TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

School of Engineering
Department of Materials and Environmental Technology
Materials and Processes of Sustainable Energetics

UTILITY OF OPEN SOURCE COMPUTATIONAL TOOLS FOR AERODYNAMIC AND STRUCTURAL BEHAVIOUR ANALYSIS OF SMALL WIND TURBINES IN ACCORDANCE TO RELATED STANDARD

Avatud Lähtekoodiga Arvutusvahendite Kasulikkus Väikeste Tuuleturbiinide Aerodünaamiliseks ja Struktuuriliseks Käitumiseks Vastavalt Valdkonnaga Seotud Standardile

MASTER'S THESIS

Student: Michael Keumatio Lontsie
Student code: 177347KAYM

Supervisor: Ivo Palu, Professor
Co-Supervisor: Drew Gertz, MASc

Tallinn, 2019

AUTHOR'S DECLARATION

Hereby I declare that this master thesis, my original investigation and achievement, submitted for the master degree at Tallinn University of Technology has not been submitted for any degree or examination.

"......." .... 2019.

Author: ....
/signature /

Thesis is in accordance with terms and requirements

"......." .... 2019.

Supervisor: ……..........................
/signature/

Co-Supervisor: ….........................
/signature/

Accepted for defence

"......."....................2019.

Chairman of theses defence commission: ....
/name and signature/

Department of Materials and Environmental Technology

THESIS TASK

Student:	Michael Keumatio Lontsie, 177347KAYM
Study programme,	KAYM09/09 - Materials and Processes for Sustainable Energetics.
main speciality:	Processes for Sustainable Energetics.
Supervisors:	Professor Ivo Palu, +3726203752
	Drew Gertz, MASc, CEO NorthWind Engineering OÜ, +37255650147

Thesis Topic:

Utility of Open Source Computational Tools for Aerodynamic and Structural Behaviour Analysis of Small Wind Turbines in Accordance to Related Standard

Avatud Lähtekoodiga Arvutusvahendite Kasulikkus Väikeste Tuuleturbiinide Aerodünaamiliseks ja Struktuuriliseks Käitumiseks Vastavalt Valdkonnaga Seotud Standardile

Thesis main objectives:

1. The use of freely available simulation tools for analysis of aerodynamic and structural response of wind turbines to wind loading for quality assurance and operation safety of the machine assembly in accordance to the IEC-61400-2 standard for small wind turbines;

2. Demonstration of the importance of standard's load calculation methodologies for turbine integrity.

3. Importance of integrating open source trending programming languages for optimization of simulation processes.

Thesis tasks and time schedule:

No	Task description	Deadline
1.	Internship, data collection, simulation analysis and result processing	26.10.2018
2.	Compilation of literature review	28.02.2019
3.	Compilation of main thesis body and submission	27.05.2019

Language: …………..……………… **Deadline for submission of thesis**: "……."………………..2019

Student: …………………………………. .... "……."………………..2019
/signature/

Supervisor: ………………………………. .... "……."………………..2019
/signature/

Co-supervisor: ……………………… .... "……."………………..2019
/signature/

Consultant: ………………………………. .... "……."………………..2019
/signature/

Table of Contents

ABBREVIATIONS, SYMBOLS USED AND SUBSCRIPTS

SWT – Small Wind Turbine

HAWT – Horizontal Axis Wind Turbine

VAWT – Vertical Axis Wind Turbine

NREL – National Renewable Energy Laboratory

FAST – Fatigue, Aerodynamics, Structural, Turbulence

HAWC2 – Horizontal Axis Wind turbine simulation Code 2nd generation

DLC – Design Load Case

ECD – Extreme Coherent Gust with Direction Change

ECG – Extreme Coherent Gust

EDC – Extreme Wind Direction Change

EOG – Extreme Operating Gust

EWC – Extreme Wind Conditions

EWM – Extreme Wind Speed Model

F – Fatigue

NTM – Normal Turbulence Model

NWC – Normal Wind Conditions

NWP – Normal Wind Profile Model

U – Ultimate

A – rotor swept area [m2]

A_{proj} – component area projected on to a plane perpendicular to the wind direction [m2]

B – number of blades [-]

c – blade chord [m]

C_d – drag coefficient [-]

C_f – force coefficient [-]

C_l – lift coefficient [-]

C_p – power coefficient [-]

C_T – thrust coefficient [-]

D – rotor diameter [m]

e_r – distance from the centre of gravity of the rotor to the rotation axis [m]

F – force [N]

F_{zB} – force in z direction on the blade at the blade root [N]

F_x – shaft axial shaft load [N]

G – acceleration due to gravity: 9,81 [m/s2]

G – multiplier for generator short circuit [-]

L – Lift force [N]

D – Drag force [N]

I_B – blade moment of inertia [kgm2]

L_{rt} – distance between the rotor centre and the yaw axis [m]

L_{rb} – distance between rotor centre and first bearing [m]

m_B – blade mass [kg]

m_r – rotor mass being the mass of the blades plus the mass of the hub [kg]

M_{xB}, M_{yB} – blade root bending moments [Nm]

M_{brake} – torque on the low speed shaft caused by the brake [Nm]

$M_{x\text{-}shaft}$ – torsion moment on the rotor shaft at the first bearing [Nm]

M_{shaft} – shaft bending moment at the first bearing [Nm]

n – rotor speed [r/min]

P – electrical power [W]

P_r – rotor power [W]

Q – rotor torque [Nm]

r – radial coordinate [m]

R – radius of the rotor [m]

R_{cog} – distance between the centre of gravity of a blade and the rotor centre [m]

V – wind speed [m/s]

V_{ave} – annual average wind speed at hub height [m/s]

V_{design} – design wind speed defined as 1,4 Vave [m/s]

V_{eN} – expected extreme wind speed (averaged over 3 s), with a recurrence time interval of N years.

