Autorot_performance

This function provides autorotation performance of a rotor: the procedure is based upon the one shown in reference (par. 8.10). For a given helicopter, advance ratio and descend angle are to be assigned by the user. Autorotation condition yields to equations (see reference), which are here dealt with by means of Matlab Symbolic Toolbox. Once solved in respect to the intake ratio, expressions of thrust coefficient, attack angle, angular velocity, freestream velocity and collective pitch angle can be evaluated. Please note that solver may not find a solution: it simply means that autorotation cannot be achieved under the assigned conditions. This is taken into account by means of an "if-else" construct, which yields a warning message (NaN is assigned to output value).

Procedure description

The procedure upon which the function is based on is now described.

- Assign advance ratio μ.
- By autorotation definition the following equation $P_c = \lambda T_c + Q_{c_0} + \mu H_{c_0} - \mu H_c = 0$ is valid. Once solved, the solution yields $\theta_0 = \theta_0(\lambda)$, thus the relations $T_c = T_c(\lambda)$, $H_c = H_c(\lambda)$ and $\alpha = \alpha(\lambda)$.
- Assign descent angle: this allows λ_c to be evaluated. Consequently, it is possible to solve $\lambda_i T_c + \lambda_c T_c + \mu D_{fus}/W T_c + P_{c_0} = 0$, obtaining λ_i and λ.
- Using the calculated λ, it is now possible to obtain θ_0 and force coefficients.
- From the equation $T_c \rho \Omega^2 R^2 A = W$ it is possible to find Ω.
- Finally, α and V_∞ can be evaluated.

syntax

The default syntax of the function is shown below; all inputs are scalar values.

```
[T_C, H_C, alpha_deg, omega, V_inf, Theta_0] = autorot_performance(mu,X,R,N,c
    ,gamma,theta_tw,f,M)
```

Where:

Input

W	Helicopter weight [kg]
R	Rotor blade radius [m]
X	descent angle [deg]
c	Chord length [m]
gamma	Lock number
theta_tw	Blade twist [deg]
f	Equivalent wet area [m^2]
mu	Advance ratio
N	Number of blades

Output

T_C	Thrust coefficient
H_C	Drag coefficient
alpha	Angle of attack [deg]
omega	Rotor angular velocity [rad/s]
V_inf	Freestream velocity [m/s]
Theta_0	Collective pitch [deg]

error indicator

For given flying conditions, it may be not possible to find autorotation conditions: this means that "vpasolve" cannot find a solution and λ will be an empty vector. If this is the case, the function shows a warning message and the output vector contains NaN elements.

example

A test case for the function is shown below. The helicopter is an Agusta Bell 206.

```
%% Agusta Bell AB 206 Helicopter in forward flight
%  Advance ratio = 0.15
%  Descent angle = 20


%User Input:

%Helicopter
W = 1120;                       % [Kg], mass
R = 5.1;                        % [m], blade radius
N = 2;                          % number of blades
c = 0.34;                       % [m], chord length
gamma = 9;                      % Lock's number
theta_tw = -13.2;               % [deg], blade twist (linear variation is
    assumed)
f = 0.007;                      % [m^2] equivalent wet area

mu = 0.15;                      %Advance ratio
X = 20;                         %Descend angle

%Function
[T_C H_C alpha_deg omega V_inf Theta_0]=autorot_performance(W,R,N,c,gamma,
    theta_tw,f,mu,X);

%Function Output
T_C        =   0.0017
H_C        =   1.4473e-05
alpha_deg  = -15.3454
omega      =  50.0588
V_inf      =  39.7107
Theta_0    =   8.9242
```

