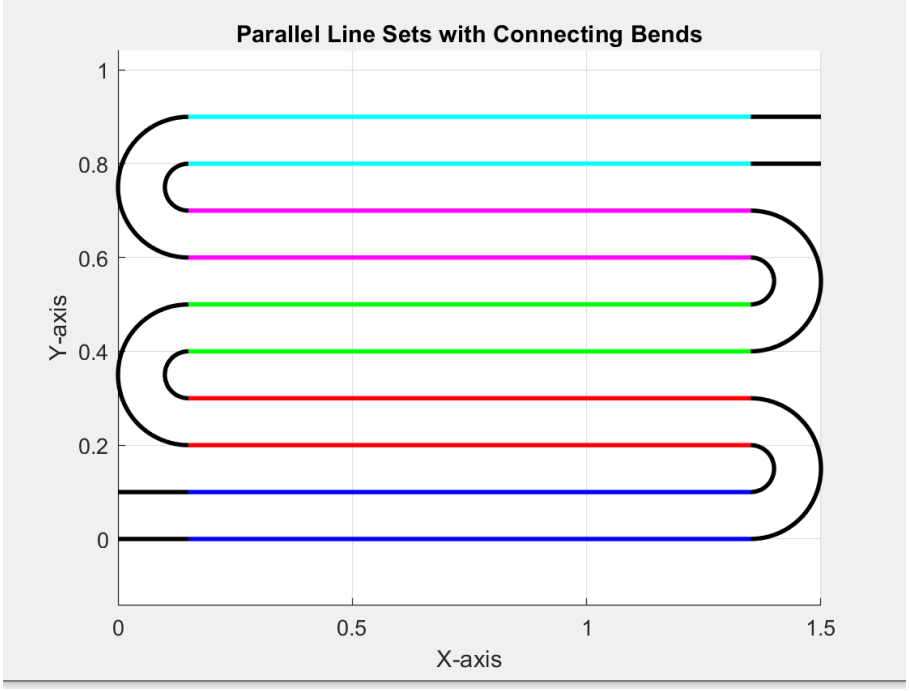
Dr. Seepersad

Shubh Raval

Part A)

Explanation of Analysis Model:

So there were 2 critical objective functions for this problem to minimize, the 1st was the Volume of the enclosure and the 2nd was the head loss of the system. An important aspect was accurately determining the total length of the piping used. For the entire system I determined that there were 3 critical design variables which guided the problem. The Pipe Diameter (D_p), the Length of the Enclosure (L_e), and the Number of turns (N) the pipe made in the enclosure. The system also had an equality constraint of ensuring that the exit temperature was 3 K higher than the input temperature. Since the straight lengths and width of the system were critical and needed to be accurately measured based on the pipe diameter, I also wrote a serpentine pipe plotting function which aided immensely in removing numerical error and logic errors in determining the geometry.



Objective Function 1 Volume:

For this objective function there were 3 terms Length, Width, and Height that had to be formulated in terms of the design variables. Length was directly taken as L_e a design variable. The height was set as an arbitrary fixed ratio of the Pipe diameter D_p in my case I used D_p * 1.10. Finally the width was constructed as a function of the pipe diameter and the number of turns. Where I defined it then as W_e = N*(D_p *S_p) + D_p, where S_p is the mandatory spacing between the pipes which is held to be equal to the Pipe Diameter.

With these three terms determined I was able to input them into the objective functions of the NSGA-II algorithm as:

```
% Dependent Variables
S_p = D_p;
H_e = 1.1 * D_p;
W_e = N * (2 * D_p) + D_p;

% Objective Function 1: Volume
f(1) = W_e * H_e * L_e;
```

Objective Function 2 Head loss:

This analysis was entirely based on the recommended sheet from the problem statement which describes both minor and major head loss via Darcy's equations. Now for this system then the critical aspect was determining the length of the Pipe so as biproduct of setting the plotting up it is apparent that in the major headloss there are 2 types of straight lengths to consider the first being L_e – 1.5*D_p which is length of the straight section at the bottom and top of the piping and L_e – 3*D_p to account for the middle sections which have bends on either side.

```
%Straight Run on the bottom Pipe and the Exit Pipe
ExposedStraightPipe_Len = (N+1)*((L_e - 1.5*D_p));  % There will be N+1 of these
if N>1
    ExposedStraightPipe_Len = ExposedStraightPipe_Len + (N-1)*(L_e-(3*D_p));
end
```

With this total length the major head loss is just a defined by equation 3-14 where v the flow

$$H_f = f \frac{L}{D} \frac{v^2}{2g}$$

velocity is a function of the mass flow rate given as .2 kg/s, the density of the water, and the

cross sectional area which is a function of the pipe Diameter.

The other consideration is the minor losses which based on the table for each return bend (180 degree turn) can be modeled by the equation k* (v^2/2*g). Now this is considering just 1 return bend hence the minor losses need to be multiplied by the N bends. Where N starts at 0 for no bends. The important inclusion here is the coefficent k which is the pipe friction factor (f) defined as 0.04 times the ratio of the equivalent length of pipe divide by the pipe diameter. In table 1 typical values are given of this L_eq/D_p ratio which for a return bend is then 50 which is what was used in the equation k = f*L_eq/D_p. With the minor losses then defined the objective function for head loss was then constructed as:

```
% Objective Function 2: Head Loss
    rho_water = 998; % kg/m^3
    m_dot = 0.2; % kg/s
    f_friction = 0.04;
    flow_v = m_dot / (rho_water * pi * (D_p / 2)^2);
    mhl_t1 = (flow_v^2) * (1 / (2 * 9.81));

    % Compute Straight Pipe Lengths
    ExposedStraightPipe_Len = (N + 1) * (L_e - 1.5 * D_p);
    if N > 1
        ExposedStraightPipe_Len = ExposedStraightPipe_Len + (N - 1) * (L_e - 3 * D_p);
    end
    U_bend_Len = N * (1.5 * D_p * pi * 0.5);

    % Total Head Loss Calculation
    major_head_loss = f_friction * (ExposedStraightPipe_Len / D_p) * mhl_t1;
    k = f_friction * 50; % Equivalent length ratio for return bends
    minor_head_loss = k * (flow_v^2) / (2 * 9.81);
    f(2) = major_head_loss + N * minor_head_loss; % Objective 2: Head Loss
```

