BEMT.m function user guide

AUTHORS: CRISPINO UMBERTO & DI MAIO DANIELE

1. Introduction and description

The following function allows the determination of the induced incidence angle α_i of the blade element, once known the geometry of the blade, the aerodynamic characteristics of the profiles used and the number of blades.

Any proposed consideration is made with reference to Figure 1.1.

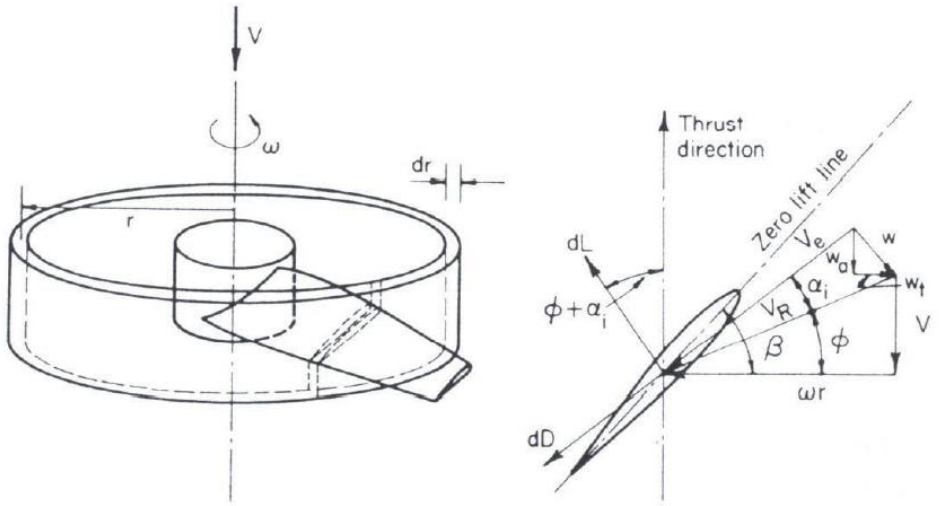

Figure 1.1: Pattern of a blade element

Three methods are available in the code to evaluate this angle:

• ***Method N1***: It is assumed that the angle of incidence is small and that the axial induced velocity is only a function of the parameter *r*. In such hypotheses I can express the elementary thrust as

$$dT = \rho(2\pi r dr)(V + \alpha_i V_R cos(\phi))2\alpha_i V_R cos(\phi)$$

which for an N-blade propeller is also given by

$$dT = N\frac{1}{2}\rho V_R^2 c C_{l\alpha}(\beta - \phi - \alpha_i)cos(\phi)dr$$

Equalizing the two expressions a second-degree equation in the unknown α_i is obtained:

$$\alpha_i^2 + \alpha_i\left(\frac{\lambda}{r} + \frac{\sigma C_{l\alpha}}{8r^2}\frac{V_R}{V_T}\right) - \frac{\sigma C_{l\alpha}}{8r^2}\frac{V_R}{V_T}(\beta - \phi) = 0$$

Choosing the positive root gives the value of the angle of incidence induced:

$$\alpha_i = \frac{1}{2}\left[-\left(\frac{\lambda}{r} + \frac{\sigma C_{l\alpha}}{8r^2}\frac{V_R}{V_T}\right) + \sqrt{\left(\frac{\lambda}{r} + \frac{\sigma C_{l\alpha}}{8r^2}\frac{V_R}{V_T}\right)^2 + \frac{\sigma C_{l\alpha}}{2r^2}\frac{V_R(\beta - \phi)}{V_T}}\right]$$

where: $\lambda = \frac{V}{\Omega R}$, $V_T = \Omega r$ e $V_R = V_T\sqrt{\lambda^2 + r^2}$

• ***Method N2***: Starting from the results of *A-Method*, the corrective function of Prandtl *F* is applied, obtaining again an equation of second degree in the unknown α_i that resolved returns the value of:

$$\alpha_i = \frac{1}{2}\left[-\left(\frac{\lambda}{r} + \frac{\sigma C_{l\alpha}}{8rFcos(\phi)}\right) + \sqrt{\left(\frac{\lambda}{r} + \frac{\sigma C_{l\alpha}}{8rFcos(\phi)}\right)^2 + \frac{\sigma C_{l\alpha}}{2rFcos(\phi)}(\beta - \phi)}\right]$$

where: $F(r) = \frac{2}{\pi}\arccos\left[e^{\frac{N}{2\lambda}\left(\frac{r-R}{R}\right)}\right]$

• ***Method N3***: No simplifying hypothesis is made, so an implicit equation to derive α_i must be solved. The equation is solved in an iteratively; the first attempt is obtained using the induced rotational velocity evaluated by the B-method while the induced axial velocity is obtained by the relation:

$$w_t(r) = \frac{\sigma V_e C_{l\alpha}}{8rF}(\beta - \alpha_i - \phi) = \frac{\sigma V_e C_{l\alpha}}{8rF}\left[\beta - \arctan\left(\frac{w_t}{w_a}\right)\right]$$

By calculating the actual velocity in these hypotheses we obtain:

$$v_e^2 = (\Omega Rr - w_t)^2 + (w_a + v_\infty)^2 = (\Omega Rr - w_t)^2 + \left[v_\infty + \frac{1}{2}\sqrt{v_\infty^2 + 4w_t(\Omega Rr - w_t)} - \frac{1}{2}v_\infty\right]^2$$

from which, resolving:

$$w_t = \frac{\Omega\sigma v_e C_{l\alpha}}{8rF}\left[\beta - tan^{-1}\left(\frac{w_t}{w_a}\right)\right]\sqrt{\left(r - \frac{w_t}{\Omega R}\right)^2 + \frac{1}{4}\left[\sqrt{\lambda^2 + 4\frac{w_t}{\Omega R}\left(r - \frac{w_t}{\Omega R}\right)} + \lambda\right]^2}$$

that must be solved iteratively in w_t. Note the induced velocities is then immediate calculation of the angle of incidence induced:

$$\alpha_i = \arctan\left(\frac{w_t}{w_a}\right) - \phi$$

2. I/O

The function is intended to take in input the following variables:

- V_∞, asymptotic velocity of the stream in [m/s];
- rpm, number of revolutions per minute in [rad/min];
- R, radius of the propeller in [m];
- r_{hub}, radius of the hub of the propeller in [m];
- N, number of blades of the propeller;
- ns, number of stations in which the blade is divided.

