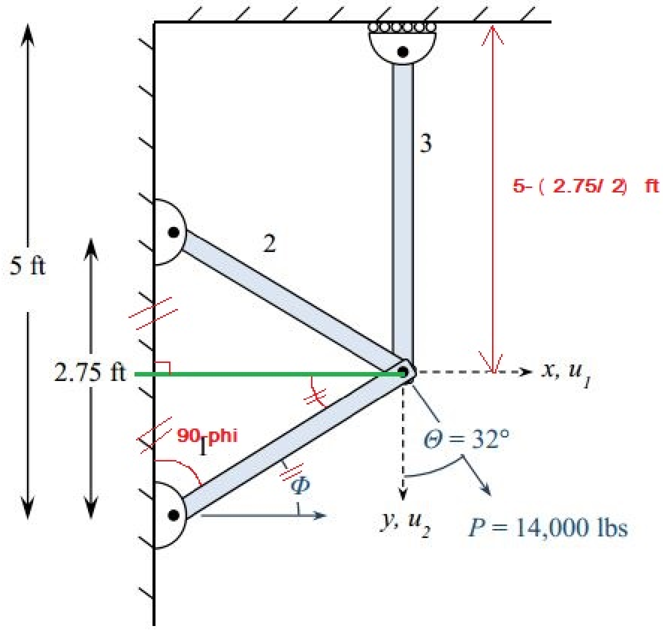
AAE550: HW1

Rahul Deshmukh
deshmuk5@purdue.edu
PUID: 0030004932

October 15, 2018

I Engineering Problem in N Variables

For this problem, We will be assuming small value of displacements u_1 and u_2. With this assumption we now have that ϕ will remain constant through out the problem and the lengths of the bars also remain constant. As shown in the following figure, We can estimate the angle ϕ using simple geometry.



Given, $L_1 = L_2$, therefore perpendicular dropped from their point of intersection will bisect the base. Thus, we obtain the length of bar 3 as $L_3 = 5 - \frac{2.75}{2}\ ft$ and $\phi = \frac{\pi}{2} - cos^{-1}(\frac{2.75/2}{3})$

1) Analytic Gradient and Hessian of the Potential energy function

$$\Pi(u) = \frac{1}{2}u^T K u - p^T u$$

$$\Rightarrow \Pi(\vec{u}) = \frac{1}{2}u_i K_{ij} u_j - p_i u_i$$

Gradient $\vec{\nabla}(\Pi(\vec{u})) = \frac{\partial \Pi(u)}{\partial u_k}\vec{e_k}$

$$\vec{\nabla}(\Pi(\vec{u})) = \frac{\partial}{\partial u_k}(\frac{1}{2}u_i K_{ij} u_j - p_i u_i)\vec{e_k}$$

$$= \frac{1}{2}\frac{\partial}{\partial u_k}(u_i K_{ij} u_j)\vec{e_k} - \frac{\partial}{\partial u_k}(p_i u_i)\vec{e_k}$$

$$= \frac{1}{2}(\delta_{ki} K_{ij} u_j + u_i K_{ij}\delta_{jk})\vec{e_k} - (p_i \delta_{ik})\vec{e_k}$$

$$= \frac{1}{2}(K_{kj} u_j + u_i K_{ik})\vec{e_k} - (p_k)\vec{e_k}$$

$$\Rightarrow \vec{\nabla}(\Pi(\vec{u})) = \frac{1}{2}(K + K^T)\vec{u} - \vec{p}$$

Hessian $H(\vec{u}) = \vec{\nabla} \otimes \vec{\nabla}(\Pi(\vec{u}))$

$$H(\vec{u}) = \frac{\partial}{\partial u_k}(\frac{1}{2}(K_{lj} u_j + u_i K_i l) + p_l)\vec{e_k} \otimes \vec{e_l}$$

$$= (\frac{1}{2}(K_{lj}\delta_{jk} + \delta_{ki} K_{il}))\vec{e_k} \otimes \vec{e_l}$$

$$= \frac{1}{2}(K_{lk} + K_{kl})\vec{e_k} \otimes \vec{e_l}$$

$$\Rightarrow H(\vec{u}) = \frac{1}{2}(K + K^T)$$

Note: For this problem, $K = K_1 + K_2 + K_3$ and all K_i are of the form $K_i = \alpha v v^T$ which means that all K_i will be symmetric (outer product of same vector gives as symmetric matrix) and therefore our global stiffness matrix K will also be a symmetric matrix.

So we can further reduce our gradient and Hessian using the property that$K^T = K$

$$\Rightarrow \vec{\nabla}(\Pi(\vec{u})) = K\vec{u} - \vec{p}$$

$$H(\vec{u}) = K$$

2) Matlab snippet for $\Pi(\vec{u})$ and gradient $\vec{\nabla}(\Pi(\vec{u}))$ and hessian $H(\vec{u})$

Function for evaluating only the Potential Energy

```
%HW1 problem (1)
%Rahul Deshmukh PUID: 0030004932
%deshmuk5@purdue.edu
%----------------begin----------------------%
%energy function
function PE=hw1_p1_PEfun(u)
%input: u is a col vector
%output: PE: scalar value for potential energy function
```

```
%definition of constants: E,A,L,phi,P
E=17.3*10^6;%psi
d1=0.65;%in
A1=pi*(d1/2)^2;%sq in
A2=A1;
d3=0.8;%in
A3=pi*(d3/2)^2;%sq in
fttoin=12;% conversion factor
L1=3*fttoin;
L2=L1;
L3=(5-(2.75/2))*fttoin;
phi=90-(180/pi)*acos((2.75/2)/3);
P=14000;%lbs

theta=32;%degrees
p=P*[sind(theta);cosd(theta)];

K1=[cosd(-1*phi);sind(-1*phi)]*(E*A1/L1)*[cosd(-1*phi),sind(-1*phi)];
K2=[cosd(phi);sind(phi)]*(E*A2/L2)*[cosd(phi),sind(phi)];
K3=[0;sind(90)]*(E*A3/L3)*[0,sind(90)];

K=K1+K2+K3;

PE=(1/2)*u'*K*u-p'*u;

end
```