Equality Constraint, Change in Temperature is 3 Kelvin:

The last aspect was the required performance of the system that is driving the entire reason for determining optimality. This was given that when the ½ of the pipe's surface area is exposed to sunlight with a solar flux of 1000W/m^2 the Outlet Temperature would be 3 Kelvin higher than the inlet Temperature. One additional remark was made is that the pipe was black. This allowed for some necessary simplifications in the heat transfer model, which were driven by certain assumptions. These assumptions were that the pipe thickness t << D_p meaning that losses from conduction resistivity are negligible. Next that since the pipe is black we can assume its absorptivity = 1 meaning all the solar energy then becomes heat. Hence we can simplify to $Q = \dot{m}c\Delta T$, since the problem statement prescribes a requirement that delta T is 3 K, the specific heat of water is taken as 4184, and

the mass flow rate of .2 is given we can solve for the required heat Q. Additionally since we assume that solar flux multiplied by the surface area is the imparted heat we can solve directly for a required Surface Area that the system must have. Now this requirement is interesting in that it is both a constraint that the pipe's system must attempt to be as close as possible to the surface area requirement without dropping below it or overly exceeding it since in either case it would be undesirable performance. Then by solving this the required surface area is 2.5104 m^2. For this system to determine the total surface area the last component needed is the length of the curved sections which is calculatable as

```
U_bend_Len = N * (1.5 * D_p * pi * 0.5);
```

or N turns multiplied by the half-circumference of a circle with radius 1.5*D_p. After this we are given that only half of the pipe is exposed so that then is half of the circumference of the pipe becoming the width. So the total surface area is then

```
Total_surface_area = (U_bend_Len + ExposedStraightPipe_Len) * (pi * D_p * 0.5);
```

.

Where I placed a requirement that the total surface area must exceed the required surface area but not by more than 10%. Where a penalty was placed if either scenario was violated to ensure that solutions found would have desired performance.

B) Formulate the multi-objective optimization

Thus to find an optimum solution we must minimize the volume of the enclosure and the head loss of the system while subject to a constraint that the outlet temperature is the inlet temperature + 3 K, which has been defined as the piping system having a surface area of approximately 2.5104 m^2. These objective functions are subject to N a number of return bends in the enclosure which can be greater than or equal to 0. They are subject to an enclosure length L_e that must be less than 1.5 meters. Lastly they are subject to a variable pipe diameter D_p which can range from 0.02 meters to .12 meters.

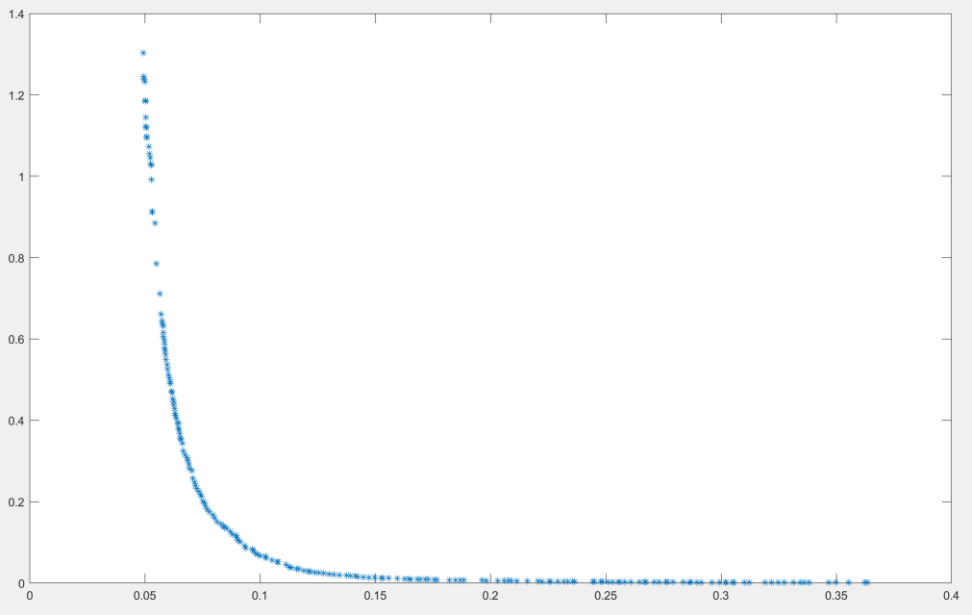
C) Devise a method to obtain the Pareto frontier

To find the pareto frontier of solutions the NSGA-II algorithm will be used. The inputs to the algorithm will be the 2 objective functions and the 3 design variables with their specific bounds. Additionally, to handle the constraints 2 penalty functions have been added as seen below.

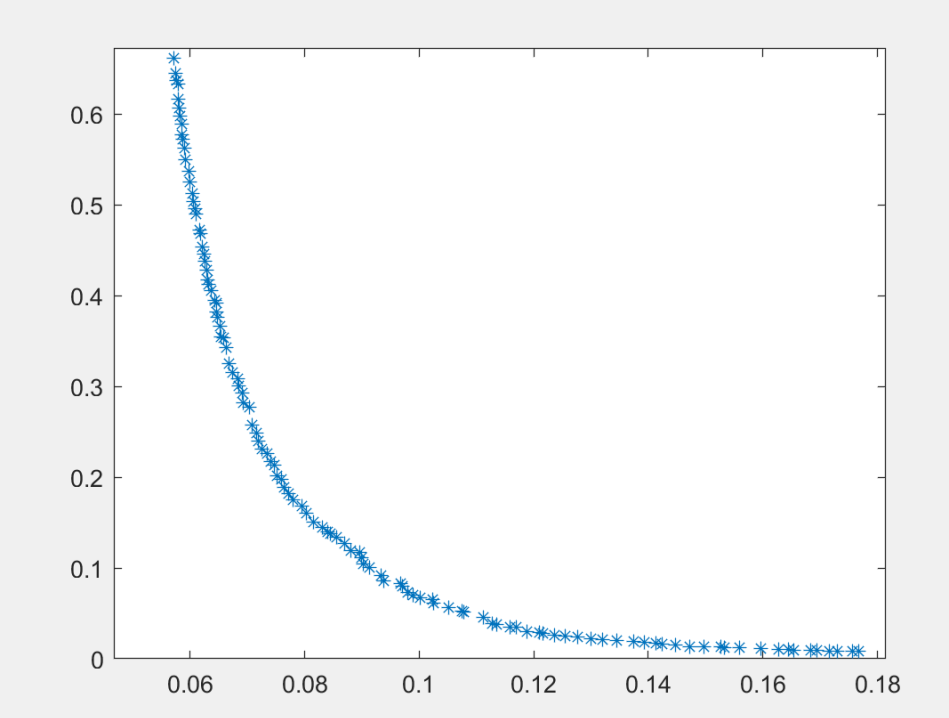
```
% If constraint is violated, add penalty
if Total_surface_area < required_SA
    penalty = 1e6 * (required_SA - Total_surface_area);
    f(1) = f(1) + penalty;
    f(2) = f(2) + penalty;
end
% If constraint is overly exceeded, add penalty
if Total_surface_area > required_SA * 1.1
    penalty = 1e6 * (Total_surface_area - required_SA);
    f(1) = f(1) + penalty;
    f(2) = f(2) + penalty;
end
```