code listing

```
function [T_C, H_C, alpha, omega, V_inf, Theta_0] = autorot_performance(W,R,N
    ,c,gamma,theta_tw,f,mu,X)


Cl_a = 2*pi;                    % [1/rad], lift coefficient gradient
Cd = 0.011;                     % Average drag coefficient along the blade
A = pi*R^2;                     % [m^2], swept area
sigma = N*c/(pi*R);             % rotor solidity
theta_tw = deg2rad(theta_tw);   % [rad], pitch twist
W = W*9.81;                     % [N], weight
rho = 1.225;                    % [Kg/m^3], density (SML)
X = deg2rad(X);                 % [rad], descent angle
lambda_c = -mu*sin(X);          % Descent ratio (<0)


%% Beginning of the procedure.
% We need to evaluate an expression of theta_0(lambda). Evaluate T_c, Q_c0,
% H_ci first:

syms theta_0 lambda real

T_c = 0.5*sigma*Cl_a*(theta_0/3*(1+3/2*mu^2)+theta_tw/4*(1+mu^2)-lambda/2);
    % thrust coefficient

Q_c0 = sigma*Cd/8*(1+mu^2);
    % parasitic torque coefficient

b_0 = gamma*(theta_0/8*(1+mu^2)+theta_tw/10*(1+5/6*mu^2)-lambda/6);
    % flapping coefficients expressions

b_1c = -2*mu*(4/3*theta_0+theta_tw-lambda)/(1-0.5*mu^2);

b_1s = -4/3*mu*b_0/(1+0.5*mu^2);

H_ci = 0.5*sigma*Cl_a*(theta_0*(-1/3*b_1c+0.5*mu*lambda)...
    % induced rotor drag coefficient
    +theta_tw*(-1/4*b_1c+1/4*mu*lambda)+3/4*lambda*b_1c...
    +1/6*b_0*b_1s+1/4*mu*(b_0^2+b_1c^2));

% We can find an expression of theta_0(lambda) then (we use capital
%subscripts from this point)

%from the definition of autorotation:
P_c = lambda*T_c+Q_c0-mu*H_ci;
    % Equation to solve in respect to theta_0

THETA_0 = vpasolve(P_c==0,theta_0);
    % Two solutions

THETA_0 = THETA_0(2);
    % We choose the second one

% It is possible now to find Tc=Tc(lambda),Hc=Hc(lambda),alfa=alfa(lambda)

T_C = 0.5*sigma*Cl_a*(THETA_0/3*(1+3/2*mu^2)...
    % Tc as a function of lambda only
    +theta_tw/4*(1+mu^2) -lambda/2);

H_C0 = sigma*Cd*mu/4;
    % parasitic rotor drag coefficient

B_0 = gamma*(THETA_0/8*(1+mu^2)+theta_tw/10*(1+5/6*mu^2)...
    % flapping coefficients as a function of lambda only
    -lambda/6);

B_1c = -2*mu*(4/3*THETA_0+theta_tw-lambda)/(1-0.5*mu^2);

B_1s_lambda = -4/3*mu*B_0/(1+0.5*mu^2);

H_Ci = 0.5*sigma*Cl_a*(THETA_0*(-1/3*B_1c...
```

```
    % induced rotor drag coefficient
    +0.5*mu*lambda)+theta_tw*(-1/4*B_1c+1/4*mu*lambda)...
    % as a function of lambda only
    +3/4*lambda*B_1c+1/6*B_0*B_1s_lambda...
    +1/4*mu*(B_0^2+B_1c^2));

alpha = atan(1/mu*(lambda-T_C/(2*sqrt(mu^2+lambda^2))));
    % angle of attack as a function of lambda only

% We can now solve an equation for lambda

lambda_i = T_C/(2*sqrt(mu^2+lambda^2));

P_C = T_C*lambda_i+lambda_c*T_C+...
    % This expression is obtained from the Pc expression
      mu*0.5*f/A*(mu/cos(alpha))^3+H_C0+Q_c0;

lambda = vpasolve(P_C==0,lambda);
    % This equation could have no solution

    % depending on flying condition. This is dealt

    % with by means of an if else construct.

if isempty(lambda)                                  % If lambda is empty,
    then previous equation has no solution.

    warning('Autorotation cannot be achieved under user specified conditions'
    )
    T_C = NaN;
    H_C = NaN;
    alpha_deg = NaN;
    omega = NaN;
    V_inf = NaN;
    Theta_0 = NaN;

else

Lambda_i  = double(subs(lambda_i));

%% Evaluating output

T_C       = double(subs(T_C))               % Thrust coefficient

H_C       = double(subs(H_Ci))+H_C0         % Rotor drag coefficient

alpha     = double(subs(rad2deg(alpha)))    % [deg] Angle of attack

omega     = double(sqrt(W/(rho*A*T_C*R^2))  % [rad/s] Angular velocity

V_inf     = mu*omega*R/cos(deg2rad(alpha))  % [m/s] Freestream velocity

Theta_0   = rad2deg(double(subs(THETA_0)))  % [deg] Collective pitch
end
end
```