University of Bath
Faculty of Engineering & Design

Word count: 2154

November 13, 2018

Advanced Helicopter Dynamics

Supervisor
D. Cleaver

Assessor

Author's Candidate Number
10838

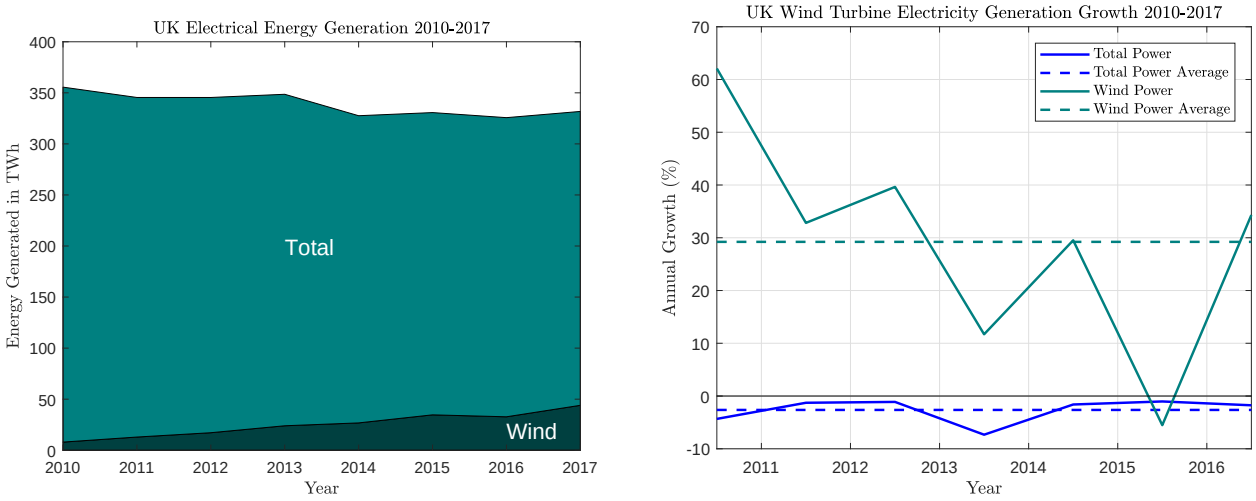
Contents

List of Figures

List of Tables

1 Introduction

Somewhat surprisingly over the period of 2010 to 2017 total electricity generation in the United Kingdom fell by 18 % [1] as consumption also fell 10% across the period [2]. This is in spite of a rising population and number of households. This has been attributed to warmer winter lowering electric heating demands as well as improvements in refrigerator technology and increased use of LED lights [5]. Using data made available by the UK Department for Business, Energy & Industrial Strategy [1] it can be seen that wind power has defied the slowdown of electricity generation in the UK, rising an average 29% year on year over the same 2010 to 2017 period.



(a) Annual Electrical Energy Generation

(b) Annual Growth of Electrical Energy Generation

Figure 1: Statistics on UK Electrical Energy Generation

Traditionally the wind sectors growth was completely reliant on large government subsidies but in recent years governments such as the UK and Germany have adopted a new approach to subsidisation which has lead to significant shift in the sector's economics. The government policy change was to make energy companies bid for projects and associated subsidies meaning the actual subsidy received is the difference between the subsidy and the bid. This competitive approach (contract for difference) has driven innovation with off-shore wind cost more than halving in the UK in just three years [6]. In some cases large wind projects have become viable with no subsidy or tax-break at all. The significance is the rate of development within the wind energy sector is now not reliant on government renewable energy policy but is shifting to large scale private sector investment. Due to the sheer size of the energy industry this could lead to significant wind turbine technology development. With the age of the electric car expected to dramatically reverse the trend of falling electricity consumption seen in Figure 1b, research into wind turbine design has never been more relevant.

The optimal design of a wind turbine blade is one which produces the most power for the lowest unit cost [4]. This is would require a vast amount of cost data which is not readily available and so this paper shall instead focus on the design of a technically optimal design. Specifically the design which has the highest annual energy production (AEP) within the parameters given in Table 1.