Function for evaluating only the Potential Energy and its gradient

```
%HW1 problem (1)
%Rahul Deshmukh PUID: 0030004932
%deshmuk5@purdue.edu
%---------------begin----------------------%
%gradient of enrgy function
function [PE,gradPE]=hw1_p1_PEwtgrad(u)
%input: u is a col vector
%output: PE: scalar value of potential energy
%        gradPE: gradient of potential energy function

%definition of constants: E,A,L,phi,P
E=17.3*10^6;%psi
d1=0.65;%in
A1=pi*(d1/2)^2;%sq in
A2=A1;
d3=0.8;%in
A3=pi*(d3/2)^2;%sq in
fttoin=12;% conversion factor
L1=3*fttoin;
L2=L1;
L3=(5-(2.75/2))*fttoin;
phi=90-(180/pi)*acos((2.75/2)/3);
P=14000;%lbs

theta=32;%degrees
```

```
p=P*[sind(theta);cosd(theta)];

K1=[cosd(-1*phi);sind(-1*phi)]*(E*A1/L1)*[cosd(-1*phi),sind(-1*phi)];
K2=[cosd(phi);sind(phi)]*(E*A2/L2)*[cosd(phi),sind(phi)];
K3=[0;sind(90)]*(E*A3/L3)*[0,sind(90)];

K=K1+K2+K3;

PE=(1/2)*u'*K*u-p'*u;
gradPE=(1/2)*(K+K')*u-p;

end
```

Function for evaluating Potential Energy, its gradient and Hessian

```
%HW1 problem (1)
%Rahul Deshmukh PUID: 0030004932
%deshmuk5@purdue.edu
%---------------begin----------------------%
%PE, gradient of PE, and Hessian of PE
function [PE,gradPE,H]=hw1_p1_HessianPEfun(u)
%input: u is a col vector
%output: PE: scalar value of potential energy
%         gradPE: gradient of potential energy function
%         H: Hessina of potential energy function: tensor

%definition of constants: E,A,L,phi,P
E=17.3*10^6;%psi
d1=0.65;%in
A1=pi*(d1/2)^2;%sq in
A2=A1;
d3=0.8;%in
A3=pi*(d3/2)^2;%sq in
fttoin=12;% conversion factor
L1=3*fttoin;
L2=L1;
L3=(5-(2.75/2))*fttoin;
phi=90-(180/pi)*acos((2.75/2)/3);
P=14000;%lbs

theta=32;%degrees
p=P*[sind(theta);cosd(theta)];

K1=[cosd(-1*phi);sind(-1*phi)]*(E*A1/L1)*[cosd(-1*phi),sind(-1*phi)];
K2=[cosd(phi);sind(phi)]*(E*A2/L2)*[cosd(phi),sind(phi)];
K3=[0;sind(90)]*(E*A3/L3)*[0,sind(90)];

K=K1+K2+K3;

PE=(1/2)*u'*K*u-p'*u;
gradPE=(1/2)*(K+K')*u-p;
H=(1/2)*(K+K');
end
```

3) Using *"fminunc"* with different options:

The script for this task is:

```
%HW1 problem (1)
%Rahul Deshmukh PUID: 0030004932
%deshmuk5@purdue.edu

% main file for problem 1
%---------------begin----------------------%
clc; clear all;close all;
format long;
%%
% (3)
% using BFGS
u0=[0;0];%initial guess: assuming at no displacement
%------------------Part A--------------------%
% using finite diffrence gradients and BFGS solver
options_3a=optimoptions('fminunc','Algorithm','quasi-newton',...
    'SpecifyObjectiveGradient',false,'Display','iter');
[u_star,pe_star,exitflag,output,grad,hessian]=fminunc(@hw1_p1_PEfun,...
    u0,options_3a)
%-----------results--------%
% u0=[0;0];u_star=[0.029448184643855;0.044483086047248];
% f(u_star)= -3.733027694338580e+02;
% grad=  1.0e-05*[0;-0.762939453125000]; num_iter=3; funcCount=18; ...
    exitflag=1;
%---------------------------%

%-----------------Part B-----------------------%
%  solve using analytic gradients
options_3b=optimoptions(@fminunc,'Algorithm','quasi-newton',...
    'SpecifyObjectiveGradient', true, 'Display', 'iter');
[u_star,pe_star,exitflag,output,grad,hessian]=fminunc(@hw1_p1_PEwtgrad,...
    u0,options_3b)
%-----------results--------%
% u0=[0;0];u_star=[0.029448192103366;0.044483093518566];
% f(u_star)=  -3.733027694338724e+02;
% grad=[0;0]; num_iter=3; funcCount=6; exitflag=1;
%---------------------------%
```

4) Using *"fminunc"* with different options:

The script for this task is:

```
%%
% (4)
% using DFP and Steepest Descent
u0=[0;0];%initial guess: assuming at no displacement
%------------------Part A--------------------%
% using analytic gradient with DFP update
options_4a=optimoptions(@fminunc,'Algorithm','quasi-newton',...
    'SpecifyObjectiveGradient',true,'Display','iter',...
    'HessUpdate','dfp');
[u_star,pe_star,exitflag,output,grad,hessian]=fminunc(@hw1_p1_PEwtgrad,...
```

```
    u0,options_4a)
%-----------results--------%
% u0=[0;0];u_star=[0.029448192103366;0.044483093518566];
% f(u_star)=  -3.733027694338725e+02;
% grad=[0;0]; num_iter=3; funcCount=6; exitflag=1;
%---------------------------%

%-----------------Part B-----------------------%
%  solve using analytic gradients with Steepest Descent
options_4b=optimoptions(@fminunc,'Algorithm','quasi-newton',...
'SpecifyObjectiveGradient',true,'Display','iter','HessUpdate','steepdesc');
[u_star,pe_star,exitflag,output,grad,hessian]=fminunc(@hw1_p1_PEwtgrad,...
    u0,options_4b)
%-----------results--------%
% u0=[0;0];u_star3a=[0.029448178767296;0.044483073374022];
% f(u_star3a)=  -3.733027694337959e+02;
% grad=[-0.003359750103300;-0.005376640310715]; num_iter=4; ...
    funcCount=25; exitflag=1;
%---------------------------%
```

5) Using *"fminunc"* with different options:

The script for this task is:

```
%%
% (5)
% using newtons method
u0=[0;0];%initial guess: assuming at no displacement

% using newtons mehod with specified hessian  and gradient
options_5=optimoptions(@fminunc,'Algorithm','trust-region',...
    'SpecifyObjectiveGradient',true,'Display','iter',...
    'HessianFcn','objective');
[u_star,pe_star,exitflag,output,grad,hessian]=fminunc(@hw1_p1_HessianPEfun,...
    u0,options_5)
%-----------results--------%
% u0=[0;0];u_star=[0.029448192103366;0.044483093518566];
% f(u_star)=  -3.733027694338725e+02;
% grad= 1.0e-11*[-0.090949470177293;-0.363797880709171]; num_iter=1; ...
    funcCount=2; exitflag=1;
%---------------------------%
```

6) Excel results

Problem setup screen shots:

B6 =(0.5)*($D$15*$B$3^2+($D$16+$E$15)*($B$3*$B$4)+$E$16*$B$4^2)-($D$3*$B$3+$D$4*$B$4)

