DOCUMENTATION OF *PresRot.m* FUNCTION

Giampietro Cinque M53001141
Verdiana Bottino M53001056

April 2021

Table of contents

1. Introduction

The function *RotPerf.m* implements the blade element theory to evaluate the thrust and torque coefficients, given the main geometric and aerodynamic parameters as inputs. For every blade element, both the lift and drag coefficient vectors are derived using the *XfoilParser.m* function, which invokes the *CdCl_xfoil.m* function, already contained in the Eli-TAARG repository. *XfoilParser.m* has been introduced to overcome Xfoil convergence problems: it performs an interpolation over an interval centred around the analysed angle of attack, so that the relative lift coefficient is always accurate.

Syntax

```
[ Tc, Qc ] = RotPerf( Rhub, Rtip, thetaHubdegree,
deltaThetadegree, omega, chord, N, Vc, rho, airfoil, varargin )
```

2. I/O

Inputs and outputs are listed in the two following subsections.

2.1 Inputs

- R_h [m]: hub radius;
- R_T [m]: tip radius;
- thetaH [°]: collective pitch;
- deltaTheta [°]: rotor geometric twist;
- omega [rad/s]: angular velocity *(Ω)*;
- chord [m]: geometric airfoil chord *(c)*;
- N: number of rotor blades;
- V_C [m/s]: rate of climb;
- Rho [kg/m^3]: air density *(ρ)*;
- Airfoil: airfoil “filename” which must be placed in the running directory.

2.2 Outputs

- T_C: thrust coefficient;
- Q_C: torque coefficient.

3. Code description

First the input angles are converted from degrees to radians and the number of sections N_S in which the rotor must be divided is set. Then, $C_{l\alpha}$ and *viscosity* are set as constant values, and μ is computed through the following expression: $\mu = \frac{V_C}{\Omega R_T}$

```
function [ Tc, Qc ] = RotPerf( Rh, Rt, thetaHd, deltaThetad, omega,
chord, N, Vc, rho, airfoil, varargin )

%-----------------------------------------------------------------------
% Conversion to rad
%-----------------------------------------------------------------------

thetaH = thetaHd / 180 * pi;
deltaTheta = deltaThetad / 180 * pi;

%-----------------------------------------------------------------------
% VARIABLES
%-----------------------------------------------------------------------
Ns = 50;                                                 % Number of stations
thetaVet = linspace( thetaH, thetaH - deltaTheta, Ns );
rVet = linspace( Rh / Rt, 1, Ns );
dr = rVet( 2 ) - rVet( 1 );
mu = Vc / ( omega * Rt );
Cla = 2 * pi;
viscosity = 1.78e-5;
```

Once the *blade solidity* σ is computed, all the vectorial variables are allocated and the air density is checked, a cycle begins and lambda vector is assembled, solving a second order equation whose coefficients are *a*, *b* and *c* (only the positive root must be accepted!).

```
%-----------------------------------------------------------------------
% VARIABLE INPUT
%-----------------------------------------------------------------------
n = length( varargin );

if isempty( varargin )
    cVet = chord * ones( 1, Ns );
else

end
%-----------------------------------------------------------------------
% ALLOCATE VARIABLES
%-----------------------------------------------------------------------
sigmaVet = N * cVet / ( pi * Rt );
lambdaVet=zeros(1,Ns);
phiVet=zeros(1,Ns);
alphaVet=zeros(1,Ns);
ReVet=zeros(1,Ns);
ClVet=zeros(1,Ns);
CdVet=zeros(1,Ns);
dTdr=zeros(1,Ns);
dQdr=zeros(1,Ns);

%-----------------------------------------------------------------------
% Checks
%-----------------------------------------------------------------------
if ( not( exist( 'rho', 'var' ) ) )
    rho = 1.225;
end
%-----------------------------------------------------------------------
```

```
% Calculate Tc & Qc
%-----------------------------------------------------------------------

for i = 1 : Ns

    theta = thetaVet( i );
    r = rVet( i );
    chordi = cVet( i );
    sigma = sigmaVet( i );

    a = 1;
    b = mu + Cla *(sigma/8);
    c = -(r * Cla *(sigma/8)*(theta-(mu/r)));

    % Assemble lambda vector, accept only the positive root
    lambdaVet(i) = max( roots( [ a b c ] ) );
```

