Nasti Giuseppe	M53001106
Tomasso Armando Diego	M53001087

ELI-TAARG Documentation

Characteristics curves of Horizontal Axis Wind Turbines

January 2021

1 Algorithm Introduction

The function returns the characteristics curves of horizontal axis wind turbines: $C_T(\lambda)$, $C_Q(\lambda)$, $C_P(\lambda)$ according to the Blade Element Momentum Theory. The procedure used to obtain these curves is shown in [**3**]. For a given windmill geometry, fixing arbitrarily α, the algorithm calculates the gradients of thrust $\frac{dC_T(\bar{r})}{d\bar{r}}$, torque $\frac{dC_Q(\bar{r})}{d\bar{r}}$ and power $\frac{dC_P(\bar{r})}{d\bar{r}}$. For assigned speed ratio λ, it integrates gradients from $\bar{r} = 0$ to $\bar{r} = 1$ in order to obtain $C_T(\lambda)$, $C_Q(\lambda)$, $C_P(\lambda)$.

$$C_T = \int_0^1 \frac{dC_T(\bar{r})}{d\bar{r}} d\bar{r} \tag{1}$$

$$C_Q = \int_0^1 \frac{dC_Q(\bar{r})}{d\bar{r}} d\bar{r} \tag{2}$$

$$C_P = \int_0^1 \frac{dC_P(\bar{r})}{d\bar{r}} d\bar{r} \tag{3}$$

2 Algorithm description

The algorithm uses the equations shown in [**1**] that are similar to the ones shown in [**3**] adapting to windmill convention. In the first code line a vector of α is assigned. For every blade element and α, the algorithm calculates C_l, C_d and inflow angle ϕ.

```
v_alpha=convang(linspace(2,50,100),'deg','rad');

rs=r/R;      % adimensional radius

for i=1:length(r)

    v_input_cl(i,:)=[Cl_max(i),Cl_min(i),alpha_zero_lift(i),...
                    cl_alpha(i),dcldalfastall(i)];

    v_input_cd(i,:)=[Cd_min(i),dCd_dCl2(i),Cl_Cd_min(i),...
                    Re_ref(i),Re_inf(i),f(i),Cl_max(i),Cl_min(i)];

    for j=1:length(v_alpha)

        Cl(i,j)=lift_curve(v_input_cl(i,:),...
        convang(v_alpha(j),'rad','deg'));

        [~,~,Cd(i,j)]=ClCd_XRotor(v_input_cd(i,:), Cl(i,j));
        phi(i,j)=v_alpha(j)+beta(i);
```

On these basis it determines C_x, C_y and so a, a'. From that equations explaining λ:

$$\lambda = \frac{1-a}{1+a'} \frac{1}{tan\phi} \frac{1}{\bar{r}} \tag{4}$$

The algorithm calculates:

$$\frac{dC_T(\bar{r})}{d\bar{r}} = \frac{2\sigma C_x \bar{r}(1-a)^2}{sin^2\phi} \tag{5}$$

$$\frac{dC_Q(\bar{r})}{d\bar{r}} = \frac{2\sigma C_y \bar{r}^2 (1-a)^2}{sin^2\phi} \tag{6}$$

$$\frac{dC_P(\bar{r})}{d\bar{r}} = \frac{dC_Q(\bar{r})}{d\bar{r}}\,\lambda \tag{7}$$

```
        Cx(i,j)=Cl(i,j)*cos(phi(i,j))+Cd(i,j)*sin(phi(i,j));
        Cy(i,j)=Cl(i,j)*sin(phi(i,j))-Cd(i,j)*cos(phi(i,j));

        sigmar(i,j)=N*c(i)/(2*pi*r(i));  % solidity

        a(i,j)=(sigmar(i,j)*Cx(i,j)/(4*(sin(phi(i,j)))^2))/...
               (1+(sigmar(i,j)*Cx(i,j)/(4*(sin(phi(i,j)))^2)));

        ap(i,j)=(sigmar(i,j)*Cy(i,j)/(4*sin(phi(i,j))...
                *cos(phi(i,j))))/(1-(sigmar(i,j)*Cy(i,j)/...
                 (4*sin(phi(i,j))*cos(phi(i,j)))));

        lambda(i,j)=((1-a(i,j))/(1+ap(i,j)))...
                  *(1/tan(phi(i,j)))*(1/rs(i));

        dCtdrs(i,j)=2*sigmar(i,j)*Cx(i,j)*(r(i)/R)*(1-a(i,j))^2...
                    /((sin(phi(i,j)))^2);

        dCqdrs(i,j)=2*sigmar(i,j)*Cy(i,j)*(r(i)/R)^2*(1-a(i,j))^2...
                   /((sin(phi(i,j)))^2);

        dCpdrs(i,j)=dCqdrs(i,j)*lambda(i,j);

    end

end
```