So to obtain the pareto cover the NSGA-II algorithm used a population of 200 with 100 generations. That then produced this curve where the x axis is f(1) the volume, and the y axis is f(2) the headloss.

Full frontier:



Frontier Zoomed in:

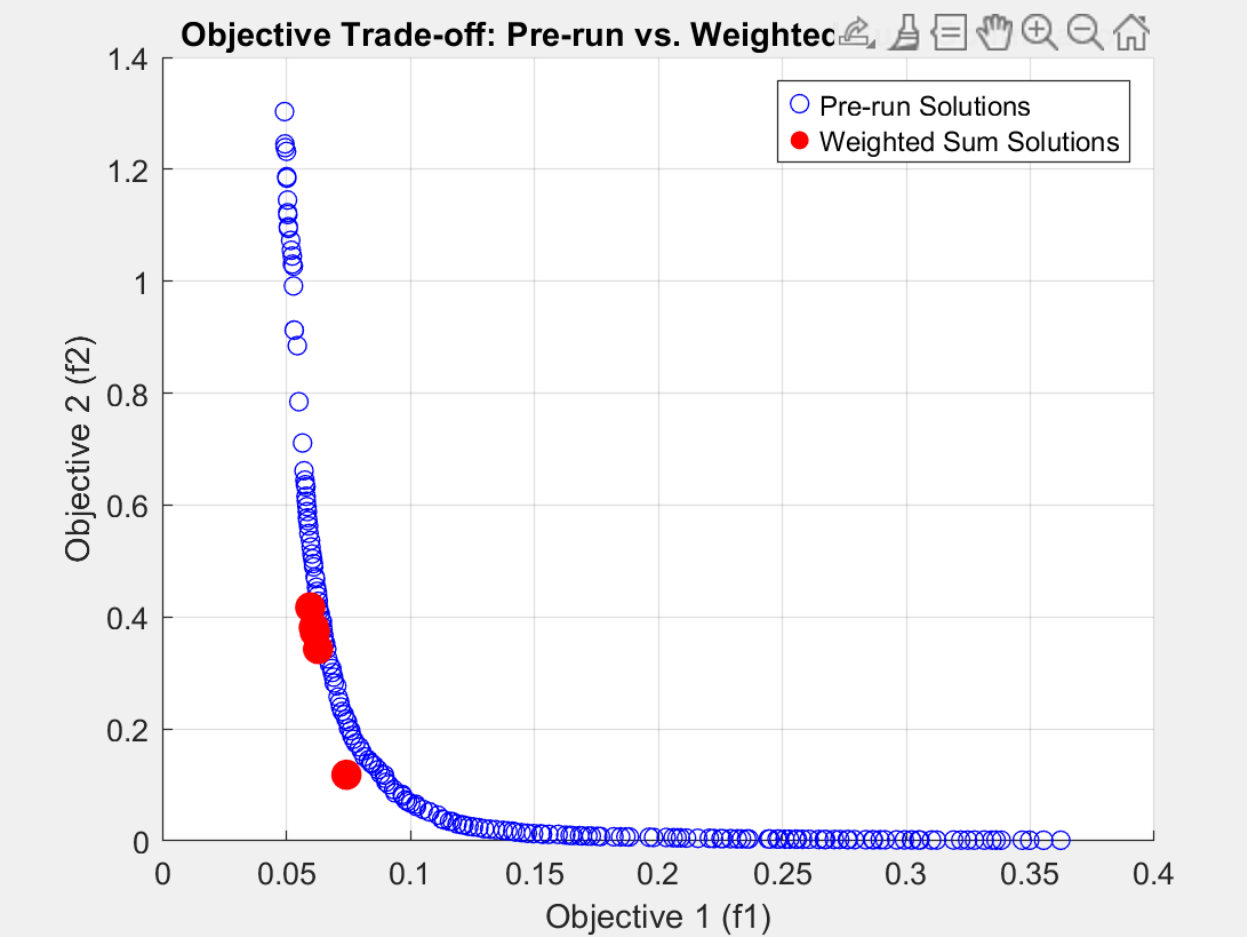


d) Apply weighted sum technique to identify at least 5 Pareto Solutions

To apply the weighted sum technique I took the f1 max and f2 max from the outputted solutions file of the non-normalized solutions these were f1_max = .36340 m^3 and f2_max= 1.3 m. Thus I applied this write a new function with calculated this weighted sum function. Then I just wrote a script which created a set of 5 different weights for w1 and w2 and used fmincon to use gradient descent to find solutions of the weight objective function Z. These were then:

w1	w2	D_p	L_e	N	Z	f1	f2	c
0.1	0.9	0.0395	0.8911	23.8178	0.102067	0.074263	0.118192	0.000001
0.3	0.7	0.0319	0.7341	35.7176	0.273377	0.059652	0.417229	0.000000
0.5	0.5	0.0326	0.7017	36.7078	0.230378	0.061107	0.381313	0.000001
0.7	0.3	0.0334	0.6725	37.5207	0.199883	0.062803	0.342840	0.000000
0.9	0.1	0.0328	0.6920	37.0572	0.180903	0.061497	0.373084	0.000001