V_{e1} and V_{e50} for 1 year and *50* years, respectively [m/s]

V_{hub} – wind speed at hub height averaged over 10 min [m/s]

V_{tip} – speed of the blade tip [m/s]

W – relative wind speed [m/s]

Δ – range [-]

η – efficiency of the components between the electric output and the rotor (typically generator, gearbox and conversion system) [-]

λ – tip speed ratio [-]

λ_{e50} – tip speed ratio at Ve50 [-]

ρ – air density, here assumed 1,225 [kg/m3]

ψ – Azimuth angle of the rotor (0° is blade vertically up) [°]

ω_n – rotational speed of the rotor [rad/s]

ω_{yaw} – yaw rate [rad/s]

γ_f – Partial safety factor

Subscripts:

ave – average

B – blade

design – input parameter for the simplified design equations

hub – hub height

max – maximum

min – minimum

proj – projected

r – rotor

shaft – shaft

INTRODUCTION

Wind power production is currently one of the most promising electricity generation method among renewable energy sources for reduction of fossil fuel dependence. Its current capacity is estimated to be around 539 GW in 2017 following hydropower production leading with a capacity of 1114 GW, REN21 Report, 2018 [1]. From the first electricity-generating 17 m height wind turbine built in 1888 (Cleveland, United States) with a capacity of 12 kW, the wind industry has noticed a considerable growth in technology to its current world largest GE Renewable Haliade-X 12 MW wind turbine standing 260 m high [2]. This evolution in wind industry has been characterized by the strong desire for optimization of turbine performance through implementation of cutting edge technology to improve the efficiency of turbine electric drive train, the structural design of the rotor enabling the harness of more power from the wind as well as the structural design of the rotor-tower-foundation ensuring turbine stability and integrity.

Wind turbine rotor has faced a tremendous improvement in terms of design and efficiency in the last two decades. This has been made possible through the development of diverse sophisticated computational tools that provide a better understanding of the interaction between environmental conditions and structural design of rotor components. Experience has shown that the understanding of this interaction leads to proper design of blades and tower, capable of not only effectively capturing power from wind but also withstanding harshly fluctuating wind loading. Developed computational tools are therefore actively used in the industry for turbine performance analysis as well as electrical and dynamic load forecasting through numerical simulation analysis. However, the cost of licenses for most of these tools generally constitutes a non-negligible additional financial investment in wind development projects which can appear to be too expensive for small wind turbine manufacturers and interested parties in wind turbine computational analysis.

Consequently, this work investigates on two main subjects: firstly, the use of freely available simulation tools for analysis of aerodynamic and structural response of wind turbines to wind loading for quality assurance and operation safety of the machine assembly in accordance to the IEC-61400-2 standard for small wind turbines; Secondly, the importance of integrating open source trending programming languages for optimization of simulation processes. The work will therefore be structured starting with overview of turbine dynamic and its connection to rotor performance, followed by representation of the standard requirement for load analysis. The second part will be the illustration of major existing simulation tools used in the industry, with a focus on main tools considered in this work. In the following part, will be implemented methods proscribed by the standard for calculation of loads applied on a small wind turbine using open source tools. The fourth part will provide a comparison between those simulation methods in terms of resulting load from simulations. Lastly will be provided a comparative analysis of both open source solutions and available license-based computational tools serving the same purposes. All evidences of conducted investigation, simulation results, generates graphs and written scripts will be given in appendixes.

1. OVERVIEW OF SMALL WIND TURBINE AERODYNAMICS AND STRUCTURAL BEHAVIOUR TO WIND LOADING

1.1. Aerodynamics of Horizontal Axis Wind-Turbines

An exposed wind turbine rotor is subjected to the power of the wind flowing from different direction with diverse turbulence intensities. The kinetic energy of the wind, by and interaction with the rotor-blades, is transferred by the wind turbine drive train to the generator and converted to useful electrical energy. Practical experience has demonstrated that the mean power output and mean loads generated during wind-rotor interaction mainly characterize wind turbine performance and are dependent on the generated aerodynamic forces. Recurrent aerodynamic forces generated by wind shear, angular winds, and rotor revolution as well as randomly fluctuating forces provoked by turbulence and dynamic effects are the source of turbine fatigue and ultimate loads [3]. The conservation of linear momentum theory for an incompressible, one-dimensional, steady flow presented in [3], defines the thrust, *F*, as the force of the wind on an ideal rotor inversely proportional to the change in momentum of air stream and being the key acting element in turbine aerodynamic.

The performance of a wind turbine can be characterized by the way power, torque and thrust vary with wind speed. The power represents the amount of energy captured by the rotor in a given. The torque developed determines the size of the gear box. The rotor thrust has great influence on the structural design of the tower [4]. These performance parameters are highly dependent of blade shape and airfoil characteristics and are determined by the aerodynamic forces generated by the mean wind V_∞. With wind flow, two resulting forces are created around the blade element airfoil: the lift force *L*, perpendicular to the direction of an effective, or relative, wind *W*, and the drag force *D*, parallel to the direction of *W*. See Figure 1.1 for airfoil velocities and forces (lift and drag).

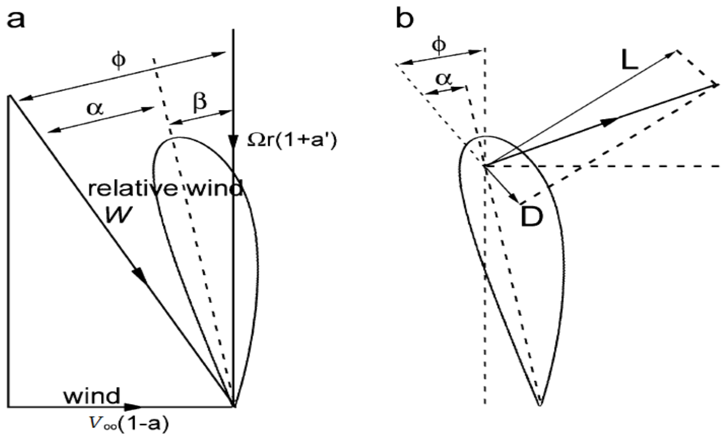


Figure 1.1 Blade element velocities (a) and forces (b) [4].

