REPORT ON
"OPTIMIZATION IN MANUFACTURING ENGINEERING"

Submitted by,

BIER Luiz
HETAVKAR Saifullah
VISAL

Under the guidance of,

Prof. Fouad BENNIS
Prof. Stephane CARO

6th December 2020

Exercise 1: Manufacturing problem with Excel & MATLAB

1.1 Problem Definition:

Find the optimal number of parts to be manufactured per week. Every part must be treated on the three machines, use the data Given in Table 01.

	Manufacturing time		maxi time per week
Type of machine	part 1	part 2	
machine 1	10	5	2500
machine 2	4	10	2000
machine 3	1	1,5	450
Profit / unity	50€	100€	

Table. 01 Manufacturing Problem input data

1.2 Overview

A manufacturing enterprise always has an aim to maximise the profits with the available resources. There are many physical constraints involved such as labour cost, electricity consumption on each machine and workload capacity per week. There are 3 machines and 2 parts to be manufactured through all 3 machines with different production lead timings.

1.3 Algorithm for the Problem

The inequality constraint is given as:

$$10*\text{Part 1} + 5*\text{Part 2} < 2500$$

$$4*\text{Part1} + 10*\text{Part2} < 2000$$

$$1*\text{Part1} + 1.5*\text{Part2} < 450$$

Our objective is to maximize the function given by:

$$50*\text{Part 1} + 100*\text{Part 2}$$

1.4 Results

After computation, the results produced are as follows:

Part 1: 187

Part 2: 125

Profit: 21850 €

Exercise 2 : Dichotomous optimization

2.1 Problem Definition

Apply the algorithm to find the minima of : **f(x)** = $\frac{1}{3}\ x^3 - 2x^2 + 3x + 1$

Given an interval [x min, x max], like the one shown in Figure 1, where we are sure our minimum is located, a dichotomous optimization method is applied to find the minimum value of the function. The method consists of splitting the interval [x min, x max] according to the desired level of precision (ε), and evaluating in which interval the minimum is located. The new interval replaces [x min, x max] and the process is repeated in a loop until the difference x max – x min reaches the desired precision:

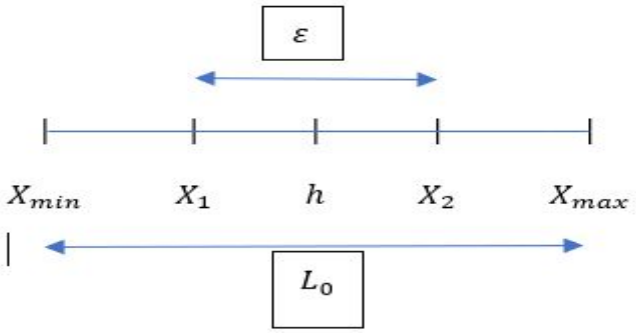


Figure 1. Interval representation

$$L_0 = X_{max} - X_{min}$$

$$\text{h} = X_{min} + \frac{L_0}{2}$$

$$X_1 = X_{min} + (\frac{L_0 - \varepsilon}{2}) \text{ and } X_2 = X_{min} + (\frac{L_0 + \varepsilon}{2})$$

$$\text{if f}(X_2) > f(X_1) \rightarrow Minimum\ is\ in\ [\ X_{min}, X_2\ ]$$

$$X_{max} = X_2$$

$$\text{if f}(X_2) < f(X_1) \rightarrow Minimum\ is\ in\ [\ X_1, X_{max}\ ]$$

$$X_{min} = X_1$$

2.2 Objective

Dichotomous search method is a one dimensional, iterative optimization approach without derivative. This method initially identifies the interval of uncertainty within the optimum solution. The optimum value of the function can be located by reducing progressively the range of uncertainty until obtaining sufficiently small range.

Figure 2 shows the objective function created in Matlab:

```
function f=f1(x)
f=(1/3)*(x.^3)-2*(x.^2)+3*x+1;
end
```

Figure 2: Objective function problem 2

2.3 Main function

Using the above objective function we can write a main program which gives us the minimized point / optimal point.

In the main function we define the interval to be considered using the objective function. The stopping criterion is started using the **WHILE** loop in the program. The complete main function is in Appendix B.

2.4 Results

The function has local minimum value in x=3, and goes towards infinite in the x+ direction, and negative infinite in the x- direction. Therefore, xmin will not change if we choose a negative initial value, and will give x=3 otherwise, where the function value is equal to 1.

Exercise 3: Extension of dichotomous method to multivariable optimization problem

3.1 Problem definition

Apply the extended algorithm to the 2 variables optimization problem :

$f(\mathbf{X}) = (3X_1 + 2X_2 - 1)^2 + (X_1 - X_2 + 1)^2$ $[-10,\ 10]$, $X_0 = (0, 1)$ and d=(0.5,1)

$X_1 and\ X_2$ are the two components of **X** = (X_1, X_2)

For a given point **X**0 and a given direction **d**, one can find all the point of the line D defined by (**X**0, and **d**) **X** = **X**0 + x **d**. f(**X**0 + x **d**) is the only function of the single variable x.

The procedure is similar to the previous exercise, but since there are two variables, it is necessary to define the points **X1** and **X2**:

$$X_1 = X_{min} + (\tfrac{L_0 - \varepsilon}{2}) \text{ and } X_2 = X_{min} + (\tfrac{L_0 + \varepsilon}{2})$$

$$\boldsymbol{X_1 = X_0 + X1 * d}$$

$$\boldsymbol{X_2 = X_0 + X2 * d}$$

The evaluation of f(**X**) is done and the new interval is defined, until the interval [X1, X2] achieves the desired resolution. Then, the average value between X1 and X2 is defined to obtain the equation of the line:

$$x = (X_1 + X_2)/2$$

X = **X**0 + x ***d**

3.2 Objective

The difference between the one with multiple variables and the single variable problem is the usage of contours instead of a single function which creates a 1D figure.

The objective function in Matlab is shown in figure 3:

```
function f=equation(x)
f=(3*x(1)+2*x(2)-1).^2+(x(1)-x(2)+1).^2;
end
```

Figure 3: Objective function problem 3

3.3 Main function

The complete main function is in Appendix C.

3.4 Results

After the implementation of the objective function and the main program we get the plot shown in Figure 4 where we can see the minimum point on the contour of the plot along the line defined in the problem definition. These points are represented by the symbol '*' in blue and red.

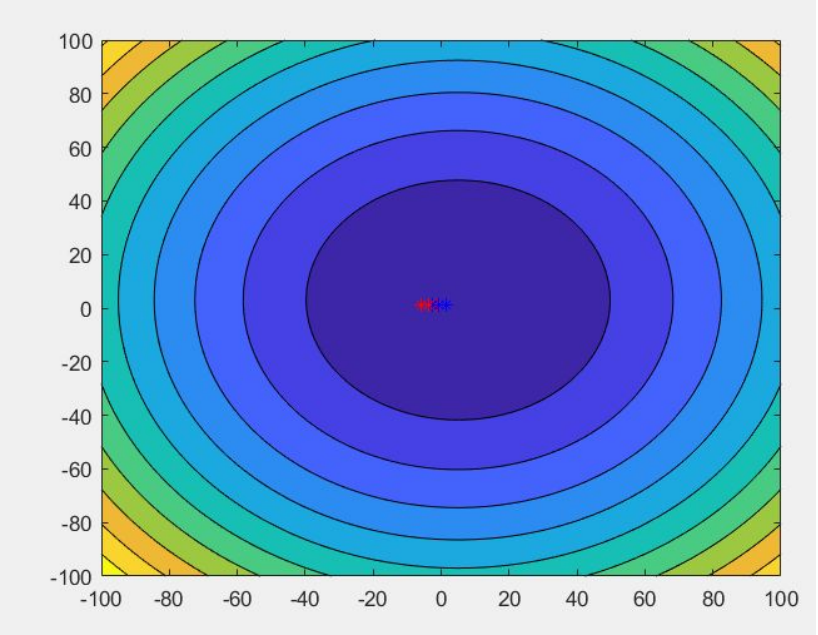
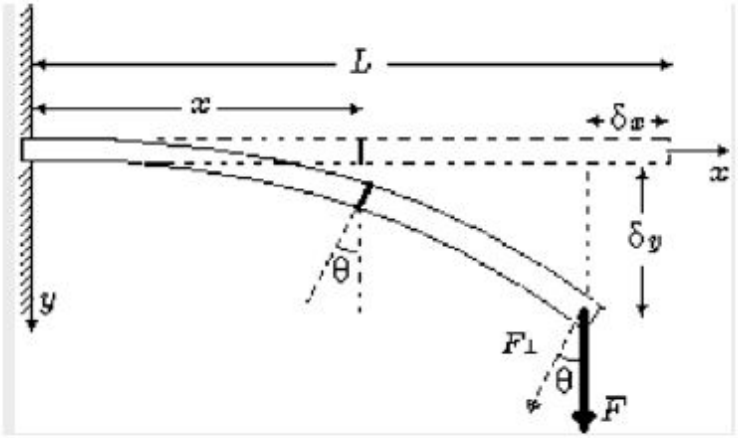


Figure 4: Representation of contour for multi variable dichotomous

Exercise 4: Application to the cantilever beam cross section optimization

4.1 Problem definition

The objective of the lab work is to make a global analysis and synthesis in order to find the best section for the problem the beam can support a concentrated force of F = 20kN applied to the free end of the cantilever beam, as shown in figure 5.



Beam parameter :

Young modulus	E= 2. 1.e11	Pa
Density (mass/volume)	ρ = Ro = 7800	Kg / m3
Admissible stress yield (max)	σ_a =sigma= 2 1.e8	Pa
Length of the beam	L = 2	m
Concentrated Force	F = 20 000	N
Admissible deflection (max)	da = Δ_a = 0.01	m
Admissible mass (max)	ma = μ_a = 300	kg

General equations of the flexion of the cantilever beam

stress :	$\sigma_{max} <= \sigma_a$	$\sigma_{max} = (FL/\mathbf{I}) * v$ (v=h/2)
Deflection	$\Delta <= \Delta_a$	$\Delta = FL^3 / (3 E \mathbf{I})$
Masse	m <= ma	$m = \rho$ L S (S = section)

Where **I** represent the Second moment of the cross section and S is the section.
For the bounds it is accepted that the h>L/5.

