

---

## Table of Contents

Import data from text file. ....	1
Initialize variables. ....	1
Format string for each line of text: ....	1
Open the text file. ....	2
Read columns of data according to format string. ....	2
Close the text file. ....	2
Post processing for unimportable data. ....	2
Allocate imported array to column variable names ....	2
Clear temporary variables ....	3
3-d Wing Optimization: ....	3
Lifting Line Theory - Validation of the results ....	6
Convert degree values to radians ....	6
Initialize work arrays ....	6
Angles for station locations ....	6
Create the system of equations for solving LLT/Monoplane Equation ....	7
Solve ....	7
Extract quantities of interest ....	7

## Import data from text file.

Script for importing data from the following text file:

```
%C:\Users\Prateek\Documents\MATLAB\Airfoil_opt\airfoil_optim
\xfoil_a00.500_cpwr.dat
%
% To extend the code to different selected data or a different text
file,
% generate a function instead of a script.
```

## Initialize variables.

```
%%PLEASE CHANGE THE DIRECTORY OF THE FILE, DEPENDING ON WHERE YOU'RE
%%RUNNING THIS MATLAB CODE. ENTER THE DIRECTORY FOR,
'xfoil_a00.500_cpwr.dat'.

filename = '\\ad.uillinois.edu\engr\instructional\paintal2\documents
\MATLAB\413_Project\xfoil_a00.500_cpwr.dat';
startRow = 4;
```

## Format string for each line of text:

```
column1: double (%f)

% column2: double (%f)
% column3: double (%f)
% For more information, see the TEXTSCAN documentation.
formatSpec = '%10f%9f%f%[^\\n\\r]';
```

---

## Open the text file.

```
fileID = fopen(filename, 'r');
```

## Read columns of data according to format string.

This call is based on the structure of the file used to generate this code. If an error occurs for a different file, try regenerating the code from the Import Tool.

```
dataArray = textscan(fileID,  
    formatSpec, 'Delimiter', ',', 'WhiteSpace', ' ', 'HeaderLines'  
    ,startRow-1, 'ReturnOnError', false);
```