	A	B	C	D	E	F	G	H	I	J	K
1						HW1 Problem 1 part (6)					
2		u (in)		p (lbf)					constants		
3	u1	0.029448192035596	p1	=$J$7*SIN($J$6)		E	=17.3*10^6	psi	d1	0.65	in
4	u2	0.044483093342771	p2	=$J$7*COS($J$6)		A1	=PI()*($J$3/2)^2	sq in	d3	0.8	in
5						A2	=PI()*($J$3/2)^2	sq in	phi	=PI()/2-ACOS((2.75/2)/3)	radians
6	Pi(u)	=(0.5)*($D$15*B	foot pound force			A3	=PI()*($J$4/2)^2	sq in	theta	=32*(PI()/180)	radians
7						L1	=3*12	in	P	14000	lbf
8						L2	=3*12	in	EA1/L1	=$G$3*$G$4/$G$7	
9						L3	=12*(5-(2.75/2))	in	EA3/L3	=$G$3*$G$6/$G$9	
10	u0										
11	0					sin(phi)	=SIN($J$5)				
12	0					cos(phi)	=COS($J$5)				
13											
14				K		K1		K2		K3	
15				=F15+H15+J15	=G15+I15+K15	=$J$8*$G$12^2	=-$J$8*$G$11*$G$12	=$J$8*$G$12^2	=$J$8*G	0	0
16				=F16+H16+J16	=G16+I16+K16	=$G$15	=$J$8*$G$11^2	=$I$15	=$J$8*G	0	=$J$9

Figure 1: Problem setup with formulas (Note: The formula for the objective function can be seen in the formula bar)

HW1 Problem 1 part (6)										
	u (in)		p (lbf)					constants		
u1	0.029448192	p1	7418.87		E	17300000	psi	d1	0.65	in
u2	0.044483093	p2	11872.7		A1	0.3318307	sq in	d3	0.8	in
					A2	0.3318307	sq in	phi	0.47611906	radians
Pi(u)	-373.3027694	foot pound force			A3	0.5026548	sq in	theta	0.55850536	radians
					L1	36	in	P	14000	lbf
					L2	36	in	EA1/L1	159463.098	
					L3	43.5	in	EA3/L3	199906.401	
u0										
0					sin(phi)	0.4583333				
0					cos(phi)	0.8887804				
			K		K1		K2		K3	
			251930	0	125965	-64958.52	125965	64958.52	0	0
			0	266903	-64959	33498.324	64959	33498.32	0	199906.4015

Figure 2: Problem setup after running the solver

Microsoft Excel 16.0 Answer Report
Worksheet: [P1_part6.xlsx]Sheet1
Report Created: 15-10-2018 12:24:46
Result: Solver has converged to the current solution. All Constraints are satisfied.
Solver Engine
Engine: GRG Nonlinear
Solution Time: 11.64 Seconds.
Iterations: 7 Subproblems: 0
Solver Options
Max Time Unlimited, Iterations Unlimited, Precision 0.000001, Use Automatic Scaling, Show Iteration Results
Convergence 0.0001, Population Size 100, Random Seed 0, Derivatives Forward, Require Bounds
Max Subproblems Unlimited, Max Integer Sols Unlimited, Integer Tolerance 1%

Objective Cell (Min)

Cell	Name	Original Value	Final Value
$B$6	Pi(u) u (in)	0	-373.3027694

Variable Cells

Cell	Name	Original Value	Final Value	Integer
$B$3	u1 u (in)	0	0.029448192	Contin
$B$4	u2 u (in)	0	0.044483093	Contin

Constraints

NONE

Figure 3: Answer report

Microsoft Excel 16.0 Sensitivity Report
Worksheet: [P1_part6.xlsx]Sheet1
Report Created: 15-10-2018 12:24:47

Variable Cells

Cell	Name	Final Value	Reduced Gradient
$B$3	u1 u (in)	0.029448192	0
$B$4	u2 u (in)	0.044483093	0

Constraints
NONE

Figure 4: Sensitivity report

7) Comparison Table:

Method / Program	$x^0(in)$	$x^*(in)$	$f(x^*)$ (foot pound force)	$\nabla f(x^*)$	No. iterations	No. f^n evals	exitflag
BFGS / Matlab *fminunc*, numerical gradient	0 0	0.029448184643855 0.044483086047248	-3.733027694338580e+02	0 -0.762939453125000 e-05	3	18	1
BFGS / Matlab *fminunc*, userdefined gradient	0 0	0.029448192103366 0.044483093518566	-3.733027694338724e+02	0 0	3	6	1
DFP / Matlab *fminunc*, userdefined gradient	0 0	0.029448192103366 0.044483093518566	-3.733027694338725e+02	0 0	3	6	1
Steepest Descent / Matlab *fminunc* user-defined gradient	0 0	0.029448178767296 0.044483073374022	-3.733027694337959e+02	-0.003359750103300 -0.005376640310715	4	25	1
Modified Newton's method/ Matlab *fminunc*, user-defined gradient	0 0	0.029448192103366 0.044483093518566	-3.733027694338725e+02	-0.090949470177293e-11 -0.363797880709171e-11	1	2	1
Quasi-Newton method / Excel Solver	0 0	0.029448192077456 0.044483093495682	-373.3027694	0 0	7	N/A	N/A

Conclusions for Unconstrained Minimization :
Comparison for 1^{st} order methods: BFGS & DFP can be seen to produce the same result with identical number of iterations and function calls. However, when using Steepest Descent we have poor performance as the gradient is still not $\vec{0}$ and the method took 4 iterations with 25 function calls to give a result. Steepest descent is the mostly costly 1^{st} order method.

Gradient- Numerical Vs User-Defined: On comparing the performance of BFGS for the two cases, as expected we get better results with less computational cost for User-Defined gradient. When using numerical gradient, our final gradient value has not reached absolute 0 and evaluating numerical gradient is costly because of which we have 18 function evaluations.

Out of all the methods, the $2nd$ order method is the fastest as we have a gradient value at absolute 0 with only 1 iteration and 2 function evaluations.

Specifically for this problem, as we are able to find out an analytic gradient and hessian for the function, we should have approach the minimization problem directly with a second order method with user defined gradient. However, this is not always the case in general for all the problems.

8) Solution by solving $K\vec{u} = \vec{p}$:

Conditions for Optimality for unconstrained minimization:

(i) Condition on Gradient: $\vec{\nabla} f = \vec{0}$,
(ii) Condition on Hessian $x^T H x \geq 0 \; \forall x \in \Re^n$ i.e. Hessian should be positive definite to have a minima.

