Nasti Giuseppe	M53001106
Tomasso Armando Diego	M53001087

ELI-TAARG Documentation

Characteristics curves of Horizontal Axis Wind Turbines

January 2021

Chapter 1

Documentation

1.1 Algorithm Introduction

The function returns the characteristics curves of horizontal axis wind turbines: $C_T(\lambda)$, $C_Q(\lambda)$, $C_P(\lambda)$ according to Blade Element Momentum Theory. The procedure used to obtain these curves is shown in [2]. 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 $C_T(\lambda)$, $C_Q(\lambda)$, $C_P(\lambda)$.

$$C_T = \int_0^1 \frac{dC_T(\bar{r})}{d\bar{r}} d\bar{r} \tag{1.1}$$

$$C_Q = \int_0^1 \frac{dC_Q(\bar{r})}{d\bar{r}} d\bar{r} \tag{1.2}$$

$$C_P = \int_0^1 \frac{dC_P(\bar{r})}{d\bar{r}} d\bar{r} \tag{1.3}$$

The algorithm simplifies the aerodynamics of the blade elements assuming $C_{l\alpha} = 2\pi$ and $C_d = 0.01$

1.2 Algorithm description

The algorithm uses the equations shown in [1] that are similar than ones shown in [2] 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 and inflow angle ϕ. On these basis it determines C_x, C_y and so a, a'. From that equations explaining λ:

$$\lambda = \frac{1-a}{1+a'} \frac{1}{tan\phi} \frac{1}{\bar{r}} \tag{1.4}$$

The algorithm calculates:

$$\frac{dC_T(\bar{r})}{d\bar{r}} = \frac{2\sigma C_x \bar{r}(1-a)^2}{sin^2\phi} \tag{1.5}$$

$$\frac{dC_Q(\bar{r})}{d\bar{r}} = \frac{2\sigma C_y \bar{r}^2 (1-a)^2}{sin^2\phi} \tag{1.6}$$

$$\frac{dC_P(\bar{r})}{d\bar{r}} = \frac{dC_Q(\bar{r})}{d\bar{r}}\lambda \tag{1.7}$$

```
v_alpha=convang(linspace(0.5,60,100),'deg','rad');

rs=r/R;        % adimensional radius

for i=1:length(r)
    for j=1:length(v_alpha)
        Cl(i,j)=Cl_alpha*v_alpha(j);
        phi(i,j)=v_alpha(j)+beta(i);

        Cx(i,j)=Cl(i,j)*cos(phi(i,j))+cd*sin(phi(i,j));
        Cy(i,j)=Cl(i,j)*sin(phi(i,j))-cd*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. 1.4). 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 1.1, 1.2, 1.3 are calculated using MATLAB function *"trapz"* for every λ.

Da scrivere la parte sullo STOP.

```
v_lambda=linspace(lambdainiziale,lambdafinale,150);
STOP=ones(length(r),length(v_lambda));

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');

        if aa(i,k)>0.5
            STOP(i,k)=k;
        else
            STOP(i,k)=length(v_lambda);
        end

        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

    CT(k)=trapz(rs,DCtdrs(:,k));
    CQ(k)=trapz(rs,DCqdrs(:,k));
    CP(k)=CQ(k)*LAM;