Once inserted these data, *ns* elements will be required in order to obtain:

- c, the chord vector of each station in [m];
- $C_{l\alpha}$ vector of each station in [1/rad];
- β, the vector of pitch angle referred to the zero-lift line in [deg].

Input data inserting mode is very strict: the function must be carefully called because the order of the input vectors can only be the following one. Input data given in a different order could result in wrong outputs.

The outputs of this function are the distributions of the induced incidence angle along the blade radius obtained from the three resolution methods; also the vectors entered by the user are returned to output in order to check the consistency of the data.

3. Algorythm description

The algorithm uses the equations shown in [**1**]. First of all, the function requires input details *ns* times about blade geometry and aerodynamic characteristics. The function will ask the user to insert the chord in meters, the $C_{l\alpha}$ coefficent of the blade element and the pitch in degrees for each station.

```
p=1;
while p< ns+1
s=input(['INSERT THE VALUE OF THE CHORD [m] AT THE STATION
',num2str(p), ' : ']);
c(p)=s;
p = p+1;
end

p=1;
while p< ns+1
f=input(['INSERT THE VALUE OF THE Cl_ALPHA [1/rad] AT THE STATION
',num2str(p), ' : ']);
```

```
Cl_alpha(p)=f;
p = p+1;
end

p=1;
while p< ns+1
d=input(['INSERT THE VALUE OF THE PITCH ANGLE [deg] AT THE STATION
',num2str(p), ' : ']);
beta_deg(p)=d;
p=p+1;
end
```

After that, the function asks the user which method he wants to use.

```
disp('Which theory do you want to use?')
disp('1 - IMPULSIVE THEORY')
disp('2 - WHIRLING THEORY WITH SMALL DISTURBANCES')
disp('3 - WHIRLING THEORY')

l= input(['']);
```

Next, the function converts input data in such a way as to use the correct dimensions during the development of the chosen method.

```
omega=convangvel(rpm, 'rpm', 'rad/s');
r=linspace(r_hub, R, ns);
beta=convang(beta_deg, 'deg', 'rad');
r_ndim = r/R;
x=linspace(r_hub,R,100);
lam= V/(omega*R);
sigma= N*c./(pi*R);
Vt= omega*R;
Vr= sqrt((V^2)+ (omega*R.*r_ndim).^2);
phi= atan ((V)./(omega*r));
```

At this point the function is split into various methods.

Method N1. It can be used if and only if the angle of incidence is small and the axial induced velocity is only a function of the parameter *r*.

```
%% METHOD N1
if l==1;

alpha_i= 0.5*(-(lam./r_ndim +
sigma.*Cl_alpha.*(Vr./Vt)./(8*r_ndim.^2)) +...
    sqrt((lam./r_ndim +
sigma.*Cl_alpha.*(Vr./Vt)./(8*r_ndim.^2)).^2 +...
    sigma.*Cl_alpha.*Vr.*(beta-phi)./(2*(r_ndim.^2).*Vt)));
```

```
alpha_i_spline=interp1(r,alpha_i,x,'spline');
else
```

Method N2. It applies the corrective function of Prandtl F, outputting the angle of incidence among the blade. A control about the value of ϕ at the tip is done.

```
%% METHOD N2

if l==2

   if phi(ns)== 0
    F= ones (1, ns)
   else
F=(2/pi)*acos(exp(N/(2*lam)*((r-R)/R)));
   end

alpha_i= 0.5*(-(lam./r_ndim +
sigma.*Cl_alpha./(8*r_ndim.*F.*cos(phi))) ...
         + sqrt((lam./r_ndim +
sigma.*Cl_alpha./(8*r_ndim.*F.*cos(phi))).^2 ...
         + sigma.*Cl_alpha.*(beta-phi)./(2*r_ndim.*F.*cos(phi))));

alpha_i_spline=interp1(r,alpha_i,x,'spline');
else
```

Method N3. In this method no simplification is done. This one starts from the values of method N2.

```
%% METHOD N3

if l==3
    warndlg('WARNING:THIS METHOD MAY BE INCORRECT');

       if phi(ns)== 0
       F= ones (1, ns)
       else
       F=(2/pi)*acos(exp(N/(2*lam)*((r-R)/R)));
       end
alpha_i_B= 0.5*(-(lam./r_ndim +
sigma.*Cl_alpha./(8*r_ndim.*F.*cos(phi))) ...
         + sqrt((lam./r_ndim +
sigma.*Cl_alpha./(8*r_ndim.*F.*cos(phi))).^2 ...
         + sigma.*Cl_alpha.*(beta-phi)./(2*r_ndim.*F.*cos(phi))));

w_t_B= Vr.*alpha_i_B.*sin(phi+alpha_i_B); % formula C-8
w_t_C= sigma.*Cl_alpha./(8*r.*F).*Vr.*(beta- phi-alpha_i_B); %
w_a_B= Vr.*alpha_i_B.*cos(phi+alpha_i_B); % formula C-9

alfaisa= alpha_i_B;
wtsa= w_t_B;
wasa= w_a_B;
```

To enter the cycle, it needs to inizialize the control term and the error term. In addition a term about the max number of iterations is introduced to avoid infinite loop in case of any bugs.

```
value= 10* ones(1, ns);
epsi= 1e-10;
max_iter=1000;
```

