Documentation of Axial_Descent_Ascent_Operating_Curves_Rotor Function

Colledà Moreno M53001072
Veneruso Salvatore M53001082

21 December, 2020

Contents

Chapter 1

Documentation

1.1 Algorithm Introduction

The function plots $w(V_\infty)$ and $P(V_\infty)$ curves according to rotor's simply impulsive theory.
To achieve this objective in the first step we defined the following vectors:

- ALTITUDE
- ASCENT VELOCITY
- DESCENT VELOCITY

Density is then calculated from the altitude vector with the MATLAB function *atmosisa*.
After calculated the density, the axial hovering induction is obtained from the following relation derived by rotor's simply impulsive theory:

$$w_h = \sqrt{\frac{Mg}{2\rho(h)\pi r^2}} \tag{1.1}$$

From this value, we calculated the non-dimensional variables:

- $\widetilde{V} = \frac{V_\infty}{w_h}$
- $\widetilde{w}_{ascent} = -\frac{\widetilde{V}}{2} + \sqrt{\frac{\widetilde{V}}{4}^2 + 1}$
- $\widetilde{w}_{descent} = -\frac{\widetilde{V}}{2} + \sqrt{\frac{\widetilde{V}}{4}^2 - 1}$
- $\widetilde{P}_{ascent} = \widetilde{V} + \widetilde{w}_{ascent}$
- $\widetilde{P}_{descent} = \widetilde{V} + \widetilde{w}_{descent}$

Then, we used some cycles to obtain the matrices including the values of induction and power that are two variables functions.
Subsequently these functions are plotted. In the code the user has also the possibility to insert the interest altitude.

1.2 Algorithm Description

The code begins at line 40 with the function call, where are defined the inputs as well as described in section 1.3.
Dimensional variables, as the altitude and the velocity, are defined in the lines [42-45].
The density as a function of altitude is calculated in line 49 through the MATLAB function *atmosisa*. In line 51 this variable is interpolated in a function handle to obtain a numerical law to use whit any altitude that the user choice in line 152.
From the line 55 to line 59 the code calculates the non-dimensional variables as mentioned in section 1.1.
At lines [61-62] the matrices which will contain the numerical value of induction as a function of velocity and altitude are initialized . Subsequently (lines [66-81]) those matrices are filled with *for loops* as show below, where you can see that is respected the limit of simply impulsive rotor theory:

```
for i = 1 : length(hh)
    for j = 1 : length(VVs)
        WTS(i,j) = w_tilde_salita(VVs(j),hh(i));
    end
end

for i = 1 : length(hh)
    for j = 1 : length(VVd)
        if V_tilde(VVd(j),hh(i)) < -2
        WTD(i,j) = w_tilde_discesa(VVd(j),hh(i));
        aa(1,i) = j;
        else
            WTD(i,j) = 0;
        end
    end
end
```

Subsequently (lines [88-108]) the same thing is done for the non-dimensional power
The first type of outputs are the 3D plots of the induction and power as a function of altitude and velocity, which are in line [113-144]. After that, in lines [148-152] the user has the possibility to choose the altitude of interest to obtain the 2d plots of induction and power as function of only velocity at the entered altitude. In the following lines [154-185] the matrices containg the numerical value of induction and power are initialized and till as function of velocity at the entered altitude as show below:

```
prompt = {'Insert interest altitude in metres [min=0,Max=6000]: '};
dlgtitle = 'Altitude';
dims = [1 35];
answer = inputdlg(prompt,dlgtitle,dims);
hnew = str2double(answer{1});

WTSnew = zeros(1,length(VVs));
WTDnew = zeros(1,length(VVd));

% Matrices fill
for j = 1 : length(VVs)
    WTSnew(1,j) = w_tilde_salita(VVs(j),hnew);
end

for j = 1 : length(VVd)
    if V_tilde(VVd(j),hh(i)) < -2
        WTDnew(1,j) = w_tilde_discesa(VVd(j),hnew);
        aanew = j;
    else
        break
    end
end

PTSnew = zeros(1,length(VVs));
PTDnew = zeros(1,length(VVd));


for j = 1 : length(VVs)
    PTSnew(1,j) =  P_tilde_salita(VVs(j),hnew);
end

for j = 1 : length(VVd)
    if V_tilde(VVd(j),hh(i)) < -2
        PTDnew(1,j) = P_tilde_discesa(VVd(j),hnew);
        bbnew = j;
    else
        break
    end
end
```

The values of induction and power as function of velocity at the entered altitude are plotted through the command in line[188-207].