In the second part of the code is assigned a vector of tip speed λ. From previous listing, the algorithm calculates the gradients $\frac{dC_T(\bar{r})}{d\bar{r}}$, $\frac{dC_Q(\bar{r})}{d\bar{r}}$, $\frac{dC_P(\bar{r})}{d\bar{r}}$ for every α and blade station and the corresponding tip speed λ (see eq. 4). In order to obtain gradients distributions along adimensional radius fixed λ, in the following loop the algorithm implements an interpolation of datas. In the final part of the code, known gradients distributions, the integrals 1, 2, 3 are calculated using MATLAB function *"trapz"* for every λ within the validity limits of the Momentum Theory (a<0.5).

```
v_lambda=linspace(firstlambda,lastlambda,150);

for k=1:length(v_lambda)
    LAM=v_lambda(k);
    for i=1:length(r)
        aa(i,k)=interp1(lambda(i,:),a(i,:),LAM,'pchip');
        aap(i,k)=interp1(lambda(i,:),ap(i,:),LAM,'pchip');


        DCtdrs(i,k)=interp1(lambda(i,:),dCtdrs(i,:),LAM,'pchip');
        DCqdrs(i,k)=interp1(lambda(i,:),dCqdrs(i,:),LAM,'pchip');
        DCpdrs(i,k)=interp1(lambda(i,:),dCpdrs(i,:),LAM,'pchip');

    end
    if aa(:,k)<0.5
        Ct(k)=trapz(rs,DCtdrs(:,k));
        Cq(k)=trapz(rs,DCqdrs(:,k));
        Cp(k)=Cq(k)*LAM;
    else
        break
    end
end
```

3 Input & Output

The function requires input:

- **N** : number of blades,
- **r** : vector of dimensional radius [m] of the blade element stations,
- **Aero_Matrix** : matrix of aerodynamic characteristics,
- **c** : vector of dimensional chords [m] of the blade element stations,
- **beta** : vector of the pitch angle [deg] of the blade element stations,
- **firstlambda** : first value of tip speed,
- **lastlambda** last value of tip speed.

Aero_matrix is requested as follows:

- each row represents a blade element section,
- the columns show in the following order:
  - $C_{d,min}$: minimum blade element's drag coefficient,
  - $\frac{dC_d}{dC_l^2}$: quadratic coefficient of the parable that approximates the $C_d(C_l)$ curve,
  - $C_l(C_{d,min})$: blade element's lift coefficient for which the drag coefficient assumes its minimum value,
  - Re_{ref} : blade element's Reynolds number computed taking into account the radius at which the blade element is intended to be(i.e. the reference velocity is Ωr where Ω is the windmill's angular velocity and r is the radial position of the blade element taken into account),
  - Re_∞ : asymptotic Reynolds number of the phenomena,
  - f: Reynolds number scaling exponent, according to XRotor documentation [**4**],
  - $C_{l,max}$: maximum blade element's lift coefficient,
  - $C_{l,min}$: minimum blade element's lift coefficient,
  - α_{zl}: blade element's alpha zero lift,
  - $C_{l,\alpha}$: gradient of the linear section of the lift curve,
  - $C_{l,\alpha poststall}$: gradient of the lift curve after the stall,

The function returns in output:

- λ : vector of tip speed for which it is valid the momentum theory (a<0.5),
- C_P : vector of Power Coefficient,
- C_Q vector of Torque Coefficient,
- C_T : vector of Thrust Coefficient.