Where φ - angle between the rotation plane and the relative wind vector, a_r and a_r' - are the axial and angular inductor factors at a radius r defined in [2], α - the angle of attack representing the angle between the chord line and the relative wind W.

The angle of attack is a major factor in pressure distribution across the both top and bottom blade surface. It is considered as one of the two mechanism used in lift generation [5]. The aerodynamic lift and drag forces are then responsible for the rate of change of axial and angular momentum originating the kinematic of blade motion. The blade element momentum theory (BEM) with the use of the conservation of linear momentum provides a detailed demonstration of the relation between lift and drag forces with the blade aerodynamic [3]. The thrust not only influences the rotor aerodynamic but also the structural dynamic of turbine exposed components.

1.2. Structural Dynamic

The wind turbine structural dynamic generally refers to blade and tower deflections due wind loading. The rotor thrust has great influence on the structural dynamic of turbine blades and tower [4]. The figure 1.2 below represents the main turbine top structural dynamic parameters reflecting the degree of freedom of the assembly.

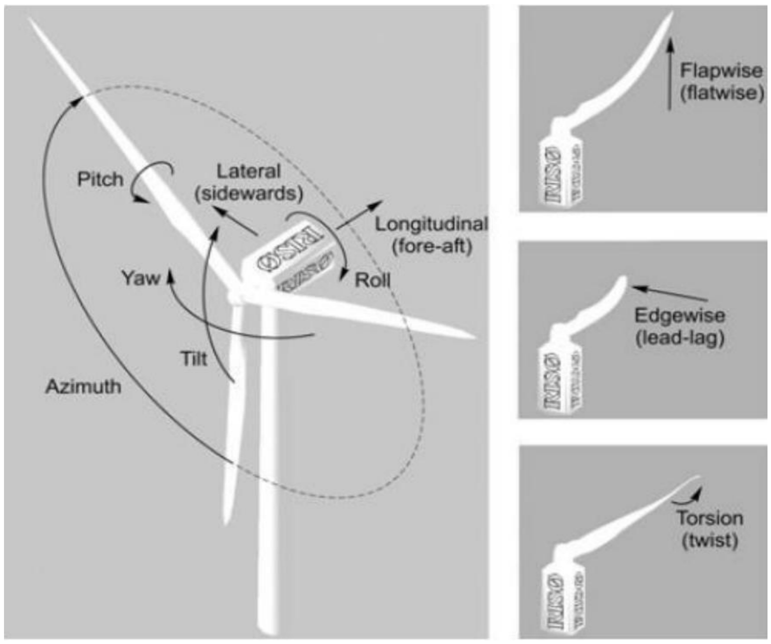


Figure 1.2 Wind turbine top degree of freedom representing blade aero-elasticity [6].

The pitch, yaw, tilt and roll are the main structural parameters considered when defining turbine degree of freedom hugely important for turbine multibody analysis. The pitch is defined as blade rotation about the axis perpendicular to airfoil cross-section. Whereas the yaw is the rotation of the rotor-nacelle assembly around tower vertical axis. On the other hand, the tilt is the rotation of the rotor-nacelle assembly about axis perpendicular to the tower vertical axis. And the roll is denoted as the rotation of the rotor-nacelle assembly about rotor axis. These rotations induce moments considered in turbine design analysis.

The flapwise (flatwise) deflection accounts for the blade flapwise bending moment generated by the thrust force, that causes the blades to bend upwind or downwind. Flapwise deflection is an important design parameter carefully studied during blade design as an over-deflection backward can cause the blades to hit the tower leading to rotor damage [3]. The edgewise (lead-lag) deflection accounts for the blade edgewise bending moment towards the direction parallel to the rotor plane. It is a non-negligible turbine design parameter as it increases the power-producing torque [3].

The torsion (twist) is the torsional deflection of blades about the pitch axis. Torsional deflections are generally not considered for a fixed pitch wind turbine. Whereas for variable pitch wind turbine, they can cause fluctuating loads in the active pitch control mechanism [3].

The lateral (side-to-side) deflection generally accounts for the tower sideward bending moment. This deflection is mostly caused by the constant change in wind direction. Whereas the longitudinal (fore-aft) deflection generally accounts for the tower frontward and backward bending moment mainly induced by the thrust force.

All the above mentioned forces, rotation moments and bending moments characterizing wind turbine structural dynamic have been deeply studied by many researchers in the field of wind energy [6, 8 and 9] for load calculation and turbine safety in operation enabling the limitation of component damages. The safety concern of wind turbines has pushed the International Electrotechnical Commission (IEC) to developed a general standards regularizing turbine load calculations for quality assurance, the implementation of which is demonstrated in this work on a SWT using open source computational tools.

1.3. IEC-64100-2 Standard for Small Wind Turbines

International Electrotechnical Commission (IEC) standard 61400 Part 2 is a version of the IEC-61400 standards that provides engineering design requirements for small wind turbine (SWT) to ensure safety and reliability of operation throughout projected lifetime, withstanding environmental and electrical hazards capable of originating component failure. The standard describes external condition in terms of wind field model to be considered in design depending on the wind farm type. The latest is classified according to wind speed and turbulence parameters used to determine wind fluctuations and extreme wind events that can serve as input into Aeroelastic models allowing to engineers the prediction of the performance and structural loading on turbines for a given site wind condition. The standard describes a SWT as wind machine with a swept rotor area of 200 m^2 or less and classifies on SWT classes basis in terms of wind speed and turbulence parameters [7]. See table 1.1 below for SWT classification.