For the beam of square section : $\mathbf{I} = a^4 / 12$ and $S = a^2$
so that : $\sigma_{max} = 6FL / a^3$ $\Delta = 4 FL^3 / (E a^4)$ and Mass = ρ L a^2

For the beam of disc section : $\mathbf{I} = \pi D^4 / 64$ and $S = \pi D^2 / 4$
so that : $\sigma_{max} = 32$ F L / πD^3 , $\Delta = 64 FL^3 / (3 E \pi D^4)$ and Mass = ρ L $\pi D^2 / 4$

Figure 5: Cantilever problem

4.2 Case 1: Problems with one variable (Square and disc)

P1: Mass minimization subject to the constraints ($\sigma_{max} \leq \sigma_a$, $\Delta \leq \Delta_a$)

• Formulate the problem of minimizing the mass subject to the constraint of maximum deflection and stress yield.
• Without using the optimization tool, find by hand the value of corresponding value of "a" (the square side length) and "D" (the diameter of the disc section) that minimize the mass (s.t. the constraints).
• Use MATLAB to find the same solution (select the appropriate lower and upper bounds),
• Plot the mass curve and show on the curve the limit corresponding the stress yield constraint, deflection and bounds.

P2: Deflection minimization subject to the constraints (($\sigma_{max} \leq \sigma_a$ **, mass** $< ma$ **)**

• Apply the same steps as 1.1/ for the minimizing the Deflection subject to the constraint of maximum mass and stress yield.

4.2.1 Objective

The objective of this program is minimizing the mass subject to the constraint of maximum deflection.

We represent mass of the beam as m = ρ LS, where:

ρ being the density of the beam material
L, the length of the beam
S, the cross sectional area of the beam

We are supposed to find the position of the optimized cross sectional area S such that the beam does not fail under load. In, the case of the square and the disc, we represent S = a^2 and S = Πd^2 respectively.

4.2.2 Constraints

Here, being 2 constraints represented by:

$$g(1) = \left(\frac{FL}{I}\right) V - \sigma_{max}$$

Simplified as:

$$g(1) = \frac{6FL}{a^3} - \sigma_{max}$$

$$\text{and } g(2) = \frac{FL^3}{3EI} - \Delta_a \text{ or } g(2) = \frac{4FL^3}{Ea^4} - \Delta_a$$

We have to compute the value of 'a'. We equate the constraint equations to 0.

Therefore, equating g(1):

$$\frac{6FL}{a^3} - \sigma_{max} = 0$$

Substituting the values for F, L and σ_{max} , we obtain a = 0.0485.

Similarly, substituting the the values of F, L, Young's Modulus E and $\Delta_{admissible}$ in the second constraint equation g(2) = 0, we get the value of a = 0.1321.

Case 2: Disc

$$g(1) = \frac{32\,FL}{\Pi D^3} - \sigma_{max}$$

$$g(2) = \frac{64\,FL^3}{3E\Pi D^4} - \Delta_a$$

As before when g(1) = 0:

$$\frac{3\,FL}{\Pi D^3} - \sigma_{max} = 0$$

$$F = 20KN$$

$$L= 2$$

$$\sigma_{max} = 2.1* 10^8\ N/m^2$$

Solving for D, we get D = 0.05665.

Similarly for the second constraint, g(2) = 0

$$\frac{64\,FL^3}{3E\Pi D^4} - \Delta_a = 0$$

Again solving for D, we get D = 0.2007

Therefore, we choose D= 0.2007 as it is the larger of the 2 values of D.

Deflection Minimization

Objective:

$$\Delta = \frac{FL^3}{3EI}$$

Constraints:

$$g(1) = \left(\frac{FL}{I}\right) * V - \sigma_{max}$$

$$g(2) = \rho LS - ma$$

Case 3: Square

$$g(1) = \frac{6FL}{a^3} - \sigma_{max}$$

$$g(2) = \rho L a^2 - m_a$$

$$g(1) = 0$$

$$\frac{6FL}{a^3} - \sigma_{max} = 0$$

$$a = 0.1045$$

$$g(2) = 0$$

$$\rho L a^2 - 3m_a^2 = 0$$

$$a = 0.13867$$

We choose a=0.13867 as it is bigger.

Case 4: Disc

$$g(1) = \frac{32FL}{\Pi D^3} - \sigma_{max}$$

$$g(2) = \frac{\rho L \Pi D^2}{4} - m_a$$

$$g(1) = 0$$

$$\frac{32FL}{\Pi D^3} - \sigma_{max} = 0$$

$$D = 0.057$$

$$g(2) = 0$$

$$\frac{\rho L \Pi D^2}{4} - m_a$$

$$D = 0.156$$

We choose D = 0.156 as diameter

Figures 6 and 7 show the plot results for the single variable problem:

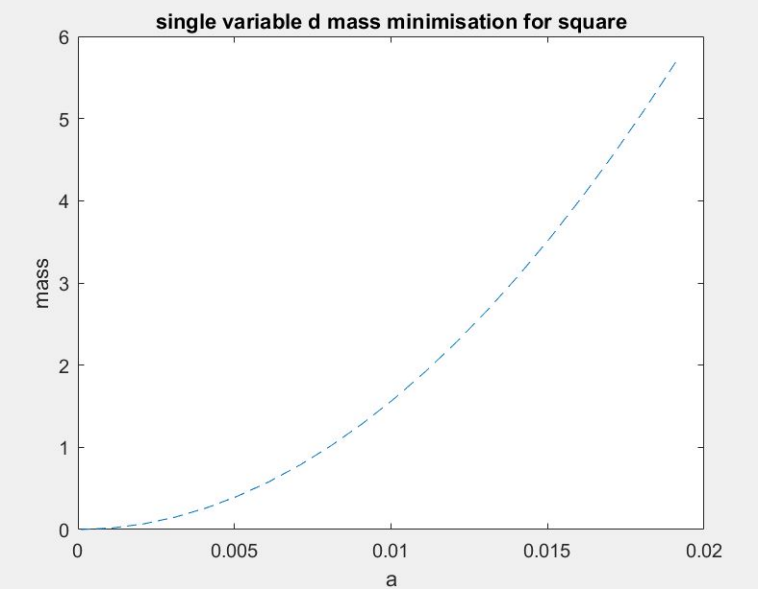


Figure 6: Plot square section mass minimisation

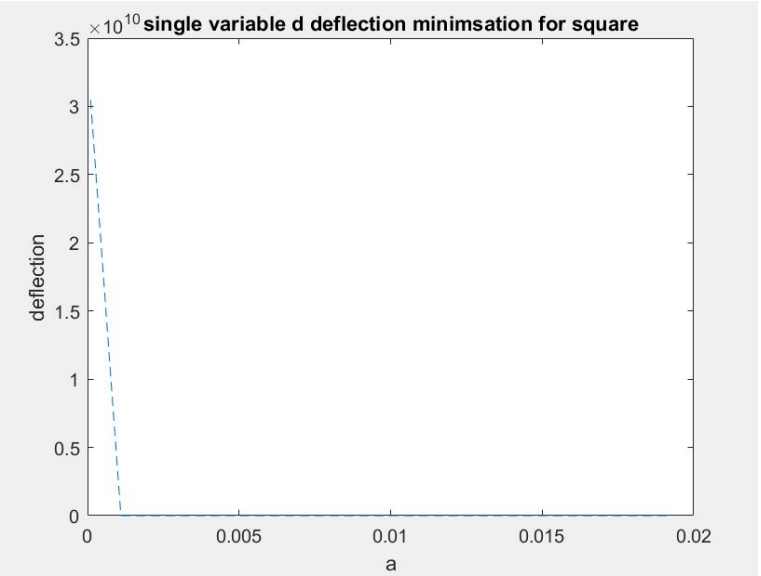


Figure 7: Plot square section mass minimisation

Figures 8 and 9 show the same results for the disc section:

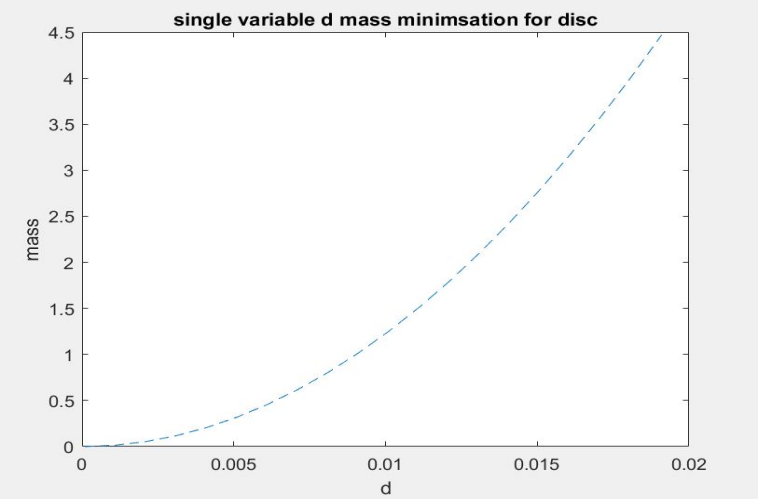


Figure 8: Plot square section mass minimisation

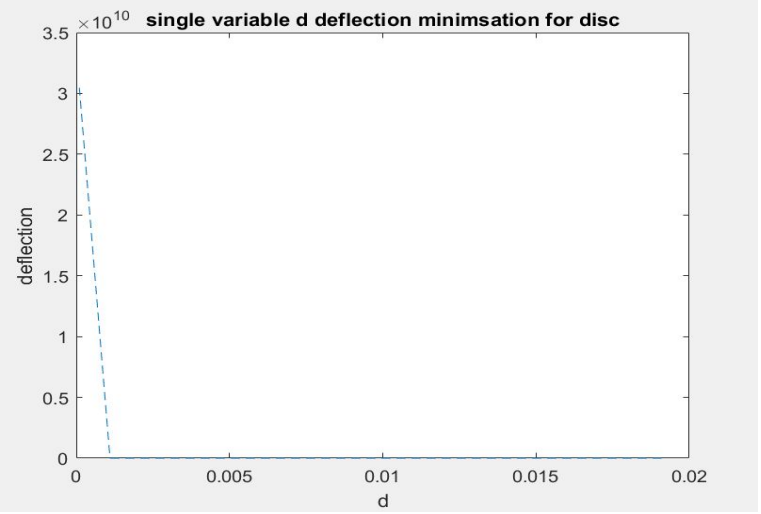


Figure 9: Plot square section mass minimisation

PROBLEM 4.2: Problem with two variables: (Hollow disc, Hollow square, Rectangle and plus section)
• For each section write the equations of the mass, deflection stress yield
• Formulate the two optimization problems (Mass and deflection : as in 1.1)
• Use MATLAB to find the optimum (select the appropriate lower and upper bounds),
• Plot the contour of the objective function and the line constraint of the constraints;

2/ Problem with two variables (Hollow disc, Hollow square, Rectangle and plus section)
• For each section write the equations of the mass, deflection stress yield
• Formulate the two optimization problems (Mass and deflection : as in 1.1)
• Use MATLAB to find the optimum (select the appropriate lower and upper bounds),
• Plot the contour of the objective function and the line constraint of the constraints

TWO VARIABLE

For the rectangular section we are asked to find the optimum length and breadth with mass minimisation. The objective function is mass. We did the matlab program with mass minimisation while keeping the stress constraints and deflection. The difference between this problem and the previous problem was that we had to give two values for input and output, as shown in figure 10:

```
x0=[0.05 0.2];
lb=[0.02 0.1];
ub=[0.3 0.3];
```

Figure 10: Inputs for two variable problem

We also created a function for the first moment of inertia so that we could easily change the expression for different geometry shapes, as well for the cross section area, as shown in figure 10.

```
function I=Ir(x)
I= (x(1)*x(2)^3)/12;

end

function m=Section(x)
m=x(1)*x(2);
end
```

Figure 10: Inputs for two variable problem

For plotting the graph we used the Matlab function “contourf”. Figure 11 shows the contours lines obtained for mass optimization. After optimization the result was obtained as b=0.0500 , H= 0.2000.