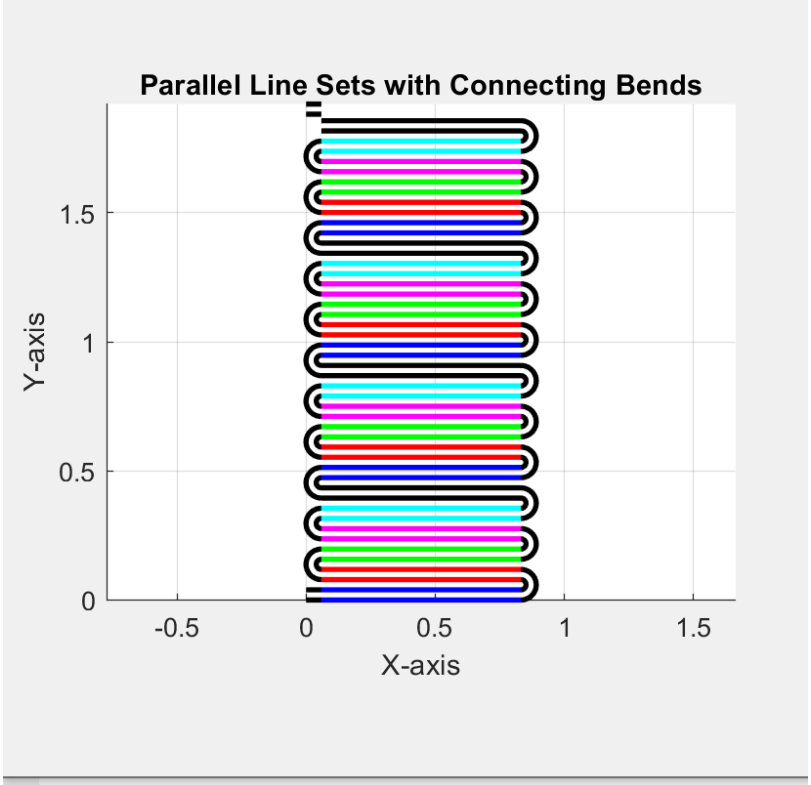
Which when plotted relative to pareto frontier found by NSGA-II were:



The critical item is that none of these solutions found violate the constraint on a minimum surface area and this is primarily achieve by just continuing the same penalty weight even

when formulating the Z function and ensures that only feasible solutions are found by fmincon. This is show as c which is taken as the difference between the computed Surface Area and the required surface area is always positive but below the 10% threshold of .251 m^2.

So when visualized solution A appears roughly like so:




```
The volume is 7.438196e-02
    0.0744

The minimum SA is 2.510400e+00
The current SA is 2.512288e+00
The total Head loss is 1.177645e-01
```

First its seen that the pipe diameter has very little fluctuations between the weights suggesting that this is an optimal point for it and that this range is close to its lower limit. So as expected the Number of turns will increase as less emphasis is placed on the head loss and the length will shrink allowing for a thicker stack that minimizes the volume with more emphasis. The inverse is true when the weight is placed on the head loss objective function. Both of these outcomes do align with the analytical understanding of the underlying equations.

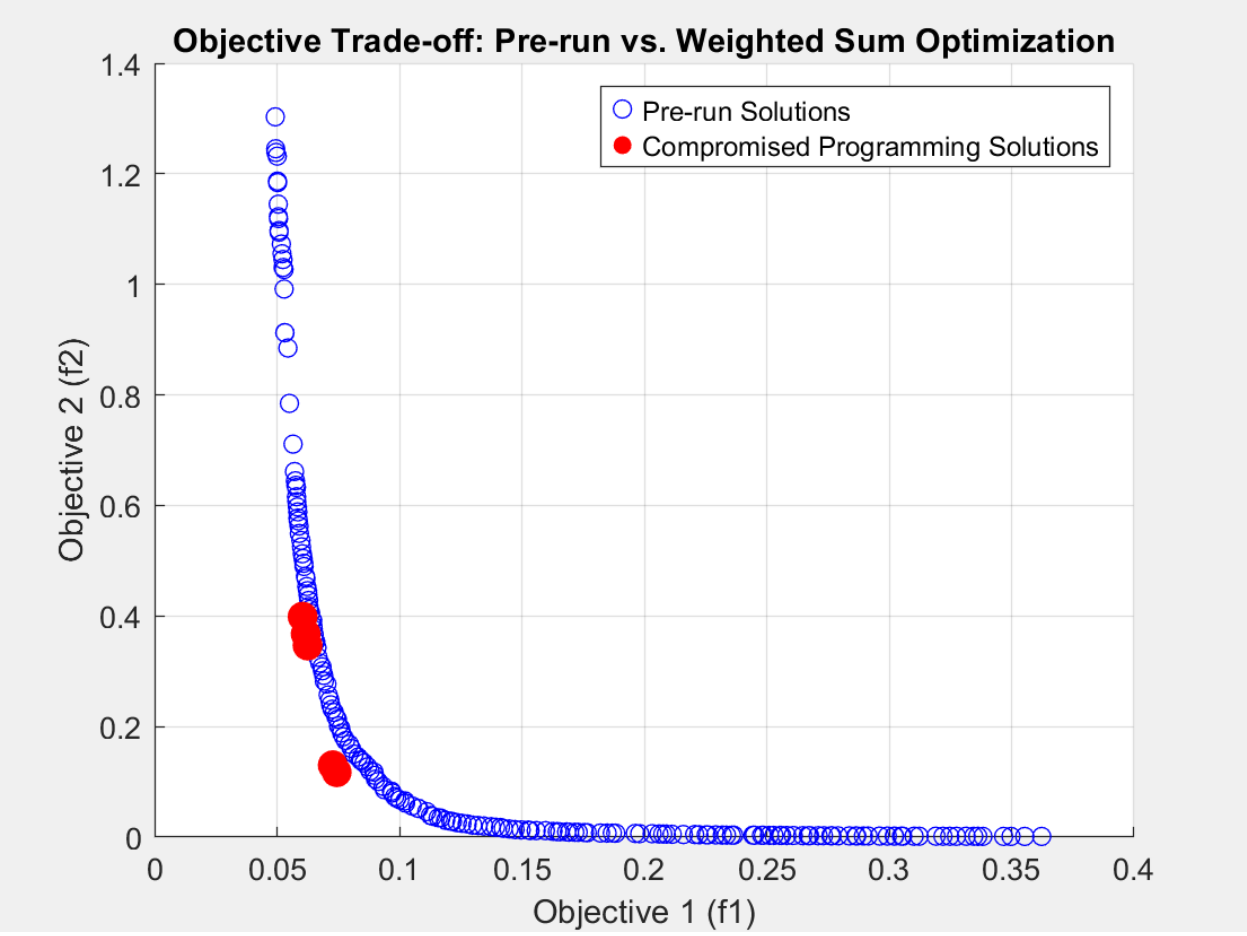
e) Apply 1 Additional technique:

So here the the compromised programming technique was handled in a very similar manner to how the weighted sum was done. Which is where a function was made and then in a script fmincon was used so gradient descent can find an optimal solution. Now the main items of interest were what P was set to and what was used for the target f1 and the target f2. For P I used 2 which was to use the Euclidean distance. For target f1 and f2 the minimum values from the NSGA-II Pareto and those were used. The logic here was that the target for either objective function truly was to minimize them as much as possible. Hence this would be appropriate. Thus the optimal solutions from this method were:

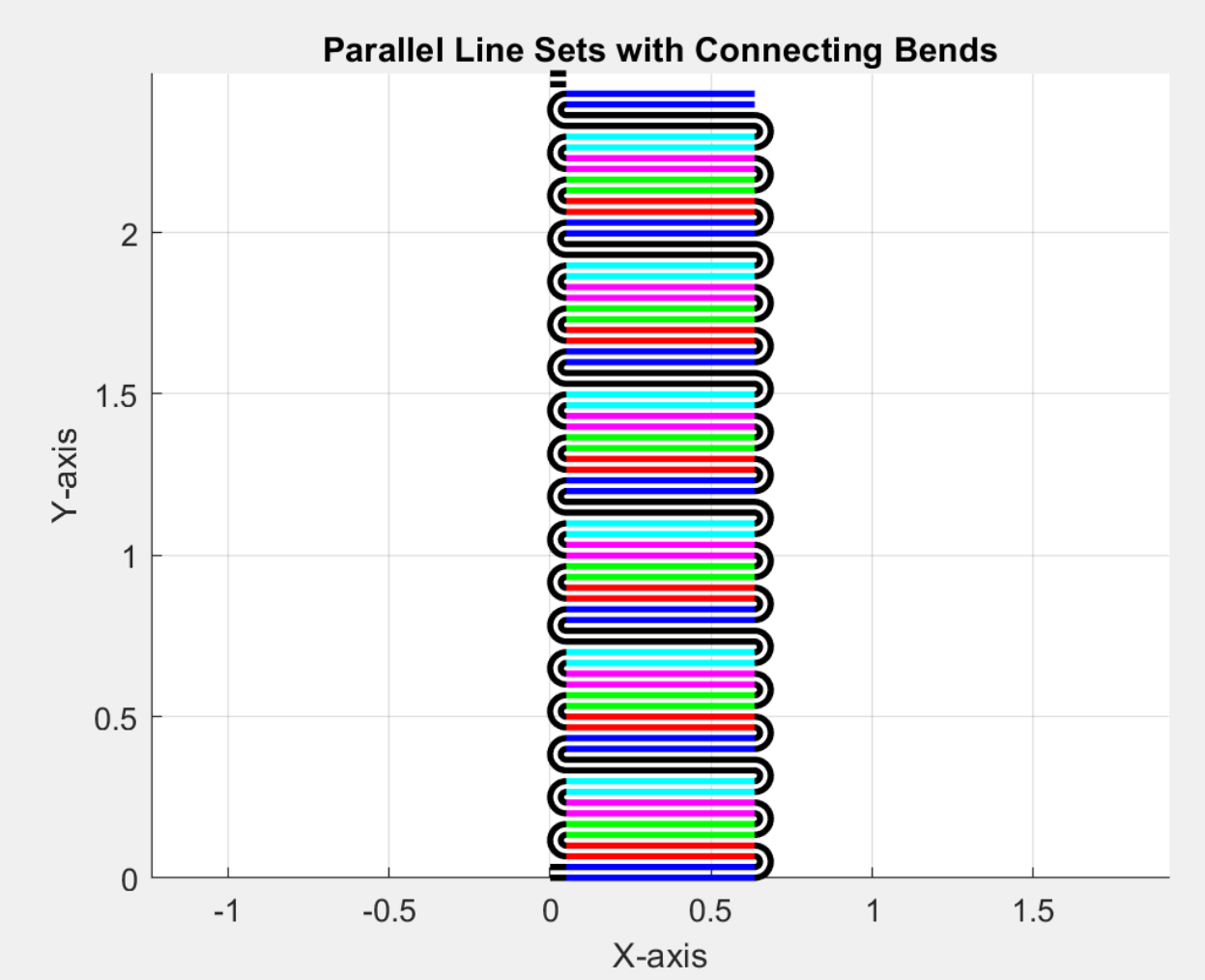
w1	w2	D_p	L_e	N	Z	f1	f2	c
0.1	0.9	0.0387	0.9023	23.9303	0.089212	0.072773	0.129990	0.000001
0.3	0.7	0.0395	0.8904	23.8109	0.065813	0.074361	0.117468	0.000001
0.5	0.5	0.0333	0.6850	36.9422	0.133793	0.062489	0.346615	0.000001
0.7	0.3	0.0323	0.7106	36.5537	0.094018	0.060464	0.398703	0.000002
0.9	0.1	0.0329	0.6885	37.1419	0.041676	0.061721	0.367673	0.000001

Where each of them satisfies the required constraints and attempt to be as tightly constrained to the required surface area as possible.

When plotted along the Pareto frontier these are then:



Then taking one of the solutions which is at (0.062489, 0.346615) which has parameters (0.0333,0.6850,36.9422) = (D_p ,L_e, N) this is visualized as:



Now when looking through the trends here is that one observation is that the target value of f1 and f2 can have a significant result on the final Z function we can see here as expected if there is additional weight on the head loss minimization then the number of N turns is lower and the Length is longer since as expected more turns will greatly increase the head loss. Its also seen that the D_p value does also remain quite consistent and is seen towards the bottom range of the parameter. On the note of N and L_e with respect to which

objective function is weighted, we see that when volume gains more emphasis than head loss a less long and more densely nest of pipe bends will be what the solution tends towards.

f) Compare results between d and e

Something that suggests confidence in the results is that the same trends appeared for both methods and that their solution trends were generally in agreeance when it came to the stability of D_p to be within 0.03 and 0.04. Additionally that the trend of N growing when more weight was placed on the Volume minimization with L_e shrinking as well when the weight moved in the favor of volume. That being said there are differences in the minimal Z solution of both methods.