For our problem, using the first condition we have:

$$\vec{\nabla}(\Pi(\vec{u})) = \vec{0}$$

$$\Rightarrow K\vec{u} - \vec{p} = \vec{0}$$

$$\Rightarrow K\vec{u} = \vec{p}$$

and as for the second condition, we have $H(\vec{u}) = K$, where K is a symmetric matrix and therefore positive definite. Therefore on solving the equation $K\vec{u} = \vec{p}$, we will get our optimum solution $u^* = \begin{bmatrix} 0.029448192103366 \\ 0.044483093518566 \end{bmatrix}$(in) and $\Pi(\vec{u^*}) = -3.733027694338724e+02$(foot pound force)

The script for this task is:

```
%%
% (8)
```

```
% solving for u using Ku=p
[K,p]=hw1_p1_tangent();
u_star=K\p
%-----------results------------%
% u_star=[0.029448192103366;0.044483093518566];
% in one step.
%--------------------------------%
```

Comments on single step method : The solution obtained with this method is identical to that obtained with BFGS/DFP with user-defined Gradient.

II <u>Engineering Application of SUMT Approach</u>

1) Problem Formulation For this problem, our objective is to minimize the Induced drag with c_r, b, α as design variables. With the list of formulas given in the problem description, I have found out the expressions for objective function and constraints using symbolics in MATLAB. The script for the task is as follows:

```
% AAE:550 HW1 P2
% Rahul Deshmukh
% PUID: 0030004932
%%
clc; clear all;
format long;
%----------------------begin-----------------------------------%
%define symbols for design variables
syms cr b a; %root chord, wing span,angle of attack

%define related independent terms
V=53.6;% m/sec
rho=1.134;%kg/m^3

Tr=0.4; %taper ratio
ct=Tr*cr; % tip chord
C_la=2*pi;% lift curve slope
e=0.9; %efficiency
a_L0= -3;
S=((ct+cr)/2)*b;
AR=b^2/S;

a1=0.14;
k=(1-(1+ (pi*AR/C_la) )*a1)/((1+ (pi*AR/C_la) )*a1);

C_La=C_la/( (1+ (C_la/(pi*AR)) )*(1+k));

C_L=C_La*(pi/180)*(a-a_L0); %need to convert angles to radians here

q=0.5*rho*V^2;

C_Di=(C_L^2)/(pi*AR*e);

%find symbolic expressions for objective and constraints
%objective function min Di
Di=q*S*C_Di;
vpa(simplify(Di))  % 0.033949298754935466466210448881266*b^2*(a + 3.0)^2

% constraint on L≥9500 N
L=q*S*C_L;
vpa(simplify(L)) % 12.50454558912777413506277959447*b^2*(a + 3.0)

%constraint on 0.7=<C_L≤0.9
vpa(simplify(C_L)) % (0.01096622711232150957648276777764*b*(a + 3.0))/cr
```

By executing the above script we get the following expressions:

$$D_i = \gamma_f * b^2 * (\alpha + 3.0)^2$$

$$L = \gamma_L * b^2 * (\alpha + 3.0)$$

$$C_L = \gamma_{lc} * b * (\alpha + 3.0))/cr$$

where,$\gamma_f, \gamma_L, \gamma_{lc}$ are constants with values, $\gamma_f = 0.033949298754935466466210448881266$, $\gamma_L = 12.504545589127774135062779594447$, $\gamma_{lc} = 0.010966227112321509576482767777764$

As we can see from the above equations, for the expression of Lift coefficient(C_L) we have c_r in the denominator, therefore we need to remove the design variable from the denominator as for some of the penalty methods we might have a value of 0 for c_r and then the Lift coefficient(C_L) expression cannot be evaluated.

To avoid the above stated issue I have formulated the problem with new design variable $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{c_r} \\ b \\ \alpha \end{bmatrix}$.

The Optimization problem for design variable $\vec{x}$ then becomes:

a) Objective function

$$\min_{\vec{x}} f(x) = \gamma_f * x_2^2 * (x_3 + 3.0)^2$$

b) Inequality Constraints ($g_j(x)$):

i) Lift Coefficient bounds

$$g_7(x) = \gamma_{lc} x_2 (x_3 + 3) x_1 / 0.9 - 1 \leq 0$$

$$g_8(x) = 1 - (\gamma_{lc} x_2 (x_3 + 3) x_1)/0.7 \leq 0$$

ii) Lift Constraint

$$g_9(x) = 1 - \gamma_L (x_2^2)(x_3 + 3)/9500 \leq 0$$

c) Design variable Bounds

Bound on $x_1 = 1/c_r$

$$g_1(x) = 1 - 3x_1 \leq 0$$

$$g_2(x) = 0.8x_1 - 1 \leq 0$$

Bound on $x_2 = b$

$$g_3(x) = x_2/14 - 1 \leq 0$$

$$g_4(x) = 1 - x_2/8 \leq 0$$

Bound on $x_3 = \alpha$

$$g_5(x) = x_3/10 - 1 \leq 0$$

$$g_6(x) = -1 - x_3/5 \leq 0$$

d) Matlab script for evaluating objective function and constraints:

Script for Objective function:

```
% AAE:550 HW1 P2
% Rahul Deshmukh
% PUID: 0030004932
%%
function f = hw1SUMTfun(x)
% funciton computes the objective function value
% input:  x is a col vector
% output: f is a scalar value of the objective function

% x=[1/cr,b,a]
% objective function
y_f=0.033949298754935466466210448881266;%scalar multiplier
f = y_f*(x(2)^2)*(x(3)+3)^2;
end
```

Script for Constraints:

```
% AAE:550 HW1 P2
% Rahul Deshmukh
% PUID: 0030004932
%%
function g = hw1SUMTcon(x)
% this function computes the inequality constraint function values
% input: x is a column vector
% output: g: col vector of constraint values [bounds; gj]

% x=[x1,x2,x3]=[1/cr,b,a]
% constraints
g=zeros(9,1);%col vector
% bounds
g(1)=1-3*x(1);%for cr
g(2)=0.8*x(1)-1;

g(3)=x(2)/14-1;%for b
g(4)=1-x(2)/8;

g(5)=x(3)/10-1;%for alpha
g(6)= -1 - x(3)/5;

% inequality constraints
% for lift coeff
y_lc=0.010966227112321509576482767777764;
g(7)=y_lc*x(2)*(x(3)+3)*x(1)/0.9-1;
g(8)=1-(y_lc*x(2)*(x(3)+3)*x(1))/0.7;

% for Lift
y_L=12.504545589127774135062779594447;
g(9)=1-y_L*(x(2)^2)*(x(3)+3)/9500;
end
```

2) Optimization using different penalty function approaches:
For all these methods, I have used *fminunc* with default BFGS update and numerical gradients was chosen. Also the tolerance on relative error of objective function and error in constraint function value was chosen as $\epsilon = 10^{-6}$ for all cases.