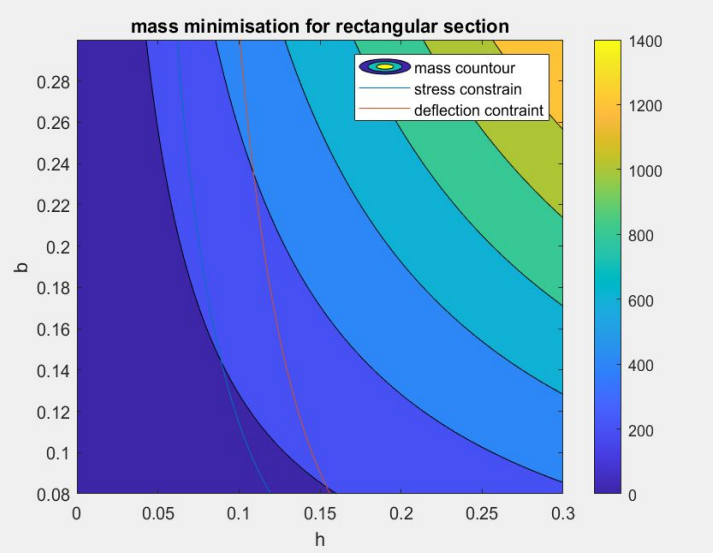


Figure 11: Mass minimisation for rectangular section

The green line represents the stress constraints while the red line shows deflection constraints. As we reduce b and h, the mass decreases but sigma_max and deflection also steadily increase and beyond a certain value of b and h, they exceed the max admissible sigma_max and deflection. Hence, all the mass contours can only lie to the right of the stress and deflection constraint lines.

The stress constraint lines and deflection constraints are obtained by the relation between b and h:

$$b = \frac{6FL}{\sigma_a h^2}, b = \frac{4FL^3}{h\Delta_{max}}$$

So we use “plot” function for the graph (Figure 12):

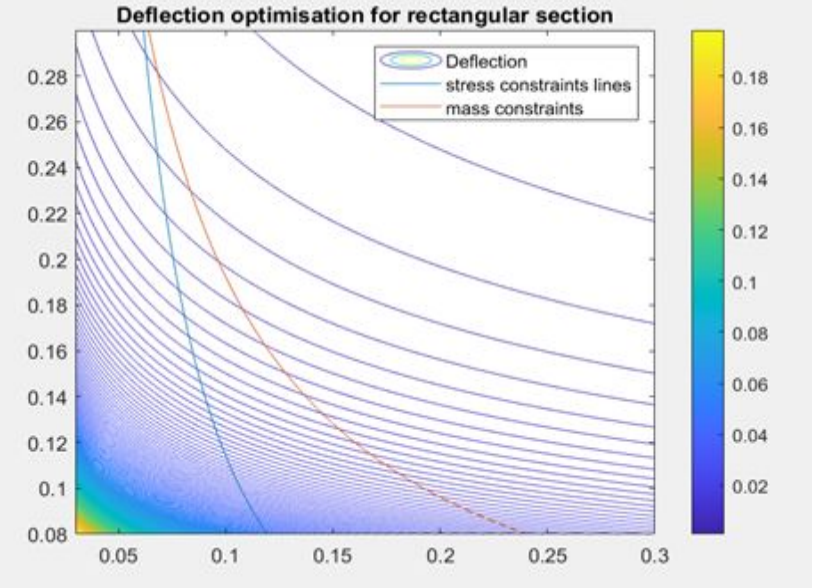


Figure 12: Deflection optimization for rectangular section

The desired solution is between the stress constraints lines and mass constraints lines.

For the Deflection contour line similarly, we got a relationship between b and h as:

$$b = \frac{4FL^3}{h\Delta_{max}}$$

$$b = \frac{m_a}{\rho h L}$$

HOLLOW DISC

The results plot for the hollow disc are shown in figure 13:

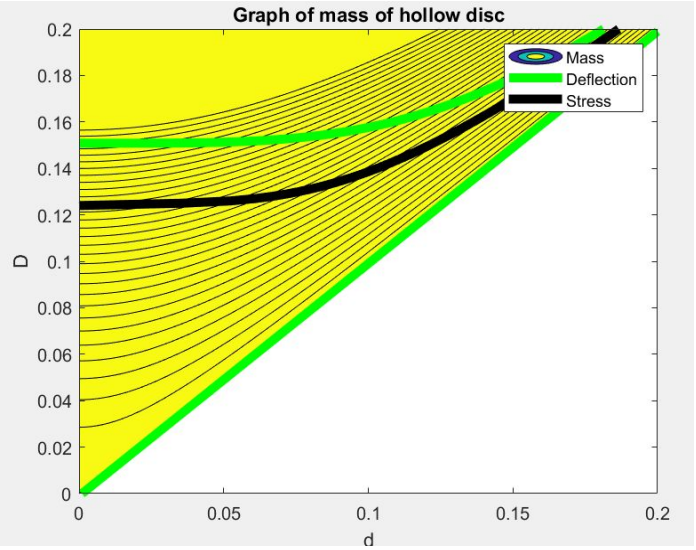


Figure 13: Mass plot hollow disc

For plotting hollow disc, we found an easier way to plot the contour constraints using the Matlab function "contour":

```
contour(d1,D1,Deflection,[def_a def_a],'g-','LineWidth',5);
```

The above code only plots the deflection contour lines when the deflection is equal to admissible deflection. In this manner we were able to easily plot the contour lines.

But for plotting the stress contour we were experimenting with it to plot with the previous equation method as in rectangular section. Interestingly, we came across imaginary numbers in our set of plotting values. What MATLAB did was, it plotted the points ignoring the imaginary value of the coordinates .i.e. 10+0.001i is an imaginary number so it plots at point 10 without considering 0.001. So what we did was we made an new array and imported only the non imaginary number in to this array and plotted only the non imaginary numbers, as shown in figure 14:

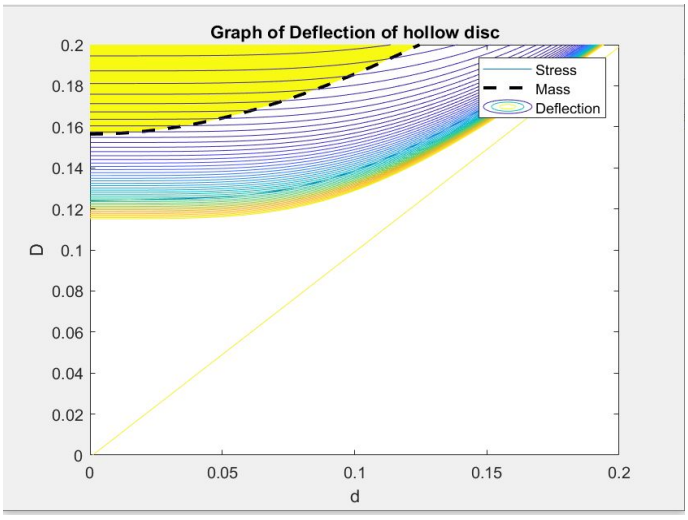


Figure 14: Deflection hollow disc

For the deflection graph we used the contour to plot the constraints as similar to the previous methods.

PLUS SECTION

The mass contour lines lie in the 2 regions as shown in figure 15:

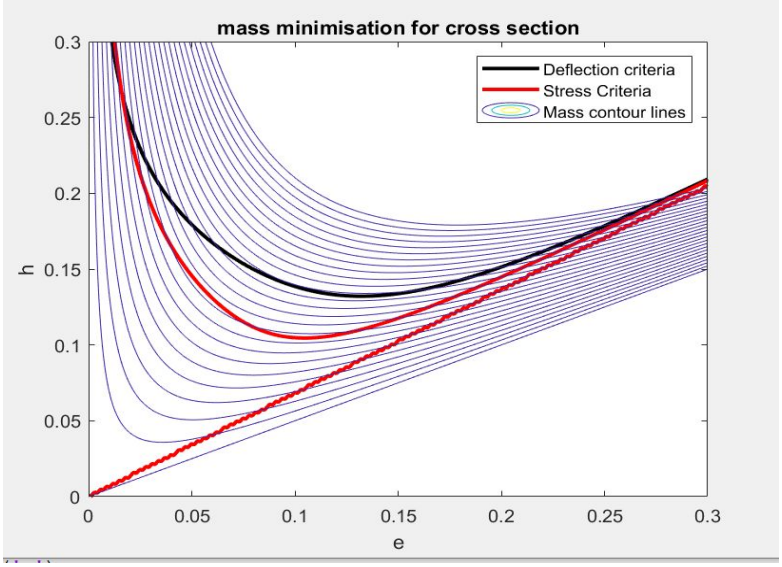


Figure 15: Mass minimisation for cross section

Region 1: As e >> h, the numerator for sigma_max is relatively small and the denominator for sigma_max is relatively large. Hence, sigma_max is well below the max admissible stress. For the same e, as h steadily increases, sigma_max increases up to the stress contour line, beyond which, it exceeds admissible stress.

