DOCUMENTATION OF TURBINE DARRIEUS TUBO FLUSSO MULTIPLO FUNCTION

Luciano Roncioni - Camilla Scotto di Carlo

December 2020

Contents

1 Theory

This function implements the multiple streamtubes theory which gives a more accurate prediction of the wind velocity variations across the Darrieus rotor with respect to the single streamtube model.
The multiple streamtubes theory is derived as a generalization of the single stremtube one. The induction a is considered variable with the radius,

$$a = a(r) \tag{1}$$

where

$$r = R\, sin(\phi) \tag{2}$$

The induction comes from the equations of drag force from the differential momentum theory,

$$dD_R = 2\,\rho\, V_\infty^2\,(1-a)\,a\,R\,cos(\phi)\,d\phi \tag{3}$$

end the drag force from the blade element theory.

$$dD_R = \frac{1}{2}\,\rho\, V^2\, c\, Cl\, cos(\phi+\alpha)\,d\phi \tag{4}$$

Matching equations (3) and (4), as in eq. (5), and substituting the working velocity of the airfoil, eq. (6), and the angle of attack expression, eq. (7), the induction is obtained.

$$2\,\rho\, V_\infty^2\,(1-a)\,a\,R\,cos(\phi)\,d\phi = \frac{1}{2}\,\rho\, V^2\, c\, Cl\, cos(\phi+\alpha)\,d\phi \tag{5}$$

$$\frac{V}{V_\infty} = \sqrt{[\lambda +\ (1-a)\, sin(\phi)]^2 + (1-a)^2\, cos(\phi)^2} \tag{6}$$

$$\alpha = arctan\frac{(1-a)\,cos(\phi)}{\lambda +\ (1-a)\,sin(\phi)} \tag{7}$$

Moreover, the C_P and the C_Q coefficients are derived by integrating the forces acting on the blade element during the rotation, as in single streamtube theory.

$$C_P = \frac{N\,c\,\lambda}{4\,\pi\,R}\int_0^{2\pi}\left(\frac{V}{V_\infty}\right)^2 Cl\, sin(\alpha)\,\left(1-\frac{Cd}{Cl}\,cot(\alpha)\right)\,d\phi \tag{8}$$

$$C_Q = \frac{C_P}{\lambda} \tag{9}$$

The characteristic $\lambda - C_P$ curves obtained through the proposed method is not valid for any λ. It is possible to define a λ_{min} when $\alpha = \alpha_{max}$, at the stall of the airfoil and a λ_{max} by imposing that the turbine must provide power.

2 Input and Output

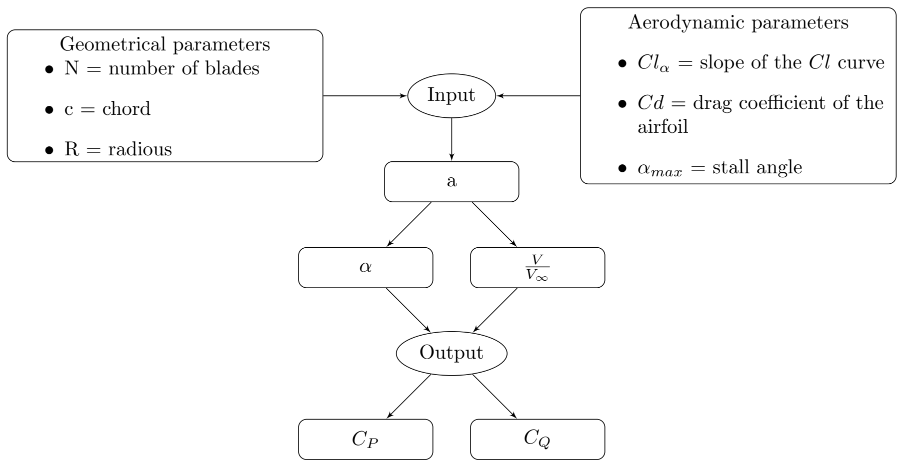

Figure 1 Flow chart of the function code.

3 Algorithm Description

The purpose of this section is to explain the code in every details.
Firstly, the function accept in input the following parameters:

- $\alpha_{\max}$, that is the angle of stall of the profile.
- c, that is the chord of the profile.
- R, that is the radius of the rotor.
- c_{l_α}, that is the slope of the Cl-α curve of the profile.
- N, that is the number of blades.
- Cd, that is the drag coefficient of the profile.