a) Exterior Penalty Method
The script for evaluating the pseudo-objective function is:

```
function phi = hw1SUMTphi(x,r_p)
% This function is the pseudo-objective function using the exterior ...
    penalty.
% In this function, r_p is a "parameter", x are the variables.  This
% does not include constraint scaling parameters, c_j.

% compute values of the objective function and constraints at the ...
    current
% value of x
f = hw1SUMTfun(x);
g = hw1SUMTcon(x);

% exterior penalty function
P =max(0,g);  % note: no c_j scaling parameters
P=P'*P;

phi = f + r_p * P;
end
```

SUMT Exterior Penalty						
P	1	2	3	4	5	6
r_p	1	5	25	125	625	3125
x^0	0 0 0	1.276757470185778 14.282241785987278 -2.921714161272035	1.367313681237115 15.246682636432020 -2.644356099023211	1.506572258345380 16.838371086757036 -1.870468397277544	1.427422323989274 16.206891583266160 -0.802253513342547	1.303404309890892 14.966946773637224 0.119199519530206
x*	1.276757470185778 14.282241785987278 -2.921714161272035	1.367313681237115 15.246682636432020 -2.644356099023211	1.506572258345380 16.838371086757036 -1.870468397277544	1.427422323989274 16.206891583266160 -0.802253513342547	1.303404309890892 14.966946773637224 0.119199519530206	1.245616259723662 14.388663156694799 0.561694734918270
f(x*)	0.042441427986762	0.998187804502612	12.280830393731572	43.071049159105527	73.991792246489879	89.163085342838670
g1(x*) g2(x*) g3(x*) g4(x*) g5(x*) g6(x*) g7(x*) g8(x*) g9(x*)	-2.830272410557335 0.021405976148623 0.020160127570520 -0.785280223248410 -1.292171416127204 -0.415657167745593 -0.982605869505924 0.977636117936188 0.978980601697539	-3.101941043711345 0.093850944989692 0.089048759745144 -0.905835329554002 -1.264435609902321 -0.471128780195358 -0.909661346819279 0.883850303053359 0.891179580663644	-3.519716775036141 0.205257806676304 0.202740791911217 -1.104796385844629 -1.187046839727754 -0.625906320544491 -0.650857147798434 0.551102047169415 0.578456062297739	-3.282266971967823 0.141937859191420 0.157635113090440 -1.025861447908270 -1.080225351334255 -0.839549297331491 -0.380495404354177 0.203494091312514 0.240161390982396	-2.910212929672675 0.042723447912713 0.069067626688373 -0.870868346704653 -0.988080048046979 -1.023839903906041 -0.258569839380015 0.046732650631447 0.080283680647589	-2.736848779170986 -0.003506992221070 0.027761654049628 -0.798582894586850 -0.943830526508173 -1.112338946983654 -0.222185320500602 -0.000047445070655 0.029396406892983
No. or inter	28	11	12	12	11	16
exit-flag	1	1	1	1	1	1

SUMT Exterior Penalty (contd.)						
P	7	8	9	10	11	12
r_p	15625	78125	390625	1953125	9765625	48828125
x^0	1.245616259723662 14.388663156694799 0.561694734918270	1.245616259723662 14.086828912272756 0.804461916983293	1.245616259723662 14.017801197861015 0.861374375049149	1.245616259723662 14.003578018675936 0.873170186707270	1.245616259723662 14.000715961212642 0.875546689456223	1.245616259723662 14.000141871099446 0.876023217302232
x*	1.245616259723662 14.086828912272756 0.804461916983293	1.245616259723662 14.017801197861015 0.861374375049149	1.245616259723662 14.003578018675936 0.873170186707270	1.245616259723662 14.000715961212642 0.875546689456223	1.245616259723662 14.000141871099446 0.876023217302232	1.245616259723662 14.000028148183382 0.876117920663847
f(x*)	97.508790704715992	99.465946895373833	99.871598089342427	99.953324735102143	99.969707396335551	99.972968423565220
g1(x*) g2(x*) g3(x*) g4(x*) g5(x*) g6(x*) g7(x*) g8(x*) g9(x*)	-2.736848779170986 -0.003506992221070 0.006202065162340 -0.760853614034094 -0.919553808301671 -1.160892383396658 -0.186597549678368 -0.045803150413527 0.006280001254589	-2.736848779170986 -0.003506992221070 0.001271514132930 -0.752225149732627 -0.913862562495085 -1.172274875009830 -0.178474966383863 -0.056246471792176 0.001274791082862	-2.736848779170986 -0.003506992221070 0.000255572762567 -0.750447252334492 -0.912682981329273 -1.174634037341454 -0.176801461345889 -0.058398121126714 0.000255738700570	-2.736848779170986 -0.003506992221070 0.000051140086617 -0.750089495151580 -0.912445331054378 -1.175109337891244 -0.176464711465342 -0.058831085258847 0.000051180151172	-2.736848779170986 -0.003506992221070 0.000010133649960 -0.750017733887431 -0.912397678269777 -1.175204643460446 -0.176397224231205 -0.058917854559879 0.000010241746917	-2.736848779170986 -0.003506992221070 0.000002010584527 -0.750003518522923 -0.912388207933615 -1.175223584132769 -0.176383791316146 -0.058935125450669 0.000002055002479
No. or inter	11	9	6	10	7	6
exit-flag	1	1	1	2	2	2

SUMT Exterior Penalty (contd.)			
P	13	14	15
r_p	244140625	1.220703125000000e+09	6.103515625000000e+09
x^0	1.245616259723662 14.000028148183382 0.876117920663847	1.245616259723662 14.000028468800194 0.876124113338654	1.245616259723662 13.999945578745367 0.876171322149313
x*	1.245616259723662 14.000028468800194 0.876124113338654	1.245616259723662 13.999945578745367 0.876171322149313	1.245616259723662 13.999945706938167 0.876171575563591
f(x*)	99.973292446196567	99.974543837296977	99.974558740329954
g1(x*) g2(x*) g3(x*) g4(x*) g5(x*) g6(x*) g7(x*) g8(x*) g9(x*)	-2.736848779170986 -0.003506992221070 0.000002033485728 -0.750003558600024 -0.912387588666135 -1.175224822667731 -0.176382456604978 -0.058936841507885 0.000000411554760	-2.736848779170986 -0.003506992221070 -0.000003887232474 -0.749993197343171 -0.912382867785069 -1.175234264429863 -0.176377301906091 -0.058943468977884 0.000000073690674	-2.736848779170986 -0.003506992221070 -0.000003878075845 -0.749993213367271 -0.912382842443641 -1.175234315112718 -0.176377240518085 -0.058943547905320 -0.000000010000120
No. or inter	2	3	1
exit-flag	2	2	2