## Close the text file.

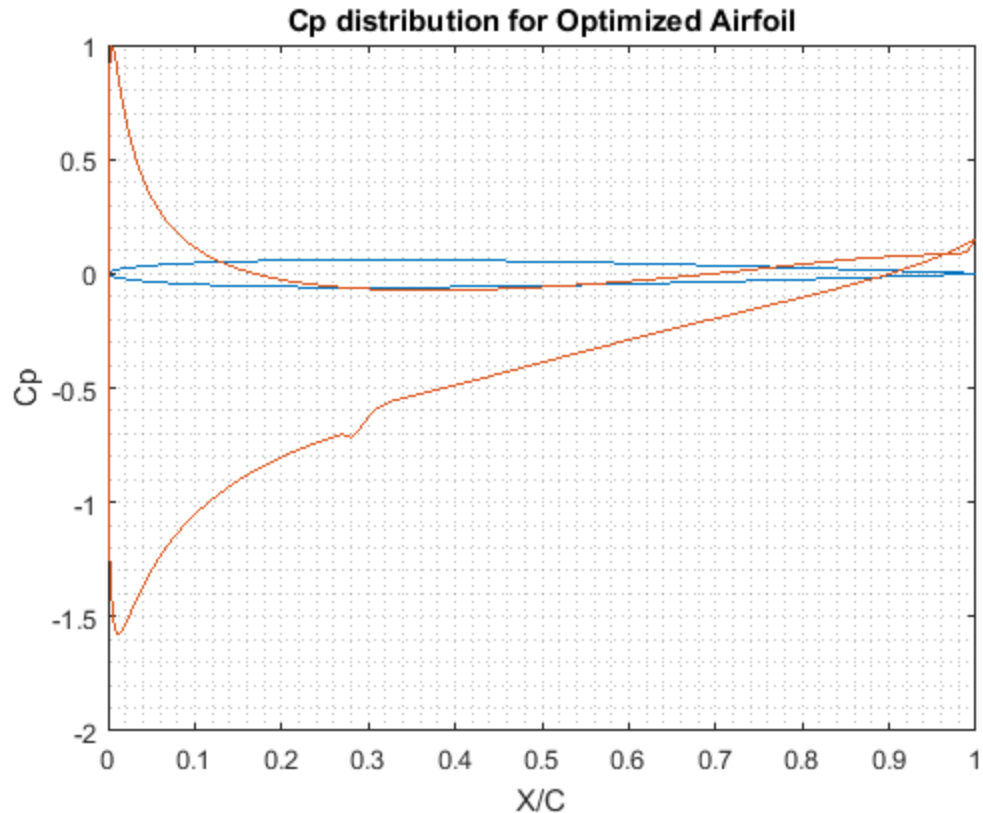
```
fclose(fileID);
```

## Post processing for unimportable data.

No unimportable data rules were applied during the import, so no post processing code is included. To generate code which works for unimportable data, select unimportable cells in a file and regenerate the script.

## Allocate imported array to column variable names

```
X = dataArray{:, 1};  
Y = dataArray{:, 2};  
Cp = dataArray{:, 3};  
Cp = Cp;  
figure  
plot(X,Y)  
hold on  
plot(X,Cp)  
grid minor  
xlabel('X/C')  
ylabel('Cp')  
title('Cp distribution for Optimized Airfoil')  
%axis equal
```



## Clear temporary variables

```
clearvars filename startRow formatSpec fileID dataArray ans;
```

## 3-d Wing Optimization:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Optimizing Wing Parameters%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc;
options = optimset('MaxFunEvals ',
    10000, 'MaxIter', 7000, 'LargeScale', 'on');
A = []; B = []; Aeq = []; beq = [];
lb = [0 0 0.2 10 15 0]; ub = [10 10 0.85 35 200 5];
x0 = [2.8 1.45 0.9 14 30 1];

% Establishing Constants and wing parameters:
global Cl, global pi, global e, global Mr, global Lb, global rh
global Nlift, global sigmaMax, global sigmaMaxSh, global t, global g
global tcap, global tweb, global tau, global Icap, global wbar, global
rhoCap, global rhoWeb

%Defining Constants and Parameters:
Mr = 64156.67; %Upper Limit Moment at the root chord of the wing.
Lb = 5200; %Weight of the Aircraft excluding wing.
Nlift = 6.0; %Wing loading Multiplier.
```

---

```

sigmaMax = 25e7; % Maximum allowable Stress
sigmaMaxSh = 167e6; % Maximum allowable Shear Stress
Cl=0.65; % Desired Lift Coefficient
pi = 3.14159265; % Value of 'pi'
e = 0.95; % Wing psan efficiency
t = 0.25; % 't' ranges from 0.117 to 0.312m Maintaining constant
    thickness of the airfoil
tcap = 0.00427310; % Thickness of the spar cap per unit chord
tweb = 0.000591; %shear web thickness per unit chord
wbar =0.5; %The ratio of wing box length/Chord_Length
rhoCap =2700; %Density of the Cap
g = 9.81; % Gravitaionl Constant
rhoWeb = 2700; %Density of the web
rh= 0.75; % Wing parameter constant
Icap = 1.908e-5; %Moment of Inertia Per unit span.
tau = 0.15; %Ratio of max thickness of wing section to local chord.

%The decision variables are as follows:
%Y=[p, q, lambda, b, S, v(lambda)]

%Compute the objective function and optimum values for the wing:
warning off
global xopt;
warning off
%[xopt,fval] = run(ms,problem,100);
% Implementing GA:
opts = optimoptions(@ga,'PlotFcn',{@gaplotbestf,@gaplotstopping});
%thestate = rng;
fun = @(x)(Cl^2*(pi*e*x(4)^2/x(5))^(-1));
rng(thestate);
[x,Fvalue,exitFlag,Output] = ga(@(x)(Cl^2*(pi*e*x(4)^2/x(5))^(-1)),6,[],
[],[],[],[0 0 0.2 10 15 0],[10 10 0.85 32 60 5],...
    @Wingdrag_Con,opts);

fprintf('The number of generations was : %d\n', Output.generations);
fprintf('The number of function evaluations was : %d\n',
    Output.funccount);
fprintf('The best function value found was : %g\n', Fvalue);

display('Y=[p, q, lambda, b, S, v(lambda)]:');
display(x');