Table 1: Design Parameters for Wind Turbine Design [4]

	Quantity	Symbol	Value	Units
DIMENSIONS	Hub Height	H_{hub}	35	m
	Hub-Tower Separation	d_{hub}	3	m
	Root Radius	R_{min}	1	m
	Maximum Radius	R_{max}	20	m
	Mean Chord	c_{mean}	1	m
FORCES	Maximum Root Bending Moment	$M_{root,max}$	0.5	MNm
	Maximum Vertical Hub Force	$F_{Y,max}$	70,000	N
QUANTITIES	Blade Density	ρ_{blade}	2,000	kg/m^3
	Blade Stiffness	$(EI)_{blade}$	-	GPa
	Cut-In Speed (assumed)	V_{min}	5	m/s
	Cut-Out Speed (assumed)	V_{max}	25	m/s
	Weibull Coefficient	A	7	-
	Weibull Coefficient	k	1.8	-
	Rotational Speed	ω	30	rpm

2 Methodology

To find the optimal design blade element momentum theory was used to calculate power output for a given velocity and parameter set. Using Weibull's wind speed distribution and power output for a given velocity, AEP was calculated. Part A of the flow diagram of Figure 2 shows the flow of information for the computer program used to calculate the AEP.

The design parameter's which could be changed for blade design were inclination at root: θ_0, rate of twist: θ_{tw} and chord gradient: c_g. The theoretical maximum power output for a given wind speed, V_0, and reference area, A, is the called the Betz limit and is given by equation 1. The maximum theoretical AEP can be found using the Weibull distribution and the Betz limit. Using a minimum finding function from Matlab's library the combination of the three parameters which gives the least difference to the Betz limit AEP can be found thus finding the optimal design for the given parameter set.

$$P_{Betz} = \frac{16}{27}\left(\frac{1}{2}\rho V_0^3 A\right) \tag{1}$$

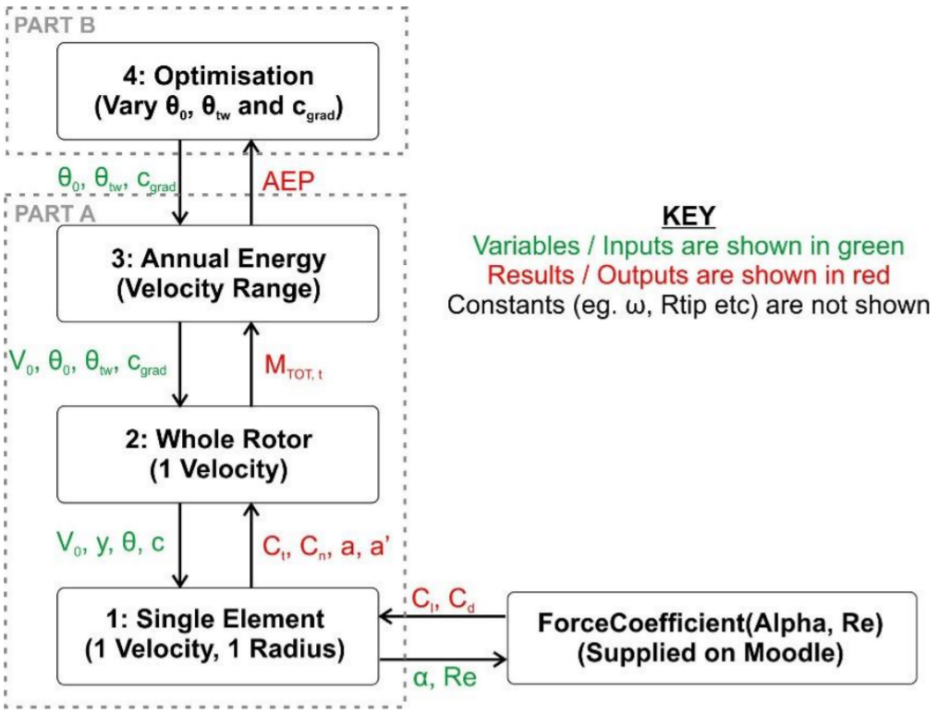


Figure 2: Flow Diagram of Method Used to Find Annual Energy Production for the Turbine Design [4]

3 Results and Discussion

3.1 A: Optimal Design

3.2 B: Limitations of Model

4 Conclusions

References

[1] Department for Business, Energy & Industrial Stratergy, 2018. *Historical Electricity Data:1920 to 2017.*
Available: www.gov.uk/government/statistical-data-sets/historical-electricity-data

[2] Digest of United Kingdom Energy Statistics (DUKES), 2018. *Digest of UK Energy Statistics (DUKES): electricity.* Available:https://www.gov.uk/government/collections/electricity-statistics

[3] Cleaver, D., 2018. *Slides: WT Coursework.*
University of Bath.