1.3 Input and Output

The function takes in input:

- Rotorcraft's mass
- Rotor's radius

The outputs are the plots of induction and power and these are reported in section *Test Case*

1.4 Test Case

The numerical values in input for those plots are:

- Rotorcraft's mass = 5000kg
- Rotor's radius = 7m

The outputs of the test case are the following plots:

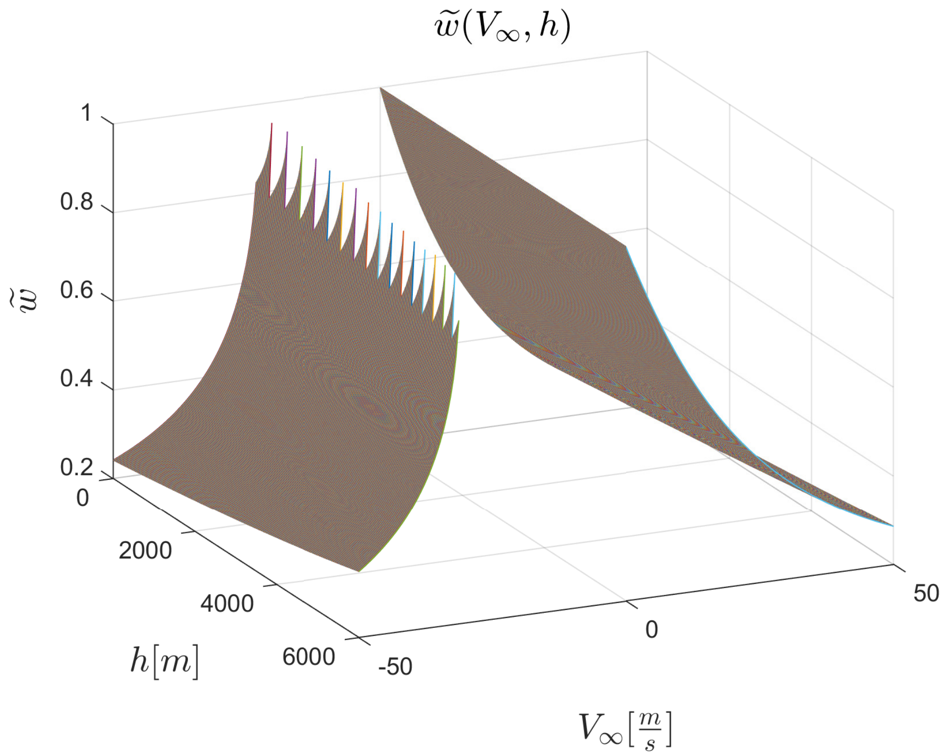

Figure 1.1: *Induction*

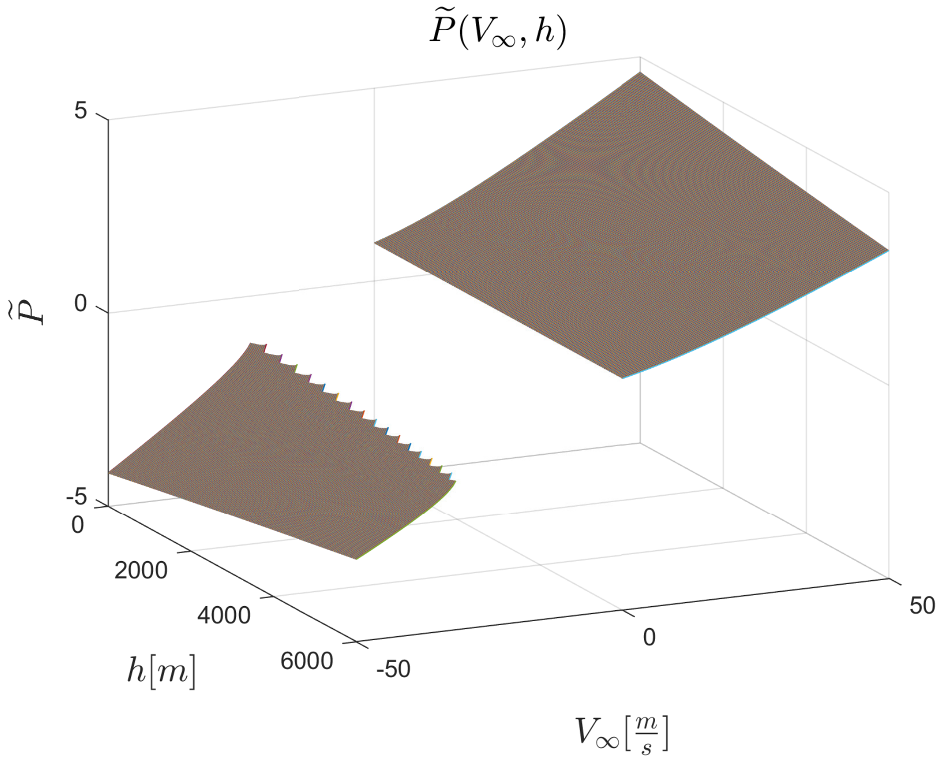

Figure 1.2: *Power*

In the code the user has also the possibility to insert the interest altitude: for example, choosing an altitude of 2000 metres we are obtained the following curves:

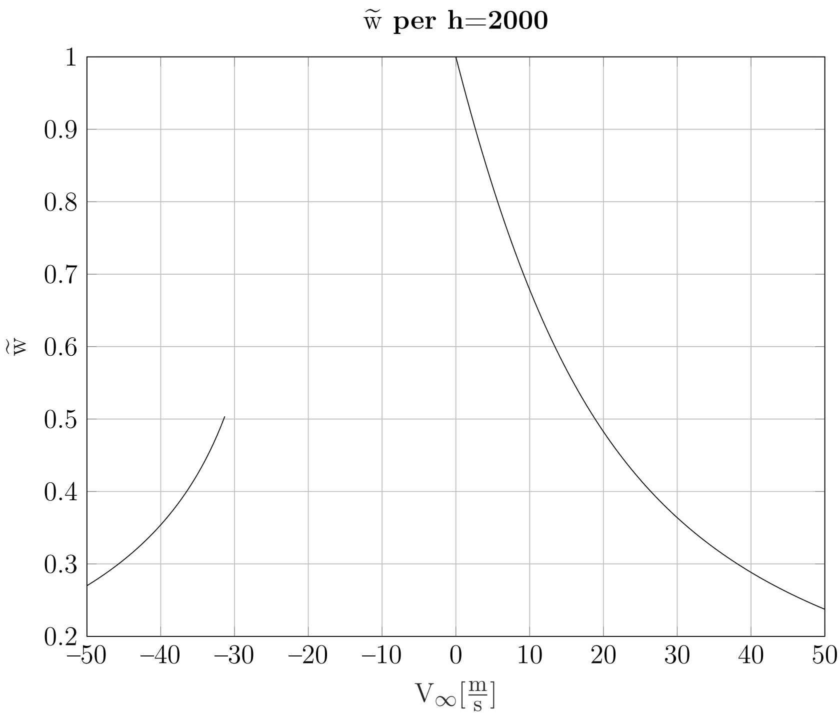

Figure 1.3: *Induction*

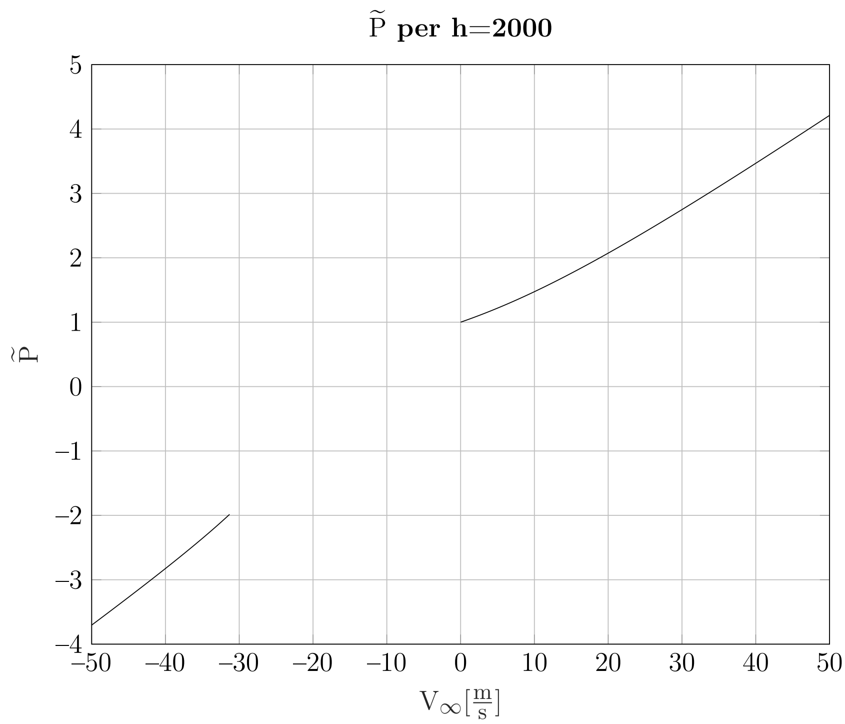

Figure 1.4: *Power*

Appendices

Appendix A

Function's code

```
%% \Axial_Descent_Ascent_Operating_Curves_Rotor.m
%  \brief: the function plots w(V_infty) and P(V_infty) curves according to
%  Impulsive theory.
%  aerodynamic model
%  \author: ColledÃ  Moreno, Veneruso Salvatore
%  \version: 1.00
%
% Eli-TAARG is free software; you can redistribute it and/or
% modify it under the terms of the GNU General Public
% License as published by the Free Software Foundation; either
% version 3 of the License, or (at your option) any later version.
%
% Eli-TAARG is developed by the TAARG Educational organization for
% educational purposes only.
% Theoretical and Applied Aerodynamic Research Group - University of Naples Federico II.
%
% Eli-TAARG GitHub link: <https://github.com/TAARG-Education/Eli-TAARG>
%
% Eli-TAARG is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
% General Public License for more details.
% <http://www.gnu.org/licenses/>.
%
-------------------------------------------------------------------------------
% Name         : Axial_Descent_Ascent_Operating_Curves_Rotor.m
% Author       : ColledÃ  Moreno, Veneruso Salvatore
%                University of Naples Federico II.
% Version      : 1.00
% Date         : 21/12/2020
% Modified     : 21/12/2020
% Description  : the function plots w(V_infty) and P(V_infty) curves according to
%                 Rotor Simply Impulsive theory.
% Reference    : Renato Tognaccini. Appunti Aerodinamica dell'ala rotante.
%                 UniversitÃ  degli studi di Napoli Federico II. a.a.2020/2021
% Input        : * the inputs must be Mass of rotorcraft and radius of rotor
% Output       : w(V_infty) and P(V_infty) plots
% Note         :
-------------------------------------------------------------------------------
function [Power,Induction]=Axial_Descent_Ascent_Operating_Curves_Rotor(M,R)
%Variables Definition
hh     = linspace(0,6000,1000);
VVs    = linspace(0, 50, 100);
VVd    = linspace(-50, 0, 100);
g      = 9.81;

%Axial Induction

[~, ~, ~, rho1] = atmosisa(hh);

rho = @(h) interp1(hh,rho1,h, 'pchip');
wh  = @(h) sqrt((M*g)/(2*rho(h)*pi*R^2));

% Non-Dimensional Variables Definition
V_tilde = @(V,h) V/wh(h);

%Non Dimensional Induction
w_tilde_salita  = @(V,h) (-V_tilde(V,h)/2) + sqrt((V_tilde(V,h)/2)^2+1);
w_tilde_discesa = @(V,h) (-V_tilde(V,h)/2) - sqrt((V_tilde(V,h)/2)^2-1);

WTS = zeros(length(hh),length(VVs)); %Ascent Induction
WTD = zeros(length(hh),length(VVd)); %Descent Induction
aa  = zeros(1,length(hh));           %Control Parameter

% Matrices fill
for i = 1 : length(hh)
    for j = 1 : length(VVs)
        WTS(i,j) = w_tilde_salita(VVs(j),hh(i));
    end
end

for i = 1 : length(hh)
    for j = 1 : length(VVd)
        if V_tilde(VVd(j),hh(i)) < -2            %Validity limit of simply impulsive theory;