Table 1.1 Parameters for standard SWT classification [7]

SWT class	**I**	**II**	**III**	**IV**	**S**
V_{ref}, m/s	50	42.5	37.5	30	Values to be specified by the designer
V_{ave}, m/s	10	8.5	7.5	0	
I_{15}	0.18	0.18	0.18	0.18	
a	2	2	2	2	

Where I_{15} is the dimensionless characteristic value of the turbulence intensity at 15 m/s and a is the dimensionless slope parameter. All presented values apply at hub height.

The turbine class is chosen by the manufacturer based on the wind characteristics of the sites they are to be installed. The corresponding basic parameters from the table are then used, together with other secondary parameter which will be presented afterwards, for load calculation.

The standard proposes 3 methods for wind turbine loads assessment for structural integrity and reliability of SWT:

- Simplified load methodology SLM;
- Aeroelastic model;
- Full scale load measurements.

The thesis focuses exclusively on the two first design methods. As for the third method, it is used when the design loads are obtained through direct load measurements, which is not considered in this work. In the following sections will be described the simulation methodology of the first and second methods followed by their implementation on 25 kW sample wind turbine for load calculation.

1.3.1. Simplified Load Methodology (SLM) for load calculation

Design load representation

The model uses a set of developed conservative equations physically explainable for assessment of essential load applied on SWT during normal operation, extreme condition and stand-still conditions. Depending on the external conditions, the standard predefined a set of situations where loads are generated on machine component throughout operation lifecycles. These situations are named Load Cases. The latest are subdivided in groups of design situations and into two main categories depending on the type of analysis to be conducted - Fatigue or Ultimate analysis. Table 1.2 from [7] shows classification of design load cases for SLM according to design situations.

Table1.2. Design Load cases for SLM calculation [7]

Design situation	**Load cases**		**Wind inflow**	**Type of analysis**	**Remarks**
Power production	A	Normal operation		F	
	B	Yawing	$V_{hub} = V_{design}$	U	
	C	Yaw error	$V_{hub} = V_{design}$	U	
	D	Maximum thrust	$V_{hub} = 2{,}5\ V_{ave}$	U	Rotor spinning but could be furling or fluttering
Power production plus, occurrence of fault	E	Maximum rotational speed		U	
	F	Short at load connection	$V_{hub} = V_{design}$	U	Maximum short-circuit generator torque
Shutdown	G	Shutdown (Braking)	$V_{hub} = V_{design}$	U	
Extreme wind Loading	H	Extreme wind loading	$V_{hub} = V_{e50}$	U	The turbine may be parked (idling or standstill) or governing. No manual intervention has occurred.
Parked and fault conditions	I	Parked wind loading, maximum exposure	$V_{hub} = V_{ref}$	U	Turbine is loaded with most unfavourable exposure
Transport assembly, maintenance and repair	J	To be stated by manufacturer		U	

Where F – Fatigue analysis,

U – Ultimate analysis.

For a better description of each design load cases in connection with our particular case studied, some specificities related to the turbine on which the model will be implemented on will be given.

Input Data

Exist 6 main parameters so called first order parameters that describe the turbine and constitute the foundation of the SLM model, all formulas are taken from [7]:

- Design rotational speed or angular velocity

$$\omega_n = \frac{2\pi n}{60} = \frac{\pi n}{30};[rad/s], \quad (1.1)$$

where n is the rotor rotational speed [r/min];

- Design wind speed

$$V_{design} = 1.4V_{ave}; m/s, \tag{1.2}$$

where V_{ave} – is the average wind speed defined by the wind class in Table 9.2 of [7];

- Tip speed ratio

$$\lambda = \frac{V_{tip}}{V_{hub}} = \frac{\omega_n R}{V_{hub}} = \frac{R}{V_{hub}} \frac{\pi n}{30} \tag{1.3}$$

Where V_{tip} – the speed of the blade tip [m/s],

V_{hub} – the wind speed at hub height [m/s],

R – the radius of the rotor [m].

- Design shaft torque

$$Q_{design} = \frac{P_r}{\omega_n} = \frac{P}{\eta\omega_n} = \frac{30P}{\eta\pi n}, [Nm] \tag{1.4}$$

Where P_r – the rotor power [W],

P – the electrical power [W].

- Maximum yaw rate, $\omega_{yaw,\max}$
- Maximum rotational speed $\omega_{n,\max}$.

Description of load cases

- **Load Case A- Fatigue loads on blades and shaft**

The main consideration here is the assessment of loads as peak-to-peak values around the design parameters (wind speed, torque and rotor speed). Fatigue analysis are then evaluated at a range of rotor rpm from 0.5 - 1.5 n_{design} for centrifugal force and from 0.5 - 1.5 Q_{design} for bending moments. In our specific case, loads are quantified between the range of 32.5-97.5 *rpm* and 1901 *N* - 5703 *N*. This enables a sufficient coverage of possible fatigue occurrence situation as the maximum rotational speed of the turbine set by the manufacturer is equal to $\omega_{n,\max} = 70 rpm$.

Blade fatigues are generated by both the centrifugal force and the bending moment components. Centrifugal force at blade root-hub junction is determined using the following equation:

$$\Delta F_{zB} = 2m_B R_{cog}\omega^2_{n,design}, N \tag{1.5}$$

Where R_{cog}, m - the distance from the blade centre of mass to the rotor axis, m_B, kg - blade mass,

Lead-lag or edgewise moment in the direction of rotation and the flapwise moment in the direction of the wind apply to the part of the blade root having the lowest ultimate material strength and are determined using the following conservative equation.

$$\Delta M_{xB} = {Q_{design}}/{B} + 2m_B g R_{cog}, Nm \tag{1.6}$$

$$\Delta M_{yB} = {\lambda_{design} Q_{design}}/{B}, Nm. \tag{1.7}$$

Where *B* – the number of blades.