[4] Cleaver, D., 2018. *ME40343 Helicopter Dynamics: Coursework.*
University of Bath.

[5] Vaughan, A., 2017. *Electric cars will fuel huge demand for power, says National Grid.* The Guardian.

[6] Arie, S., 2018. *Renewables are primed to enter the global energy race.* The Financial Times.

Appendices

A Induced Calculations Function

```
function [a_out, adash_out, phi, Cn, Ct] = ...
    WTInducedCalcs(a_in, adash_in, V0, omega, y, theta, chord, B)
%1: SINGLE ELEMENT: use an iterative solution to find the ...
    values of a,
%adash, phi, Cn and Ct at a particular radius.

%% Set constants
rho = 1.225;          %density of air in kg/m^3
mu = 18.81e-6;        %kinematic viscosity in Pa.s

error_tol = .0001;  % setting allowable error
error = error_tol + 1; %set first error to initiate while loop

%% loop over until error goes below tolerance
i = 0;       %Initialize counter, will need to break loop if i ...
    exceeds 100
while error ≥ error_tol && i <101
    i = i + 1;

    %% Calculate  angles
    phi = atan(((1-a_in)*V0) / ((1+adash_in)*omega*y));

    alpha = phi - theta;

    %% Calculate Reynolds number
    V_rel = ((V0*(1-a_in))^2 + ((omega*y)*(1 + adash_in))^2)^.5;

    Re = (rho*V_rel*chord) / mu;

    %% Get non-dimensional lift and drag coefficients and convert
    [Cl, Cd] = ForceCoefficient(alpha, Re);
    Cn = Cl*cos(phi) + Cd*sin(phi);          %Normal Force ...
        Coefficient
    Ct = Cl*sin(phi) - Cd*cos(phi);          %Tangential Force ...
        Coefficient

    %% Calculate new values for a and a'
    sigma = (B*chord) / (2*pi()*y);
    a_out = 1 / (((4*sin(phi)^2)/(sigma*Cn)) + 1);
```

```
35      adash_out = 1 / (((4*sin(phi)*cos(phi))/(sigma*Ct)) - 1);
36
37      %% Calculate error
38      error = abs(a_out - a_in) + abs(adash_out - adash_in);
39
40      %% If error greater than allowable error apply relaxation ...
           factor,
41      %else the solution has been found and while loop can be ...
           broken
42      if error ≥ error_tol
43          a_in = 0.1*(a_out - a_in) + a_in;
44          adash_in = 0.1*(adash_out - adash_in) + adash_in;
45      else
46          break
47      end
48
49  end
50
51  %% If counter exceed 100 set a' to zero and solve for a
52
53  if i > 100
54
55      a_in = 0;
56      adash_out = 0;
57      while error ≥ error_tol
58
59          %% If counter reaches 500 the error tolerance
60          % can be decreased to get a result.
61
62          i = i+1;    %Increase counter
63
64          if i ≥ 500              %Decrease error tol after 500 ...
               interations
65              error_tol = 0.01;
66          end
67          %% Calculate  angles
68          phi = atan(((1-a_in)*V0) / ((1)*omega*y));
69
70          alpha = phi - theta;
71
72
73          %% Calculate Reynolds number
74          V_rel = ((V0*(1-a_in))^2 + ((omega*y)*(1))^2)^.5;
75
76          Re = (rho*V_rel*chord) / mu;
```

```
77
78          %% Get non-dimensional lift and drag coefficients and ...
               convert
79          [Cl, Cd] = ForceCoefficient(alpha, Re);
80
81          Cn = Cl*cos(phi) + Cd*sin(phi);          %Normal Force ...
               Coefficient
82          Ct = Cl*sin(phi) - Cd*cos(phi);          %Tangential ...
               Force Coefficient
83
84          %% Calculate new values for a and a'
85          sigma = (B*chord) / (2*pi*y);
86          a_out = 1 / (((4*sin(phi)^2)/(sigma*Cn)) + 1);
87
88          %% Calculate error
89          error = abs(a_out - a_in);
90
91          %% If error greater than allowable error apply ...
               relaxation factor,
92          % else the solution has been found and while loop can ...
               be broken
93          if error ≥ error_tol
94              a_in = 0.1*(a_out - a_in) + a_in;  %Relaxation ...
                   facto k=.1 added
95          else
96              break
97          end
98
99      end
100
101 end
102 end
```