The Final solution in terms of original parameters is:

$$cr = 0.802815467599828m$$

$$b = 13.999945706938167m$$

$$\alpha = 0.876171575563591degree$$

and Minimum Drag $D_i = 99.974558740329954N$

b) Interior Penalty Method

For this problem, It was observed that the Log-Barrier method worked better than the classical barrier. The script for evaluating the pseudo-objective function is:

```
function phi = hw1SUMTphi_Int(x,r_p)
% This function is the pseudo-objective function using interior
% penalty method.
% input: x: col vector of design variables
%        r_p: penalty multiplier, x are the variables.  This
% does not include constraint scaling parameters, c_j.

% compute values of objective function and constraints at current x
f = hw1SUMTfun(x);
g = hw1SUMTcon(x);

% Interior penalty function
% classic
% P=sum(-1./g);

% Log barrier
P=sum(-log(-g));

phi = f + r_p * P;
end
```

Also, for Interior Penalty Method we need an initial feasible solution, I found out this using random function in matlab. The script for finding an initial feasible solution is:

```
%AAE:550 HW1 P2
% Rahul Deshmukh
% PUID: 0030004932
%%
%script to find x0 for interior penalty function
% x=[1/cr,b,a]
clc; clear all;
format long;
notfound=1;
Nmax=10^4;%maximum number of iterations for the while loop
i=0;%counter
while notfound && i≤Nmax
    % x = a+ (b-a)*x_rand;
    %pick random value for cr,b,a conforming their bounds
    x1= 0.8 + (3-0.8)*rand(1);%cr
    x2= 8 + (14-8)*rand(1);%b
```

```
    x3= -5 + (10+5)*rand(1); %a
    if x1≠0
       x1=1/x1;%1/cr
       x=[x1;x2;x3];
       g=hw1SUMTcon(x);
       %check if all are negative
       sat = find(g≤0);%satisfied constraint
       if length(sat)==length(g)
           notfound=0;
           display(x);
       end
    end
    i=i+1;
end
i-1
%{
result
x0=[0.475058013560474;...
   13.235315387885350;...
   9.002524127606579];
i=4
%}
```

SUMT Interior Penalty						
P	1	2	3	4	5	6
r_p	1	0.200000000000000	0.040000000000000	0.008000000000000	0.001600000000000	3.200000000000000e-04
x^0	0.475058013560474 13.235315387885350 9.002524127606579	1.242051716019787 13.441551104374039 1.207219398288423	1.21034451759675 13.985678615012258 0.888073797550397	1.211402342892526 13.997185781437357 0.878480256183915	1.211402819590726 13.999439081569248 0.876607277522905	1.211402860049516 13.999886801893286 0.876235124597706
x*	1.242051716019787 13.441551104374039 1.207219398288423	1.210344517596750 13.985678615012258 0.888073797550397	1.211402342892526 13.997185781437357 0.878480256183915	1.211402819590726 13.999439081569248 0.876607277522905	1.211402860049516 13.999886801893286 0.876235124597706	1.211402860049516 13.999886801893286 0.876235124597706
f(x*)	1.085725163937203e+02	1.003845543265858e+02	1.000542248860218e+02	99.989798142317724	99.976995576143040	99.976995576143040
g1(x*) g2(x*) g3(x*) g4(x*) g5(x*) g6(x*) g7(x*) g8(x*) g9(x*)	-2.726155148059362 -0.006358627184170 -0.039889206830426 -0.680193888046755 -0.879278060171158 -1.241443879657685 -0.144147441149831 -0.100381861378789 -0.000548728235836	-2.631033552790250 -0.031724385922600 -0.001022956070553 -0.748209826876532 -0.911192620244960 -1.177614759510079 -0.198060182424997 -0.031065479739289 -0.001027242204078	-2.634207028677577 -0.030878125685979 -0.000201015611617 -0.749648222679670 -0.912151974381608 -1.175696051236783 -0.198680986541584 -0.030267303017964 -0.000201145923403	-2.634208458772178 -0.030877744327419 -0.000040065602197 -0.749929885196156 -0.912339272247709 -1.175321455504581 -0.198938704839049 -0.029935950921224 -0.000040033071945	-2.634208580148548 -0.030877711960387 -0.000008085579051 -0.749985850236661 -0.912376487540229 -1.175247024919541 -0.198989963161419 -0.029870047363890 -0.000007989634798	-2.634208580148548 -0.030877711960387 -0.000008085579051 -0.749985850236661 -0.912376487540229 -1.175247024919541 -0.198989963161419 -0.029870047363890 -0.000007989634798
No. or inter	45	28	16	8	8	1
exit-flag	0	5	1	2	2	2

The Final solution in terms of original parameters is:

$$cr = 0.825489218309362m$$

$$b = 13.999886801893286m$$

$$\alpha = 0.876235124597706degree$$

and Minimum Drag $D_i = 99.976995576143040N$

c) Extended Linear Interior Penalty Method
The script for evaluating the pseudo-objective function is:

```
function phi = hw1SUMTphi_LinExt(x,r_p,tr_e)
% This function is the pseudo-objective function using Linear
% extended interior penalty method.
% input: x: col vector of design variables
%        r_p: penalty multiplier
%        tr_e: transition eps
% This does not include constraint scaling parameters, c_j.

% compute values of objective function and constraints at current x
f = hw1SUMTfun(x);
g = hw1SUMTcon(x);

% Linear extended Interior Penalty function
P=0;
for i=2:length(g)
    if g(i)<=tr_e
        g_hat=-1/g(i);
    else
        g_hat= - (2*tr_e-g(i))/tr_e^2;
    end
    P=P+g_hat;
end

phi = f + r_p * P;
end
```

SUMT Linear Extended Interior Penalty						
P	1	2	3	4	5	6
r_p	25.470164726578144	5.094032945315629	1.018806589063126	0.203761317812625	0.040752263562525	0.008150452712505
x^0	0.475058013560474 13.235315387885350 9.002524127606579	0.834131139714169 11.963258962706560 3.941625283990999	1.000652394693517 12.236051713002341 2.853394922611828	1.115365666879351 13.021288169458897 1.821546258167441	1.170535609943452 13.539475377703932 1.285556001618355	1.194427876487765 13.797706316840070 1.049162277714346
x*	0.834131139714169 11.963258962706560 3.941625283990999	1.000652394693517 12.236051713002341 2.853394922611828	1.115365666879351 13.021288169458897 1.821546258167441	1.170535609943452 13.539475377703932 1.285556001618355	1.194427876487765 13.797706316840070 1.049162277714346	1.204528077550375 13.910737956511738 0.951239741438898
f(x*)	2.341273493200837e+02	1.741522415104640e+02	1.338170291786038e+02	1.143006844268324e+02	1.059680557512164e+02	1.025646990423054e+02
g1(x*) g2(x*) g3(x*) g4(x*) g5(x*) g6(x*) g7(x*) g8(x*) g9(x*)	-1.502393419142507 -0.332695088228665 -0.145481502663817 -0.495407370338320 -0.605837471600900 -1.788325056798200 -0.155966408891668 -0.085186045710712 -0.307689036998923	-2.001957184080550 -0.199478084245187 -0.125996306214119 -0.529506464125293 -0.714660507738817 -1.570678984522365 -0.126732905568099 -0.122771978555302 -0.153545536283816	-2.346097000638052 -0.107707466496520 -0.069907987895793 -0.627661021182362 -0.817845374183256 -1.364309251633488 -0.146757879358040 -0.097025583682521 -0.076065102486295	-2.511606829830356 -0.063571512045238 -0.032894615878291 -0.692434422212991 -0.871444399838164 -1.257111200323671 -0.172423083865238 -0.064027463601837 -0.034082425370034	-2.583283629463295 -0.044457698809788 -0.014449548797138 -0.724713289605009 -0.895083772228565 -1.209832455542869 -0.186894804084932 -0.045420966176516 -0.014666412977479	-2.613584232651124 -0.036377537959700 -0.006375860249162 -0.738842244563967 -0.904876025856110 -1.190247948287779 -0.193294151858642 -0.037193233324603 -0.006417151777776
No. or inter	30	21	15	12	11	12
exit-flag	1	1	1	2	2	2