The inflow angle, the angle of attack, the effective velocity V_{eff} and the Reynolds number *Re* are computed as follows:

$$\varphi = \frac{\lambda_i}{\bar{r}} \qquad \alpha = \theta - \varphi \qquad V_{eff} \approx \frac{\Omega_r}{\cos\varphi} \qquad R_e = \frac{\rho V_{eff}\, c}{\mu}$$

where this time μ indicates the air viscosity.

```
phiVet(i) = lambdaVet(i)/ rVet(i);      % inflow angle
alpha = theta - phiVet(i);              % effective aoa
alphaVet(i) = alpha;
alphad = alpha * 180 / pi;

% Effective velocity computation through the blade element theory
Ve = ( omega * r ) / ( cosd( phiVet( i ) ) );
% Reynolds number computation
Re = round( ( rho * Ve * chordi ) / ( viscosity ) );
ReVet( i ) = Re;


% Print to terminal
disp( "Section n°" + num2str( i ) +"/" + num2str(Ns) +"..." );
disp( "alfa = " + num2str( alphad ) );
disp( "Re = " + num2str( Re ) );
fprintf( "Calculating..." );
```

At this point, C_d and C_l vectors are derived through the function *XfoilParser.m*, and they are used to calculate the two ratios *dT/dr* and *dQ/dr*. Then, thrust and torque coefficients are obtained integrating the arrays *dT/dr* and *dQ/dr* respectively; this operation physically corresponds to the sum of all the single blade elements contributions.

```
[ flag, ClVet( i ), CdVet( i ) ] = ...
        XfoilParser( airfoil, alphad, Re, 'plot' );
    switch flag
        case 1
            fprintf( 'OK!\n\n' );
        case 2
            warning( 'Be carefull! Cl & Cd extrapolation!' );
            fprintf( '\n' );
    end

    dTdr(i)=1/2*sigma*ClVet(i)*rVet(i)^2;
    dQdr(i)=1/2*sigma*(ClVet(i)*phiVet(i)+CdVet(i))*(rVet(i)^3);

end
dTdrf = @( x ) spline( rVet, dTdr, x ) ;
Tc = integral( dTdrf, rVet( 1 ), rVet( end ));

dQdrf = @( x ) spline( rVet, dQdr, x ) ;
Qc = integral( dQdrf, rVet( 1 ), rVet( end ));

end

function [ Ni ] = inputN ( name )

switch name
    case 'variableChord'
        Ni = 2;
    case 'variableTheta'
        Ni = 2;
    otherwise
        error(message('MATLAB:PresRot:wrongVariableName'));
end

end
```

4. Test cases

- First, the correct working of the *RotPerf.m* function has been proved taking as a reference the exercise conducted on October 26th during the AAR lesson. Particularly, the same input data were used to run the Matlab code, and the resulting thrust coefficient was compared with the one obtained with the Excel calculation; then, an error margin was computed to validate the result's truthfulness.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; clear; close all;

Rt = 11.9 / 2;
Rh = Rt * .2 ;
thetaHd = 9;
deltaThetad = 4;
omega = 240 / Rt;
chord = .385;
N = 4;
Vc = 0;
rho = 1.225;
airfoil = 0012;

[ Tc , Qc ] = ...
    RotPerf( Rh, Rt, thetaHd, deltaThetad, omega,
chord, N, Vc, rho, airfoil )

T = Tc * rho * omega^2 * pi * Rt^4;
Ct = T / ( rho * (rpm/59)^2 * (2*Rt)^4 )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The Excel calculation returned a thrust coefficient $\mathbf{T_C=0.0038}$, while the *RotPerf.m* function estimated that $\mathbf{T_C=0.0037}$, with an error margin of 2.6%.

- For the second test the rotor geometric characteristics were extracted from a technical report entitled *"An analytical and experimental investigation of helicopter rotor hover performance and wake geometry characteristics"*, released by United Aircraft Corporation Research Laboratories of Connecticut in 1971.