The value of angle of incidence is computed through Newton-Raphson iterative method, that goes on until the difference between the values of two consecutive iterations is lower than an error. During each iteration a control about the arguments of square roots, that must be greater than zero, is done. We used a cycle *for* to apply the right number of iterations at each station. In this way each element blade doesn't depend from others computation and in case of bug it's easy to identify in which station the problem has arisen.

```
for j= 1:ns
  i=0;
if value(j) < epsi
w_t_B(j)= w_t_B(j)-(y(j)/dy(j));
else

while [value(j)> epsi , i<max_iter+1]
    i= i+1;
    test1(j)= V^2+ (4*w_t_B(j).*((omega*R*r_ndim(j))- w_t_B(j)));
%radice della C-15          %OK
    test2(j)= lam^2+ (4*w_t_B(j)./(Vt)).*(r_ndim(j)-
(w_t_B(j)./(Vt))); %radice della C-17     %OK

 if (test1 <0 | test2 <0)
     w_t_B(j)=wtsa(j);
     w_a_B(j)=wasa(j);
 else
term1(j)= (sqrt(test1(j))-V);  %2*w_a_C dalla C-15
term2(j)= lam+ sqrt(test2(j)); %seconda radice di C-17

aa(j)= beta(j)-(atan((w_t_B(j)*2)./(term1(j)))); % termine in
parentesi nella C-14
bb(j)= sqrt((0.25*(term2(j).^2)+((r_ndim(j)-w_t_B(j)./Vt)).^2));
%tutto il termine sotto radice della C-17
cc(j)= 8*r_ndim(j).*F(j).*w_t_B(j)./Vt; % ha portato al primo
membro tutti i termini della C-17 tranne sigma,..
                      % Cl_alpha, aa, bb
da(j)= ((-2)./(((term1(j)).^2)+
4*w_t_B(j).*w_t_B(j))).*(term1(j)...
    -(w_t_B(j).*(((2*omega*R*r_ndim(j))-
(4*w_t_B(j)))./(term1(j)+V))));
```

```
db(j)= (((term2(j).*((r_ndim(j)./Vt)-
((2*w_t_B(j))./((Vt).^2))))...
    ./(2*(term2(j)- lam)))+ (w_t_B(j)./((Vt).^2))-
(r_ndim(j)./Vt))./bb(j);
dc(j)= 8*r_ndim(j).*F(j)./Vt;
y(j)= (sigma(j).*Cl_alpha(j).*aa(j).*bb(j))- cc(j);
dy(j)= (sigma(j).*Cl_alpha(j).*((aa(j).*db(j))+(bb(j).*da(j))))-
dc(j);
value(j)= abs(y(j)./dy(j));
w_t_B(j)= w_t_B(j)-(y(j)./dy(j));
alpha_i(j)= beta(j)-aa(j)-phi(j);
end
end
end
I(j)=i;
end

alpha_i_spline=interp1(r,alpha_i,x,'spline');

if i>max_iter
disp('There is no value for w_t');
end
end
end
end
alpha_i_deg= convang(alpha_i,'rad','deg');
alpha_i_deg_spline= convang(alpha_i_spline,'rad','deg');
```

At the end of the computation the function converts the angle of incidence to degree allowing a clearer vision of it and plots it among the blade.

```
alpha_i_deg= convang(alpha_i,'rad','deg');
alpha_i_deg_spline= convang(alpha_i_spline,'rad','deg');

figure
plot(x, alpha_i_deg_spline, 'k');
xlabel('r [m]');ylabel('\alpha_i [deg]');
axis ([0 R min(alpha_i_deg)-1 max(alpha_i_deg)+1]);
```

4. Example

A test case for the function is shown below.

A comparison is made between the value of the angle of incidence induced for the 3 methods. Furthermore, in order to validate the function, the Wieck propeller was analyzed by making a comparison (with the same geometry, velocity and rpm) between the C_l obtained with the *XROTOR* software and the C_l obtained with the three methods implemented in the function.

The values of C_l at each station is read by Matlab through the file "*myprop_xrotor*".

Apart from some discrepancies around the hub, it can be seen that the function is very consistent.

The results obtained are shown in the following graphs.

```
%% Test_case

load("myprop_xrotor.dat");
Cl_xrotor = myprop_xrotor(:,5)';
r_xrotor  = myprop_xrotor(:,2)';

V     = 20;    %m/s
rpm   = 600;
R     = 1.58;  %m
r_hub = 0.2;   %m
N     = 3;
ns    = 30;

[alpha_i, beta_deg, c, Cl_alpha]=BEMT(V, rpm, R, r_hub, N, ns);
```