end
```

1.3 Input & Output

1.4 Test Case

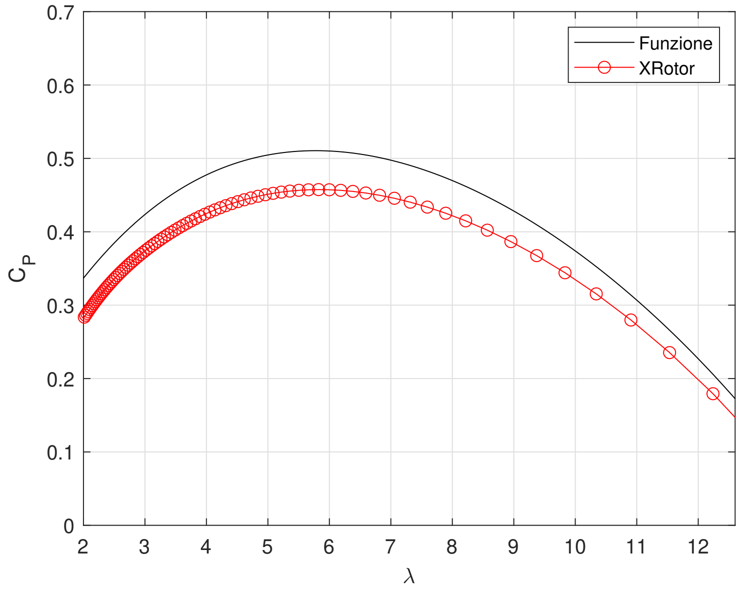

Figure 1.1

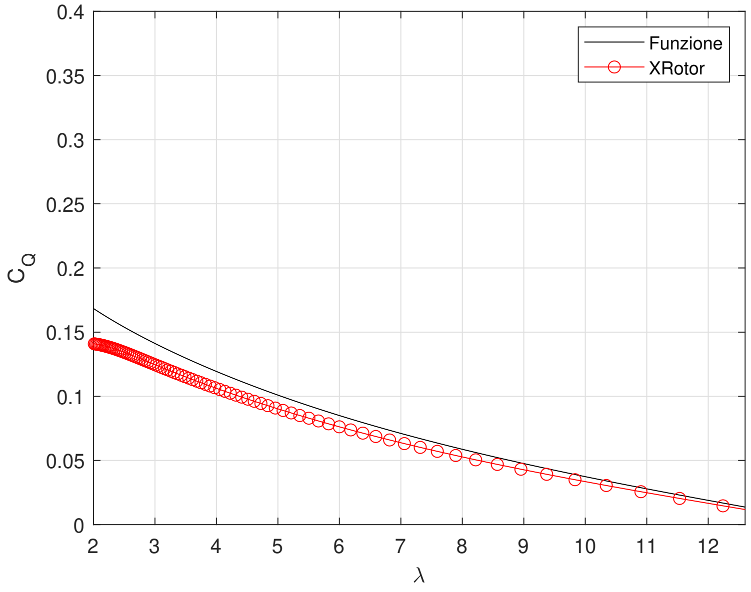

Figure 1.2

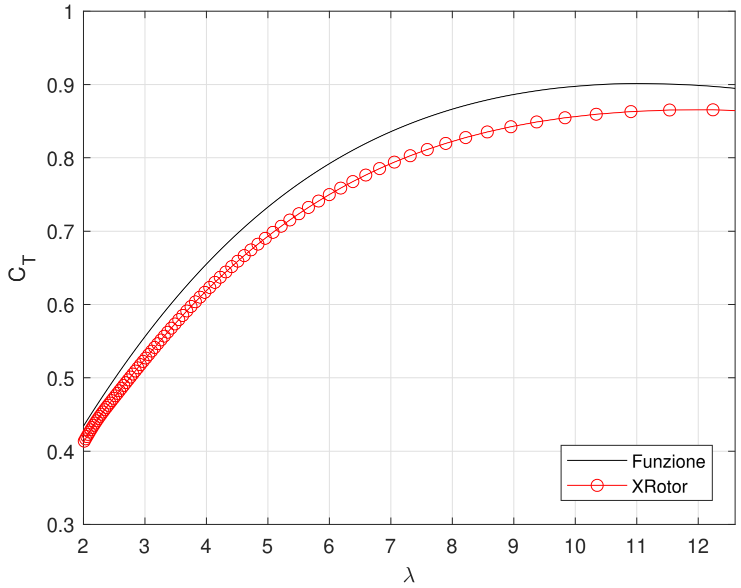

Figure 1.3

Bibliography

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

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