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 $\mathrm{w}(\mathrm{V}_\infty)$ and $\mathrm{P}(\mathrm{V}_\infty)$ 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:

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

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

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

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.
At line 44 altitude vector is defined to allow the calculation of the 3D curves outputs, as show in section 1.5, and the calculation of the density. The density as a function of altitude is calculated in line 48 through the MATLAB function *atmosisa*. In line 50 this variable is interpolated in a function handle to obtain a numerical law to use whit any altitude the user isert in input. From the line 54 to line 57 the code calculates the non-dimensional variables as mentioned in section 1.1.
At line 59 if the user has insert only 3 variables the code calculated the power and the induction distributions. Indeed At lines [65-66] the matrices which will contain the numerical value of induction as a function of velocity and altitude are initialized . Subsequently (lines [70-85]) 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 [85-105]) 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 [112-146]. After that, in lines [150-182] the matrices containg the numerical value of induction and power are inizialized and till as a function of velocity at the altitude insert in input, as show below:

```
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[185-202].
While, in lines [207-229], if the user insert four inputs (rotorcraft's mass, rotor's radius, altitude and ascent or descent velocity) the code calculated and shows only the power and the induction at the altitude at the velocity inserted as inputs.

1.3 Input and Output

The function takes in input:

- Rotorcraft's mass
- Rotor's radius
- Altitude
- Ascent or Descent Velocity

If the inputs values are three (Rotorcraft's mass, Rotor's radius and altitude) the outputs are the plots of induction and power and these are reported in section *Test Case*. If the inputs values are all four the outputs are only the value of power and induction at the altitude and velocity inserted as inputs.

1.4 Error Indicators

The plots in axial discent operating condition have been obtained within the validity limits of simple impulsive rotor theory, that's until when $V_\infty < -2w$. Indeed for these values in inputs:

- $V_\infty = -10[m/s]$
- $h = 1000[m]$

the code shows this error message:

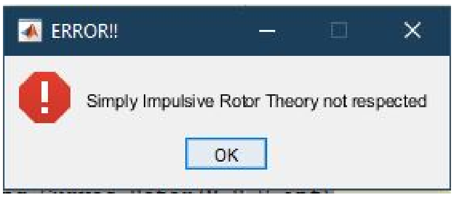


Figure 1.1: *Error Message*

1.5 Test Case

The numerical values in input for those plots are:

- Rotorcraft's mass = 5000kg
- Rotor's radius = 7m
- Altitude = 0m

The outputs of the test case are the following plots:

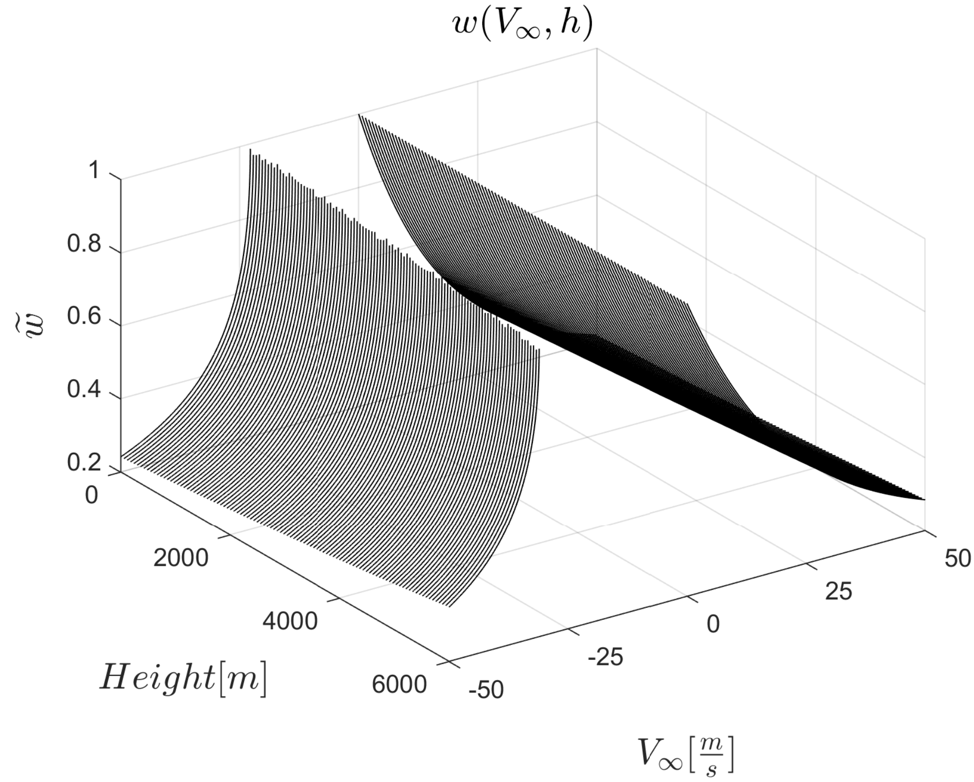


Figure 1.2: *Induction*

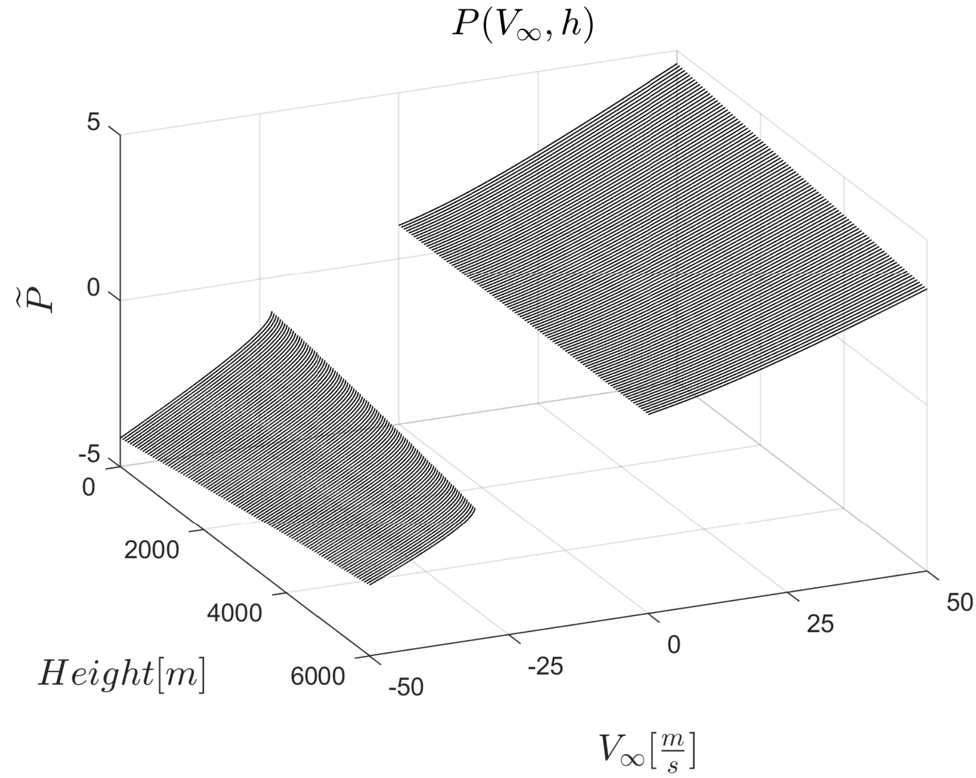


Figure 1.3: *Power*

At sea level we are obtained the following curves. In order to prove the validity of the code the curves were compared with some examples reported in [1] as follow.