4 Test Case

A test case is presented in order to validate the code.The results of the code were compared with the results produced from the software 'XRotor'[**4**]. The aerodynamic parameters of every blade section are reported in 2.

r[m]	chord[m]	β[deg]
2.05	3.00	28.5
6.20	1.61	9.19
10.8	0.88	2.27
17.3	0.54	0.01
20.5	0.47	0.00

N	3
firstlambda	2
lastlambda	15

Table 1: Function's input.[**2**]

$C_{d,min}$	$\frac{dC_d}{dC_l^2}$	$C_l(C_{d,min})$	Re_{ref}	Re_∞	f	$C_{l,max}$	$C_{l,min}$	α_{zl}	$C_{l\alpha}$	$C_{l\alpha,stall}$
0.0080	0.004	0.3	4000000	4100000	-0.2	1.5	0.8	-3	0.11	0
0.0080	0.004	0.3	4000000	4100000	-0.2	1.5	0.8	-3	0.11	0
0.0080	0.004	0.3	4000000	4100000	-0.2	1.5	0.8	-3	0.11	0
0.0080	0.004	0.3	4000000	4100000	-0.2	1.5	0.8	-3	0.11	0
0.0080	0.004	0.3	4000000	4100000	-0.2	1.5	0.8	-3	0.11	0

Table 2: Aeromatrix.