```

```
        WTD(i,j) = w_tilde_discesa(VVd(j),hh(i));
        aa(1,i) = j;
        else
            WTD(i,j) = 0;
        end
    end
end

%Non-Dimensional Power

P_tilde_salita   = @(V,h) V_tilde(V,h) + w_tilde_salita(V,h);
P_tilde_discesa  = @(V,h) V_tilde(V,h) + w_tilde_discesa(V,h);

PTS = zeros(length(hh),length(VVs)); %Ascent Power
PTD = zeros(length(hh),length(VVd)); %Descent Power
bb  = zeros(1,length(hh));           %Control Paramenter

% Matrices fill
for i = 1 : length(hh)
    for j = 1 : length(VVs)
        PTS(i,j) = P_tilde_salita(VVs(j),hh(i));
    end
end

for i = 1 : length(hh)
    for j = 1 : length(VVd)
        if V_tilde(VVd(j),hh(i)) < -2            %Validity limit of simply impulsive theory;
        PTD(i,j) = P_tilde_discesa(VVd(j),hh(i));
        bb(1,i) = j;
        else
            break
        end
    end
end


%%
% 3D Plots of Induction [output]
figure(1)
 plot3(hh(1)*ones(1,length(WTS(1,:))),VVs,WTS(1,:))
 hold on
 for i = 2 : length(hh)
     plot3(hh(i)*ones(1,length(WTS(i,:))),VVs,WTS(i,:))
 end
 for i = 1 : length(hh)
     plot3(hh(i)*ones(1,length(WTD(i,1:aa(i)))),VVd(1,1:aa(i)),WTD(i,1:aa(i)))
 end
xlabel('$Quota$','Interpreter','latex', 'FontSize',15)
ylabel('$V_{\infty}$','Interpreter','latex', 'FontSize',15)
zlabel('$w$','Interpreter','latex', 'FontSize',15)
grid on
title('$w (V_{\infty}, h)$','Interpreter','latex', 'FontSize', 15)
view(71,32)

% 3D Plots of Power [output]
figure(2)
 plot3(hh(1)*ones(1,length(PTS(1,:))),VVs,PTS(1,:))
 hold on
 for i = 2 : length(hh)
     plot3(hh(i)*ones(1,length(PTS(i,:))),VVs,PTS(i,:))
 end
 for i = 1 : length(hh)
     plot3(hh(i)*ones(1,length(PTD(i,1:bb(i)))),VVd(1,1:bb(i)),PTD(i,1:bb(i)))
 end
xlabel('$Quota$','Interpreter','latex', 'FontSize',15)
ylabel('$V_{\infty}$','Interpreter','latex', 'FontSize',15)
zlabel('$P$','Interpreter','latex', 'FontSize',15)
grid on
title('$P (V_{\infty}, h)$','Interpreter','latex', 'FontSize', 15)
view(71,32)

%% Insertion of Interes Altitude;

prompt = {'Insert interest altitude in metres [min=0,Max=6000]: '};
dlgtitle = 'Altitude';
dims = [1 35];
answer = inputdlg(prompt,dlgtitle,dims);
hnew = str2double(answer{1});

WTSnew = zeros(1,length(VVs));  %Vector Inizialization of axial ascent induction related to velocity
WTDnew = zeros(1,length(VVd));  %Vector Inizialization of axial descent induction related to velocity

% Matrices fill
for j = 1 : length(VVs)
    WTSnew(1,j) = w_tilde_salita(VVs(j),hnew);
end

for j = 1 : length(VVd)
    if V_tilde(VVd(j),hh(i)) < -2                   %Validity limit of simply impulsive theory;
        WTDnew(1,j) = w_tilde_discesa(VVd(j),hnew);
        aanew = j;
    else
        break
    end
end

PTSnew = zeros(1,length(VVs));
PTDnew = zeros(1,length(VVd));

for j = 1 : length(VVs)
    PTSnew(1,j) =  P_tilde_salita(VVs(j),hnew);
end