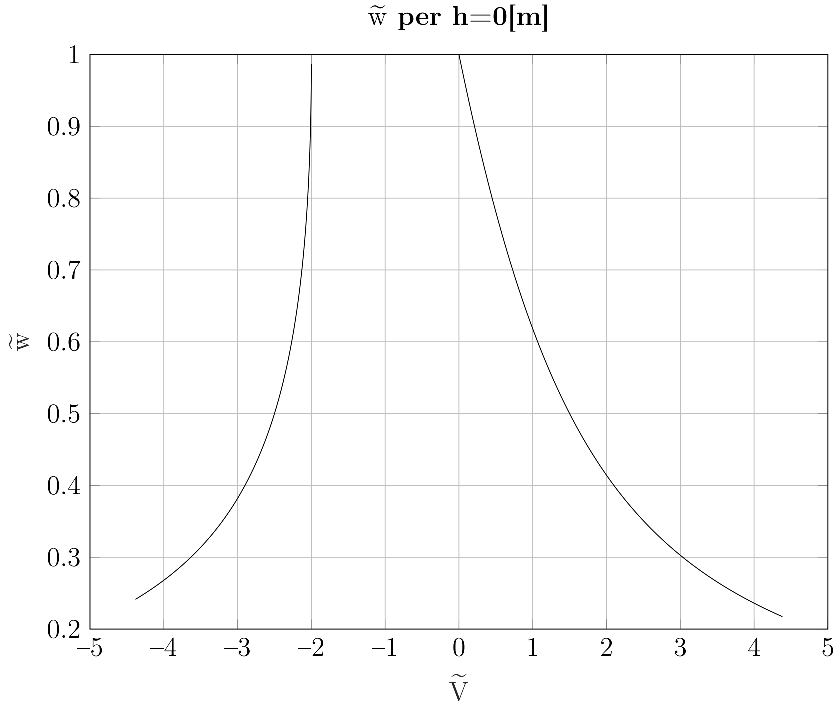
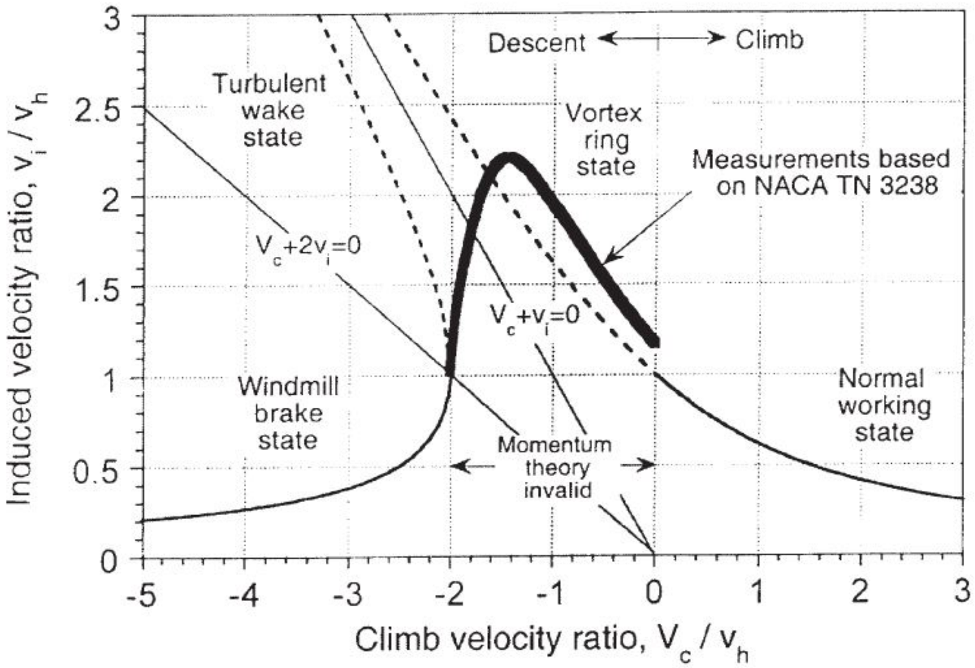


Figure 1.4: *Induction*



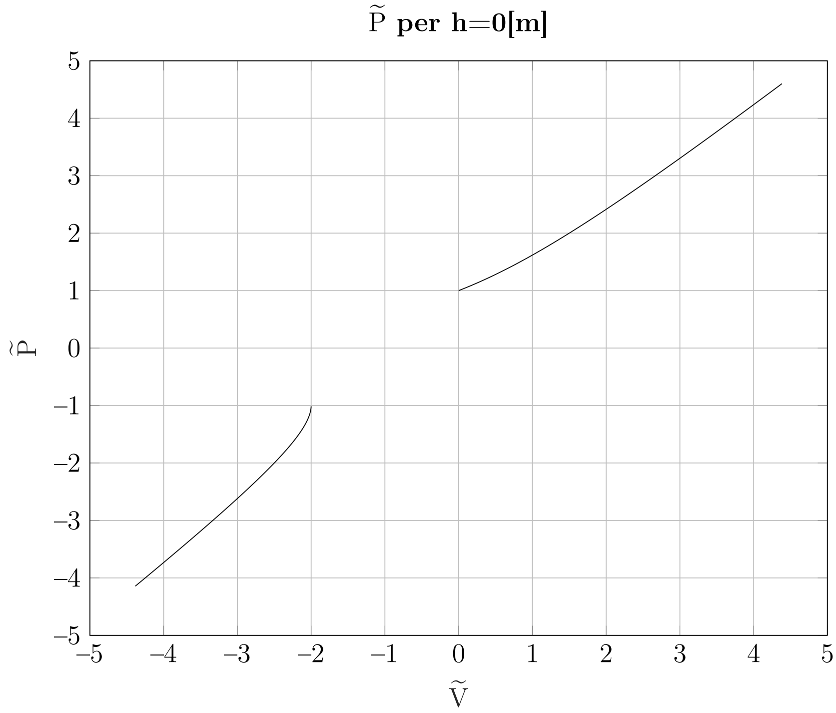
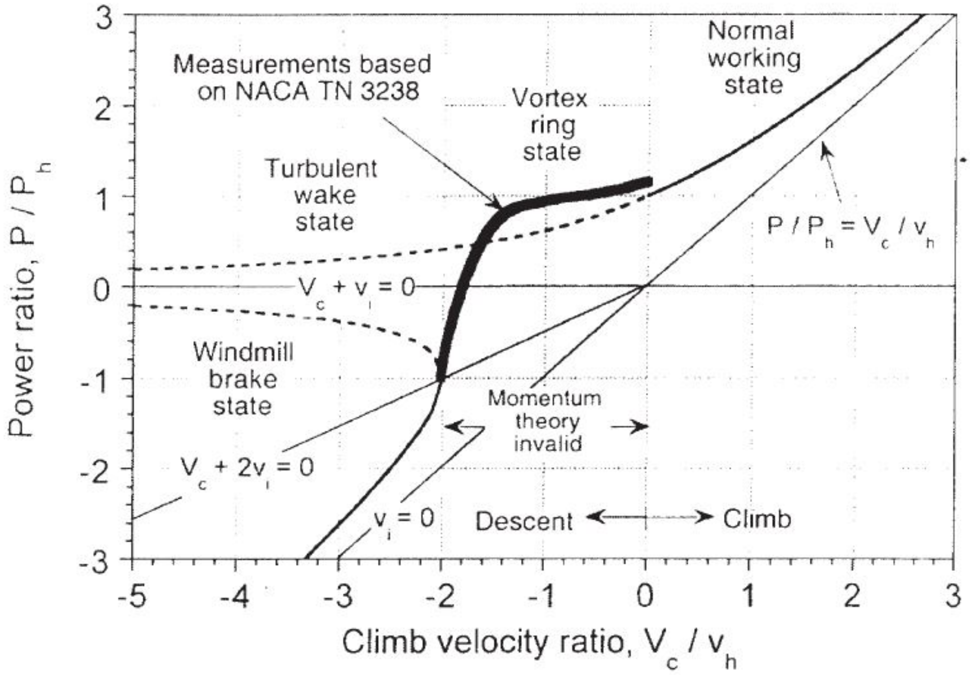


Figure 1.5: *Power*



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,hnew,V_inf)

g = 9.81;
%Altitude Range
hh    = linspace(0,6000,100);

%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);

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);

if nargin==3
    VVs   = linspace(0, 50, 100);
    VVd   = linspace(-50, 0, 7000);

    %Non Dimensional Induction

    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
```

```matlab

    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,:),'k')
    hold on
    for i = 2 : length(hh)
        plot3(hh(i)*ones(1,length(WTS(i,:))),VVs,WTS(i,:),'k')
    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)),'k')
    end
    yticks([-50 -25 0 25 50])
    xlabel('$Height[m]$','Interpreter','latex', 'FontSize',15)
    ylabel('$V_{\infty}[\frac{m}{s}]$','Interpreter','latex', 'FontSize',15)
    zlabel('$\widetilde{w}$','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,:),'k')
    hold on
    for i = 2 : length(hh)