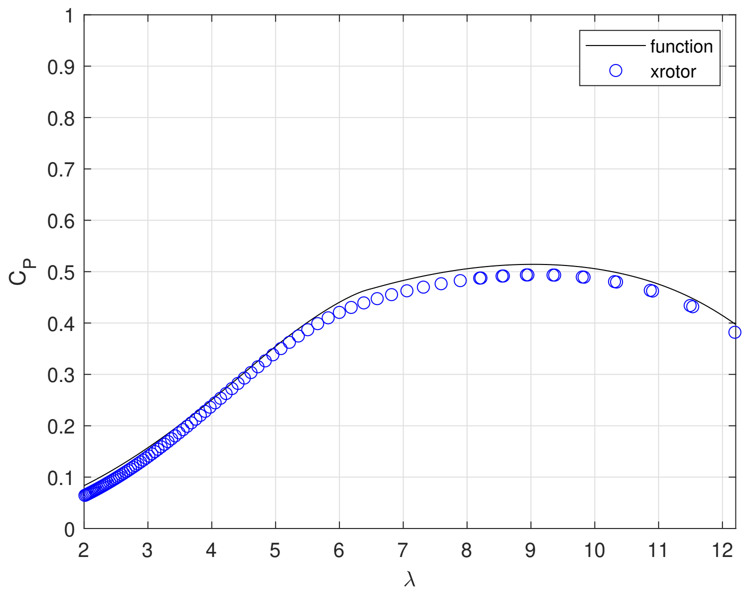

Figure 1: Power Coefficient

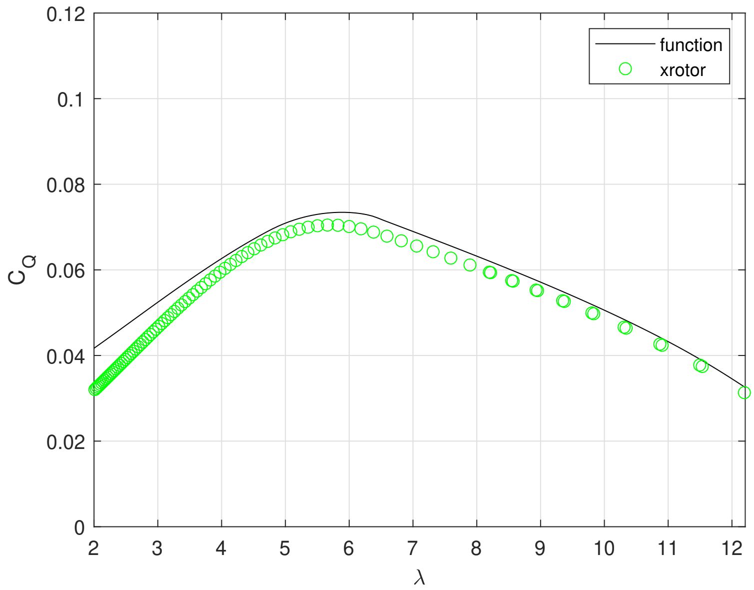

Figure 2: Torque Coefficient

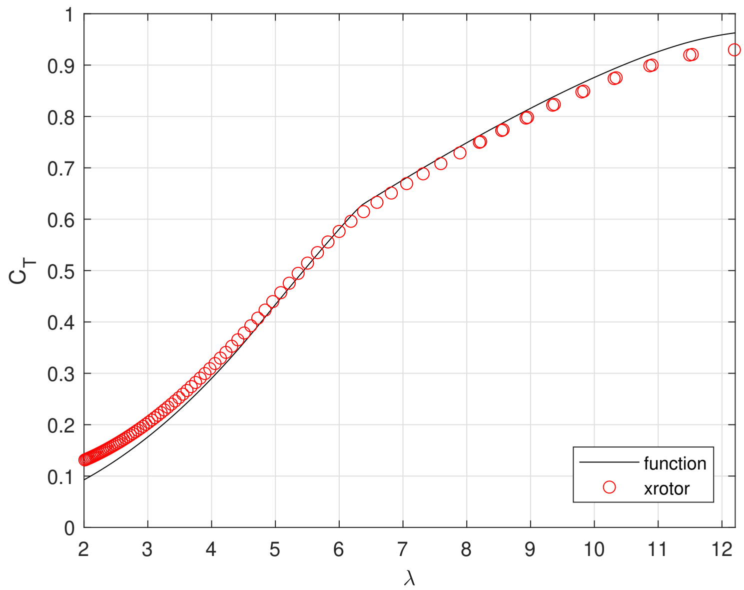

Figure 3: Thrust Coefficient.

5 Subroutine

5.1 "lift_curve"

This section introduces the "lift_curve" subroutine which allows to evaluate lift's coefficient based on the angle of attack.
The function takes the following values as input:

- $C_{l,max}$:maximum blade element's lift coefficient,
- $C_{l,min}$:minimum blade element's lift coefficient,
- α_{zl} :blade element's alpha zero lift,
- $C_{l,\alpha}$:gradient of the linear section of the lift curve,
- $C_{l,\alpha,poststall}$:gradient of the lift curve after the stall,
- ***angle*** :angle of attack.

In the first part of the subroutine the lift curve is constructed assuming:

- linear trend between $C_{l,max}$ and $C_{l,min}$ with a gradient equal to $C_{l,\alpha}$,
- linear trend after the stall with a gradient equal to $C_{l,\alpha,poststall}$.

```
% This function approximates lift curve with three segments:
% - linear part of the lift-curve
% - post stall sector with Cl>0
% - post stall sector with Cl<0

function CL=lift_curve(v_input,angle)

%Input mangament
cl_max=v_input(1);
cl_min=v_input(2);
alpha_zero_lift=v_input(3);      %deg
cl_alpha=v_input(4);             %1/deg
dcldalfastall=v_input(5);        %1/deg

v_alpha=linspace(-60,60,300);

%Ideal section of the lift curve
cl=cl_alpha*(v_alpha-alpha_zero_lift);

alphastallmax=(cl_max/cl_alpha)+alpha_zero_lift; % Ideal case
alphastallmin=(cl_min/cl_alpha)+alpha_zero_lift; %Ideal case

for i=1:length(cl)

    if cl(i)>cl_max %Correction for post stall with Cl>0

    cl(i)=cl_max+dcldalfastall*(v_alpha(i)-alphastallmax);

    elseif cl(i)<cl_min %Correction for post stall with Cl<0

    cl(i)=cl_min+dcldalfastall*(v_alpha(i)-alphastallmin);

    end

end

%Interpolation
CL=interp1(v_alpha,cl,angle,'linear');

end
```

In the final part of the subroutine the data are interpolated and the C_l is returned for the corresponding angle of attack.

```
CL=interp1(v_alpha,cl,angle,'linear');
end
```

5.2 "ClCd_XRotor"

See ELI-TAARG Function documentation.

References

[1] BURTON, T., JENKINS, N.,SHARPE, D., BOSSANYI, E.,(2011), *Wind Energy Handbook*, 2nd Edition, Wiley and Sons.

[2] Carcangiu C.E., (2008), *CFD-RANS Study of Horizontal Axis Wind Turbines.*

[3] TOGNACCINI, R., (2019), *Lezioni di Aerodinamica dell'Ala Rotante.*

[4] XRotor software user guide `http://web.mit.edu/drela/Public/web/xrotor/xrotor_doc.txt`.