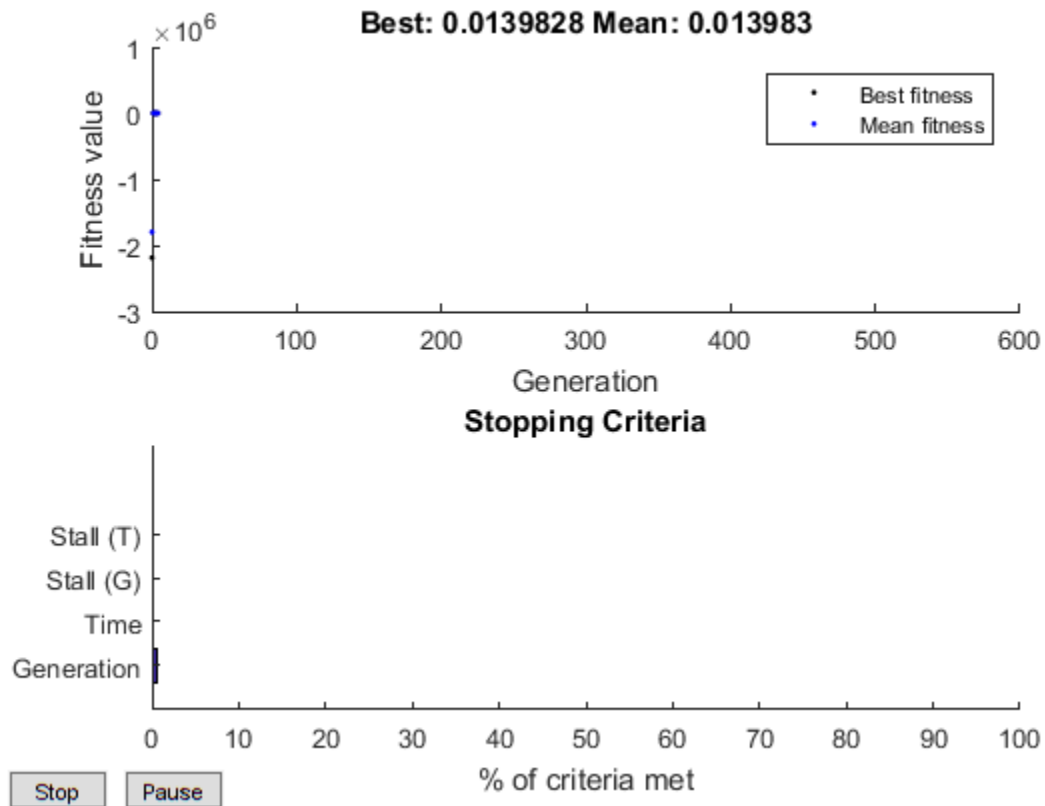
Optimization terminated: average change in the fitness value less than
    options.FunctionTolerance
    and constraint violation is less than options.ConstraintTolerance.
The number of generations was : 4
The number of function evaluations was : 67600
The best function value found was : 0.013983
Y=[p, q, lambda, b, S, v(lambda)]:
    1.563228098424743
    1.281593058667785
    0.281594582685219
    19.198476821882366
    36.406759636739380

```

---

---

0.828620919974883



```
options = optimset('MaxFunEvals',20000,'MaxIter',7000,'LargeScale','on');
%options = optimoptions('fmincon','Display','iter','Algorithm','sqp');
A = []; B = []; Aeq = []; beq = [];
lb = [0 0 0.2 10 15 0]; ub = [10 10 0.85 35 200 5];
x0 = [x(1) x(2) x(3) x(4) x(5) x(6)];

