Table of Contents


```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                   %
%                                                                   %
% Justin Millsap | ARO 3011 | Computer Assignment | Dr. Tony Lin %
%                                                                   %
%                                                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A) Develop a computer program that uses numerical lifting-line theory to calculate the following aerodynamic characteristics.

1) Total wing lift coefficient C_L 2) Total wing induced drag coefficient C_di 3) Spanwise lift distribution normalized with total wing lift coefficient C_l (y) / C_L 4) Spandwise distribution of bound circulation GAMMA(y) 5) Delta = (pi*A*C_Di)/(C_L)^2 - 1

A.1) Rectangular Wing - Pilatus PC-6 Turbo Porter

```
clear;clc; clear all

%%%%%   INPUTS [ EDIT ] %%%%%

epsilon_t_deg = -3;                            % Twist [ deg ]
epsilon_t_rad = epsilon_t_deg*pi/180;          % Twist [ rad ]
C_L_alpha_rad = 2*pi*0.96;                     % Section lift curve slope [ 1 /
 rad ]
SemiSpanLength = 1;                           % b/2
b = SemiSpanLength*2;                         % Wing Span
AR = 8;                                        % Aspect Ratio
AOA_absolute_deg = 5;                         % Wing Absolute AOA (at center
 section) [ deg ]
AOA_absolute_rad = AOA_absolute_deg*pi/180;  % Wing Absolute AOA (at center
 section) [ rad ]
```

```
rho = 1;
V = 1;
q = (1/2)*rho*V^2;                                          % Dynamic Pressure
N = 128;                                                     % Number of itterations

%%%%%% Equations [ DO NOT EDIT ]    %%%%%
% Total wing lift coefficient C_L

c = b/AR;                                                   % Chord Length
S = b*c;                                                    % Wing Area

i = 1:N;
j = 1:N;

%  Determining k_ij values
%  if i => j -----> k = 1
%  if i < j  -----> k =0

k = zeros(N,N);
k = k(i,j);
for i = 1:N;
    for j = 1:N;
        if i>= j;
            k(i,j) = 1;
        else i < j ;
            k(i,j) = 0;
        end
    end
end



% Cosine Spacing y_vi

for i = 1:N;
   m = 2*(N-1) + 1;
   deltaTheta = pi/m;
   y_v(i) = (b/2)*cos((1 - i + (1/2)*(m-1))*deltaTheta);
end

% Determining y_ci  Span Location of computation of downwash

 for i = 2:N;

y_c(1) = 0;
y_c(i) = (1/2)*(y_v(i) + y_v(i-1));

 end
 % y_c_128 = y_c

 % save('y_c_128.mat','y_c_128')

% Determining AOA_j
for j = 1:N
```

```
    for i = 1:N

AOA(i) = epsilon_t_rad*y_c(i) + AOA_absolute_rad;

    end
end


% Determining C_ij values
for i = 1:N;
    for j = 1:N;

C(i,j) = (1/(2*pi)) * (y_v(i)) / ( (y_v(i))^2 - (y_c(j))^2 );

    end
end

% Defining Chord Length "c"

for j = 1:N

    c(j) = b/AR ;

end

% Defining A

for i = 1:N
    for j = 1:N
        A(j,i) = (1/2)*(c(j))*(C_L_alpha_rad)*C(i,j) + k(i,j);
    end

end


% Defining B

for j = 1:N
    B(j) = (1/2)*c(j)*(C_L_alpha_rad)*AOA(j);
end
B = B' ;

% Defining gamma_lower
gamma_lower = (A)^-1 * B;


% Defining w ( down - wash velocity )
w = zeros(N,1);
for j = 1:N
    for i = 1:N

        w(j) = w(j) + C(i,j)*gamma_lower(i);

    end
```

```
end


% Defining AOA_dw ( Down-wash Angle of Attack)
AOA_dw = zeros(N , 1);
for j = 1:N;
    for i = 1:N;
        AOA_dw(j) = w(j)/V;
    end
end


% Define delta_y

delta_y = zeros(N,1);

delta_y(1) = (y_v(1)+y_v(1));

for i = 2:N
        delta_y(i) =2*( y_v(i) - y_v(i-1));
end


% Calculate lift coefficients C_Li

for j = 1:N;