We plotted the graph using “contour”. As an optimization result we got h=0.0127 and e=0.3000. The red line is the stress constraint line while the black is the deflection constraint.

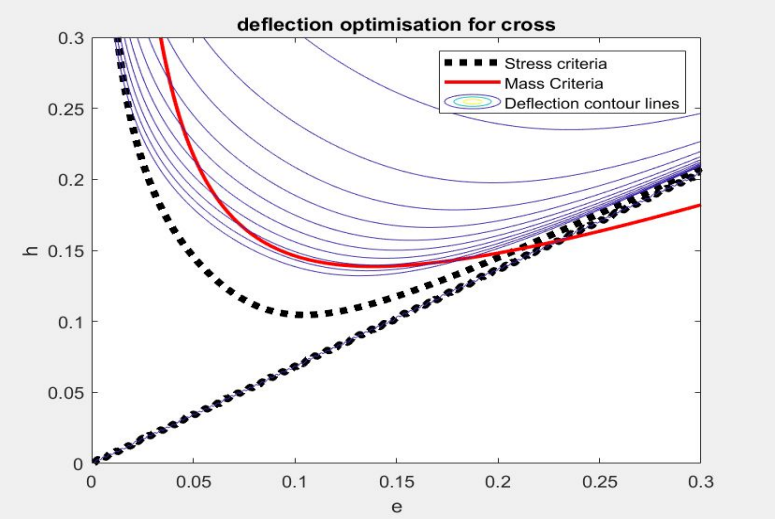


Figure 16: Deflection optimization for cross section

The mass contour lines lie in the 2 regions shown in figure 16.

Region 1: As e >> h, the numerator for sigma_max is relatively small and the denominator for sigma_max is relatively large. Hence, sigma_max is well below the max admissible stress. For the same e, as h steadily increases, sigma_max increases up to the stress contour line, beyond which, it exceeds admissible stress.

HOLLOW SQUARE

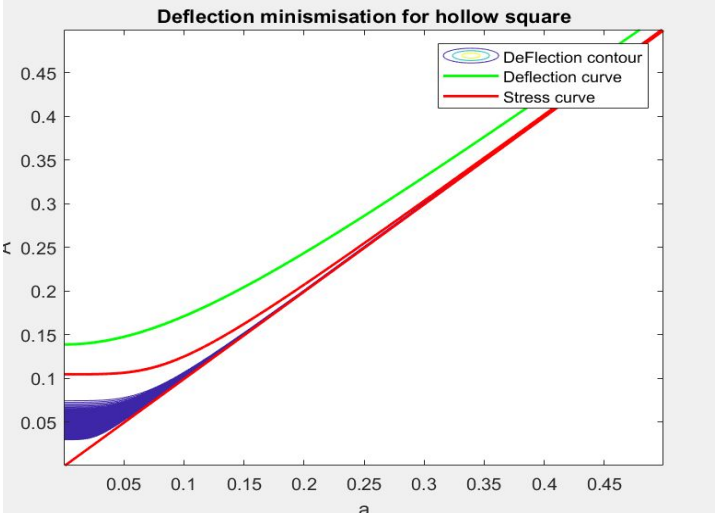


Figure 17: Deflection minimisation for hollow square

Figure 17 shows the plot for deflection minimisation.As done before we plotted it using contour.Desired solution of the problem is between the deflection curve (green) and stress curve(red). The blue lines are the contour lines. Mass contour is decreasing downwards as A drops since there is less material. However it cannot drop below the deflection constraint line because beyond this I is too low and hence deflection exceeds allowable deflection.

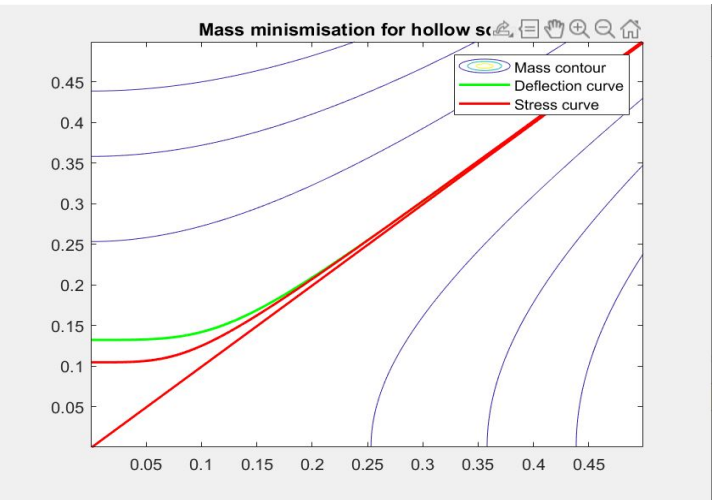


Figure 18: Mass minimisation for hollow square

Figure 18 shows the plot for mass minimisation. As done for the previous graphs this was also plotted using the “contour” command.

2/ Problem with four variables (Hollow square, and IPN)

Use MATLAB to find the optimum (select the appropriate lower and upper bounds), for the two minimization problems (mass and deflection).

Objective of the function was to optimize the dimension for the two geometries: The hollow square and I section.

Hollow square:

The results for mass minimisation are:
b=0.0723 h= 0.2250 B=0.0846 H=0.2135

The results for deflection minimisation are:
B=0.0500 H= 0.2000 b=0.0300 h= 0.1000

I section:

The results for mass minimisation are:
B=0.0495 H= 0.2010 b=0.0304 h=0.1006

The results for deflection minimisation are:
B= 0.0500 H= 0.2000 b=0.0300 h=0.1000

Exercise 5 - Trajectory optimization

5.1 Problem Definition

Consider a room of size 10x10. The room has obstacles in the form of four circles of definite radii and centers. Two rectangular walls, each of size 4x2 with their origins at (4, 0) and (4, 6) separates the test region into two rooms. A trajectory AB has to travel from the left side of the room (preferably from the top) to the right side of the room (preferably to the bottom). The trajectory has to pass through 4 to 7 intermediate points before it has to reach its destination B. The objective of the project is to optimize this trajectory containing the intermediate points (guess points) in such a way that they pass either tangentially to the circles or away from the circles as well as pass in between the walls as it tries to go from the left to the right side of the room. The discretization has to be done for each individual segment of the trajectory with n number of points. For example, if there is one guess point, the discretization has to be done between A and the guess point followed by discretization between the guess point and final point B. If multiple guess points are initialized, successful discretization on the segments between the guess points has to be done.

The objective is to find the best trajectory between point A and B avoiding obstacles like circles and two rectangles representing the walls separating two rooms.

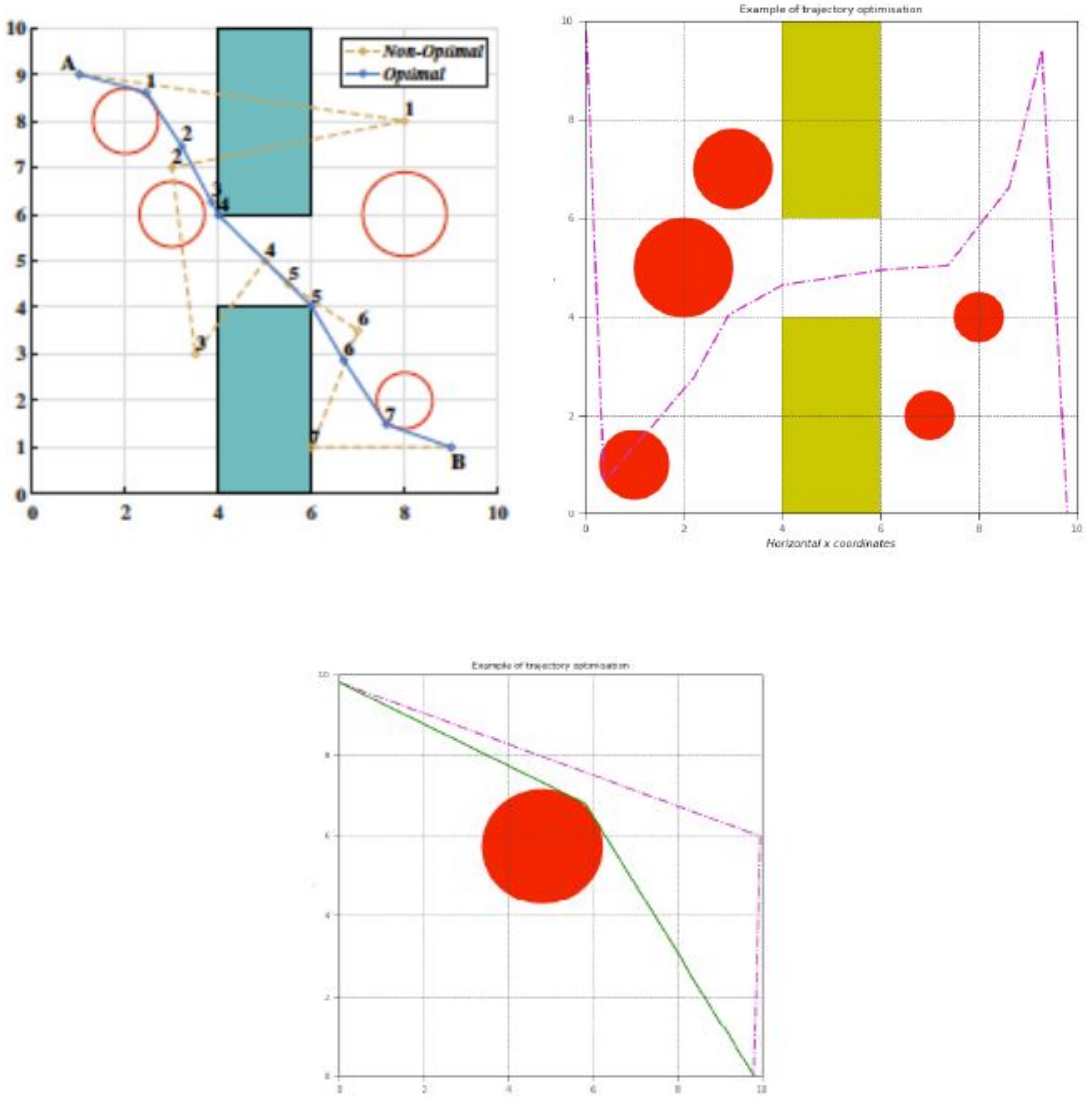


Figure 19: Trajectory optimization problem

5.2 Case - 1

For a given point A(1, 5) and B(9,5) and a circle centered at (5,5) with r = 2, find the best trajectory from A to B avoiding the circle (no wall) :

a/ The trajectory can be represented by two segments;

b/ the trajectory can be represented by 3 segments;

c/ and n segments.