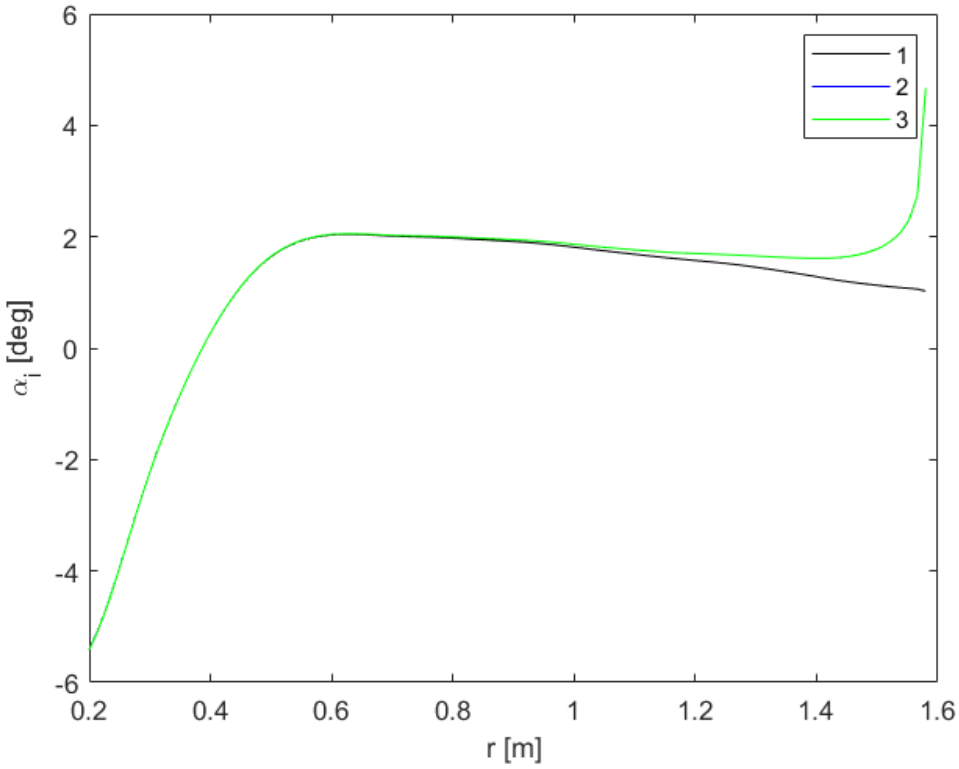

Figure 1.2: α *vs* r

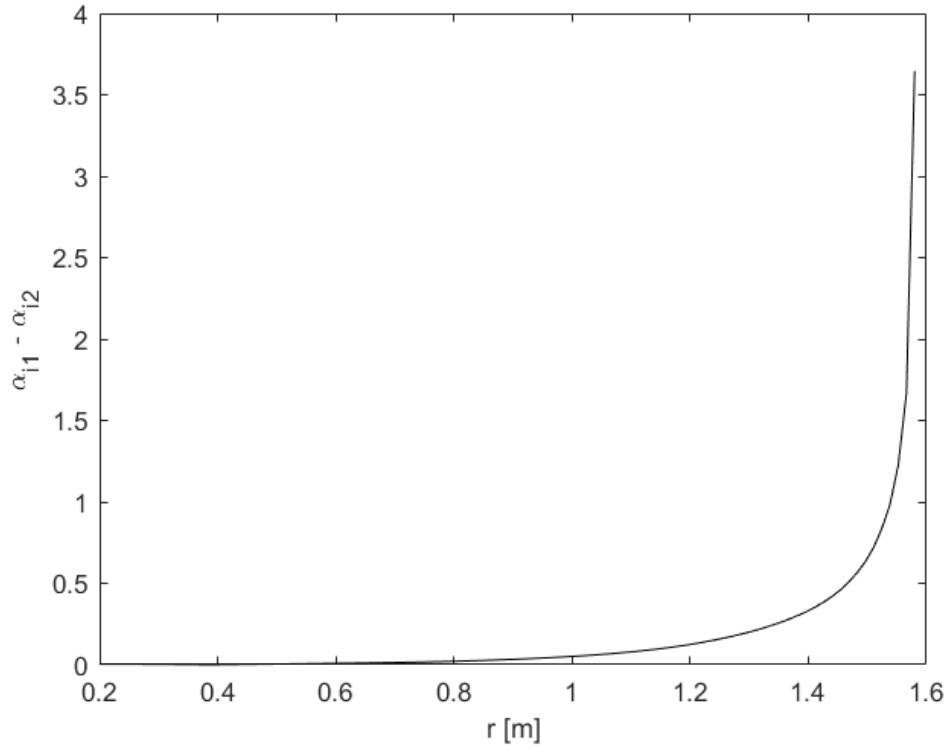

Figure 1.3: α_1-α_2 *vs* r

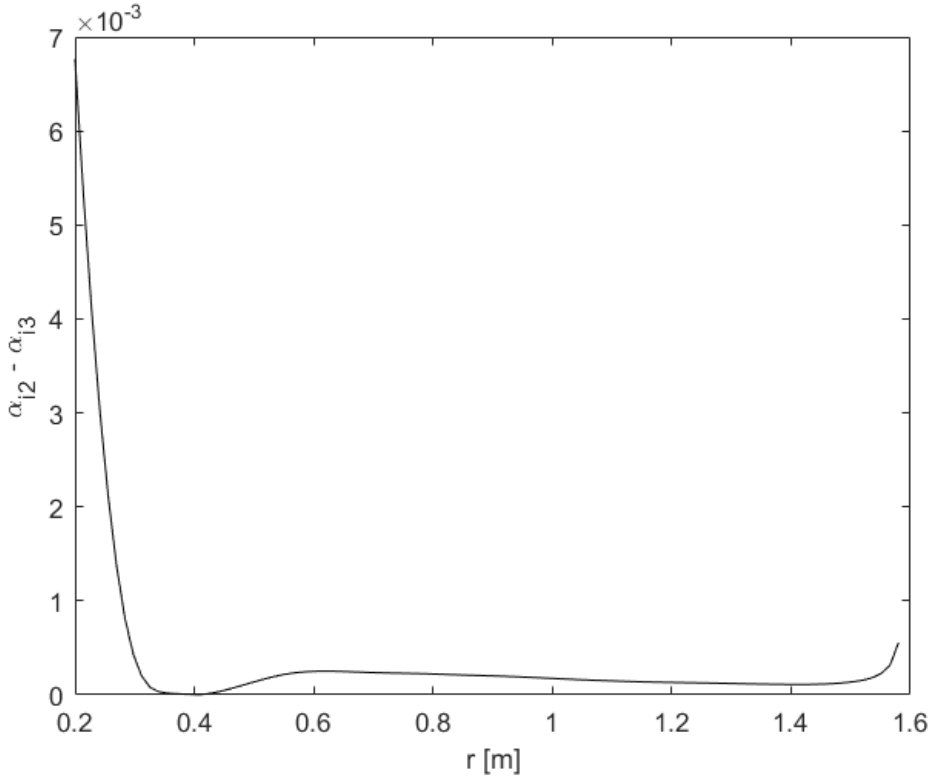

Figure 1.4: α_2-α_3 *vs* r

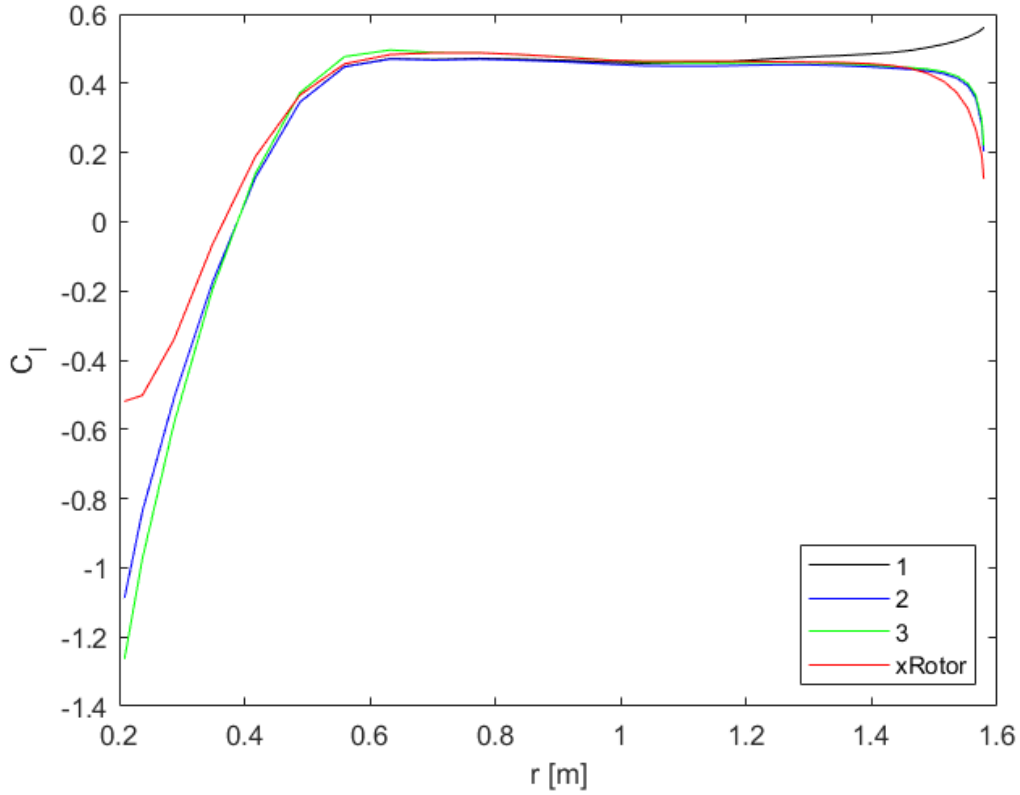

Figure 1.5: Comparison between the obtained C_l

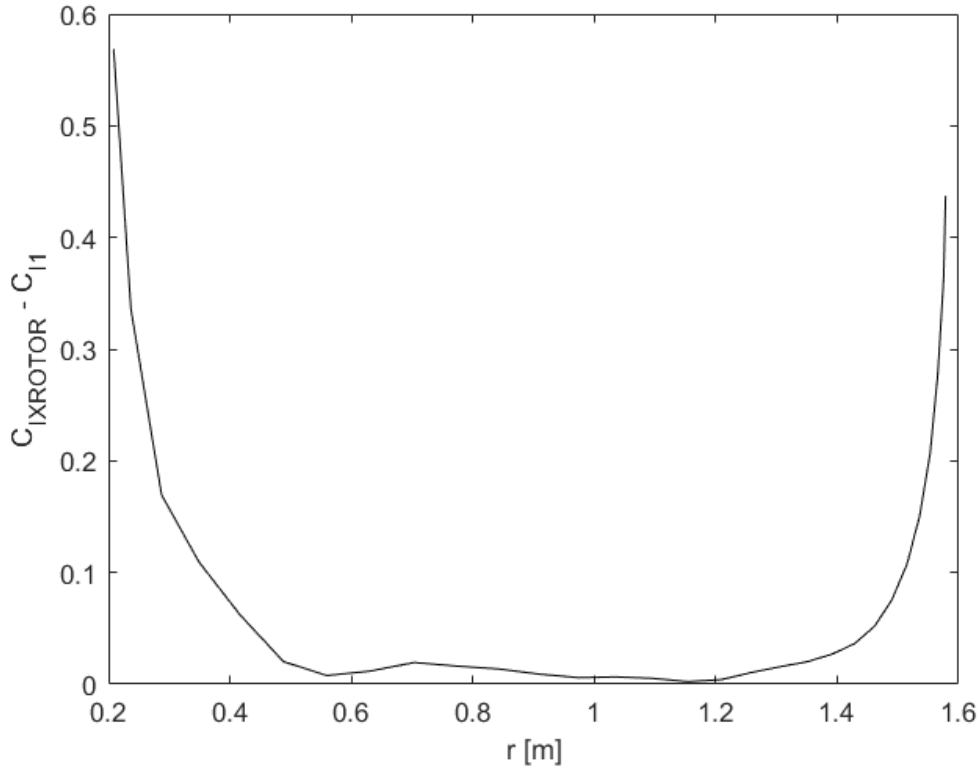

Figure 1.6: $C_{l\_XROTOR} - C_{l1}$ *vs* r

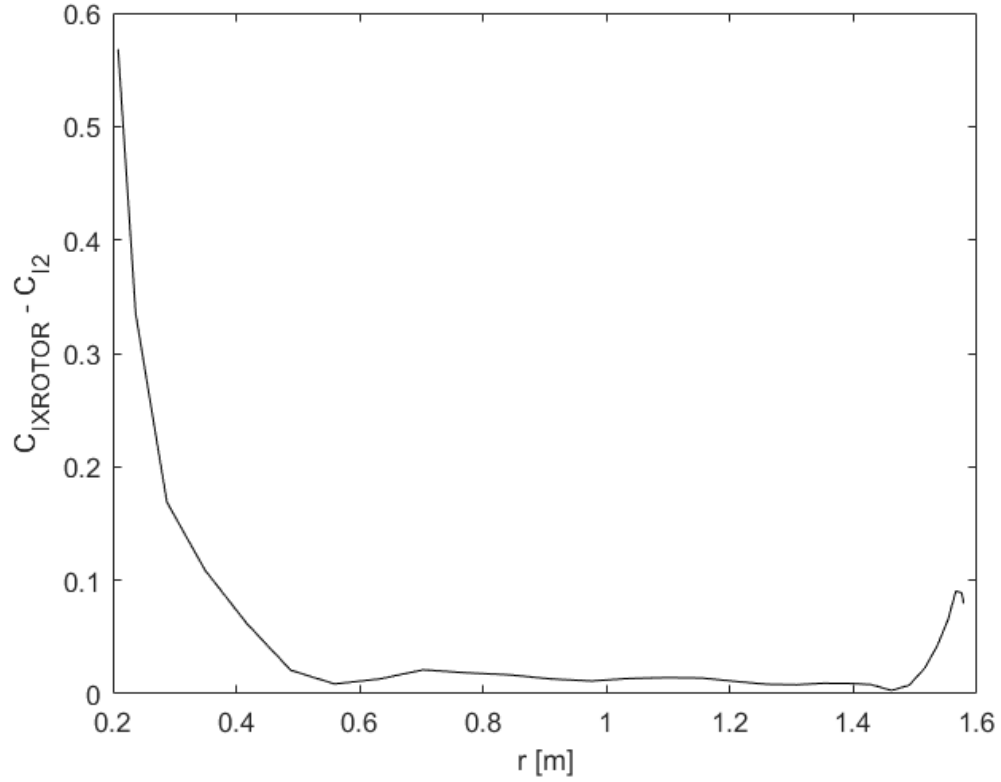

Figure 1.7: $C_{l\_XROTOR} - C_{l2}$ *vs* r

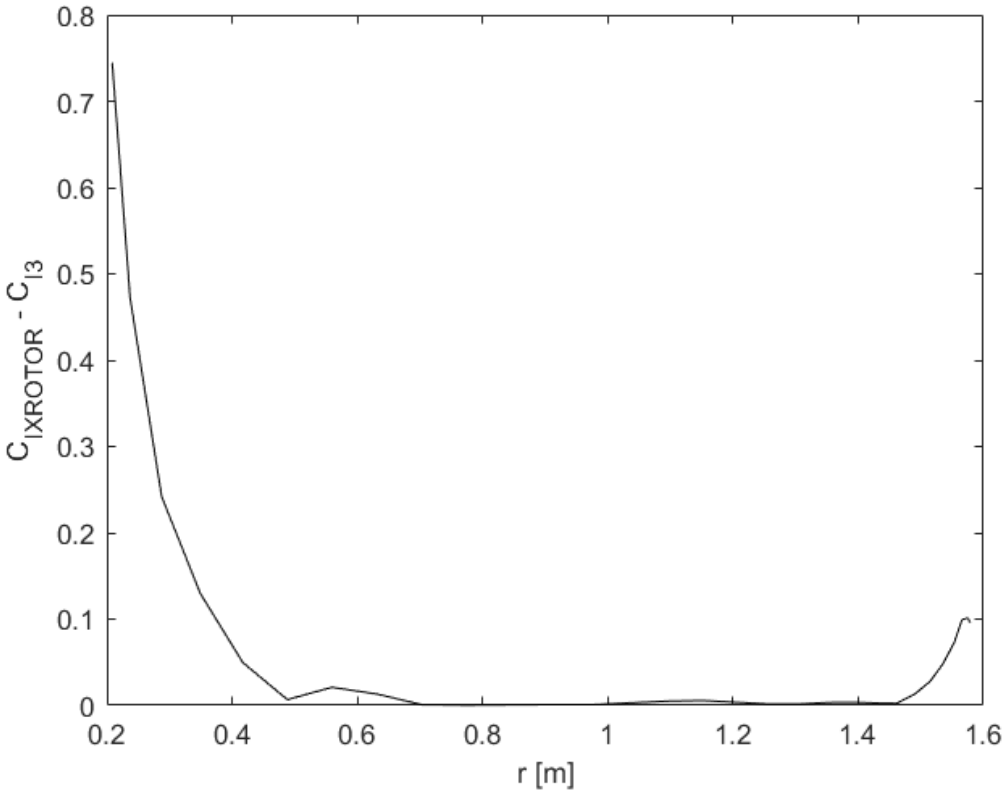

Figure 1.8: $C_{l\_XROTOR} - C_{l3}$ *vs* r

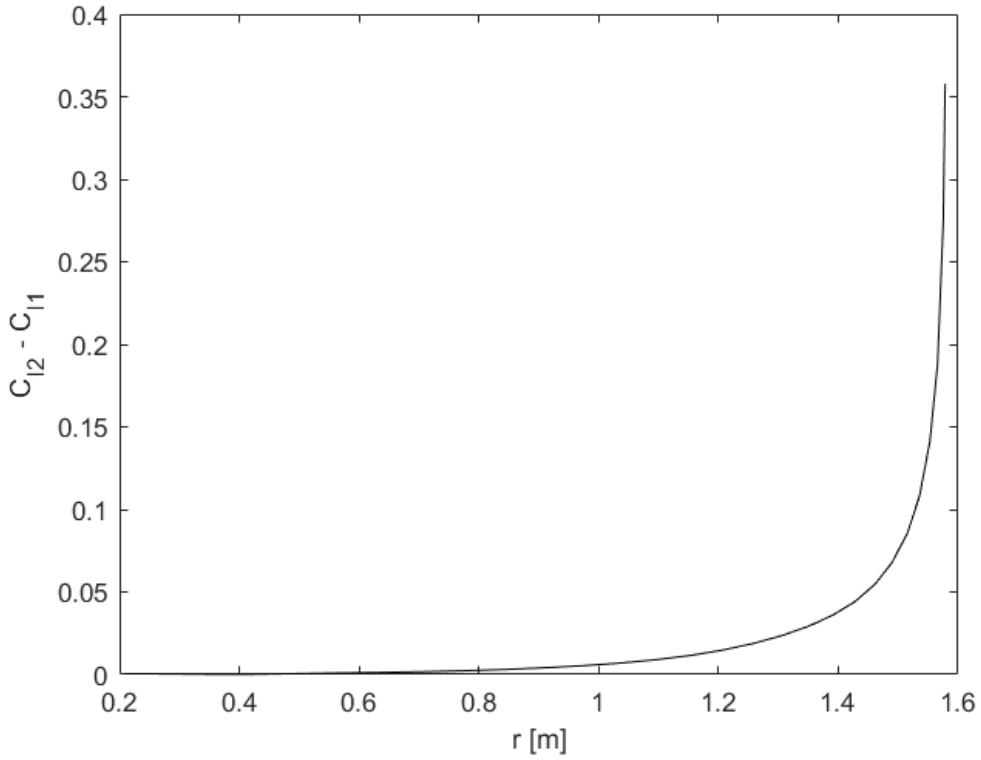

Figure 1.9: $C_{l1} – C_{l2}$ *vs* r

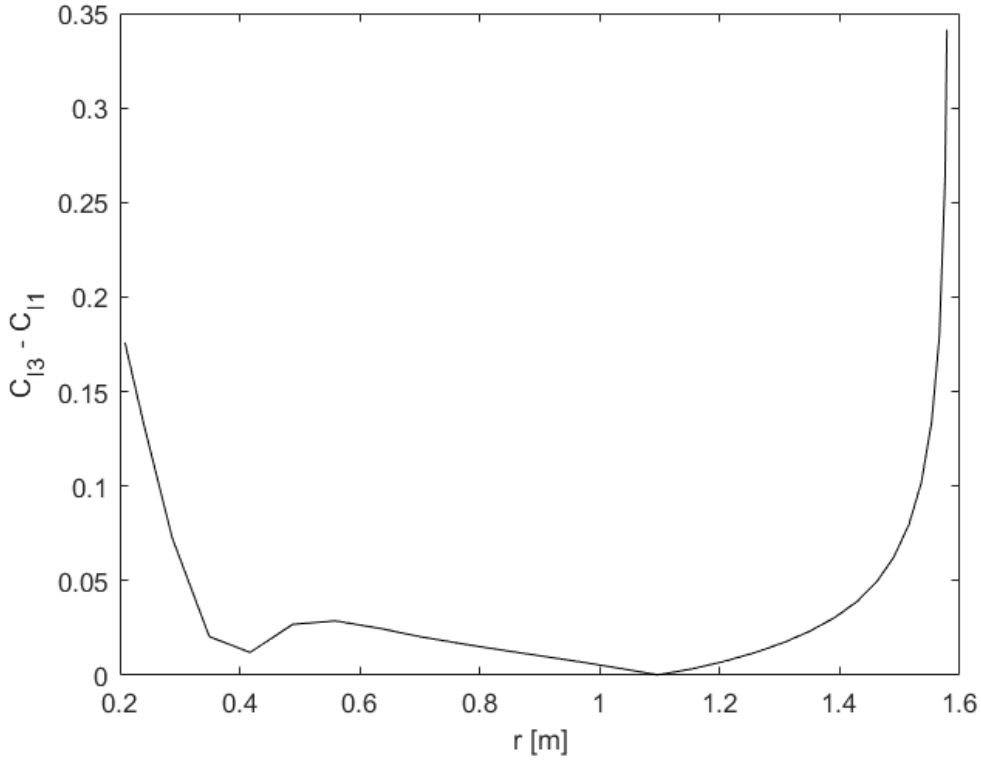

Figure 1.10: $C_{l1} – C_{l3}$ *vs* r

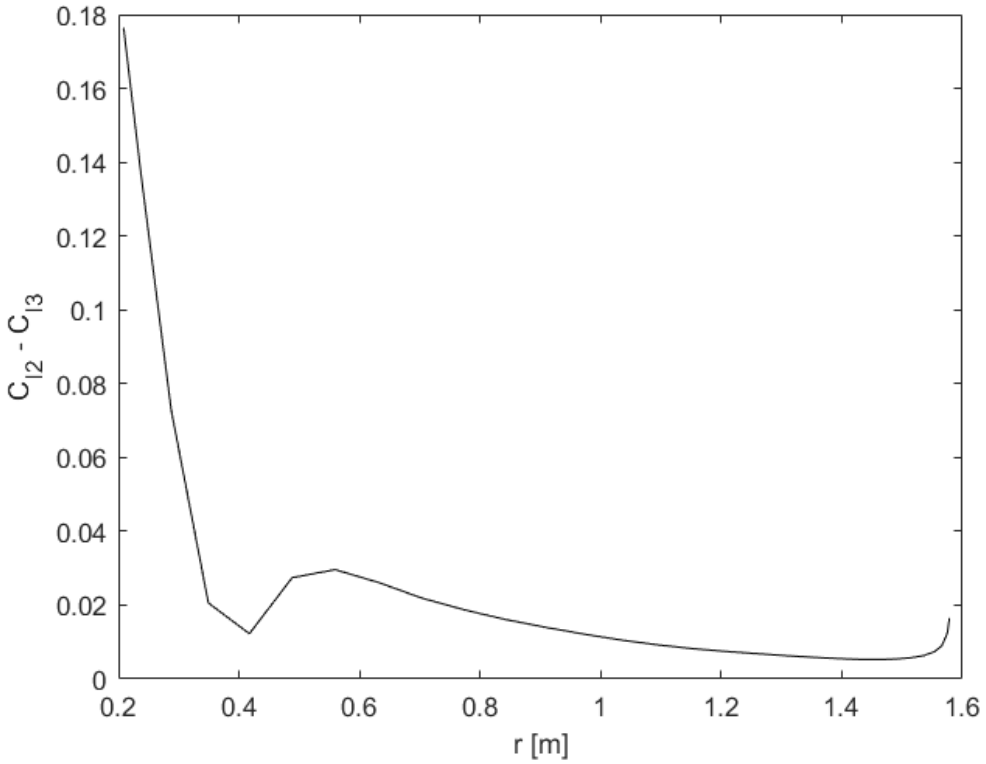

Figure 1.11: $C_{l3} - C_{l2}$ *vs* r

5. Complete script of the function

```
function [alpha_i_deg, beta_deg, c, Cl_alpha] = BEMT(V, rpm, R,
r_hub, N, ns)

p = 1;
while p < ns+1
s=input(['INSERT THE VALUE OF THE CHORD [m] AT THE STATION
',num2str(p), ' : ']);