    C_l(j) =  C_L_alpha_rad*(AOA(j) - AOA_dw(j));

end

% Define gamma_cap for a rectangular wing
gamma_cap_rect = zeros(N, 1);  % Initialize gamma_cap as a column vector of
 zeros

for i = 1:N
    gamma_cap_rect(i) = (1/2)*c(i)*C_l(i);
end

    gamma_cap_y_rect = gamma_cap_rect;
% Calculate Lift Coefficient
for i = 1:N;
    L(i) = rho*V*(gamma_cap_rect(i)*delta_y(i));
end

C_L_rect = sum(L)/(q*S);

%Calculate Drag Coefficient CD_i
C_D_rect = zeros(N,1);
    C_D_rect(1) = gamma_cap_rect(1)*y_v(1)*w(1);
for i = 2:N;
      C_D_rect(i) = gamma_cap_rect(i)* (y_v(i) - y_v(i-1))*w(i);
end
C_D_i_rect = AR*( C_D_rect(1) + sum(C_D_rect(2:N)) );
```

```
% Spanwise lift distribution C_ly
C_ly = sum(C_l);



C_ly_CL_rect =  C_l/ C_L_rect;



C_lyCl = C_ly/C_L_rect;

% Determine Delta
Delta_rect = ( (pi*AR*C_D_i_rect)/(C_L_rect^2) ) - 1;


% gamma_cap_y_rect_2 = gamma_cap_y_rect
% gamma_cap_y_rect_8 = gamma_cap_y_rect
% gamma_cap_y_rect_32 = gamma_cap_y_rect
% gamma_cap_y_rect_128 = gamma_cap_y_rect
% %
% %
%  save('Gamma_cap_rect_N_128.mat','gamma_cap_y_rect_128')
```

A.2) Tapered Wing - Cessna Cition Bravo

```
lambda = 0.3;

%%%%%% Equations [ DO NOT EDIT ]    %%%%%
% Total wing lift coefficient C_L

S = (b^2)/AR;


c_root = (2*S)/(b*(1+lambda));
c_tip = lambda*c_root;




i = 1:N;
j = 1:N;

%  Determining k_ij values
%  if i => j -----> k = 1
%  if i < j  -----> k =0

k = zeros(N,N);
k = k(i,j);
for i = 1:N;
    for j = 1:N;
        if i>= j;
            k(i,j) = 1;
```

```
        else i < j ;
            k(i,j) = 0;
        end
    end
end



% Cosine Spacing y_vi

for i = 1:N;
   m = 2*(N-1) + 1;
   deltaTheta = pi/m;
   y_v(i) = (b/2)*cos((1 - i + (1/2)*(m-1))*deltaTheta);
end



% Determining AOA_j
for j = 1:N
    for i = 1:N

AOA(i) = epsilon_t_rad*y_c(i) + AOA_absolute_rad;

    end
end


% Determining C_ij values
for i = 1:N;
    for j = 1:N;

C(i,j) = (1/(2*pi)) * (y_v(i)) / ( (y_v(i))^2 - (y_c(j))^2 );

    end
end

% Define chord length for a Tapered Wing
c_tapered = zeros(N, 1);
for i = 1:N;
    y = y_c(i);
    c_tapered(i) = c_root * (1 - (2 * y / b) * (1 - lambda));
end
% Defining A

for i = 1:N;
    for j = 1:N;
        A(j,i) = (1/2)*(c_tapered(j))*(C_L_alpha_rad)*C(i,j) + k(i,j);
    end

end


% Defining B
```

```
for j = 1:N
    B(j) = (1/2)*c_tapered(j)*(C_L_alpha_rad)*AOA(j);
end
B = B' ;



% Defining gamma_lower
gamma_lower = A \ B(:);




% Defining w ( down - wash velocity )
w = zeros(N,1);
for j = 1:N;
    for i = 1:N;

        w(j) = w(j) + C(i,j)*gamma_lower(i);

    end
end


% Defining AOA_dw ( Down-wash Angle of Attack)
AOA_dw = zeros(N , 1);
for j = 1:N;
    for i = 1:N;
        AOA_dw(j) = w(j)/V;
    end
end


% Define delta_y

delta_y = zeros(N,1);

delta_y(1) = (y_v(1)+y_v(1));

for i = 2:N
        delta_y(i) =2*( y_v(i) - y_v(i-1));
end





% Calculate lift coefficients C_Li

for j = 1:N;