5.2.1 Objective

Our objective is to travel from A to B at a minimum distance through one point. We divide the trajectory into two lines Ax and xB. Summing up the lengths of these lines will give us the least distance which is the Euclidean distance from each segment. We can find the implementation code in the name of objective function. Norm is a function that we use to find the distance between two points in MATLAB.Hence the objective function for this case can be defined as :

$$f(x) = y = \sqrt{(A-x)^2} + \sqrt{(x-B)^2}$$

Figure 20 shows the objective function:

```
function f=obj(x,A,B)

f=norm(A-x)+norm(B-x);

end
```

Figure 20: Objective function trajectory problem

5.2.2 Constraints

It is required that the minimum distance of the line segment Ax and xB from the circle should be greater than or equal to the radius of the circle. Figure 21 shows the constraint function we use in our program.

```
function [g,h]=constr(x,A,B,C,R)
      n = 20;
     %g is unequality constraints
     g(1)=R-distance_segment_circle(A,x,C,n);
    g(2)=R-distance_segment_circle(x,B,C,n);
       h=[]; %Equality constraints
end
```

Figure 21: Constraints trajectory problem

5.2.3 Discretization

We have used the discretization function, to discretize the line segment, in order to minimize the distance between the discretized point and the circle center. We use a loop in our program to decrease the length of the function, as shown in figure 22:

```
function D=distance_segment_circle(A,B,C,n)
% A et B two point of the segement
% C center of the circle R the radius
% n number of discretization
for i=0:n
 M=A+(i/n)*(B-A); % M is a given point between A and B
 d(i+1)=norm(M-C);
end
D=min(d);
 end
```

Figure 22: Discretization trajectory problem

5.2.4 Main function

We use the function **'Fmincon'** to run the program in the main function. Fmincon is a non-linear programming solver. It is a gradient - based method that is designed to work on problems where the objective and constraint functions are both continuous and have continuous first derivatives. The main function has access to the objective and the constraint functions. It also has the codes for plotting. The main function is in Appendix E.

5.2.5 Results

In figure 23, we can see that the sum of the two line segments for the initial path (blue dotted line) is way larger than that for the final optimized trajectory represented by the solid red lines tangent to the circle. The Euclidean sum of the red lines represents the optimized path. Therefore, we were able to find the optimized path using 1 circle and 1 intermediate point.

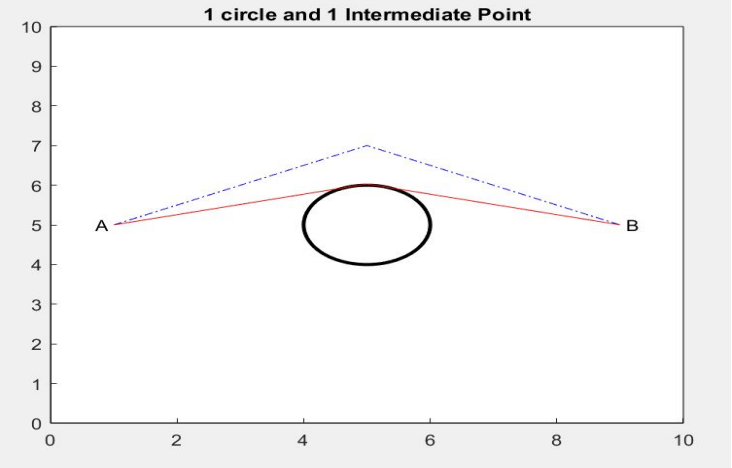


Figure 23: Results trajectory problem

5.3 Case - 2

For a given point A and B with one circle and two intermediate points, find the best trajectory from A to B avoiding the circles (no wall)

5.3.1 Objective

To travel from point A to B at a minimum distance through a given point by avoiding crossing over a circle and to find the best trajectory. The objective function is shown in figure 24:

```
function f=obj(x,A,B)

f=norm(A-x(1,:)) + norm(x(1,:)-x(2,:)) + norm(x(2,:)-B);

end
```

Figure 24: Objective function trajectory problem case 2

5.3.2 Constraints

It is required that the minimum distance of each line segment between A and B from the circle center should be greater than or equal to the radius of the circles. Figure 25 shows the implemented constraint code in our program. Since there line segments there are three constraints (constraints of line segments AX1,X1X2,X2B):

```
function [g,h]=constr(x,A,B,C,R)
     n = 20;
    %g is unequality constraints
    %X=[A; x; B]
   g(1)=R-distance_segment_circle(A,x(1,:),C,n);
   g(2)=R-distance_segment_circle(x(1,:),x(2,:),C,n);
   g(3)=R-distance_segment_circle(x(2,:),B,C,n);
    h=[]; %Equality constraints
end
```

Figure 25: Constraints trajectory problem case 2

5.3.3 Discretization

We have used the discretization function, to discretize the line segment, in order to minimize the distance between the discretized point and the circle centers, as shown in figure 26:

```
 function D=distance_segment_circle(A,B,C,n)
% A et B two point of the segement
% C center of the circle R the radius
% n number of discretization
for i=0:n
 M=A+(i/n)*(B-A); % M is a given point between A and B
 d(i+1)=norm(M-C);
end
D=min(d);
 end
```

Figure 26: Discretization trajectory problem case 2

5.3.4 Main function

We have added one more circle. The main function has access to the objective and the constraint functions. It also has the codes for plotting and can be found in Appendix E.

5.3.5 Results

In figure 27, we can see that the sum of the two line segments for the initial path (blue dotted line) is way larger than that for the final optimized trajectory represented by the solid red lines tangent to the circle. The Euclidean sum of the red lines represents the optimized path. Therefore, we were able to find the optimized path using a circle and two intermediate points.

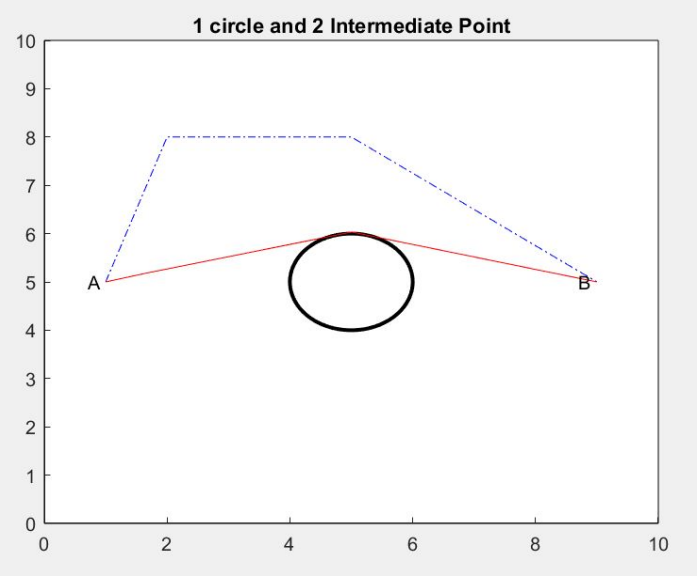


Figure 27: Results trajectory problem case 2

5.4 Case - 3

For a given point A and B and a 2 circles, find the best trajectory from A to B avoiding the circles (no wall).

5.4.1 Objective

To travel from point A to B at a minimum distance through a given point by avoiding crossing over **"n"** circles and to find the best trajectory. Our objective function is defined as shown in figure 28:

```
function f = obj(x,A,B)
pts = [A;x;B];
f = 0;
for i=1:length(pts)-1
f =f+norm(pts(i,:)-pts(i+1,:));
end

end
```

Figure 28: Objective function trajectory problem case 3

5.4.2 Constraints

We are supposed to minimize the distance of each line segment between A and B from the circle centers should be greater than or equal to the radius of the circles. The constraint function is as shown in figure 29:

```
function [g,h]=constr(x,A,B,C,R)
     n = 20;
     pts = [A;x;B];
    %g is unequality constraints
    %X=[A; x; B]
    for j=1:size(C,1)

    for i=1:length(pts)-1
   %progressively gives constraints to all the circles
   if (j==2)%constraints  for the second circle
       g(i+5)=R(j)-distance_segment_circle(pts(i,:),pts(i+1,:),C(j,:),n);
   else %constraints  for the first circle
       g(i)=R(j)-distance_segment_circle(pts(i,:),pts(i+1,:),C(j,:),n);

   end

    end
    end

    h=[]; %Equality constraints
end
```

Figure 29: Constraints trajectory problem case 3

5.4.3 Discretization

We have used the discretization function shown in figure 30, to discretize the line segment, in order to minimize the distance between the discretized point and the circle centers, where 'n' represents the number of discretizations.

```
function D=distance_segment_circle(A,B,C,n)
% A et B two point of the segement
% C center of the circle R the radius
% n number of discretization
for i=0:n
 M=A+(i/n)*(B-A); % M is a given point between A and B
 d(i+1)=norm(M-C);
end
D=min(d);
 end
```

Figure 30: Constraints trajectory problem case 3

5.4.4 Main function

We have added one more circle. The main function has access to the objective and the constraint functions. It also has the codes for plotting and it can be found in Appendix E.

5.4.5 Results

In figure 31, we can see that the sum of the two line segments for the initial path (blue dotted line) is way larger than that for the final optimized trajectory represented by the solid red lines tangent to the circle. The Euclidean sum of the red lines represents the optimized path. Therefore, we were able to find the optimized path using 2 circles and 1 intermediate point.

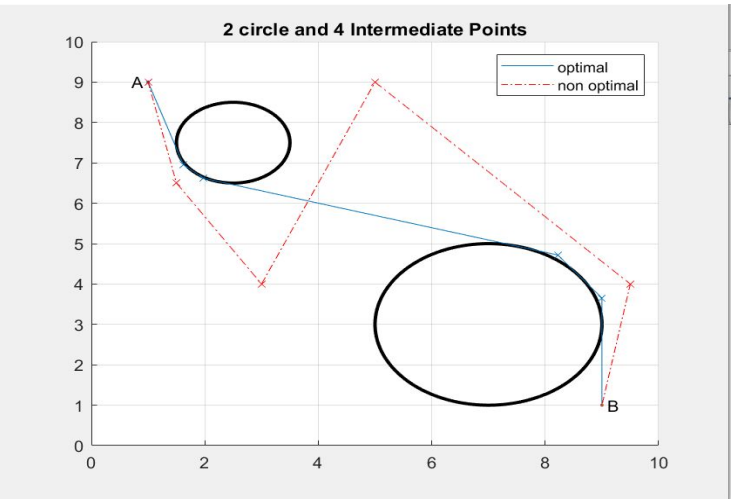


Figure 31: Results trajectory problem case 3

5.5 Case - 4

For a given point A and B and a n circles, find the best trajectory from A to B avoiding the circles and the wall.