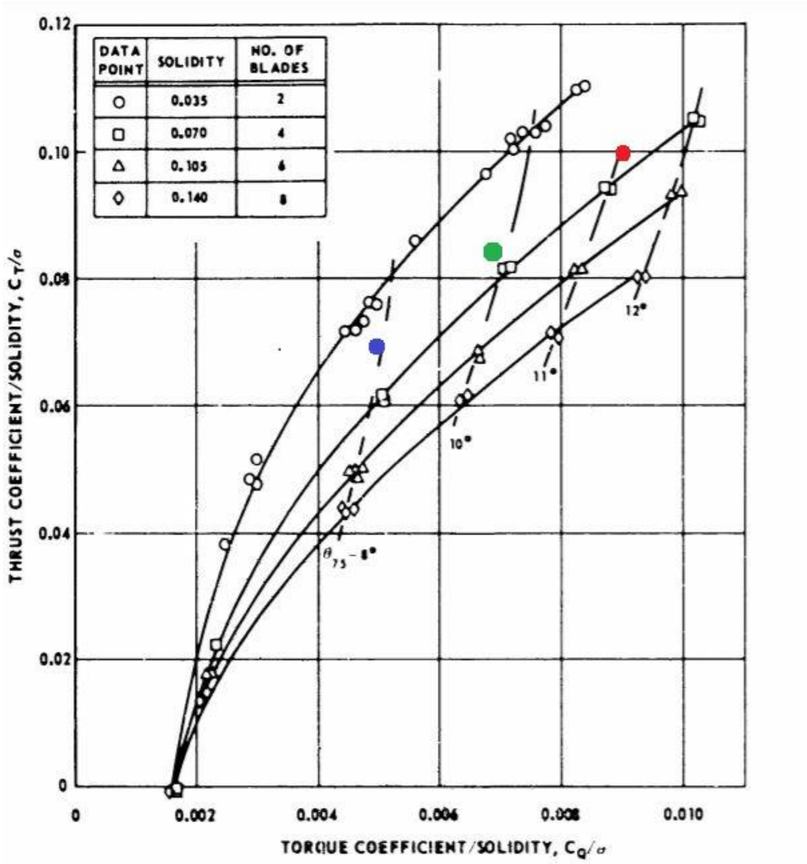
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; clear; close all;

Rt = convlength( 26.75, 'in', 'm' );
AR = 18.2;
chord = Rt / AR;
Rh = Rt * .148;
thetaHd = 8;
deltaThetad = 0;
omega = 45.72;
N = 4;
Vc = 0;
rho = 1.225;
airfoil = 0012;
sigma = chord * N / ( pi * Rt );

[ Tc , Qc ] = ...
    RotPerf( Rh, Rt, thetaHd, deltaThetad, omega,
chord, N, Vc, rho, airfoil )

Ts = Tc / sigma
Qs = Qc / sigma
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The thrust and torque coefficients were derived for three values of thetaHd and reported in the table below; then, these results were compared with the experimental ones reported in the paper and represented in the following Figure.

θ [°]	**T_C/σ**	**Q_C/σ**	Colour
6	0.066	0.0048	
8	0.085	0.0067	
10	0.102	0.0088	

Table 1: T_C/σ and Q_C/σ for different values of θ; the colour identifies the corresponding result in the successive Figure 1.

Since a number of blades equal to four has been chosen for our analysis, the second curve from the top must be considered: it can be noticed that our prediction was quite optimistic; indeed, the Matlab analysis led to a higher thrust and torque with respect to the experimental ones.



(b) ΩR = 525 FPS

Figure 1: Experimental results, T_C/σ versus Q_C/σ for different values of θ and σ, [3]

- The third test was conducted using the geometric parameters reported in "*Numerical Simulation Of Model Helicopter Rotor In Hover*", by Piotr Doerffer And Oskar Szulc. The chosen operating conditions are reported in Figure 2, and the results in Table 2.

$$\Omega = 1250 \text{ rpm}$$
$$M_{tip} = 0.434$$
$$C_T = 0.00213$$

Figure 2: Operating conditions and experimental thrust coefficient, [4]

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; clear; close all;
Rt = 11.9 / 2;
Rh = Rt * .2 ;
thetaHd = 9;
deltaThetad = 4;
omega = 240 / Rt;
chord = .385;
N = 4;
Vc = 0;
rho = 1.225;
airfoil = 0012;

[ Tc , Qc ] = ...
    RotPerf( Rh, Rt, thetaHd, deltaThetad, omega,
chord, N, Vc, rho, airfoil )

T = Tc * rho * omega^2 * pi * Rt^4;
Ct = T / ( rho * (rpm/59)^2 * (2*Rt)^4 )
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Experimental T_C	**Predicted T_C**
0.0020	0.0032

Table 2: Experimental and predicted thrust coefficient, [4]

5. Bibliography

[1] Tognaccini R., *“Lezioni di Aerodinamica dell’Ala Rotante”* (2019)

[2] Di Giorgio G., *“Teoria del volo dell’elicottero”*, Aracne Editrice

[3] *“An analytical and experimental investigation of helicopter rotor hover performance and wake geometry characteristics”*, by Anton J. Landgrebe, United Aircraft Corporation Research Laboratories of Connecticut in 1971

[4] *“Numerical Simulation Of Model Helicopter Rotor In Hover“,* Piotr Doerffer And Oskar Szulc, Institute Of Fluid-Flow Machinery Pas, Fiszera 14, 80-952 Gdansk, Poland