    C_l(j) =  C_L_alpha_rad*(AOA(j) - AOA_dw(j));
```

```
end

% Define gamma_cap for a rectangular wing
gamma_cap_taper = zeros(N, 1);  % Initialize gamma_cap as a column vector of
 zeros

for i = 1:N;
    gamma_cap_taper(i) = (1/2)*c_tapered(i)*C_l(i);
end;

    gamma_cap_y_taper = gamma_cap_taper;

% Calculate Lift Coefficient
for i = 1:N;
    L(i) = rho*V*(gamma_cap_taper(i)*delta_y(i));
end

C_L_taper= sum(L)/(q*S);

%Calculate Drag Coefficient CD_i

C_D_taper = zeros(N,1);
    C_D_taper(1) = gamma_cap_taper(1)*y_v(1)*w(1);
for i = 2:N;
      C_D_rect(i) = (gamma_cap_taper(1) * y_v(1) * w(1) + gamma_cap_taper(i)*
 (y_v(i) - y_v(i-1))*w(i));
end

C_D_i_taper = AR*( C_D_taper(1) + sum(C_D_taper(2:N)) );


% Spanwise lift distribution C_ly
C_ly = sum(C_l);




C_ly_CL_taper =  C_l/ C_L_taper;




C_lyCl = C_ly/C_L_taper;

% Determine Delta
Delta_taper= ( (pi*AR*C_D_i_taper)/(C_L_taper^2) ) - 1;

% Determine gamma_cap_y
gamma_cap_y_taper = gamma_cap_taper;
%  gamma_cap_y_taper_2 = gamma_cap_y_taper
% gamma_cap_y_taper_8 = gamma_cap_y_taper
% gamma_cap_y_taper_32 = gamma_cap_y_taper
% gamma_cap_y_taper_128 = gamma_cap_y_taper
% %
%  save('Gamma_cap_taper_N_128.mat','gamma_cap_y_taper_128')
```

A.3) Elliptical Wing - Supermarine Spitfire

```
%%%%%% Equations [ DO NOT EDIT ]    %%%%%
% Total wing lift coefficient C_L


S = (b^2)/AR;                                            % Wing Area

i = 1:N;
j = 1:N;

%  Determining k_ij values
%  if i => j -----> k = 1
%  if i < j  -----> k =0

k = zeros(N,N);
k = k(i,j);
for i = 1:N;
    for j = 1:N;
        if i>= j;
            k(i,j) = 1;
        else i < j ;
            k(i,j) = 0;
        end
    end
end



% Cosine Spacing y_vi

for i = 1:N;
   m = 2*(N-1) + 1;
   deltaTheta = pi/m;
   y_v(i) = (b/2)*cos((1 - i + (1/2)*(m-1))*deltaTheta);
end
% Determining y_ci  Span Location of computation of downwash

 i = 2:N;

y_c(1) = 0;
y_c(i) = (1/2)*(y_v(i) + y_v(i-1));

% y_c_32 = y_c
%
% save('y_c_32.mat','y_c_32')