Rotor shaft are subjected to thrust and moments which are generated at the first shaft bearing adjacent to the rotor and are calculated using the following equations:

$$\Delta F_{x-shaft} = {3\lambda_{design} Q_{design}}/{2R}, N \tag{1.8}$$

$$\Delta M_{x=shaft} = Q_{design} + 2m_r g e_r, Nm \tag{1.9}$$

$$\Delta M_{shaft} = 2m_r g L_{rb} + {R}/{6}\Delta F_{x-shaft}, Nm. \tag{1.10}$$

Where *m_r*– is rotor mass (blades + hub), *kg*,

e_r – rotor eccentricity,

L_{rb} – distance between rotor centre and first bearing.

- **Load Case B- Blade and Rotor Shaft Loads during Yaw**

For turbines equipped with an active yaw system enabling automatic positioning the turbine parallel to wind direction. This mechanism created additional loading due to effect of cyclic gyroscopic forces and moments at blade's root and shaft. Blade root bending moment is found by the equitation:

$$M_{x-shaft} = m_B \omega_{yaw,\max}^2 L_{rb} R_{cog} + 2\omega_{yaw,\max} I_B \omega_n + {R}/{6}\Delta F_{x-shaft}, Nm \tag{1.11}$$

where *L_{rt}* – the distance between rotor centre and yaw (tower) axis,

I_B – blade second moment of inertia.

Paragraph 9.2.2 of [8] defines the physics behind each term in this equation.

Shaft bending moment for a 2 bladed SWT is calculated using the next equation:

$$M_{x-shaft} = 2\omega_{yaw,\max} I_B \omega_{n,design} + m_r g L_{rb} + {R}/{6}\Delta F_{x-shaft}, Nm \tag{1.12}$$

For a 3 or more bladed turbine:

$$M_{x-shaft} = B\omega_{yaw,\max} I_B \omega_{n,design} + m_r g L_{rb} + {R}/{6}\Delta F_{x-shaft}, Nm \tag{1.13}$$

The logical "if/else" statement shall be used in the model considering whether the tested turbine is a 2 or more bladed type.

In both equations the gyroscopic effect creates more than 2/3 of the moment

- **Load Case C- Yaw Error Load on Blades**

Due to yaw error existing in every operating wind turbine, additional flapwise bending moments are created at blade root. The standard proposes a general yaw error of 30° for evaluation of moments in this load case. Flapwise bending moments are determined by the equation:

$$M_{yB} = \frac{1}{8}\rho A_{prof,B} C_{l,\max} R^3 \omega_{n,design}^2 \left[1 + \frac{4}{3\lambda_{design}} + \frac{1}{2}\left(\frac{1}{\lambda_{design}}\right)^2\right], Nm \tag{1.14}$$

Where $C_{l,max}$ – maximum lift coefficient, value 2 will be used for no data availability as specified in the standard,

$A_{proj,B}$ – platform area of blades projected on to a plane perpendicular to wind direction.

- **Load Case D- Maximum Thrust on Shaft**

During operation, turbines are subjected to thrust loads parallel to rotor shaft with a maximum value expressed by the equation:

$$F_{x-shaft} = C_T 0{,}5\rho(2.5 \cdot V_{ave})^2 \pi R^2, N \tag{1.15}$$

where C_T – the thrust coefficient equal to 0,5 for turbines operating at less than 2,5V_{ave}.

- **Load Case E- Maximum Rotational Speed**

During turbine operation at maximum rpm, unbalance rotor as well as centrifugal loads generates additional centrifugal load in blade root and bending moment determined using the following expressions:

$$F_{zB} = m_B \omega_{n,\max}^2 R_{cog}, N \tag{1.16}$$

$$M_{shaft} = m_r g L_{rb} + m_r e_r \omega_{n,\max}^2 L_{rb}, Nm \tag{1.17}$$

- **Load Case F- Short at Load Connection**

Possibility of occurrence of short circuit event are considered as ultimate loading factors. Significantly High moments are created at rotor shaft caused by short circuit torque of the alternator during occurrence of direct electrical short at the output of the turbine or internal short in the generator. These moments can be calculated using the equations:

$$M_{x-shaft} = GQ_{design}, Nm \tag{1.18}$$

Where *G* – short circuit torque factor. Value 2 should be taken in absence of any accurate data as specified by the standard.

$$M_{xB} = {}^{M_{x-shaft}}\!/_{B} + m_B g R_{cog}, Nm \tag{1.19}$$

- **Load Case G- Shutdown Braking**

Shutdown events occasionally occur during turbine operation, caused by multitude reasons like high turbulence wind speed above rated, low wind speed below cut-in, system failure and scheduled maintenance. [10] describes wind turbine shutdown cases and provides a comparative study of shutdown procedure with the turbine dynamic response.

Loads created during this event are highly dependent on the brake moment, for turbine equipped with an electrical or mechanical braking system within the drive-strain. The generated moment on the shaft depends not only on brake moment but also on whether the braking system is applied on high-speed or low-speed shaft. If brake is positioned on the high speed side of the gearbox, brake moment will have to be multiplied by gearbox ratio provided by manufacturer to account for the drive train dynamic, forming the gearbox efficiency:

$$M_{x-shaft} = Gear \cdot M_{brake} + Q_{design}, Nm \tag{1.20}$$

Where M_{brake} – brake moment,

Gear – gearbox ratio, strictly dependent on the existence of gear and the position of the braking system.