B Wind Turbine Single Velocity Function

```
1  function [MT, MN, MT_local, MN_local] = WTSingleVelocity(V0, ...
       theta0, theta_twist, chord_mean, chord_grad, TipRadius, ...
       RootRadius, omega, B)
2  %2: WHOLE ROTOR - loop WTInducedCalcs to find the values for ...
       all radii,
3  %then integrate these to get the normal and tangential moment ...
       at the blade
```

```
%root.

%% Set constants
rho = 1.225;        %density of air in kg/m^3

%% Set up geometry
y = 1.5:1:19.5;   %Vector of local radii (elements) (in m)
y_△ = y(2) - y(1);  %△ is difference between adjacent elements

R = TipRadius;

%% Set up empty outputs arrays to improve efficiency
MN_local = zeros(length(y),1);
MT_local = zeros(length(y),1);
%% 2: Calculate individual element moment both in plane and ...
   normal

for i = 1:length(y)

    %Calculate local chord and angle of attack
    chord_local = chord_mean + ((y(i)) - (R/2))*chord_grad;
    assert(chord_local ≥ 0 && chord_local ≤ 2, strcat('Error ...
       in chord length! Chord length = ', num2str(chord_local)))
    theta_local = theta0 + (y(i))*theta_twist;

    %Calculate Cn and Ct for the local element
    [a_out, adash_out, ¬, Cn, Ct] = WTInducedCalcs(0, 0, V0, ...
       omega, y(i), theta_local, chord_local , B);

    % Calculate Relative Velocity
    V_rel = ((V0*(1-a_out))^2 + ((omega*y(i))*(1 + ...
       adash_out))^2)^.5;

    %Calculate the local moment in normal and tangential ...
       (torque) directions
    MN_local(i) = (.5*rho*(V_rel^2)*chord_local*Cn)*y_△*y(i);
    MT_local(i) = (.5*rho*(V_rel^2)*chord_local*Ct)*y_△*y(i);

end

%% Integrate the moments to find totals

MN = sum(MN_local);
MT = sum(MT_local);