%Calculating chord length for an elliptical wing


for i = 1:N;
    y = y_c(i);
    c_elliptical(i) = (4*S/(pi*b)) * sqrt(1 - (2*y/b)^2);
```

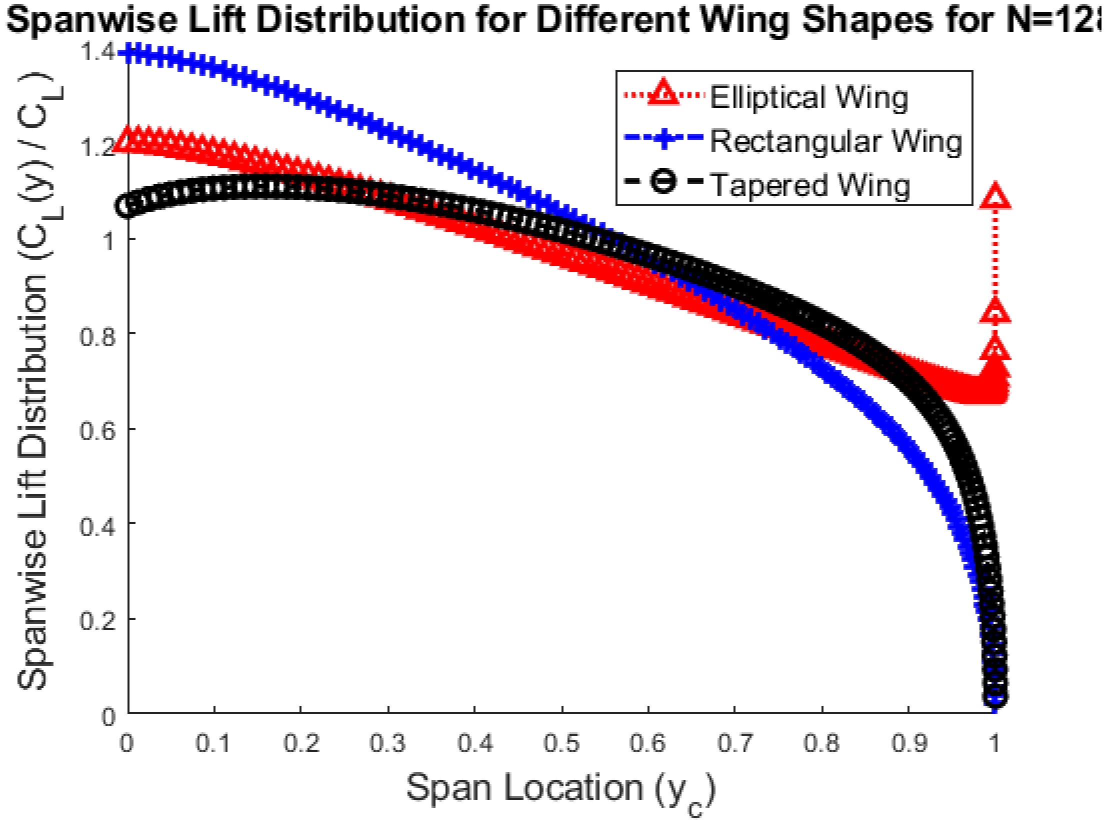
```
end



% Determining AOA_j
for j = 1:N
    for i = 1:N

AOA(i) = epsilon_t_rad*y_c(i) + AOA_absolute_rad;

    end
end




% Determining C_ij values
for i = 1:N;
    for j = 1:N;

C(i,j) = (1/(2*pi)) * (y_v(i)) / ( (y_v(i))^2 - (y_c(j))^2 );

    end
end



% Defining A

for i = 1:N
    for j = 1:N
        A(j,i) = (1/2)*(c_elliptical(j))*(C_L_alpha_rad)*C(i,j) + k(i,j);
    end

end


% Defining B

for j = 1:N
    B(j) = (1/2)*c_elliptical(j)*(C_L_alpha_rad)*AOA(j);
end
B = B' ;



% Defining gamma_lower
gamma_lower = A \ B(:);



% Defining w ( down - wash velocity )
w = zeros(N,1);
for j = 1:N;
```

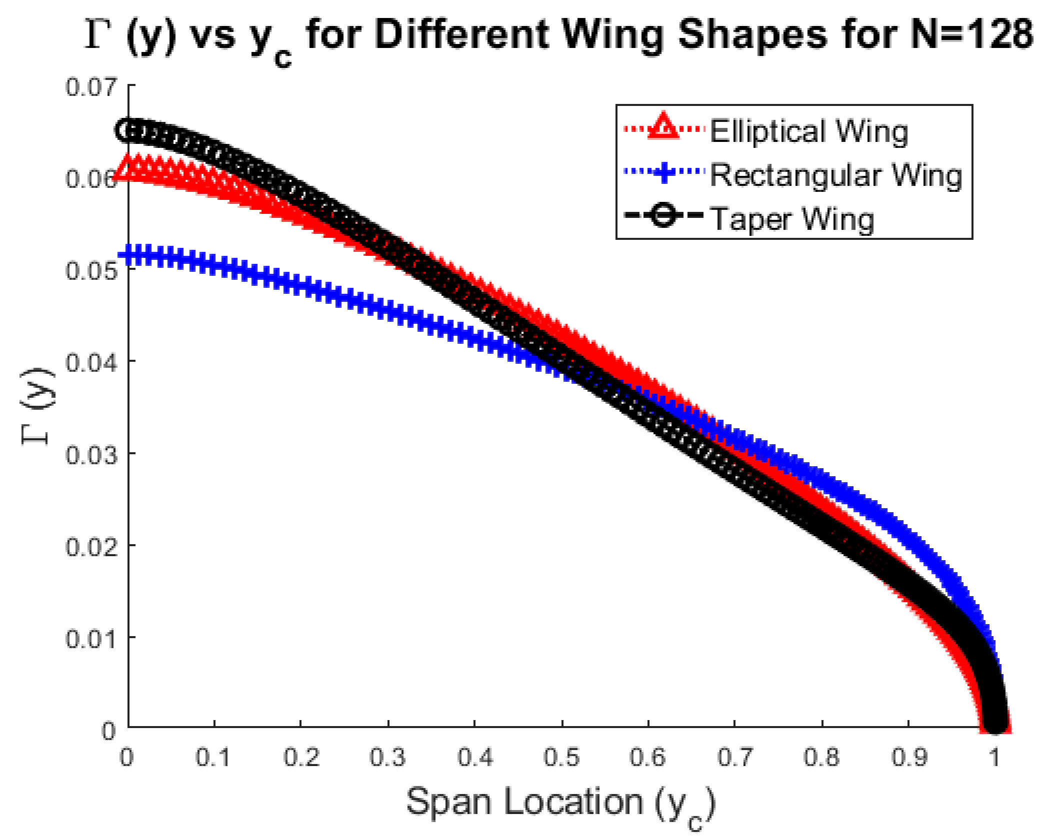
```
    for i = 1:N;