At the beginning of the function, the vector of the phi and lambda domains are created and then all the needed variables v/vinf, α, a, C_P and C_Q are initialized. After that, we enter in a while cycle where it is defined a value of lambda and only the positive values of C_P are considered because for negative C_P the rotor will not provide energy but it'll need energy.

```
while cp >= 0
    j = j +1;
    lambda = lambdav(j);
```

Inside this cycle, we enter a for cycle in which for every value of ϕ the velocity induction is evaluated thanks to the matlab function *Fzero* since the equation that has to be solved is not linear.

```
    for i = 1 : numel(phiv)
        phi = phiv(i);
        %anonymous function
        eq = @(a) ((1-a).*a) - (c/(4*R))* ...
            ( (lambda + (1-a).*sin(phi)).^2+...
            (1-a).^2.* cos(phi).^2).*cla.*...
            atan2(((1-a).*cos(phi)),(lambda + ...
            (1-a).*sin(phi))).*(cos(phi+(atan2 ...
            (((1-a).*cos(phi)),(lambda +...
            (1-a).*sin(phi)))))./cos(phi)));
        %find zero of the previous function
        a(1,i) = (fzero(eq,0.2));
```

Then v/vinf and α can be evaluated, and in particular, following the rows, these parameters change with ϕ, instead, following the column, they change with λ.

```
        v_vinf(j,:) = sqrt((lambda + (1-a).*...
            sin(phiv)).^2 + (1-a).^2 .*cos(phiv).^2);
        alpha(j,:) = atan2(((1-a).*cos(phiv)),...
            (lambda + (1-a).*sin(phiv)));
```

Once evaluated these parameters, C_P and C_Q can be evaluated with the matlab function *trapz* because to obtain both of them, integrals have to be made.

```
        cost_p = (N*c*lambda)./(4*pi*R);
        % computing Cp - numerical integration
        cp = cost_p.* trapz(phiv, (...
            v_vinf(j,:).^2*cla.*alpha(j,:).*...
            sin(alpha(j,:)).*(1 -(cd./...
            (cla.*alpha(j,:))).*cot(alpha(j,:)))));
        cq = cp/lambda; % Cq value

        % Allocating values in corresponding vectors
        cpv(j,1) = cp;
        cqv(j,1) = cq;
```

At the end of the code, there is a control on the minimum λ because the alpha of the blade elements have always to be smaller than the $\alpha_{\max}$, otherwise the wing will not work properly and will not generate any power.

```
max_v = zeros(numel(lambdav),1);
```

```
alphadeg = rad2deg(alpha); % [deg]

%Find max alpha for each row of the matrix alpha
for h = 1 : numel(phiv)
    max_v(h) = max(alphadeg(h,:));
end

%intial value for lambda min index
contatorelambdamin = 0;

%check on stall angle
for f = 1 : numel(phiv)
    if alphamax < max_v(f)
        contatorelambdamin = contatorelambdamin + 1;
    end
end
```

4 Error markers

For values of σ that are greater than 0.25, there can be possibilities of error when λ becames much larger than one.
For this reason, in the code there is a control on the induction vector that allows the function to exit the cycle when one NaN appears.
So, from the first value of λ characterized by this behavior, the C_P and C_Q are not evaluated.
In the end, this behaviour appears when lambda is really great and so for values of C_P that are close to zero.
But when a rotor is designed, the idea is to size it in order to work where C_P is close to the $C_{P,max}$, so the error of the function occurs far away from the design point.

```
    if sum(isnan(ind(j,:))) >= 1
        disp('Warning: from this lambda on,');
        disp('the results are not reliable,');
        disp('so the results are not reported');
        cp = -1;
        j = j-1;
    else
```

5 Test Case

In order to validate the function, some test cases have been run. The obtained results have been compared with the values of De Vries,[2], which refers to the single streamtube theory. The table 1 shows the whole parameters taken into account to carry out the validation procedure.

test	σ	α_{max}	c	R	Cl_α	N	Cd
1	0.1	14°	1	30	6.28	3	0.01
2	0.1	14°	1	30	6.28	3	0
3	0.2	14°	1	15	6.28	3	0.01
4	0.2	14°	1	15	6.28	3	0

Table 1 Test case values.