5.5.1 Objective

To travel from point A to B at a minimum distance through a given point by avoiding crossing over the 2 circles and to find the best trajectory, as shown in figure 32:

```
function f = obj(x,A,B)
pts = [A;x;B];
f = 0;
for i=1:length(pts)-1
f =f+norm(pts(i,:)-pts(i+1,:));
end

end
```

Figure 32: Objective function trajectory problem case 4

5.5.2 Constraints

It is required that the minimum distance of each line segment between A and B from the circle center should be greater than or equal to the radius of the circles. Figure 33 shows the implemented constraint code in our program.

```
function [g,h]=constr(x,A,B,C,R)
     n = 20;
     pts = [A;x;B];
    %g is unequality constraints
    %X=[A; x; B]

    for j=1:size(C,1)

    for i=1:length(pts)-1


       g(i+5*j)=R(j)-distance_segment_circle(pts(i,:),pts(i+1,:),C(j,:),n);

    end
    end

    h=[]; %Equality constraints
end
```

Figure 33: Constraints trajectory problem case 4

5.5.3 Discretization

We have used the discretization function, to discretize the line segment, in order to minimize the distance between the discretized point and the circle centers, as shown in figure 34:

```
function D=distance_segment_circle(A,B,C,n)
% A et B two point of the segement
% C center of the circle R the radius
% n number of discretization
for i=0:n
 M=A+(i/n)*(B-A); % M is a given point between A and B
 d(i+1)=norm(M-C);
end
D=min(d);
 end
```

Figure 34: Discretization trajectory problem case 4

5.5.4 Main function

We have added one more circle. The main function has access to the objective and the constraint functions. It also has the codes for plotting. We restricted the plotting in the regions of the wall by constraining the upper bound and lower bound, as commented in the code, which can be found in Appendix E.

5.5.5 Results

In figure 35, we can see that the sum of the two line segments for the initial path (red dotted line) is way larger than that for the final optimized trajectory represented by the solid blue lines tangent to the circle. The Euclidean sum of the blue lines represents the optimized path. Therefore, we were able to find the optimized path using **n** circles and **4** intermediate points.

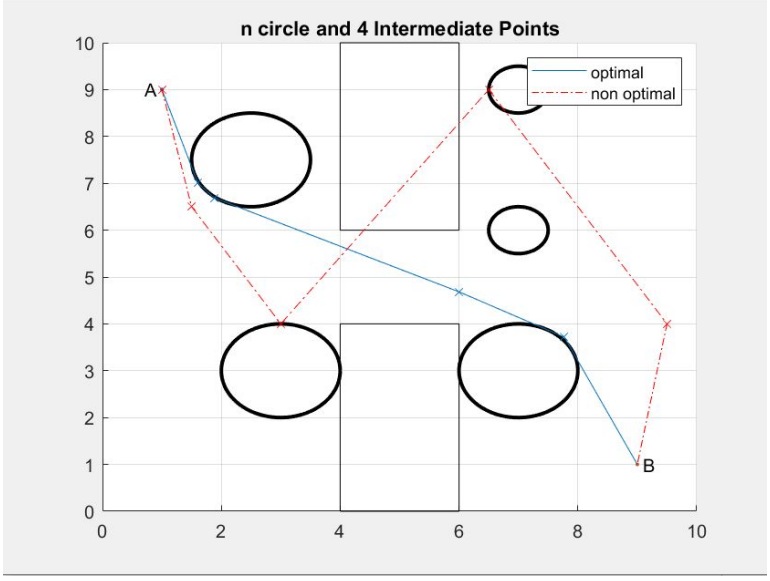


Figure 34: Results trajectory problem case 4

5.6 Conclusions

- It is important to guess proper values to avoid bad results;
- We can get better trajectory path by increasing the number of discretization points;
- We can get smooth curves as we increase the number of points.

Exercise 6: Antenna problem

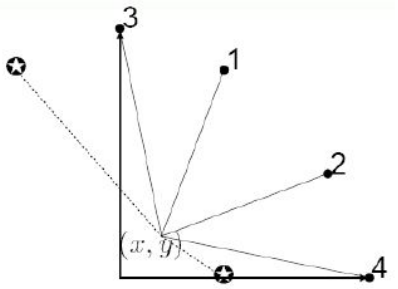
6.1 Problem definition

Find the best positioning of an antenna allowing the connection of 4 new customers, as shown in figure 35:

Priority for the "best" customers

Existing antennas: (-5,10) and (5,0)

Interdiction to place the new antenna less than 10km from the existing one.



Client	Coordinates of the client	Consummation hours
1	(5,10)	200
2	(10,5)	150
3	(0,12)	200
4	(12,0)	300

Figure 35: Antenna problem

6.2 Objective

We have a situation where a company in the telecommunication industry wants to place an antenna in a neighbourhood at such an optimized location such that the clients with more number of consummation hours benefit the most from the quality of the output generated by the antenna. In this problem, our inputs are the x and y coordinates of the antenna location and output is a cost problem of prioritizing.

6.3 Algorithm

Like in minimum distance optimization problems with obstacles, the algorithm is to find the minimum optimal solution within the frame of constraints yet, with fmincon, the algorithm can fall into a local minima. When the initial guess is changed, the solution yields a different result. To solve this issue, a simple thing can be done.

Aim of the exercise: Keep the new antenna closest to the best customers i.e. most number of consumption hours (hrs).If this is the case, distance to best customer will be smallest Let d(i) = distance to customer i.Let hrs(i) = consumption hrs of customer i.For biggest hrs(i), d(i) should be smallest. Therefore, sum (i=1:4) hrs(i)*d(i) should be the smallest.

6.4 Results and discussions

The contour plot for the problem is shown in figure 36.

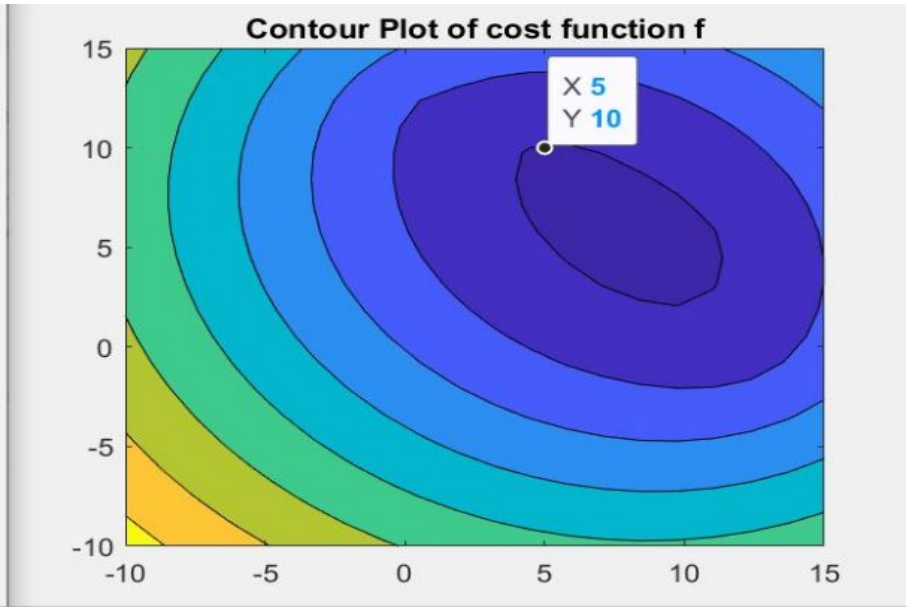


Figure 36: Antenna Problem Result

After computation, we figure optimization the new location of the antenna should be at the coordinates (5,10).

Appendix A Machines and Cost Problem

A.1 Main code

```
%Main function

%x=fmincon(Name of the objective function,...
%x0;...
%{},[],[],[],...
%lb,ub,...

clc
clear
clear all
%x0=[1,0] represents the variables x0(1) and x0(2)
%lb=[min of x0(1) min of x0(2)]
%ub=[max of x0(1) max of x0(2)]

x0=[0,0];
lb=[0,0];
ub=[500,500];
options=optimset('Display','Iter',...
    'TolX',1e-6,...
    'Tolfun',1e-6,...
    'Maxiter',100,...
    'MaxfunEval',300);

x=fmincon(@obj,x0,[],[],[],[],lb,ub,@constr,options)
```

A.2 Constraint function

```
%Constraint function
function [g,h]=constr(x)
    %g is inequality constraints
    g(1)=10*x(1)+5*x(2)-2500;
    g(2)=4*x(1)+10*x(2)-2000;
    g(3)=x(1)+1.5*x(2)-450;
    h=[]; %Equality constraints
end
```

A.3 Objective Function

```
%Objective function
function f=obj(x)
    f=50*x(1)+100*x(2);
    f=-f; %that minus is used to find the max of f.
end
```

Appendix B Single Variable Dichotomous Problem

B.1 Objective function

```
function f=fl(x)
f=(1/3)*(x.^3)-2*(x.^2)+3*x+1;
end
```

B.2 Main function

```
function min=optimex1()
xmin=0;
xmax=100;
ebs=0.0002;
L=xmax-xmin;

while (L>2.*ebs)

   L=xmax-xmin;

   x1=xmin+(L-ebs)/2;
   x2=xmin+(L+ebs)/2;

   if (fl(x1)<fl(x2))
      xmin=xmin;
      xmax=x2;

   else
      xmin=x1;
      xmax=xmax;

   end
end
min=xmin;
end
```