        w(j) = w(j) + C(i,j)*gamma_lower(i);

    end
end


% Defining AOA_dw ( Down-wash Angle of Attack)
AOA_dw = zeros(N , 1);
for j = 1:N;
    for i = 1:N;
        AOA_dw(j) = w(j)/V;
    end
end


% Define delta_y

delta_y = zeros(N,1);

delta_y(1) = (y_v(1)+y_v(1));

for i = 2:N
        delta_y(i) =2*( y_v(i) - y_v(i-1));
end



% Calculate lift coefficients C_Li

for j = 1:N;

    C_l(j) =  C_L_alpha_rad*(AOA(j) - AOA_dw(j));

end

% Define gamma_cap for a Elliptical Wing
gamma_cap_ellip = zeros(N, 1);  % Initialize gamma_cap as a column vector of
 zeros

for i = 1:N;
    gamma_cap_ellip(i) = (1/2)*c_elliptical(i)*C_l(i);
end

    gamma_cap_y_ellip = gamma_cap_ellip;

 % Calculate Lift Coefficient
for i = 1:N;
    L(i) = rho*V*(gamma_cap_ellip(i)*delta_y(i));
end

C_L_ellip = sum(L)/(q*S);
```

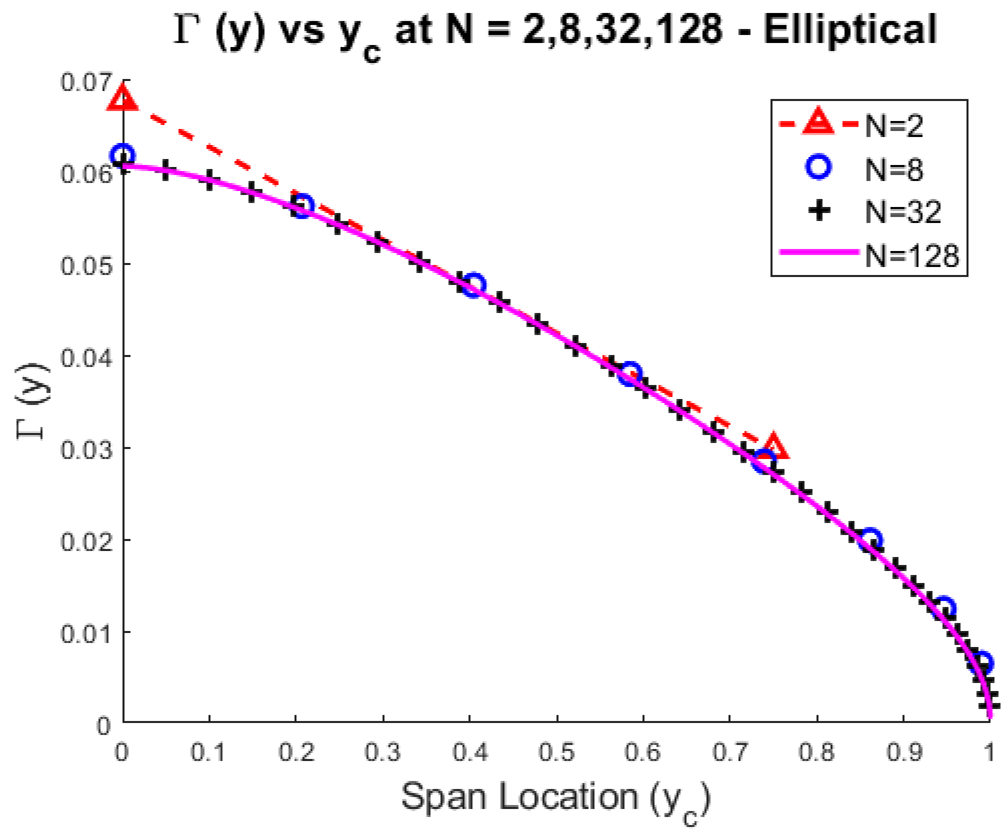
```
% Spanwise lift distribution C_ly
C_ly = sum(C_l);