SUMT Linear Extended Interior Penalty (contd.)						
P	7	8	9	10	11	12
r_p	0.001630090542501	3.260181085002002e-04	6.520362170004005e-05	1.304072434000801e-05	2.608144868001602e-06	5.216289736003204e-07
x^0	1.204528077550375 13.910737956511738 0.951239741438898	1.208901119373016 13.960166023280568 0.909416823572248	1.210840470509089 13.982165326219759 0.890992681080089	1.210847079236560 13.992015030592004 0.882781309378108	1.210847950693696 13.996426312711483 0.879111128648261	1.210848048770953 13.998400279770372 0.877469908221259
x*	1.208901119373016 13.960166023280568 0.909416823572248	1.210840470509089 13.982165326219759 0.890992681080089	1.210847079236560 13.992015030592004 0.882781309378108	1.210847950693696 13.996426312711483 0.879111128648261	1.210848048770953 13.998400279770372 0.877469908221259	1.210848059661707 13.999285769555806 0.876734896921222
f(x*)	1.011197365889512e+02	1.004848299247179e+02	1.002021856974592e+02	1.000759162584386e+02	1.000194578080266e+02	99.994191487556009
g1(x*) g2(x*) g3(x*) g4(x*) g5(x*) g6(x*) g7(x*) g8(x*) g9(x*)	-2.626703358119049 -0.032879104501587 -0.002845284051388 -0.745020752910071 -0.909058317642775 -1.181883364714450 -0.196088822665686 -0.033600085144118 -0.002853403964909	-2.632521411527267 -0.031327623592729 -0.001273905270017 -0.747770665777470 -0.910900731891991 -1.178198536216018 -0.197330979336961 -0.032003026566764 -0.001275497992858	-2.632541237709678 -0.031322336610752 -0.000570354957714 -0.749001878824000 -0.911721869062189 -1.176556261875622 -0.198456275204993 -0.030556217593580 -0.000570663546899	-2.632543852081089 -0.031321639445043 -0.000255263377751 -0.749553289088935 -0.912088887135174 -1.175822225729652 -0.198960889175684 -0.029907428202692 -0.000255284465680	-2.632544146312859 -0.031321560983238 -0.000114265730688 -0.749800034971297 -0.912253009177874 -1.175493981644252 -0.199186811785869 -0.029616956275312 -0.000114124394406	-2.632544178985120 -0.031321552270635 -0.000051016460300 -0.749910721194476 -0.912326510307878 -1.175346979384244 -0.199287959304693 -0.029486909465395 -0.000051050701659
No. or inter	14	18	11	9	7	9
exit-flag	2	1	2	2	2	2

SUMT Linear Extended Interior Penalty (contd.)						
P	13	14	15	16	17	18
r_p	1.043257947200641e-07	2.086515894401282e-08	4.173031788802563e-09	8.346063577605127e-10	1.669212715521025e-10	3.338425431042051e-11
x^0	1.210848059661707 13.999285769555806 0.876734896921222	.210848060519287 13.999680238393443 0.876407262920070	1.210848060610322 13.999856141842171 0.876260686095742	1.210848060613061 13.999935357586557 0.876195182975888	1.210848060613061 13.999933541764523 0.876186705813254	1.210848060613061 13.999933117801678 0.876183349749345
x*	.210848060519287 13.999680238393443 .876407262920070	1.210848060610322 13.999856141842171 0.876260686095742	.210848060613061 13.999935357586557 .876195182975888	1.210848060613061 13.999933541764523 0.876186705813254	1.210848060613061 13.999933117801678 0.876183349749345	1.210848060613061 13.999932794893834 0.876182044674636
f(x*)	99.982924965287467	99.977876251023361	99.975628700368688	99.975165477381864	99.974986302163842	99.974914369053806
g1(x*) g2(x*) g3(x*) g4(x*) g5(x*) g6(x*) g7(x*) g8(x*) g9(x*)	-2.632544181557862 -0.031321551584570 -0.000022840114754 -0.749960029799180 -0.912359273707993 -1.175281452584014 -0.199333068828654 -0.029428911506017 -0.000022887989043	-2.632544181830967 -0.031321551511742 -0.000010275582702 -0.749982017730271 -0.912373931390426 -1.175252137219148 -0.199353284168184 -0.029402920355192 -0.000010203999392	-2.632544181839183 -0.031321551509551 -0.000004617315246 -0.749991919698320 -0.912380481702411 -1.175239036595177 -0.199362283678222 -0.029391349556573 -0.000004621901222	-2.632544181839183 -0.031321551509551 -0.000004747016820 -0.749991692720565 -0.912381329418675 -1.175237341162651 -0.199364138501386 -0.029388964783933 -0.000002175505821	-2.632544181839183 -0.031321551509551 -0.000004777299880 -0.749991639725210 -0.912381665025065 -1.175236669949869 -0.199364855950407 -0.029388042349477 -0.000001249121595	-2.632544181839183 -0.031321551509551 -0.000004800364726 -0.749991599361729 -0.912381795532536 -1.175236408934927 -0.199365143983386 -0.029387672021360 -0.000000866300566
No. or inter	7	9	7	2	2	1
exit-flag	2	2	2	2	2	2