Appendix C Multi variable Dichotomous Problem

C.1 Main code

```
function[xs, fs]= dich2Dfun(X,AlphaMax,AlphaMin,d,eps)
contour()
X0= [0 1];
AlphaMin=-10;
AlphaMax=10;
eps=3;
d=grad(X0);
% X0 = X;
% d=grad(X0);
% eps = e;
% AlphaMax = max;
% AlphaMin = min;
L= AlphaMax-AlphaMin;
while max(d)>eps
   X0=optim(X0,d,AlphaMin,AlphaMax,eps);
   d=grad(X0);
end

i=0;
while L>2*eps&&i<70
   Alpha1= AlphaMin+((L-eps)/2);X1=X0+Alpha1*d;
   Alpha2= AlphaMin+((L+eps)/2);X2=X0+Alpha2*d;
   figure(1);hold on;
   plot(X1(1),X1(2),'*r')
   plot(X2(1),X2(2),'*b')
   f1=equation(X1);f2=equation(X2);

   if f1<f2
      AlphaMax = Alpha2;
   else
      AlphaMin = Alpha1;
   end

   L=AlphaMax-AlphaMin;

end
Alphas=(AlphaMax+AlphaMin)/2;
xs=X0+Alphas*d;
fs=equation(xs);
end
```

C.2 Objective function

```
function f=equation(x)
f=(3*x(1)+2*x(2)-1).^2+(x(1)-x(2)+1).^2;
end
```

Appendix D Cantilever Beam optimization problem

D.1 Deflection minimization for the Disc

```
function test()
%Main function

%x=fmincon(Name of the objective function,...
%x0;...
%{],[],[],[],...
%lb,ub,...

clc
clear
clear all
%x0=[1,0] represents the variables x0(1) and x0(2)
%lb=[min of x0(1) min of x0(2)]
%ub=[max of x0(1) max of x0(2)]
pho=7800;
L=2;
x0=0.1;
lb=0;
ub=1;
F=20000 ;
 E=2.1*10^11;
options=optimset('Display','Iter',...
    'TolX',1e-6,...
    'Tolfun',1e-6,...
    'Maxiter',300,...
    'MaxfunEval',1000);

a=fmincon(@(x)obj(x,F,E,L),x0,[],[],[],[],lb,ub,@(x)constr(x,F,pho),options)

x=0.0001:0.001:0.02;
f=obj(x,F,E,L);
plot(x,f,'--')
xlabel('d')
ylabel('deflection')
title('single variable d deflection minimisation for disc')
end

function f=obj(x,F,E,L)

    f=(4*F*L^3)./(E*x.^4);
end
%Constraint function
function [g,h]=constr(x, F,pho)
    %g is unequality constraints

  L=2;
   sigma=2.1*10^8;
    v=x/2;
    g(1)=(F*L * v/Isq(x)) -sigma;
    g(2)= pho*L*mass(x)-300;



    h=[]; %Equality constraints
end


function I=Isq(x)
I= (pi*x.^4)./64;
end
```

```
function m=mass(x)
m=pi*x.^2./4;
end
```

D.2 Mass minimization for the hollow disc

```
function test()
%Main function

%x=fmincon(Name of the objective functin,...
%x0;...
%{},[],[],[],...
%lb,ub,...

clc
clear
clear all
%x0=[1,0] represents the variables x0(1) and x0(2)
%lb=[min of x0(1) min of x0(2)]
%ub=[max of x0(1) max of x0(2)]
pho=7800;
L=2;
x0=0.6;
lb=0;
ub=10;
F=20000 ;
 E=2.1*10^11;
options=optimset('Display','Iter',...
    'TolX',1e-6,...
    'Tolfun',1e-6,...
    'Maxiter',300,...
    'MaxfunEval',1000);

d=fmincon(@(x) obj(x,pho,L),x0,[],[],[],[],lb,ub,@(x)constr(x, F,E),options)

x=0.0001:0.001:0.02;
f=obj(x,pho,L);
plot(x,f,'--')
xlabel('d')
ylabel('mass')
title('single variable d mass minimsation for disc')

end

function f=obj(x,pho,L)

    f=pho*L*mass(x);
end
%Constraint function
function [g,h]=constr(x, F,E)
    %g is unequality constraints

  L=2;
   sigma=2.1*10^8;
    v=x/2;
    g(1)=(F*L * v/Isq(x))  -sigma;
    g(2)=(4*F*L^3)./(E*x.^4)-0.01;


    h=[]; %Equality constraints
end


function I=Isq(x)
I= (pi*x.^4)./64;
```

```
60. end
61.
62. function m=mass(x)
63. m=pi*x.^2./4;
64. end
```

D.3 Deflection minimization for the hollow square

```
1.  function test()
2.  %Main function
3.
4.  %x=fmincon(Name of the objective functin,...
5.  %x0;...
6.  %{],[],[],[],...
7.  %lb,ub,...
8.
9.  clc
10. clear
11. clear all
12. %x0=[1,0] represents the variables x0(1) and x0(2)
13. %lb=[min of x0(1) min of x0(2)]
14. %ub=[max of x0(1) max of x0(2)]
15. pho=7800;
16. l=2;
17. x0=0.2;
18. lb=0;
19. ub=1;
20. F=20000 ;
21. options=optimset('Display','Iter',...
22.     'TolX',1e-6,...
23.     'Tolfun',1e-6,...
24.     'Maxiter',300,...
25.     'MaxfunEval',1000);
26.
27. a=fmincon(@(x) obj(x, F),x0,[],[],[],[],lb,ub,@(x)constr(x, F),options)
28.
29. [g h ]= constr(a, F)
30. x=0.0001:0.001:0.02;
31. f=obj(x,F);
32. plot(x,f,'--')
33. xlabel('a')
34. ylabel('deflection')
35. title('single variable d deflection minimsation for square')
36. end
37.
38. function f=obj(x, F)
39. % F=20000;
40.  L=2;
41.  E=2.1*10^11;
42.     f=(4*F*L^3)./(E*x.^4);
43. end
44. %Constraint function
45. function [g,h]=constr(x, F)
46.     %g is unequality constraints
47.    pho=7800;
48.    L=2;
49.  %  F=20000;
50.    sigma=2.1*10^8;
51.     v=x/2;
52.     g(1)=(F*L * v/Isq(x))  -sigma;
53.     g(2)=pho*L*mass(x)-300;
54.
55.     h=[]; %Equality constraints
56. end
57.
```

```
58.
59. function I=Isq(x)
60. I= (x^4)/12;
61. end
62.
63. function m=mass(x)
64. m=x^2;
65. end
```

D.4 Mass minimization for the square

```
1.  function test()
2.  %Main function
3.
4.  %x=fmincon(Name of the objective functin,...
5.  %x0;...
6.  %{},[],[],[],...
7.  %lb,ub,...
8.
9.  clc
10. clear
11. clear all
12. %x0=[1,0] represents the variables x0(1) and x0(2)
13. %lb=[min of x0(1) min of x0(2)]
14. %ub=[max of x0(1) max of x0(2)]
15. pho=7800;
16. L=2;
17. x0=0.6;
18. lb=0;
19. ub=10;
20. F=20000 ;
21.  E=2.1*10^11;
22. options=optimset('Display','Iter',...
23.     'TolX',1e-6,...
24.     'Tolfun',1e-6,...
25.     'Maxiter',300,...
26.     'MaxfunEval',1000);
27.
28. a=fmincon(@(x) obj(x,pho,L),x0,[],[],[],[],lb,ub,@(x)constr(x, F,E),options)
29.
30. x=0.0001:0.001:0.02;
31. f=obj(x,pho,L);
32. plot(x,f,'--')
33. xlabel('a')
34. ylabel('mass')
35. title('single variable d mass minimisation for square')
36.
37. end
38.
39. function f=obj(x,pho,L)
40.
41.     f=pho*L*mass(x);
42. end
43. %Constraint function
44. function [g,h]=constr(x, F,E)
45.     %g is inequality constraints
46.
47.   L=2;
48.    sigma=2.1*10^8;
49.     v=x/2;
50.     g(1)=(F*L * v/Isq(x))  -sigma;
51.     g(2)=(4*F*L^3)./(E*x.^4)-0.01;
52.
53.
54.     h=[]; %Equality constraints
```

```
end

function I=Isq(x)
I= (x^4)./12;
end

function m=mass(x)
m=x.^2;
end
```

Appendix E Trajectory Optimization

Case 1 A single point and a given circle (no wall)

```
function test()
%Main function
clc
clear
clear all
lb=[0 0 ];
ub=[10 10];
 A=[1,5];
 B=[9,5];
 P1=[5 7];
 x0 = [P1];
 C=[5,5];
 R=1;
options=optimset('Display','Iter',...
    'TolX',1e-8,...
    'Tolfun',1e-10,...
    'Maxiter',100,...
    'MaxfunEval',300);

x= fmincon(@(x) obj(x,A,B),x0,[],[],[],[],lb,ub,@(x)constr(x,A,B,C,R),options)
[g h]= constr(x,A,B,C,R)
figure(1); clf;
hold on
rectangle('Position',[0 0 10 10])
X0=[A; x0; B]
plot(X0(:,1), X0(:, 2), 'b-.')
text(A(1)-0.3,A(2),'A')
viscircles(C,R,'color','k')
X=[A; x; B]
plot(X(:,1), X(:, 2), 'r-')
text(B(1)+0.1,B(2),'B')

hold on
title('1 circle and 1 Intermediate Point')

end


function f=obj(x,A,B)

f=norm(A-x)+norm(B-x);

end


 function D=distance_segment_circle(A,B,C,n)
% A et B two point of the segement
% C center of the circle R the radius
% n number of discretization
for i=0:n
 M=A+(i/n)*(B-A); % M is a given point between A and B
 d(i+1)=norm(M-C);
end
D=min(d);
 end
function [g,h]=constr(x,A,B,C,R)
    n = 20;
   %g is unequality constraints
   g(1)=R-distance_segment_circle(A,x,C,n);
  g(2)=R-distance_segment_circle(x,B,C,n);
     h=[]; %Equality constraints
```

```
62. end
```