%Calculate Drag Coefficient CD_i

C_D_ellip = zeros(N,1);
    C_D_ellip(1) = gamma_cap_ellip(1)*y_v(1)*w(1);
for i = 2:N;
      C_D_ellip(i) = (gamma_cap_ellip(1) * y_v(1) * w(1) + gamma_cap_ellip(i)*
 (y_v(i) - y_v(i-1))*w(i));
end
C_D_i_ellip = AR*( C_D_ellip(1) + sum(C_D_ellip(2:N)) );


C_ly_CL_ellip =  C_l/ C_L_ellip;



C_lyCl = C_ly/C_L_ellip;

% Determine Delta
Delta_elliptical = ( (pi*AR*C_D_i_ellip)/(C_L_taper^2) ) - 1;

% Determine gamma_cap_y


%  gamma_cap_y_ellip_2 = gamma_cap_y_ellip
% gamma_cap_y_ellip_8 = gamma_cap_y_ellip
% gamma_cap_y_ellip_32 = gamma_cap_y_ellip
% gamma_cap_y_ellip_128 = gamma_cap_y_ellip
%
%  save('Gamma_cap_ellip_N_128.mat','gamma_cap_y_ellip_128')


load('Gamma_cap_rect_N_2.mat','gamma_cap_y_rect_2')
load('Gamma_cap_rect_N_8.mat','gamma_cap_y_rect_8')
load('Gamma_cap_rect_N_32.mat','gamma_cap_y_rect_32')
load('Gamma_cap_rect_N_128.mat','gamma_cap_y_rect_128')

load('Gamma_cap_taper_N_2.mat','gamma_cap_y_taper_2')
load('Gamma_cap_taper_N_8.mat','gamma_cap_y_taper_8')
load('Gamma_cap_taper_N_32.mat','gamma_cap_y_taper_32')
load('Gamma_cap_taper_N_128.mat','gamma_cap_y_taper_128')

load('Gamma_cap_ellip_N_2.mat','gamma_cap_y_ellip_2')
load('Gamma_cap_ellip_N_8.mat','gamma_cap_y_ellip_8')
load('Gamma_cap_ellip_N_32.mat','gamma_cap_y_ellip_32')
load('Gamma_cap_ellip_N_128.mat','gamma_cap_y_ellip_128')