The Final solution in terms of original parameters is:

$$cr = 0.825867449871203m$$

$$b = 13.999932794893834m$$

$$\alpha = 0.876182044674636degree$$

and Minimum Drag $D_i = 99.974914369053806N$

d) Augmented Lagrange Multiplier Method
The script for evaluating the pseudo-objective function is:

```
% AAE:550 HW1 P2
% Rahul Deshmukh
% PUID: 0030004932
%%
function A = hw1SUMTalm(x,r_p,lambda)
% This function is the pseudo-objective function using the ALM.
%Input: x: column vector of design variables
%       r_p: penalty multiplier
%       lambda: col vector, lagrange multipliers[ineq, eq]

% compute values of the objective function and constraints at the ...
    current
% value of x
f = hw1SUMTfun(x);
g = hw1SUMTcon(x); %inequality constraints column vector of size len_g
% h = hw1SUMTcon_heq(x);% equaltiy constraints column vector of ...
    size len_h

len_g=length(g);
% Fletcher's substitution only for ineqaluti constraints
psi = max(g, -lambda(1:len_g) / (2 * r_p)); % col vec size len_g

% Augmented Lagrangian function
A = f + (lambda(1:len_g))' * psi + r_p * ((psi')*psi);%+...
%     (lambda(len_g+1:end))'*h+r_p*(h'*h);
end
```

ALM						
P	1	2	3	4	5	6
r_p	1	5	25	125	625	3125
x^0	0.350113005416948 8.217596212161899 -2.530475785010287	1.276756989556942 14.282393517682941 -2.921714790819759	1.415339832204870 15.759311463018443 -2.577782321399998	1.577158568494572 17.658668240467161 -1.689690306605792	1.408995023308075 16.087967822291439 -0.481294436641496	1.265605409167405 14.591070975135302 0.444008032949034
x*	1.276756989556942 14.282393517682941 -2.921714790819759	1.415339832204870 15.759311463018443 -2.577782321399998	1.577158568494572 17.658668240467161 -1.689690306605792	1.408995023308075 16.087967822291439 -0.481294436641496	1.265605409167405 14.591070975135302 0.444008032949034	1.237628759681719 14.185728965995825 0.727283434437060
f(x*)	0.042441647159707	1.503066214958353	18.175845003904250	55.742698476949712	85.730125514583094	94.911527329259371
g1(x*) g2(x*) g3(x*) g4(x*) g5(x*) g6(x*) g7(x*) g8(x*) g9(x*)	-2.830270968670827 0.021405591645554 0.020170965548781 -0.785299189710368 -1.292171479081976 -0.415657041836048 -0.982605831141177 0.977636068610084 0.978980324117700	-3.246019496614610 0.132271865763896 0.125665104501317 -0.969913932877305 -1.257778232140000 -0.484443535720000 -0.885250947470214 0.852465503890276 0.861975812157810	-3.731475705483716 0.261726854795657 0.261333445747654 -1.207333530058395 -1.168969030660579 -0.662061938678842 -0.555346221114698 0.428302284290326 0.462183456018077	-3.226985069924226 0.127196018646460 0.149140558735103 -1.010995977786430 -1.048129443664150 -0.903741112671701 -0.304330799695254 0.105568171036755 0.141927293274670	-2.796816227502215 0.012484327333924 0.042219355366807 -0.823883871891913 -0.955599196705097 -1.088801606589807 -0.225066608955370 0.003657068656904 0.034876690638764	-2.712886279045156 -0.009896992254625 0.013266354713988 -0.773216120749478 -0.927271656556294 -1.145456686887412 -0.202649636798318 -0.025164752687877 0.012720671491172
No. or inter	14	11	10	12	11	15
exit-flag	1	1	1	1	1	2

ALM (contd.)				
P	7	8	9	10
r_p	15625	78125	390625	1953125
x^0	1.237628759681719 14.185728965995825 0.727283434437060	1.237628759681719 14.005500456065775 0.872171151912445	1.237628759681719 14.000029126364613 0.876120282958519	1.237628759681719 13.999999657710770 0.876141650918566
x*	1.237628759681719 14.005500456065775 0.872171151912445	1.237628759681719 14.000029126364613 0.876120282958519	1.237628759681719 13.999999657710770 0.876141650918566	1.237628759681719 13.999999659075149 0.876141664870233
f(x*)	99.847492383823834	99.973104250604877	99.973785629943563	99.973786369114805
g1(x*) g2(x*) g3(x*) g4(x*) g5(x*) g6(x*) g7(x*) g8(x*) g9(x*)	-2.712886279045156 -0.009896992254625 0.000392889718984 -0.750687557008222 -0.912782884808755 -1.174434230382489 -0.182178927944464 -0.051484235499974 0.000239168407692	-2.712886279045156 -0.009896992254625 0.000002080454615 -0.750003640795577 -0.912387971704148 -1.175224056591704 -0.181664664939527 -0.052145430792037 0.000001305815426	-2.712886279045156 -0.009896992254625 -0.000000024449231 -0.749999957213846 -0.912385834908143 -1.175228330183713 -0.181661876210281 -0.052149016301068 0.000000002916323	-2.712886279045156 -0.009896992254625 -0.000000024351775 -0.749999957384394 -0.912385833512977 -1.175228332974047 -0.181661873185027 -0.052149020190679 -0.000000000877958
No. or inter	11	4	4	1
exit-flag	1	2	5	5

The Final solution in terms of original parameters is:

$$cr = 0.807996737452328m$$
$$b = 13.999999659075149m$$
$$\alpha = 0.876141664870233degree$$

and Minimum Drag $D_i = 99.973786369114805N$

3) Discussion
Comparison Table for all SUMT methods used:

Method	Exterior Penalty	Interior Penalty	Extended Linear Penalty	ALM
fminunc calls	15	6	18	10
iters	145	106	197	93
cr (m)	0.802815467599828	0.825489218309362	0.825867449871203	0.807996737452328
b (m)	13.999945706938167	13.999886801893286	13.999932794893834	13.999999659075149
α(Deg)	0.876171322149313	0.876235124597706	0.876182044674636	0.876141664870233
$D_i(N)$	99.9745587403299	99.976995576143	99.9749143690538	99.9737863691148

From the above comparison table , We can see that out of all the SUMT methods, Augmented Lagrange Multiplier and Interior Penalty method has the minimum number of iterations and fminunc calls. However, with ALM we have a lower value of the induced Drag.

When comparing the final solution, we can see that for ALM we have the lowest value of induced Drag. Also, the values of the parameters b and α for all the methods are quite close, but not for c_r. Further, both exterior penalty and ALM gave us a value of c_r 0.81 and correspondingly have the lowest values of the induced drag.

In terms of ease of implementation of the pseudo objective function, the Exterior Penalty method was the easiest.

For this problem, I found that ALM method is the best tool at hand, even though constructing the pseudo function for ALM was difficult but that was the only work required. However, for Interior Penalty method, I had to write out a script to figure out a feasible initial solution and then try out classic vs log barrier performance, this required more effort in my opinion and even then we don't have a better solution (in fact it has the highest value of induced drag).
Furthermore, Extended Linear Penalty does help in giving a better solution than the interior penalty method but it turned out to be the most costly method.