Case 2 A single given point and 2 circles (no wall)

```
1.  function test()
2.  %Main function
3.
4.  %x=fmincon(Name of the objective functin,...
5.  %x0;...
6.  %{],[],[],[],...
7.  %lb,ub,...
8.
9.  clc
10. clear
11. clear all
12. %x0=[1,0] represents the variables x0(1) and x0(2)
13. %lb=[min of x0(1) min of x0(2)]
14. %ub=[max of x0(1) max of x0(2)]
15. pho=7800;
16. l=2;
17.
18. lb=[0 0 ;0 0 ];
19. ub=[10 10; 10 10];
20.
21. %Start, end points
22.  A=[1,5];
23.  B=[9,5];
24.  P1=[2 8]; P2=[5 8] ;
25.  x0 = [P1 ; P2];
26.  C=[5,5];
27.  R=1;
28.
29. options=optimset('Display','Iter',...
30.     'TolX',1e-8,...
31.     'Tolfun',1e-10,...
32.     'Maxiter',100,...
33.     'MaxfunEval',300);
34.
35. x = fmincon(@(x) obj(x,A,B),x0,[],[],[],[],lb,ub,@(x)constr(x,A,B,C,R),options)
36.
37. [g h]= constr(x,A,B,C,R)
38. figure(1); clf;
39. hold on
40. rectangle('Position',[0 0 10 10])
41. X0=[A; x0; B]
42. plot(X0(:,1), X0(:, 2), 'b-.')
43. text(A(1)-0.3,A(2),'A')
44. viscircles(C,R,'color','k');
45. X=[A; x; B]
46.  plot(X(:,1), X(:, 2), 'r-')
47. text(B(1)-0.3,B(2),'B')
48. title('1 circle and 2 Intermediate Point')
49. hold on
50.
51. end
52. function f=obj(x,A,B)
53.
54. f=norm(A-x(1,:)) + norm(x(1,:)-x(2,:)) + norm(x(2,:)-B);
55.
56. end
57.  function D=distance_segment_circle(A,B,C,n)
58. % A et B two point of the segement
59. % C center of the circle R the radius
60. % n number of discretization
61. for i=0:n
```

```
M=A+(i/n)*(B-A); % M is a given point between A and B
d(i+1)=norm(M-C);
end
D=min(d);
 end
function [g,h]=constr(x,A,B,C,R)
    n = 20;
   %g is unequality constraints
   %X=[A; x; B]
  g(1)=R-distance_segment_circle(A,x(1,:),C,n);
  g(2)=R-distance_segment_circle(x(1,:),x(2,:),C,n);
  g(3)=R-distance_segment_circle(x(2,:),B,C,n);
   h=[]; %Equality constraints
end
```

Case 3 A single given point and a set of 'n' circles (no wall)

```
function test()
close all;
clear all;
clc;
xlim([0 10]);
ylim([0 10]);
grid on;
hold on;

P1=[1,9];
P2=[9,1];


C1=[2.5,7.5];
C2=[7,3];
C = [C1;C2];
Radius=[1,2];
viscircles(C,Radius,'color','k');
hold on
plot(P1(1),P1(2),'.');
text(P1(1)-0.3,P1(2),'A')
plot(P2(1),P2(2),'.');
text(P2(1)+0.1,P2(2),'B')
title('2 circle and 4 Intermediate Points')
hold on;

x0=[1.5,6.5;3,4;5,9;9.5,4];
lb=[0,0;0,0;0,0;0,0];
ub=[10,10;10,10;10,10;10,10];

options =
optimset('Display','Iter','Tolx',1.e-6,'TolFun',1.e-6,'MaxIter',1000,'MaxFunEval',1000,'Algorithm
', 'sqp');
x=fmincon(@(x)obj(x,P1,P2),x0,[],[],[],[],lb,ub,@(x)constr(x,P1,P2,C,Radius),options);
disp(x);

path = [P1;x;P2];
initial_path = [P1;x0;P2];
for i=1:size(path,1)-1
a=plot([path(i,1) path(i+1,1)],[path(i,2) path(i+1,2)],'Color',[0 0.4470 0.7410]);
plot(path(i,1),path(i,2),'x','Color',[0 0.4470 0.7410]);
b=plot([initial_path(i,1) initial_path(i+1,1)],[initial_path(i,2) initial_path(i+1,2)],'-.r');
plot(initial_path(i,1),initial_path(i,2),'xr');
legend([a,b],'optimal','non optimal')
hold on;

end
figure(1);
```

```
47.
48. end
49.
50. function f = obj(x,A,B)
51. pts = [A;x;B];
52. f = 0;
53. for i=1:length(pts)-1
54. f =f+norm(pts(i,:)-pts(i+1,:));
55. end
56.
57. end
58.  function D=distance_segment_circle(A,B,C,n)
59. % A et B two point of the segement
60. % C center of the circle R the radius
61. % n number of discretization
62. for i=0:n
63.  M=A+(i/n)*(B-A); % M is a given point between A and B
64.  d(i+1)=norm(M-C);
65. end
66. D=min(d);
67.  end
68. function [g,h]=constr(x,A,B,C,R)
69.      n = 20;
70.      pts = [A;x;B];
71.     %g is inequality constraints
72.     %X=[A; x; B]
73.     for j=1:size(C,1)
74.
75.     for i=1:length(pts)-1
76.    %progressively gives constraints to all the circles
77.    if (j==2)%constraints  for the second circle
78.        g(i+5)=R(j)-distance_segment_circle(pts(i,:),pts(i+1,:),C(j,:),n);
79.    else %constraints  for the first circle
80.        g(i)=R(j)-distance_segment_circle(pts(i,:),pts(i+1,:),C(j,:),n);
81.
82.    end
83.
84.     end
85.     end
86.
87.     h=[]; %Equality constraints
88. end
```

Case 4 A single given point and ‘n’ circles with wall

```
1.  function test()
2.  close all;
3.  clear all;
4.  clc;
5.  xlim([0 10]);
6.  ylim([0 10]);
7.  grid on;
8.  hold on;
9.
10. P1=[1,9];
11. P2=[9,1];
12.
13.
14. C1=[2.5,7.5];
15. C2=[7,3];
16. C3=[3,3];
17. C4=[7,6];
18. C5=[7,9];
19. C = [C1;C2;C3;C4;C5];
20. Radius=[1,1,1,0.5,0.5];
```

```
viscircles(C,Radius,'color','k');
hold on
plot(P1(1),P1(2),'.');
text(P1(1)-0.3,P1(2),'A')
plot(P2(1),P2(2),'.');
text(P2(1)+0.1,P2(2),'B')
title('n circle and 4 Intermediate Points')
hold on;
pos=[4 0 2 4];
po= [4 6 2 4];
rectangle('position',pos);
rectangle('position',po);
x0=[1.5,6.5;3,4;6.5,9;9.5,4];
lb=[0,0;0,0;4,4;6,0];% the point in the wall restriction are restricted here
ub=[4,10;4,10;6,6;10,10];

options =
optimset('Display','Iter','Tolx',1.e-6,'TolFun',1.e-6,'MaxIter',1000,'MaxFunEval',1000,'Algorithm
', 'sqp');
x=fmincon(@(x)obj(x,P1,P2),x0,[],[],[],[],lb,ub,@(x)constr(x,P1,P2,C,Radius),options);
disp(x);

path = [P1;x;P2];
initial_path = [P1;x0;P2];
for i=1:size(path,1)-1
a=plot([path(i,1) path(i+1,1)],[path(i,2) path(i+1,2)],'Color',[0 0.4470 0.7410]);
plot(path(i,1),path(i,2),'x','Color',[0 0.4470 0.7410]);
b=plot([initial_path(i,1) initial_path(i+1,1)],[initial_path(i,2) initial_path(i+1,2)],'-.r');
plot(initial_path(i,1),initial_path(i,2),'xr');
legend([a,b],'optimal','non optimal')
hold on;

end
figure(1);

end

function f = obj(x,A,B)
pts = [A;x;B];
f = 0;
for i=1:length(pts)-1
f =f+norm(pts(i,:)-pts(i+1,:));
end

end
 function D=distance_segment_circle(A,B,C,n)
% A et B two point of the segement
% C center of the circle R the radius
% n number of discretization
for i=0:n
 M=A+(i/n)*(B-A); % M is a given point between A and B
 d(i+1)=norm(M-C);
end
D=min(d);
 end
function [g,h]=constr(x,A,B,C,R)
    n = 20;
    pts = [A;x;B];
   %g is unequality constraints
   %X=[A; x; B]

   for j=1:size(C,1)

   for i=1:length(pts)-1


```

```
85.         g(i+5*j)=R(j)-distance_segment_circle(pts(i,:),pts(i+1,:),C(j,:),n);
86.
87.       end
88.       end
89.
90.       h=[]; %Equality constraints
91.   end
```

Appendix F Antenna location problem

E.1 Main Function

```
close all
clc
clear

%Customer locations and their consumption hours
custloc=[5,10,200;
    10,5,150;
    0,12,200;
    12,0,300];

%Antenna location
Antloc=[-5,10;
    5,0];

%Initial guess for new antenna location
x0=[4,6];
lb=[-10,-10];
ub=[15,15];

options=optimset('Display','iter','Tolx',1.e-10,'Tolfun',1.e-10,...
    'MaxIter',200, 'MaxfunEvals',2000);

obj=@(x)mincost_objective(x,custloc);
cons=@(x)constraint(x, Antloc);

x=fmincon(obj,x0,[],[],[],[],lb,ub,cons,options);

display(x);

%to plot the contour
x1=linspace(lb(1),ub(1),20);
x2=linspace(lb(2),ub(2),20);
[X1,X2]=meshgrid(x1,x2);

mincost = zeros(length(x1),length(x2));
%Plot function
for i=1:length(x1)
    for j=1:length(x2)
        mincost(i,j)=mincost_objective([X1(i,j),X2(i,j)],custloc);
    end
end

figure(1); contourf(X1,X2,mincost); hold on; plot(x(1),x(2),'+');
title('Contour Plot of cost function f');
figure(2);
for i=1:size(custloc,1)
    plot(custloc(i,1),custloc(i,2),'*');
    hold on;
end
hold on;
for i=1:size(Antloc,1)
    plot(Antloc(i,1),Antloc(i,2),'bo')
    hold on;
    %Viscircles(Circlecenter,radius)
    viscircles(Antloc(i,:),10,'LineStyle',':','Color','b')
    hold on;
end
plot(x(1),x(2),'+');
title('Plot of Optimal New Antenna Location');
%Also shows Old Antenna locations (o) and customer locations (*)

```

```
figure(3); surf(X1,X2,mincost); hold on; plot3(x(1),x(2),mincost_objective(x,custloc),'+');
title('Surface plot of Antenna Locations')
grid on
```

E.2 Constraint Function

```
%we are trying to  make sure that your new antenna and old antenna are
%more than 10 km apart

function [g,h]=constraint(x, Antloc)

g(1)=10-norm(x-Antloc(1,:));
g(2)=10-norm(x-Antloc(2,:));

h=[];
end
```

E.3 Objective Function

```
function f=mincost_objective(x,custloc)

f=0;
for i=1:length(custloc)
    cost=custloc(i,3)*norm(custloc(i,1:2)-x);
    f=f+cost;
end

end
```