load('y_c_2.mat','y_c_2')
load('y_c_8.mat','y_c_8')
load('y_c_32.mat','y_c_32')
load('y_c_128.mat','y_c_128')
```

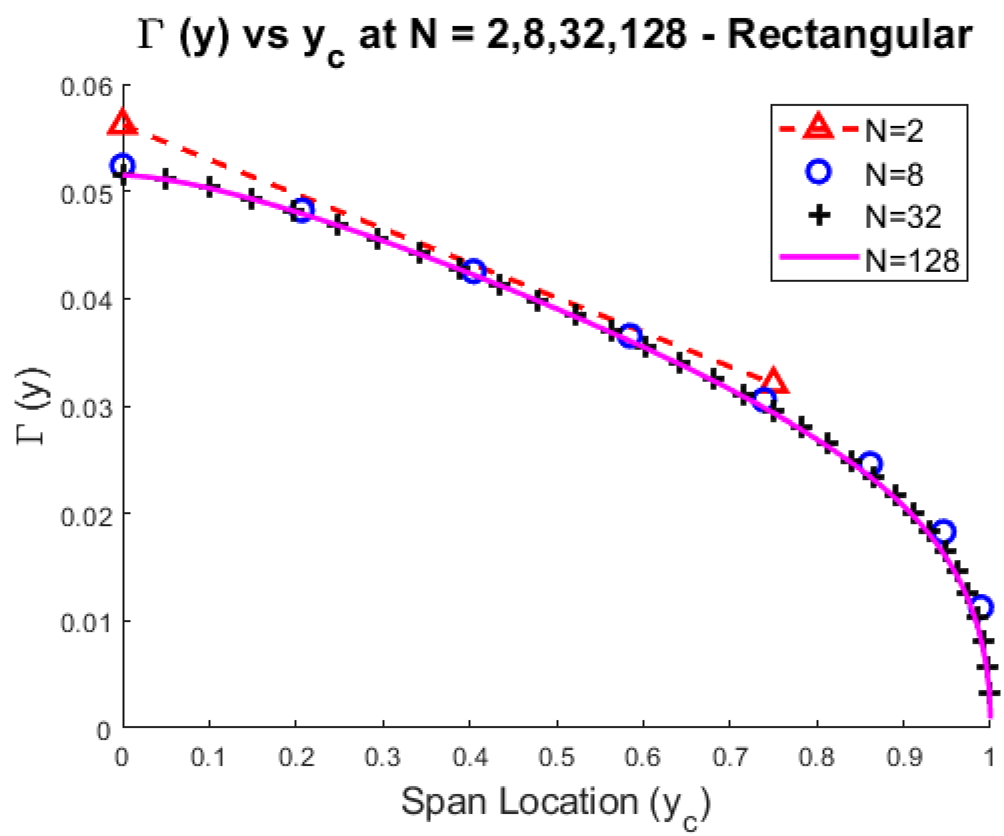
```
% Spanwise Lift distribution vs Y_c for all wing configurations

figure(1)

hold on

plot(y_c, C_ly_CL_ellip, "r:^", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "Elliptical Wing")
plot(y_c, C_ly_CL_rect, "b -.+", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "Rectangular Wing")
plot(y_c, C_ly_CL_taper, "k --o", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "Tapered Wing")
legend('show', 'FontSize', 12); % Set FontSize for legend
xlabel('Span Location (y_c)', 'FontSize', 14); % Set FontSize for xlabel
ylabel('Spanwise Lift Distribution (C_L(y) / C_L)', 'FontSize', 14); % Set
 FontSize for ylabel
title('Spanwise Lift Distribution for Different Wing Shapes for
 N=128', 'FontSize', 14); % Set FontSize for title

hold off

% Gamma(y) vs Y_c for all wing configuarions

figure(2)
hold on
plot(y_c, gamma_cap_y_ellip, "r:^", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "Elliptical Wing")
plot(y_c, gamma_cap_y_rect, "b :+", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "Rectangular Wing")
plot(y_c, gamma_cap_y_taper, "k -.o", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "Taper Wing")
legend('show', 'FontSize', 12);
xlabel('Span Location (y_c)', 'FontSize', 14);
ylabel('\Gamma (y)', 'FontSize', 14);
title('\Gamma (y) vs y_c for Different Wing Shapes for N=128', 'FontSize',
 16);