The logical “if/else” statement is used in the script to calculate moment on shaft considering existence of the gearbox and the location of brake system on the drive train. In case of the turbine that will be considered as sample for this work, gearbox exist and positioned at the high-speed side of the gearbox, M_{Brake} will then be multiply gearbox ratio.

The blade loading due to shutdown is determined by the equation:

$$M_{xBt} = {M_{x-shaft}}\big/{B} + m_B g R_{cog}, Nm \tag{1.21}$$

- **Load Case H- Extreme Wind Loads during stand still or idling**

A parked wind turbine is generally exposed to severe wind loading especially during extreme wind conditions though not producing power. Loads applied to exposed machine components in this situation are calculated with use of V_{e50} – the 50-year occurrence extreme wind speed. The following equations give the main wind loading on a turbine exposed components caused by drag.

For parked wind turbine, the blade root bending moment is expressed by the equation:

$$M_{yB} = \frac{1}{4} C_d \rho V_{e50}^2 A_{proj,B} R,\ Nm \tag{1.22}$$

Where C_d – drag coefficient, a value of 1,5 to be used according to the standard.

In case of idling, spinning blades without production, the following formula is used:

$$M_{yB} = \frac{1}{6} C_{l,\max} \rho V_{e50}^2 A_{proj,B} R,\ Nm \tag{1.23}$$

Where $C_{l,max}$ – maximum lift coefficient of blades, if no value is available for the turbine, then 2 should be taken.

In our specific case, brake is applied prior parking, meaning the rotor is at a stand-still position. The logical "if/else" statement will be used in the script to refer to the equation defining specificity of our model. Equation 1.22 will be used in occurrence. The same goes for the maximum thrust on blades determined by equation 1.24 below.

For a rotor in revolution, the thrust loading on blades is obtained by the equation:

$$F_{x-shaft} = 0.17 B \lambda_{e50}^2 \rho V_{e50}^2,\ N \tag{1.24}$$

Where λ_{e50} – is the 50-year extreme tip speed ratio at V_{e50}.

$$\lambda_{e50} = \frac{\omega_{n,\max} \pi R}{30 V_{e50}} \tag{1.25}$$

Whereas for a stationary rotor, thrust loading is fund by:

$$F_{x-shaft} = \frac{1}{2} B C_d \rho V_{e50}^2 A_{proj,B},\ N \tag{1.26}$$

The standard requires as well the calculation of the maximum tower bending moment in this case using the parked rotor shaft force equation 1.26. The thrust force on tower should be evaluated using the following equation:

$$F = \frac{1}{2} C_f \rho V_{e50}^2 A_{proj},\ N \tag{1.27}$$

Where C_f – force coefficient which is equal to 1.5. See table 3 of [7],

$A_{proj,T}$ – Tower projection area on the plane perpendicular to wind direction. The projection area at component most unfavourable position is considered.

- **Load case I – Parked wind loading, maximum exposure**

The load considers a possible occurrence of failure in yaw mechanism which will lead to exposure of turbine to extreme wind speed in turbine coming from all possible directions. Turbine will then undergo severe loading if it is at its most vulnerable position at maximum $A_{proj,B}$. This load case is not considered in our case due to very low probably of occurrence of such failure. However, paragraph 7.4.10 of [7] describes in details this load case.

- **Load case J – Transportation, assembly, maintenance and repair**.

Stresses faced by turbine during the operations should be considered by the manufacturer. These stresses are characterised by gravity loads, loads caused by special installation tools, wind loads during installation or maintenance as well as load on a tilt up tower during erection and putting to foundation. These loads being too specific with extremely low effect on general component stress are not considered in this simulation.

The next step consists of evaluating the equivalent stresses according to material strength in order to be able to later determine the strength limit of each component compare to the fatigue load lifecycle and ultimate stress faced by the turbine rotating components in normal operation as well as in described above special load cases.

Determining equivalent stresses on components

For stress calculation, most important load carriers are considered, generally blade roots and main shaft. Equivalent Stresses on these components are a combination of individual forces and moments calculated in each load cases. The resulting stress values are compared to with the allowable limit for material stress. Table 1.3 from [7] gives general formulas for equivalent stress calculation.

Table 1.3 Equivalent stress [7]

	Circular blade root	**Rectangular blade root**	**Root shaft**
Axial	$\sigma_{zB} = \frac{F_{zB}}{A_B}$	$\sigma_{zB} = \frac{F_{zB}}{A_B}$	$\sigma_{x-shaft} = \frac{F_{x-shaft}}{A_{shaft}}$
Bending	$\sigma_{MB} = \frac{\sqrt{M_{xB}^2 + M_{yB}^2}}{W_B}$	$\sigma_{MB} = \frac{M_{xB}}{W_{xB}} + \frac{M_{yB}}{W_{yB}}$	$\sigma_{M-shaft} = \frac{M_{shaft}}{W_{shaft}}$
Shear	Negligible	Negligible	$\tau_{M-shaft} = \frac{M_{x-shaft}}{2W_{shaft}}$
Combined (axial + bending)	$\sigma_{eqB} = \sigma_{zB} + \sigma_{MB}$		$\sigma_{eq-shaft} = \sqrt{\left(\sigma_{x-shaft} + \sigma_{M-shaft}\right)^2 + 3\tau_{M=shaft}^2}$

Stress level calculation takes into consideration several important factors [8]:

- Stress variation within the component
- Stress concentrations
- The direction and size of the resulting load or stress
- Variations in component dimensions and thickness
- Component surface treatment
- The type of loading on the component
- Any manufacturing effects on the components such as welding, machining etc.

Additional input data is required for this equivalent stresses on material components:

- Cross section area of shaft A_{shaft} and blade root A_B in m^2. Values obtained from component physical characteristics.

Second moment of inertia $I_{x-shaft(xB)} = \pi \cdot d^4_{shaft(B)} / 64,\ m^4$ (1.28)

Where $d_{shaft(B)}$ – diameter shaft or blade.