c(p)=s;
p = p+1;
end

p = 1;
while p < ns+1
f=input(['INSERT THE VALUE OF THE Cl_ALPHA [1/rad] AT THE STATION
',num2str(p), ' : ']);
Cl_alpha(p)=f;
p = p+1;
end

p = 1;
while p < ns+1
d=input(['INSERT THE VALUE OF THE PITCH ANGLE [deg] AT THE STATION
',num2str(p), ' : ']);
```

```
beta_deg(p)=d;
p = p+1;
end

omega  = convangvel(rpm, 'rpm', 'rad/s');
r      = linspace(r_hub, R*0.99, ns);
beta   = convang(beta_deg, 'deg', 'rad');
r_ndim = r/R;
x      = linspace(r_hub,R,100);

lam    = V/(omega*R);
sigma  = N*c./(pi*R);
Vt     = omega*R;
Vr     = sqrt((V^2)+ (omega*R.*r_ndim).^2);
phi    = atan((V)./(omega*r));

disp('Which theory do you want to use?')
disp('1 - IMPULSIVE THEORY')
disp('2 - WHIRLING THEORY WITH SMALL DISTURBANCES')
disp('3 - WHIRLING THEORY')

l= input(['']);

%% METHOD N1
if l==1

alpha_i = 0.5*(-(lam./r_ndim +
sigma.*Cl_alpha.*(Vr./Vt)./(8*r_ndim.^2)) +...
          sqrt((lam./r_ndim +
sigma.*Cl_alpha.*(Vr./Vt)./(8*r_ndim.^2)).^2 +...
          sigma.*Cl_alpha.*Vr.*(beta-phi)./(2*(r_ndim.^2).*Vt)));