When comparing the minimal Z solution for both we see that the weighted sums was:

w1	w2	D_p	L_e	N	Z	f1	f2	c
0.1	0.9	0.0395	0.8911	23.8178	0.102067	0.074263	0.118192	0.000001

with the Z minimal solution of the compromised solution was

0.9	0.1	0.0329	0.6885	37.1419	0.041676	0.061721	0.367673	0.000001

As its seen they occur on opposite sides of the weights. Where D_p does still fall within the noted range we see that there is some variation in the lengths with them following on opposite sides. But when comparing all 5 of the solutions found for both we do see that they fall in the approximate range of .65-1. Similarly pipe bends for both falls in the spectrum of 23-37 almost perfectly for both. What is critical regardless is that these 10 solutions do meet the required surface area constraint which means each design would allow for the temperature of the water to increase by 3 K.

Code:

Initial Matlab Script where the analytical functions were first put together, where the visualization function is being called, and where NSGA-II was called:

```
clear all;

close all;

clc;
```

```
%% ME6103 Assignment 3 Shubh Raval



%% Design Variables

D_p = 0.0333; %Pipe Diameter in m

L_e = .6850; % Length of the enclosure

N = 10; % Number of turns starts at 0

%% Visualization

plotParallelLines(N,L_e,D_p)

%% Dependent Variables

S_p = D_p; %Pipe Spacing = Pipe Diameter

H_e = D_p*1.1; %Slightly Larger than Pipe Diameter

W_e = N*(D_p+S_p) + D_p; % Width is the number of turns + 1 addition exit Pipe

Half_Circ = D_p*pi*.5;

%% Volume Calculation

Volume = W_e * H_e * L_e;

fprintf('The volume is %s \n', Volume)



%% Total Pipe Length

%Straight Run on the bottom Pipe and the Exit Pipe

ExposedStraightPipe_Len = (N+1)*((L_e - 1.5*D_p));  % There will be N+1 of these

if N>1
```

```
   ExposedStraightPipe_Len = ExposedStraightPipe_Len + (N-1)*(L_e-(3*D_p));
end

U_bend_Len = N*((1.5*D_p*pi*.5));


Total_Len =   U_bend_Len + ExposedStraightPipe_Len; % m
Total_surface_area = Total_Len * Half_Circ;

%% Required Heat Transfer Params and Constants
m_dot = .2 ; % kg/s
cp_water = 4184; % J*kg/K
q_sun = 1000; %W/m^2
rho_water = 998;

%% Assume pipe has 100% absortivity from sun so that removes conduction and thickness
<< D_p
Delta_T = 3; % Required minimum temp increase
Q = m_dot*cp_water*Delta_T;
minimum_SA = Q/q_sun;

fprintf('The minimum SA is %s \n',minimum_SA);
fprintf('The current SA is %s \n', Total_surface_area);

%% Total Headloss
f_friction = 0.04;
```

```
flow_v = m_dot/(rho_water*pi*(D_p*.5)^2);
mhl_t1 = ((flow_v)^2)*(1/(2*9.81));
major_head_loss = f_friction* (ExposedStraightPipe_Len/D_p)*mhl_t1;

% Using Equivalent Piping Length
ratio = 50; %Return Bend

k = f_friction* ratio;

% minor loss
minor_head_loss = k * ((flow_v)^2)/(2*9.81); %One Bend

total_head_loss = major_head_loss + N * minor_head_loss;

fprintf('The total Head loss is %s \n',total_head_loss);



%nsga_2(200,100)
% Read the solution file
data = dlmread('solution.txt');

% Extract f1 (column 4) and f2 (column 5)
f1_values = data(:, 4);
f2_values = data(:, 5);
```

```
% Compute max values
f1_max = max(f1_values);
f2_max = max(f2_values);
f1_min = min(f1_values);
f2_min = min(f2_values);
% Display the results
fprintf('f1_max = %.6f\n', f1_max);
fprintf('f2_max = %.6f\n', f2_max);
fprintf('f1_min = %.6f\n', f1_min);
fprintf('f2_min = %.6f\n', f2_min);
```

the input to the NSGA-II evaluate_objective:

```
f = [];
D_p = x(1);
L_e = x(2);
N = x(3);
% Dependent Variables
S_p = D_p;
H_e = 1.1 * D_p;
W_e = N * (2 * D_p) + D_p;

% Objective Function 1: Volume
f(1) = W_e * H_e * L_e;

V =3;
```

```
% Objective Function 2: Head Loss
    rho_water = 998; % kg/m^3
    m_dot = 0.2; % kg/s
    f_friction = 0.04;
    flow_v = m_dot / (rho_water * pi * (D_p / 2)^2);
    mhl_t1 = (flow_v^2) * (1 / (2 * 9.81));

    % Compute Straight Pipe Lengths
    ExposedStraightPipe_Len = (N + 1) * (L_e - 1.5 * D_p);
    if N > 1
       ExposedStraightPipe_Len = ExposedStraightPipe_Len + (N - 1) * (L_e - 3 * D_p);
    end
    U_bend_Len = N * (1.5 * D_p * pi * 0.5);

    % Total Head Loss Calculation
    major_head_loss = f_friction * (ExposedStraightPipe_Len / D_p) * mhl_t1;
    k = f_friction * 50; % Equivalent length ratio for return bends
    minor_head_loss = k * (flow_v^2) / (2 * 9.81);
    f(2) = major_head_loss + N * minor_head_loss;

    % Constraint Handling for Penalties
    q_sun = 1000; % W/m^2
    cp_water = 4184; % J/kg-K
    Delta_T = 3; % Required min temperature increase
    required_SA = (m_dot * cp_water * Delta_T) / q_sun;
    Total_surface_area = (U_bend_Len + ExposedStraightPipe_Len) * (pi * D_p * 0.5);
```

```
    % If constraint is violated, add penalty
    if Total_surface_area < required_SA
        penalty = 1e6 * (required_SA - Total_surface_area);
        f(1) = f(1) + penalty;
        f(2) = f(2) + penalty;
    end
    % If constraint is overly exceeded, add penalty
    if Total_surface_area > required_SA * 1.1
        penalty = 1e6 * (Total_surface_area - required_SA);
        f(1) = f(1) + penalty;
        f(2) = f(2) + penalty;
    end
```