        plot3(hh(i)*ones(1,length(PTS(i,:))),VVs,PTS(i,:),'k')
    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)),'k')
    end
    yticks([-50 -25 0 25 50])
    xlabel('$Height[m]$','Interpreter','latex', 'FontSize',15)
    ylabel('$V_{\infty}[\frac{m}{s}]$','Interpreter','latex', 'FontSize',15)
    zlabel('$\widetilde{P}$','Interpreter','latex', 'FontSize',15)
    grid on
    title('$P (V_{\infty}, h)$','Interpreter','latex', 'FontSize', 15)
    view(71,32)

    %% Insertion of Interes Altitude;

    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),hnew) <= -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),hnew) <= -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(V_tilde(VVs,hnew), WTSnew, '-k')
    hold on
    plot(V_tilde(VVd(1:aanew),hnew), WTDnew(1:aanew), '-k')
    grid on
    xlabel('$\widetilde{V}$','Interpreter','latex','FontSize',15)
    ylabel('$\widetilde{w}$','Interpreter','latex','FontSize',15)
    title(['$\widetilde{w}$ per h=' num2str(hnew) '[m]'],'Interpreter','latex')

    % 2D Plots of power [Output]
    figure(4)
    plot(V_tilde(VVs,hnew), PTSnew, '-k')
    hold on
    plot(V_tilde(VVd(1:aanew),hnew), PTDnew(1:bbnew), '-k')
    grid on
    xlabel('$\widetilde{V}$','Interpreter','latex','FontSize',15)