alpha_i_spline = interp1(r,alpha_i,x,'spline');
else

%% METHOD N2
if l==2

   if phi(ns)== 0
   F = ones (1, ns);
   else
   F =(2/pi)*acos(exp(N/(2*lam)*((r-R)/R)));
   end

alpha_i = 0.5*(-(lam./r_ndim +
sigma.*Cl_alpha./(8*r_ndim.*F.*cos(phi))) + ...
          sqrt((lam./r_ndim +
sigma.*Cl_alpha./(8*r_ndim.*F.*cos(phi))).^2 + ...
          sigma.*Cl_alpha.*(beta-phi)./(2*r_ndim.*F.*cos(phi))));

alpha_i_spline = interp1(r,alpha_i,x,'spline');
else
```

```
%% METHOD N3
if l==3
   max_iter=1000;
   warndlg('WARNING: THIS METHOD MAY BE INCORRECT...');

       if phi(ns)== 0
       F= ones (1, ns);
       else
       F=(2/pi)*acos(exp(N/(2*lam)*((r-R)/R)));
       end

alpha_i_B = 0.5*(-(lam./r_ndim +
sigma.*Cl_alpha./(8*r_ndim.*F.*cos(phi))) ...
          + sqrt((lam./r_ndim +
sigma.*Cl_alpha./(8*r_ndim.*F.*cos(phi))).^2 ...
          + sigma.*Cl_alpha.*(beta-
phi)./(2*r_ndim.*F.*cos(phi))));

w_t_B = Vr.*alpha_i_B.*sin(phi+alpha_i_B); % formula C-8
w_t_C = sigma.*Cl_alpha./(8*r.*F).*Vr.*(beta- phi-alpha_i_B);