weighted_sum function:

```
function Z = weighted_sum(x, f1_max, f2_max,w1,w2)
    % Extract decision variables
    D_p = x(1);
    L_e = x(2);
    N = x(3);

    % Dependent Variables
    S_p = D_p;
    H_e = 1.1 * D_p;
    W_e = N * (2 * D_p) + D_p;
    Half_Circ = D_p*pi*.5;
```

```
% Objective Function 1: Volume
f1 = W_e * H_e * L_e;

% Objective Function 2: Head Loss
rho_water = 998; % kg/m^3
m_dot = 0.2; % kg/s
f_friction = 0.04;
flow_v = m_dot / (rho_water * pi * (D_p / 2)^2);
mhl_t1 = (flow_v^2) * (1 / (2 * 9.81));

% Compute Straight Pipe Lengths
ExposedStraightPipe_Len = (N + 1) * (L_e - 1.5 * D_p);
if N > 1
   ExposedStraightPipe_Len = ExposedStraightPipe_Len + (N - 1) * (L_e - 3 * D_p);
end
U_bend_Len = N * (1.5 * D_p * pi * 0.5);
Total_Len =   U_bend_Len + ExposedStraightPipe_Len; % m
Total_surface_area = Total_Len * Half_Circ;
required_SA = 2.5104;
% Total Head Loss Calculation
major_head_loss = f_friction * (ExposedStraightPipe_Len / D_p) * mhl_t1;
k = f_friction * 50; % Equivalent length ratio for return bends
minor_head_loss = k * (flow_v^2) / (2 * 9.81);
f2 = major_head_loss + N * minor_head_loss; % Objective 2: Head Loss
% If constraint is violated, add penalty
if Total_surface_area < 2.5104
```

```
    penalty = 1e6 * (required_SA - Total_surface_area);
    f1 = f1 + penalty;
    f2 = f2 + penalty;
  end
  % If constraint is overly exceeded, add penalty
  if Total_surface_area > 2.5104 * 1.1
    penalty = 1e6 * (Total_surface_area - required_SA);
    f1 = f1 + penalty;
    f2 = f2 + penalty;
  end



  Z = (w1*f1 / f1_max) + (w2*f2 / f2_max);


End
```

Compromised_programming function:

```
function Z = compromised_prog(x, f1_max, f2_max,w1,w2)
  % Extract decision variables
  D_p = x(1);
  L_e = x(2);
  N = x(3);


  % Dependent Variables
  S_p = D_p;
  H_e = 1.1 * D_p;
  W_e = N * (2 * D_p) + D_p;
```

```
Half_Circ = D_p*pi*.5;

% Objective Function 1: Volume
f1 = W_e * H_e * L_e;

% Objective Function 2: Head Loss
rho_water = 998; % kg/m^3
m_dot = 0.2; % kg/s
f_friction = 0.04;
flow_v = m_dot / (rho_water * pi * (D_p / 2)^2);
mhl_t1 = (flow_v^2) * (1 / (2 * 9.81));

% Compute Straight Pipe Lengths
ExposedStraightPipe_Len = (N + 1) * (L_e - 1.5 * D_p);
if N > 1
   ExposedStraightPipe_Len = ExposedStraightPipe_Len + (N - 1) * (L_e - 3 * D_p);
end
U_bend_Len = N * (1.5 * D_p * pi * 0.5);
Total_Len =   U_bend_Len + ExposedStraightPipe_Len; % m
Total_surface_area = Total_Len * Half_Circ;
required_SA = 2.5104;
% Total Head Loss Calculation
major_head_loss = f_friction * (ExposedStraightPipe_Len / D_p) * mhl_t1;
k = f_friction * 50; % Equivalent length ratio for return bends
minor_head_loss = k * (flow_v^2) / (2 * 9.81);
f2 = major_head_loss + N * minor_head_loss; % Objective 2: Head Loss
```

```
    % If constraint is violated, add penalty
    if Total_surface_area < 2.5104
        penalty = 1e6 * (required_SA - Total_surface_area);
        f1 = f1 + penalty;
        f2 = f2 + penalty;
    end
    % If constraint is overly exceeded, add penalty
    if Total_surface_area > 2.5104 * 1.1
        penalty = 1e6 * (Total_surface_area - required_SA);
        f1 = f1 + penalty;
        f2 = f2 + penalty;
    end
    P = 2;
    f1_min = 0.049301;
    f2_min = 0.001168;
    F1 = (w1*((f1-f1_min)/f1_max))^P ;
    F2 = (w2*((f2-f2_min)/f2_max))^P ;
    Z = (F1 + F2)^(1/P);

end
```