%Compute the objective function and optimum values for the wing using
fmincon:
warning off
[xopt, fval] = fmincon('Drag_obj_fn', x0, A, B, Aeq, beq, lb, ub,...
'Wingdrag_Con',options);
display('The final design parameters are found to be as follows: ');
display(['[',num2str(xopt(1:6)),']']);
display(['Minimum Cdi value is found to be = ', num2str(fval)]);
xopt(7) =0.115; % Data obtained from wing section (airfoil)
optimization.
xopt(8) =-0.98; % Data obtained from wing section (airfoil)
optimization.
xopt(9) =3.417; % Data obtained from wing section (airfoil)
optimization.
```

*Solver stopped prematurely.*

---

*fmincon stopped because it exceeded the function evaluation limit, options.MaxFunctionEvaluations = 20000 (the selected value).*

*The final design parameters are found to be as follows:*

*[1.57576      1.28788      0.287881      24.5863      45.8294  
0.826435]*

*Minimum Cdi value is found to be = 0.010733*

## Lifting Line Theory - Validation of the results

This is a simple function for solving lifting-line theory / monoplane equation. It assumes that the wing is not maneuvering, i.e., there are no aileron deflections, and so only odd terms on the sine series are used to create a symmetric lift distribution. Inputs:  $a_0$  = Sectional lift curve slope per radians (normally  $2\pi$ ) AR = Aspect ratio of wing  $\lambda$  = Taper ratio,  $c_t/c_r$   $\alpha_{0l}$  = Sectional zero lift angle of attack in degrees  $\alpha$  = Geometric angle of attack in degrees numlocs = Number of spanwise stations / terms in sine series Outputs: CL = Wing lift coefficient CDi = Induced drag coefficient e = Span efficiency factor Created by: Joseph Derlaga Modified by: Angad Paintal

```
a0 = 2.1*pi;  
AR = xopt(4)^2/xopt(5);  
 $\lambda$  = xopt(3);  
 $\alpha_{0l}$  = xopt(8);  
 $\alpha$  = xopt(9);  
numlocs = 57;  
%a0 = 0.115; AR = xopt(6);  $\lambda$  = xopt(3);  $\alpha_{0l}$  = -3.5;  $\alpha$  = 5;  
numlocs = 57;
```

## Convert degree values to radians

Depending on your version of Matlab, you can use either of these methods

```
alphan =  $\alpha$ *pi/180;  
 $\alpha_{0lr}$  =  $\alpha_{0l}$ *pi/180;  
  
%alphan = degtorad( $\alpha$ );  
% $\alpha_{0lr}$  = degtorad( $\alpha_{0l}$ );
```

## Initialize work arrays

```
LHS = zeros(numlocs,numlocs);  
RHS = zeros(numlocs,1);  
 $\phi$  = zeros(numlocs,1);  
Cl = zeros(numlocs,1);
```

## Angles for station locations

```
for n = 1:numlocs  
     $\phi$ (n) = n*pi/(2*numlocs);  
end  
% Could vectorize the above lines
```

---

# Create the system of equations for solving LLT/Monoplane Equation

```
for a = 1:numlocs

    mu = (a0 / ( 2 * AR * ( 1 + lambda ) )) * (1 + (lambda -
1 ) * cos(phi(a)));
    RHS(a,1) = mu * ( alphas - alpha0lr ) * sin(phi(a));

    for n = 1:2:2*numlocs-1
        LHS(a,(n+1)/2) = sin(n*phi(a)) * ( n*mu + sin(phi(a)) );
    end

end

% Could vectorize some of the above lines
```

## Solve

```
A = LHS\RHS;
```

## Extract quantities of interest

```
CL = A(1)*pi*AR;
for n = 1:numlocs
    Asum = 0;
    for m = 1:numlocs
        Asum = Asum + A(m) * sin( (2*m-1) * phi(n) );
    end
    % Could replace the above for loop with:
    %Asum = sum(A(1:numlocs).*sin(2*(1:numlocs)-1)*phi(n));
    Cl(n) = 2 * AR * ( 1 + lambda ) * ...
        (1 / ( 1 + ( lambda - 1 ) * cos(phi(n)) )) * Asum;
end

einv = 0;
for n = 1:numlocs
    einv = einv + (2*n-1)*A(n)^2/A(1)^2;
end
% Could replace the above for loop with:
%einv = sum( (2*(1:numlocs)-1)*A(1:numlocs).^2/A(1)^2 );

CDi = 2.5*einv*CL^2/(pi*AR);

e = 1/einv;
%display(['Lif is found to be = ', num2str(Cl)]);
display(['Cdi for 3-D wing, using LLT = ', num2str(CDi)]);

Cdi for 3-D wing, using LLT = 0.011604
```

*Published with MATLAB® R2017a*