- Section modulus of shaft and blades,

  $W_{shaft} = 2 \cdot I_{x-shaft} / d_{shaft}, m^3$ and $W_B = I_B / C_B, m^3$, (1.29)

  where C_B – distance from blade centroid to maximum stress point. Its values depending on x or y- axis are obtained from blade's CAD model.

The values of above mentioned parameters for fatigue and equivalent stress calculation are displayed in Appendix 1.1. Using this data and formula from table 1.3, equivalent stresses where computed in accordance with the load cases.

1.3.2. Aeroelastic model for load computation

The design model implies analysis of turbine loads from aeroelastic simulation modelling at different operating conditions prescribed by the standard. These operating conditions also called design load cases (DLC) describe environmental and electrical conditions to which are exposed wind machine during life cycles. The model proposes a more realistic turbine dynamic analysis method describing in details all possible real-time scenarios occurring during turbine exposure. It recommends analysis to be conducted in a wide range of wind speed, generally from cut-in to cut-out with a wind speed step varying from 1-3 *m/s* this to ensure all possible loads are located.

Each design situations are subdivided into design load cases as described in table 1.4 below taken from the standard. Like in the previous model, the type of analysis remains similar depending on load cases they are applied to, fatigue loads analysis for evaluation of fatigue stresses and ultimate load analysis for evaluation of loads that can exceed maximum material strength, cause tip deflection or threaten turbine stability.

Table 1.4. Aero-elastic simulation design load cases (DLC) [7]

Design situation	**DLC**	**Wind condition**	**Other conditions**	**Type of analysis**
Power production	1.1	NTM $V_{in} < V_{hub} < V_{out}\ or\ 3 \cdot V_{ave}$		F, U
	1.2	ECD $V_{hub} < V_{design}$		U
	1.3	EOG$_{50}$ $V_{in} < V_{hub} < V_{out}\ or\ 3 \cdot V_{ave}$		U
	1.4	EDC$_{50}$ $V_{in} < V_{hub} < V_{out}\ or\ 3 \cdot V_{ave}$		U
	1.5	ECG $V_{hub} = V_{design}$		U

Power production plus occurrence of fault	2.1	NWP $V_{hub} = V_{design} or V_{out} or 2.5 \cdot V_{ave}$	Control system fault	U
	2.2	NTM $V_{in} < V_{hub} < V_{out}$	Control or protection system fault	F, U
	2.3	EOG1 $V_{in} < V_{out} or 2.5 \cdot V_{ave}$	Loss of electrical connection	U
Normal shutdown	3.1	NTM $V_{in} < V_{hub} < V_{out}$		F
	3.2	EOG1 $V_{hub} = V_{out} or V_{\max, shutdown}$		U
Emergency or manual shutdown	4.1	NTM to be stated by manufacturer		U
Extreme wind loading (standing still or idling; or spinning)	5.1	EWM $V_{hub} = V_{e50}$	Possible of electrical power network	U
	5.2	NTM $V_{hub} < 0.7 \cdot V_{ref}$		F
Parked and fault condition	6.1	EWM $V_{hub} = V_{e1}$		U
Transport, assembly and repair	7.1	To be stated by manufacturer		U

Where F – fatigue load analysis,

U – ultimate load analysis.

Power production: DLC 1.1-1.5

Power production assumes the turbine is in availability status and connected to the electrical grid load. Aerodynamic and structural loads are then evaluated taken into consideration several factors specified by the turbine manufacturer like rotor imbalance, yaw misalignment, maximum mass, blade pitch deviations, blade twist deviations as well as control system tracking errors.

In power production status the probability of occurrence of critical conditions generating severe loads are non-negligible. They are therefore taking into consideration in load evaluations. Such conditions are described in chapter 7.5 of [7] as:

- Atmospheric turbulence generating loads in DLC 1.1
  Here, loads are evaluated in a wind profile denoted as Normal Turbulence Model – NTM. It expresses the stochastic variations in wind speed and wind direction in a set average of 10 minutes according to the characteristic of the value of the standard deviation σ_1 and the turbulence scale parameter Λ_1. The latest parameters are obtained using the following equations:

$$\sigma_1 = I_{15}(15 + aV_{hub})/(a+1) \tag{1.33}$$

  Where *a* – the dimensionless slope parameter for the expression.

I_{15} – is the dimensionless characteristic value of the turbulence intensity at 15 m/s, where 0,18 is the minimum value that shall be used.

$$\Lambda_1 = \begin{cases} 0.7 z_{hub} \; for\, z_{hub} < 30m \\ 21m \quad for\, z_{hub} \geq 30m \end{cases} \tag{1.34}$$

The design requirement implies that load should be evaluated within the wind speed range as stated in table 1.4. In the aeroelastic model the range considered in this case is from 3 *m/s* to 25 *m/s*.

- Potentially critical transient cases with extreme conditions are considered DLC 1.2-1.5
  Extreme conditions are generally related to the variation of wind speed in terms of gusting, drastic direction change, wind turbulence intensity generating extreme wind loading on SWTs. In this design situation, these extreme conditions are taken into consideration and briefed below.

ECD – Extreme coherent gust with direction change: A gust is a sudden increase in wind speed. According to U.S. weather observing practice, gusts are observed when peak in wind speed grows up to 8.5 m/s and the variation between high peaks and low peak is at least 5 m/s. The duration gust event is most often lower than 20 seconds.