gradient descent script for weighted sums:

```
lb = [0.02, 0.01, 0]; % Lower bounds
ub = [.12, 1.5, 10000]; % Upper bounds
x0 = [.02, 1, 5]; % Initial guess
```

```
f1_max = 0.363480;
f2_max = 1.303012;

weights = [0.1, .9; 0.3, 0.7; 0.5, 0.5; 0.7, 0.3; 0.9, 0.1];

% Store solutions
solutions = zeros(size(weights,1), 9); % Store w1, w2, D_p, L_e, N, Z, f1, f2,c

% Open file to save results
fid = fopen('weighted_solutions.txt', 'w');
fprintf(fid, 'w1\tw2\tD_p\tL_e\tN\tZ\tf1\tf2\tc\n');

% Loop through weight combinations and optimize
for i = 1:size(weights,1)
   w1 = weights(i,1);
   w2 = weights(i,2);

   options = optimoptions('fmincon', 'Display', 'iter', 'Algorithm', 'sqp');

   weighted_obj = @(x) weighted_sum(x, f1_max, f2_max, w1, w2);

   [x_opt, Z_opt] = fmincon(weighted_obj, x0, [], [], [], [], lb, ub, [], options);

   [f1_opt, f2_opt, c] = compute_objectives(x_opt);

   solutions(i, :) = [w1, w2, x_opt(1), x_opt(2), x_opt(3), Z_opt, f1_opt, f2_opt, c];
```

```
   fprintf(fid, '%.1f\t%.1f\t%.4f\t%.4f\t%.4f\t%.6f\t%.6f\t%.6f\t%.6f\n', w1, w2, x_opt(1), x_opt(2), x_opt(3), Z_opt, f1_opt, f2_opt, c);
end


fclose(fid);


% Load pre-run data from solutions.txt
data = dlmread('solution.txt', '', 1, 0);
f1_values = data(:, 4);
f2_values = data(:, 5);

% Plot Pareto front
figure;
scatter(f1_values, f2_values, 'bo', 'DisplayName', 'Pre-run Solutions');
hold on;

% Plot optimized solutions from weighted sum
scatter(solutions(:, 7), solutions(:, 8), 100, 'r', 'filled', 'DisplayName', 'Weighted Sum Solutions');

% Labels and legend
xlabel('Objective 1 (f1)');
ylabel('Objective 2 (f2)');
title('Objective Trade-off: Pre-run vs. Weighted Sum Optimization');
```

```
legend;
grid on;
hold off;

function [f1, f2,c] = compute_objectives(x)
    D_p = x(1);
    L_e = x(2);
    N = x(3);

    % Dependent Variables
    S_p = D_p;
    H_e = 1.1 * D_p;
    W_e = N * (2 * D_p) + D_p;
    Half_Circ = D_p*pi*.5;

    % Objective Function 1: Volume
    f1 = W_e * H_e * L_e;

    % Objective Function 2: Head Loss
    rho_water = 998; % kg/m^3
    m_dot = 0.2; % kg/s
    f_friction = 0.04;
    flow_v = m_dot / (rho_water * pi * (D_p / 2)^2);
    mhl_t1 = (flow_v^2) * (1 / (2 * 9.81));

    % Compute Straight Pipe Lengths
```

```
    ExposedStraightPipe_Len = (N + 1) * (L_e - 1.5 * D_p);
    if N > 1
        ExposedStraightPipe_Len = ExposedStraightPipe_Len + (N - 1) * (L_e - 3 * D_p);
    end
    U_bend_Len = N * (1.5 * D_p * pi * 0.5);

    % Total Head Loss Calculation
    major_head_loss = f_friction * (ExposedStraightPipe_Len / D_p) * mhl_t1;
    k = f_friction * 50; % Equivalent length ratio for return bends
    minor_head_loss = k * (flow_v^2) / (2 * 9.81);
    f2 = major_head_loss + N * minor_head_loss;
    SA = Half_Circ * (U_bend_Len + ExposedStraightPipe_Len);
    c = SA - 2.5104;
end
```

gradient descent script for compromised programming:

```
lb = [0.02, 0.01, 0]; % Lower bounds
ub = [.12, 1.5, 10000]; % Upper bounds
x0 = [.02, 1, 5]; % Initial guess

f1_max = 0.363480;
f2_max = 1.303012;

weights = [0.1, .9; 0.3, 0.7; 0.5, 0.5; 0.7, 0.3; 0.9, 0.1];

% Store solutions
```

```
solutions = zeros(size(weights,1), 9); % Store w1, w2, D_p, L_e, N, Z, f1, f2,c

% Open file to save results
fid = fopen('compromised_programming_solutions.txt', 'w');
fprintf(fid, 'w1\tw2\tD_p\tL_e\tN\tZ\tf1\tf2\tc\n');

% Loop through weight combinations and optimize
for i = 1:size(weights,1)
   w1 = weights(i,1);
   w2 = weights(i,2);

   options = optimoptions('fmincon', 'Display', 'iter', 'Algorithm', 'sqp');

   weighted_obj = @(x) compromised_prog(x, f1_max, f2_max, w1, w2);

   [x_opt, Z_opt] = fmincon(weighted_obj, x0, [], [], [], [], lb, ub, [], options);

   [f1_opt, f2_opt, c] = compute_objectives(x_opt);

   solutions(i, :) = [w1, w2, x_opt(1), x_opt(2), x_opt(3), Z_opt, f1_opt, f2_opt, c];

   fprintf(fid, '%.1f\t%.1f\t%.4f\t%.4f\t%.4f\t%.6f\t%.6f\t%.6f\t%.6f\n', w1, w2, x_opt(1),
x_opt(2), x_opt(3), Z_opt, f1_opt, f2_opt, c);
end
```

```
fclose(fid);



% Load pre-run data from solutions.txt
data = dlmread('solution.txt', '', 1, 0);
f1_values = data(:, 4);
f2_values = data(:, 5);

% Plot Pareto front
figure;
scatter(f1_values, f2_values, 'bo', 'DisplayName', 'Pre-run Solutions');
hold on;

% Plot optimized solutions from weighted sum
scatter(solutions(:, 7), solutions(:, 8), 100, 'r', 'filled', 'DisplayName', 'Compromised
Programming Solutions');

% Labels and legend
xlabel('Objective 1 (f1)');
ylabel('Objective 2 (f2)');
title('Objective Trade-off: Pre-run vs. Weighted Sum Optimization');
legend;
grid on;
hold off;

function [f1, f2,c] = compute_objectives(x)
```

```
D_p = x(1);
L_e = x(2);
N = x(3);

% Dependent Variables
S_p = D_p;
H_e = 1.1 * D_p;
W_e = N * (2 * D_p) + D_p;
Half_Circ = D_p*pi*.5;

% Objective Function 1: Volume
f1 = W_e * H_e * L_e;

% Objective Function 2: Head Loss
rho_water = 998; % kg/m^3
m_dot = 0.2; % kg/s
f_friction = 0.04;
flow_v = m_dot / (rho_water * pi * (D_p / 2)^2);
mhl_t1 = (flow_v^2) * (1 / (2 * 9.81));

% Compute Straight Pipe Lengths
ExposedStraightPipe_Len = (N + 1) * (L_e - 1.5 * D_p);
if N > 1
   ExposedStraightPipe_Len = ExposedStraightPipe_Len + (N - 1) * (L_e - 3 * D_p);
end
U_bend_Len = N * (1.5 * D_p * pi * 0.5);
```

```
    % Total Head Loss Calculation
    major_head_loss = f_friction * (ExposedStraightPipe_Len / D_p) * mhl_t1;
    k = f_friction * 50; % Equivalent length ratio for return bends
    minor_head_loss = k * (flow_v^2) / (2 * 9.81);
    f2 = major_head_loss + N * minor_head_loss; % Objective 2: Head Loss
    SA = Half_Circ * (U_bend_Len + ExposedStraightPipe_Len);
    c = SA - 2.5104;
end
```