```

```
for j = 1 : length(VVd)
    if V_tilde(VVd(j),hh(i)) < -2                    %Validity limit of simply impulsive theory;
        PTDnew(1,j) = P_tilde_discesa(VVd(j),hnew);
        bbnew = j;
    else
        break
    end
end
%%
% 2D Plots of induction [Output]
figure(3)
plot(VVs, WTSnew, '-k')
hold on
plot(VVd(1:aanew), WTDnew(1:aanew), '-k')
grid on
xlabel('$V_{\infty}$','Interpreter','latex','FontSize',15)
ylabel('$w$','Interpreter','latex','FontSize',15)
title(['$w$ per h=' num2str(hnew)],'Interpreter','latex')
matlab2tikz('D:\UniversitÃ \Magistrale\AerodinamicaAlaRotante\Matlab\EliiTAARG\Figure\Induzione2d.tex');

% 2D Plots of power [Output]
figure(4)
plot(VVs, PTSnew, '-k')
hold on
plot(VVd(1:aanew), PTDnew(1:bbnew), '-k')
grid on
xlabel('$V_{\infty}$','Interpreter','latex','FontSize',15)
ylabel('$P$','Interpreter','latex','FontSize',15)
title(['$P$ per h=' num2str(hnew)],'Interpreter','latex')
matlab2tikz('D:\UniversitÃ \Magistrale\AerodinamicaAlaRotante\Matlab\EliiTAARG\Figure\Potenza2d.tex');
%%
% Numerical Output of Power and Induction for the interest altitude;
Power = [PTDnew(1:bbnew), PTSnew];
Induction = [WTDnew(1:aanew), WTSnew];
end
----------------------------------------------------------------
```

Bibliography

[1] Renato Tognaccini, *Appunti Aerodinamica dell'Ala Rotante*. Univeristà degli studi di Napoli Federico II, a.a. 2020/2021