5.1 Test 1

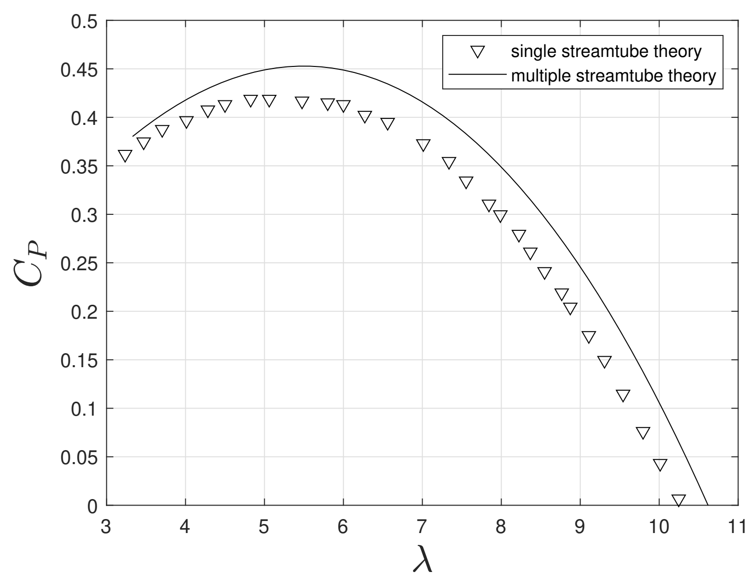

Figure 2 Test 1. Values compared with single streamtube theory from De Vries, [2].

5.2 Test 2

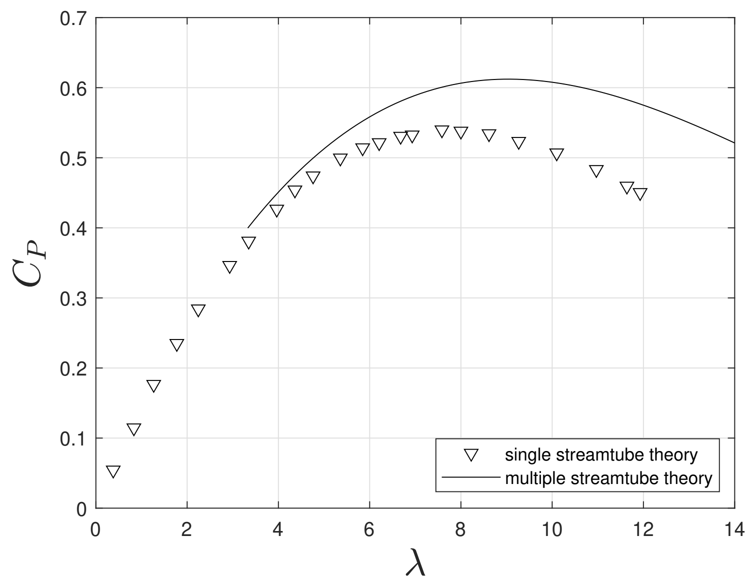

Figure 3 Test 2. Values compared with single streamtube theory from De Vries, [2].

5.3 Test 3

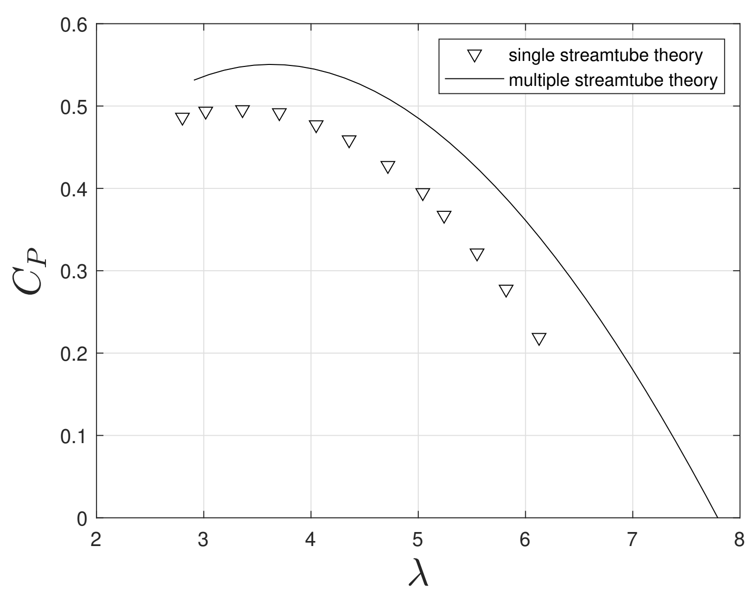

Figure 4 Test 3. Values compared with single streamtube theory from De Vries, [2].