A Appendix: code

```
function [Cp Cq Ct Lambda]=Characteristics_Curve_HO_Windmill(N,r,Aero_matrix,
    beta,c,firstlambda,lastlambda)

%% Input management

R=r(end); %Radius of the blade
rs=r/R; % adimensional radius station
v_alpha=convang(linspace(2,50,100),'deg','rad');
beta=convang(beta,'deg','rad'); %Blade Pitch

%Extrapolation of the aerodynamic characteristics
Cd_min    = Aero_matrix(:,1);
dCd_dCl2  = Aero_matrix(:,2);
Cl_Cd_min = Aero_matrix(:,3);
Re_ref    = Aero_matrix(:,4);
Re_inf    = Aero_matrix(:,5);
f         = Aero_matrix(:,6);
Cl_max    = Aero_matrix(:,7);
Cl_min    = Aero_matrix(:,8);
alpha_zero_lift=Aero_matrix(:,9); %      [deg]
cl_alpha=Aero_matrix(:,10); %            [1/deg]
dcldalfastall=Aero_matrix(:,11); %       [1/deg]

%% Gradient coefficents calculation

for i=1:length(r)

    %Creation of input vectors for the two subroutine lift_curve and
    ClCd_XRotor
    v_input_cl(i,:)=[Cl_max(i),Cl_min(i),alpha_zero_lift(i),cl_alpha(i),
    dcldalfastall(i)];
    v_input_cd(i,:)=[Cd_min(i),dCd_dCl2(i),Cl_Cd_min(i),Re_ref(i),Re_inf(i),f
    (i),Cl_max(i),Cl_min(i)];

    for j=1:length(v_alpha)

        %Calculation of Cl and Cd for the blade station at angle v_alpha(i)
        Cl(i,j)=lift_curve(v_input_cl(i,:),convang(v_alpha(j),'rad','deg'));
        [~,~,Cd(i,j)]=ClCd_XRotor(v_input_cd(i,:), Cl(i,j));

        phi(i,j)=v_alpha(j)+beta(i); %inflow angle

        Cx(i,j)=Cl(i,j)*cos(phi(i,j))+Cd(i,j)*sin(phi(i,j));
        Cy(i,j)=Cl(i,j)*sin(phi(i,j))-Cd(i,j)*cos(phi(i,j));

        sigmar(i,j)=N*c(i)/(2*pi*r(i)); %blade solidity

        a(i,j)=(sigmar(i,j)*Cx(i,j)/(4*(sin(phi(i,j)))^2))/(1+(sigmar(i,j)*Cx
    (i,j)/(4*(sin(phi(i,j)))^2)));
        ap(i,j)=(sigmar(i,j)*Cy(i,j)/(4*sin(phi(i,j))*cos(phi(i,j))))/(1-(
    sigmar(i,j)*Cy(i,j)/(4*sin(phi(i,j))*cos(phi(i,j)))));

        lambda(i,j)=((1-a(i,j))/(1+ap(i,j)))*(1/tan(phi(i,j)))*(1/rs(i));

        dCtdrs(i,j)=2*sigmar(i,j)*Cx(i,j)*(r(i)/R)*(1-a(i,j))^2/((sin(phi(i,j
    )))^2);
        dCqdrs(i,j)=2*sigmar(i,j)*Cy(i,j)*(r(i)/R)^2*(1-a(i,j))^2/((sin(phi(i
    ,j)))^2);
        dCpdrs(i,j)=dCqdrs(i,j)*lambda(i,j);

    end

end
%% Calculation of the characteristics curve
Lambda=linspace(firstlambda,lastlambda,150);

for k=1:length(Lambda)
    LAM=Lambda(k);
```

```
    for i=1:length(r)

        %Interpolation
        aa(i,k)=interp1(lambda(i,:),a(i,:),LAM,'pchip');
        aap(i,k)=interp1(lambda(i,:),ap(i,:),LAM,'pchip');

        DCtdrs(i,k)=interp1(lambda(i,:),dCtdrs(i,:),LAM,'pchip');
        DCqdrs(i,k)=interp1(lambda(i,:),dCqdrs(i,:),LAM,'pchip');
        DCpdrs(i,k)=interp1(lambda(i,:),dCpdrs(i,:),LAM,'pchip');

    end

    if aa(:,k)<0.5 %Condition for Momentum theory validation

        %Coefficient claculation
        Ct(k)=trapz(rs,DCtdrs(:,k));
        Cq(k)=trapz(rs,DCqdrs(:,k));
        Cp(k)=Cq(k)*LAM;
    else
        break %exit of the cycle for invalidation of the Momentum Theory (a
    >0.5)
    end

end

%Lambda selection in which momentum theory is valid
Lambda=Lambda(1:length(Cp));

%% Plot section



 end
```