```

```
44 end
```

C Wind Turbine Velocity Range Function

```
1  function [Diff] = WTVelocityRange(Parameters)
2  %3: ANNUAL ENERGY - loop WTSingleVelocity to find the moments ...
       across the
3  %entire velocity range. Combine this with the frequency ...
       information to get
4  %the AEP. Parameters = [theta0, theta_twist, chord_grad]
5
6  %% Setting as constants%%%%
7  A = 7;
8  k = 1.8;
9  omega = 3.14;
10 chord_mean = 1;
11 TipRadius = 20;
12 RootRadius = 1;
13 B = 3;
14 MinV0 = 5;
15 MaxV0 = 25;
16
17 %% Constants
18 rho = 1.225;          %density of air in kg/m^3
19 Area = (TipRadius^2)*(pi);            %Area in m^2
20
21 %% Calculate total tangential moment for each speed
22 ?_V0 = 1;
23 V0 = MinV0:?_V0:MaxV0;       % Set up wing velocity range in m/s
24 MT = zeros(length(V0), 1);       % Initialise tangential moment ...
       vector for each wind speed
25 MN = zeros(length(V0), 1);       %Initialise normal moment ...
       vector for each wind speed
26 Power = zeros(length(V0), 1);    %Initialise power vector for ...
       each wind speed
27
28 P_speed = zeros(length(V0)-1, 1);            %Initialise vector ...
       of speed probablilities
29 AEP_speed = zeros(length(V0)-1, 1);          %Initialise vector ...
       of annual power at each speed
30 Power_midpoint = zeros(length(V0)-1, 1);     %Initialise vector ...
       of power inbetween speeds
```

```
31 AEP_speed_ideal = zeros(length(V0)-1, 1);
32 Power_ideal = zeros(length(V0)-1, 1);
33
34 %% Calculate power for each speed
35
36 [MT(1), MN(1)] = WTSingleVelocity(V0(1), Parameters(1), ...
      Parameters(2), chord_mean, Parameters(3), TipRadius, ...
      RootRadius, omega, B);
37
38 Power(1) = MT(1)*B*omega;
39
40 for i = 2:length(V0)
41
42     [MT(i), MN(i), ¬, ¬] = WTSingleVelocity(V0(i), ...
         Parameters(1), Parameters(2), chord_mean, ...
         Parameters(3), TipRadius, RootRadius, omega, B);
43
44     Power(i) = MT(i)*B*omega;
45
46     %%%Calculating power and probablilities in the speed ...
         intervals
47     P_speed(i-1) = exp(-(V0(i-1)/A)^k) - exp(-(V0(i)/A)^k); ...
         %Calculate probability of speed between Vi and Vi+1
48
49     Power_midpoint(i-1) = 0.5*(Power(i-1) + Power(i)); ...
                %Power at mid point is average of lower and ...
         upper bounds
50     Power_ideal(i-1) = (16/27) * ...
         0.5*rho*(((V0(i-1)+V0(i))/2)^3)*Area;  %Theoretical ...
         maximum power defined by Betz limit
51
52     AEP_speed(i-1) = Power_midpoint(i-1) * P_speed(i-1) * ...
         8760;          %Calculate predictied anual power ...
         generated at each windspeed range
53     AEP_speed_ideal(i-1) = Power_ideal(i-1) * P_speed(i-1) * ...
         8760;   %Calculate the ideal anual power generated at ...
         each windspeed range
54
55 end
56
57 AEP = sum(AEP_speed);
58 AEP_ideal = sum(AEP_speed_ideal);
59
60 tip_deflection = WTBendingDeflection(Parameters(1), ...
      Parameters(2), Parameters(3));
```

```
disp(strcat('Tip deflection: ', num2str(tip_deflection)));
if  tip_deflection > 3
    AEP = 0;
end
Diff = AEP_ideal - AEP;
assert(Diff > 0, 'Error!Predicted power greater than ideal.')

end
```

D Wind Turbine Velocity Range Function

```
function [x] = TurbineOptimisation()
%%%%Script to find the optimal solution for theta0, ...
   theta_twist and
%chrod gradient

%OPTIMISE FIT - 3 variables: Theta0, Theta_grad and Chord_grad
opts = optimset('fminsearch');
opts.Display = 'iter'; %What to display in command window
opts.TolX = 0.001; %Tolerance on the variation in the parameters
opts.TolFun = 1e7;  %Tolerance on the error
opts.MaxFunEvals = 150; %Max number of iterations

%fminsearch inputs: (function, initial guess, lower limits on ...
   variables,
% upper limits on variables, options)

%Limit for Theta0 is set to +/- 20 degrees
%Limit for rate of twist is set so 45 degrees twis over blade ...
   is the
%maximum
% Limit for chord grad set so there is at least 20cm chord at ...
   root for
% attatchment and so chord is positive at R = 20

[x] = fminsearchbnd(@WTVelocityRange, [5*pi/180 -.3*pi/180 ...
   .0], [-20*pi/180 -2.36*pi/180 -0.105], [20*pi/180 ...
   2.36*pi/180 (8/95)], opts);

%disp(strcat('OPTIMAL VALUES: Theta0 = ', ...
   num2str(x(1)*180/pi), ' Twist Rate = ', ...
   num2str(x(2)*180/pi), ' Taper Rate = ', num2str(x(3))))
```

```
24
25 %% Check bending moment does not Exceed 0.5MNm
26 %WTSingleVelocity inputs => V0, theta0, theta_twist, ...
      chord_mean, chord_grad, TipRadius, RootRadius, omega, B
27
28 %[MT, ¬] = WTSingleVelocity(25, x(1), x(2), 1, x(3), 20, 1, ...
      30, 3);
29
30 %disp(MT)
31
32 %% Compute and display how
33 BetzLimit = 1.99684E+09;
34 diff = WTVelocityRange(x);
35 eff = (BetzLimit - diff) / BetzLimit * 100;
36
37 disp(strcat('Percentage of Betz Limit = ', num2str(eff), ' %'))
38
39
40
41 end
```

E Flow Chart of FEM Solver