hold off

% Elliptical Wing

figure(3)
hold on

plot(y_c_2, gamma_cap_y_ellip_2, "r^--", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "N=2")
plot(y_c_8, gamma_cap_y_ellip_8, "bo", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "N=8")
plot(y_c_32, gamma_cap_y_ellip_32, "k +", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "N=32")
```

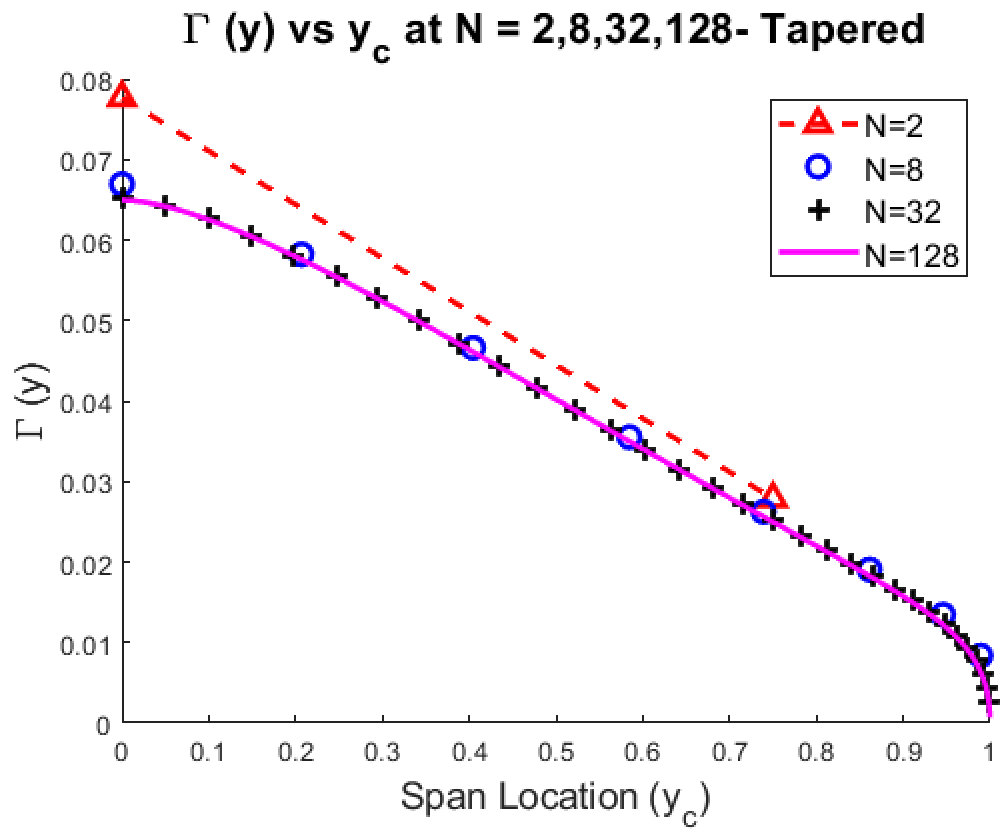
```
plot(y_c_128, gamma_cap_y_ellip_128, "m -", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "N=128")
xlabel('Span Location (y_c)', 'FontSize', 14);
ylabel('\Gamma (y)', 'FontSize', 14);
title('\Gamma (y) vs y_c at N = 2,8,32,128 - Elliptical', 'FontSize', 16);
legend('show', 'FontSize', 12);

hold off

% Rectangular Wing

figure(4)
hold on
plot(y_c_2, gamma_cap_y_rect_2, "r^--", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "N=2")
plot(y_c_8, gamma_cap_y_rect_8, "bo", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "N=8")
plot(y_c_32, gamma_cap_y_rect_32, "k +", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "N=32")
plot(y_c_128, gamma_cap_y_rect_128, "m -", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "N=128")
xlabel('Span Location (y_c)', 'FontSize', 14);
ylabel('\Gamma (y)', 'FontSize', 14);
title('\Gamma (y) vs y_c at N = 2,8,32,128 - Rectangular', 'FontSize', 16);
legend('show', 'FontSize', 12);

hold off

% Tapered Wing

figure(5)
hold on

plot(y_c_2, gamma_cap_y_taper_2, "r^--", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "N=2")
plot(y_c_8, gamma_cap_y_taper_8, "bo", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "N=8")
plot(y_c_32, gamma_cap_y_taper_32, "k +", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "N=32")
plot(y_c_128, gamma_cap_y_taper_128, "m -", "LineWidth", 2, "MarkerSize",
 8, "DisplayName", "N=128")
xlabel('Span Location (y_c)', 'FontSize', 14);
ylabel('\Gamma (y)', 'FontSize', 14);
title('\Gamma (y) vs y_c at N = 2,8,32,128- Tapered', 'FontSize', 16);
legend('show', 'FontSize', 12);

hold off
```


Spanwise Lift Distribution for Different Wing Shapes for N=128
Spanwise Lift Distribution ($C_L(y) / C_L$)
Span Location (y_c)
Elliptical Wing
Rectangular Wing
Tapered Wing


Γ (y) vs y_c for Different Wing Shapes for N=128
Γ (y)
Span Location (y_c)
Elliptical Wing
Rectangular Wing
Taper Wing


Γ (y) vs y_c at N = 2,8,32,128 - Elliptical
N=2
N=8
N=32
N=128
Γ (y)
Span Location (y_c)
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Γ (y) vs y_c at N = 2,8,32,128 - Rectangular
N=2
N=8
N=32
N=128
Γ (y)
Span Location (y_c)
0
0.01
0.02
0.03
0.04
0.05
0.06
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



Published with MATLAB® R2021b