5.4 Test 4

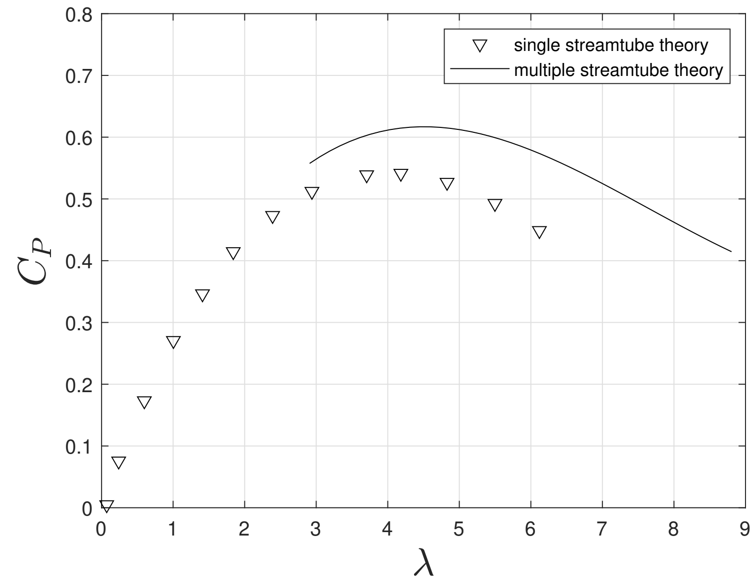

Figure 5 Test 4. Values compared with single streamtube theory from De Vries, [2].

6 Appendix

Listing 1

```

function [cp,cq,lambdav] =...
    turbine_Darrieus_tubo_flusso_multiplo...
    (alphamax,c,R,cla,N,cd)
%% Initialization of vectors used in the code
phiv = linspace(0,360,100); %phi domain [deg]
phiv = deg2rad(phiv); % [rad]
lambdav = linspace(0.1,14,100); %tip speed
cpv = zeros(numel(lambdav),1); %cp vector
cqv = zeros(numel(lambdav),1); %cq vector
a = zeros(1,numel(phiv)); %induction
v_vinf = zeros(numel(phiv),numel(phiv)); %v/v_inf
alpha = zeros(numel(phiv),numel(phiv)); %alpha
ind = zeros(numel(phiv),numel(phiv)); %inductions vector
%% Values for entering the loop
cp = 0;
j = 0;
%% a, CP, CQ computing
```

```
while cp >= 0
    j = j +1;
    lambda = lambdav(j);
    for i = 1 : numel(phiv)
        phi = phiv(i);
        %anonymous function
        eq = @(a) ((1-a).*a) - (c/(4*R))* ...
            ( (lambda + (1-a).*sin(phi)).^2+...
            (1-a).^2.* cos(phi).^2).*cla.*...
            atan2(((1-a).*cos(phi)),(lambda + ...
            (1-a).*sin(phi))).*(cos(phi+(atan2...
            (((1-a).*cos(phi)),(lambda +...
            (1-a).*sin(phi))))./cos(phi)));
        %find zero of the previous function
        a(1,i) = (fzero(eq,0.2));
    end

    ind(j,:) = a;
    % Check on the reliability of the computed induction
    % When ind is NaN, the cycle is left
    if sum(isnan(ind(j,:))) >= 1
        disp('Warning: from this lambda on,');
        disp('the results are not reliable,');
        disp('so the results are not reported');
        cp = -1;
        j = j-1;
    else

        v_vinf(j,:) = sqrt((lambda + (1-a).*...
            sin(phiv)).^2 + (1-a).^2 .*cos(phiv).^2);
        alpha(j,:) = atan2(((1-a).*cos(phiv)),...
            (lambda + (1-a).*sin(phiv)));
        cost_p = (N*c*lambda)./(4*pi*R);
        % computing Cp - numerical integration
        cp = cost_p.* trapz(phiv, (...
            v_vinf(j,:).^2*cla.*alpha(j,:).*...
            sin(alpha(j,:)).*(1 -(cd./...
            (cla.*alpha(j,:))).*cot(alpha(j,:)))));
        cq = cp/lambda; % Cq value

        % Allocating values in corresponding vectors
        cpv(j,1) = cp;
        cqv(j,1) = cq;

        %Condition to exit the cycle when lambda
        % is the last value of the
```

```
        %vector
        if lambda == lambdav(end)
            cp = -1;
        end
    end
end
%Allocating vector of max values
max_v = zeros(numel(lambdav),1);
alphadeg = rad2deg(alpha); % [deg]

%Find max alpha for each row of the matrix alpha
for h = 1 : numel(phiv)
    max_v(h) = max(alphadeg(h,:));
end

%intial value for lambda min index
contatorelambdamin = 0;

%check on stall angle
for f = 1 : numel(phiv)
    if alphamax < max_v(f)
        contatorelambdamin = contatorelambdamin + 1;
    end
end

%cp, cq and lambdav are downsized according to conditions
% of stall and positive power value
cq = cqv(contatorelambdamin:j,1);
cp = cpv(contatorelambdamin:j,1);
lambdav = lambdav(1,contatorelambdamin:j);
end
```

7 Bibliography

References

[1] Tognaccini R., (2019), "Lezioni di Aerodinamica dell’ala rotante".

[2] . De Vries O., (1979), "Fluid Dynamic Aspects of Wind Energy Conversion", AGARD-AG-243