w_a_B = Vr.*alpha_i_B.*cos(phi+alpha_i_B); % formula C-9

alfaisa = alpha_i_B;
wtsa    = w_t_B;
wasa    = w_a_B;
w_t_B   = w_t_B-(0.1*w_t_B);
value   = 10* ones(1, ns);
epsi    = 1e-10;

for j= 1:ns
    i=0;
if value(j) < epsi
   w_t_B(j)= w_t_B(j)-(y(j)/dy(j));
else

while [value(j)> epsi , i<max_iter+1]
    i= i+1;
    test1(j)= V^2+ (4*w_t_B(j).*((omega*R*r_ndim(j))- w_t_B(j)));
%radice della C-15
    test2(j)= lam^2+ (4*w_t_B(j)./(Vt)).*(r_ndim(j)-
(w_t_B(j)./(Vt))); %radice della C-17

if (test1 <0 | test2 <0)
     w_t_B(j)=wtsa(j);
     w_a_B(j)=wasa(j);
else

term1(j)= (sqrt(test1(j))-V);  %2*w_a_C dalla C-15
term2(j)= lam+ sqrt(test2(j)); %seconda radice di C-17

aa(j)= beta(j)-(atan((w_t_B(j)*2)./(term1(j)))); % termine in
parentesi nella C-14
```

```
bb(j)= sqrt((0.25*(term2(j).^2)+((r_ndim(j)-w_t_B(j)./Vt)).^2));
%tutto il termine sotto radice della C-17
cc(j)= 8*r_ndim(j).*F(j).*w_t_B(j)./Vt; % ha portato al primo
membro tutti i termini della C-17 tranne sigma,..