    ylabel('$\widetilde{P}$','Interpreter','latex','FontSize',15)
    title(['$\widetilde{P}$ per h=' num2str(hnew) '[m]'],'Interpreter','latex')
    %% Numerical Output of Power and Induction for the interest altitude;
    Power = [PTDnew(1:bbnew), PTSnew];
    Induction = [WTDnew(1:aanew), WTSnew];

elseif nargin==4 % In Output only value of Power and Induction
    % at altitude and velocity of interest;

    if V_tilde(V_inf,hnew) >= 0
        w = w_tilde_salita(V_inf,hnew)*wh(hnew);
        P   = (M*g)*(V_inf + w);
    elseif V_tilde(V_inf,hnew)<= -2
        w= w_tilde_discesa(V_inf,hnew)*wh(hnew);
        P  = (M*g)*(V_inf + w);
    else
        errordlg('Simply Impulsive Rotor Theory not respected','ERROR!!');
        %Output
        Power = [];
        Induction = [];
        return
    end
    ff = msgbox(sprintf('Power= %d [kW], \n Induction= %d [m/s]', P/1000, w),...
        'Power and Induction at the altitude and velocity of interest');
    set(ff, 'position', [500 250 400 65]);
    %Output
    Power = P;
    Induction = w;
end

end
----------------------------------------------------------------
```

Bibliography

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