                      % Cl_alpha, aa, bb

da(j)= ((-2)./(((term1(j)).^2)+
4*w_t_B(j).*w_t_B(j))).*(term1(j)... %derivata di aa
     -(w_t_B(j).*(((2*omega*R*r_ndim(j))-
(4*w_t_B(j))./(term1(j)+V))));
db(j)= (((term2(j).*((r_ndim(j)./Vt)-
((2*w_t_B(j))./((Vt).^2))))...  %derivata di bb
    ./(2*(term2(j)- lam)))+ (w_t_B(j)./((Vt).^2))-
(r_ndim(j)./Vt))./bb(j);
dc(j)= 8*r_ndim(j).*F(j)./Vt;  %derivata di cc

y(j)= (sigma(j).*Cl_alpha(j).*aa(j).*bb(j))- cc(j);
dy(j)= (sigma(j).*Cl_alpha(j).*((aa(j).*db(j))+(bb(j).*da(j))))-
dc(j);

value(j)= abs(y(j)./dy(j));
w_t_B(j)= w_t_B(j)-(y(j)./dy(j));

alpha_i(j)= beta(j)-aa(j)-phi(j);

end
end
end
I(j)=i;
end

alpha_i_spline=interp1(r,alpha_i,x,'spline');

if i>max_iter
   disp('There is no value for w_t');
end
end
end
end

alpha_i_deg        = convang(alpha_i,'rad','deg');
alpha_i_deg_spline = convang(alpha_i_spline,'rad','deg');

%% Chart
figure
plot(x, alpha_i_deg_spline, 'k')
xlabel('r [m]'); ylabel('\alpha_i [deg]');
axis ([0 R min(alpha_i_deg)-1 max(alpha_i_deg)+1]);
end
```