DLC 1.2 considers the increase in wind speed occurring simultaneously with the change of a direction θ_{cg} defined by the relation:

$$\theta_{cg} = \begin{cases} 180^\circ \; for V_{hub} < 4m/s \\ \dfrac{720^\circ}{V_{hub}} \; for\, 4m/s \leq V_{hub} \leq V_{ref} \end{cases} . \tag{1.35}$$

This change in direction of the wind or yaw error generates specific load which have to be inserted in the design increasing the efficiency of the model. Wind turbines are recommended to be tested at V_{hub} lower than V_{design} = 10.5 m/s. In the model V_{hub} of 9 m/s and 11 m/s were considered for more accuracy.

EOG – Extreme operating gust: This wind condition characterises DLC 1.3 accounting the recurrence period of *N* for such event. This implies the consideration of a transient extreme event that usually occur once in *N* years. Therefore, the gust wind V_{gustN} will depend on the recurrence period *N* and giving by the formula:

$$V_{gustN} = \beta \left(\frac{\sigma_1}{1 + 0.1\left(\dfrac{D}{\Lambda_1}\right)} \right) \tag{1.36}$$

Where σ_1 – the standard deviation described above,

Λ_1 – the turbulence scale parameter according to equation 2.34,

D – the diameter of the rotor,

β_1 = 4.8 for *N*=1 year and 14.0 for *N*=50 years.

The wind speed in this case is defined for the same recurrence of *N* years by the equation:

$$V(t) = \begin{cases} V(z) - 0.37V_{gustN} \sin(3\pi t/T)(1-\cos(2\pi t/T)) & for\, 0 \le t \le T \\ V(z) & for\, t < 0\, and\, t > T \end{cases} \tag{1.37}$$

Where *V(z)* – defined in equation 1.41,

T = 10.5 for *N*=1 year and 14.0 for *N*=50 years.

In the model the V_{hub} range considered in this case is from 3 m/s to 25 m/s at a gust wind speed of V_{gustN} with a recurrence period of 50 years.

The term one-year and 50-years extreme wind is described in the Wind Energy Handbook by David Sharpe [9] as the expected one-time occurrence of the considered severe wind transient in a period of one year or 50 years. Subsequently, this event has and unknown occurrence time that needs to be considered during design of wind machine for quality assurance.

ECG – Extreme coherent gust: considered as wind condition in DLC 1.5 implies a change in wind speed with a magnitude of V_{cg} = 15 m/s in a rise time *T* = 10 s as described in the formula below:

$$V(t,z) = \begin{cases} V(z) & for\, t < 0 \\ V(z) + 0.5 \cdot V_{cg}(1-\cos(\pi t/T)) & for\, 0 \le t \le T \\ V(z) + V_{cg} & for\, t > T \end{cases} \tag{1.38}$$

Such high increase in wind velocity incontestably generates extreme loads on machine component which therefore needs to be considered in the design, increasing the efficiency and accuracy of the model. In the model V_{hub} = 11 *m/s* was considered for load evaluation at this condition.

EDC – Extreme direction change: This condition considers the change in wind direction with yaw misalignment considered in DLC 1.4 in terms of magnitude ϑ_{eN} for a recurrence period of N years using the following relation:

$$\theta_{eN}(t) = \pm\beta \arctan\left(\frac{\sigma_1}{V_{hub}\left(1 + 0.1\left(\frac{D}{\Lambda_1}\right)\right)} \right) \tag{1.39}$$

Where ϑ_{eN} – angle of direction change limited to the range of ±180°,

Λ_1 – the turbulence scale parameter according to equation 1.34,

D – the diameter of the rotor,

β = 4.8 for *N*=1 year and 6.4 for *N*=50 years

The direction change is characterized by a transient for a recurrence period of N years, *$\vartheta_N(t)$*. The transient is the angle by which the wing velocity changes direction in a time *t*. Equation 1.40 from [7] defines its dependence over time *t* as:

$$\theta_N(t) = \begin{cases} 0 & for\, t < 0 \\ 0.5 \cdot \theta_{eN}(1 - \cos(\pi t / T)) & for\, 0 \le t \le T \\ \theta_{eN} & for\, t > T \end{cases} \tag{1.40}$$

Where T = 6 s is the duration transient at this case condition.

A graphical representation of wind behaviour characterising EOG and ECG are displayed in figure 3.11 and figure 3.13 respectively. Other characteristic graphs describing wind direction change for other transient events are illustrated in [7].

Power production plus occurrence of fault: DLC 2.1-2.3

During power production, the probability of occurence of fault in the control system and in electrical system is existant with a frequency depending on turbine quality and wind farm characteristics. The design load case is subdivided into 3 different cases according to wind conditions:

- DLC 2.1, where loads generated during fault in control system considered as normal event are analyzed. The wind condition is characterized as Normal Wind Profile – NWP. The turbine operates at an average wind speed that depends on the height, z, above the ground level. The wind profile is then defined by the power law:

  $$V(z) = V_{hub}(z / z_{hub})^{\alpha} \tag{1.41}$$

  Where α – is the power law or the wind shear coeffitient generally assumed to be 0.2.

  The turbine here is recommended to be tested at wind speed V_{hub} = V_{design} = 10.5 m/s or at V_{hub} = 2.5V_{ave} = 18.7 m/s. In the model V_{hub} of 18.7 m/s is considered as it gives more probability of occurrence of such fault compare to V_{design}.
- DLC 2.2, where loads generated due to faults in protection or internal electrical systems not significant to cause turbine quick shutdown are studied. In the model the load case will be analysed upon NTM at a wind speed standard range of cut-in to cut-out, 3 m/s to 25 m/s.
- DLC 2.3, where the loads generated during shutdown due to loss of electrical connection combined with one-year extreme operating gust EOG_1 is evaluated. In the model, this load cases considers the existence of 3 fault control systems for passively controlled turbines, the furling system where the nacelle of the turbine is turned off the wind direction, blade pitch system and the tip-brake system to reduce wind loading. The 25 kW burbine's blades are equiped with tip-brake system cut-off of rotational motion of the rotor during turbine shutdown. Therefore, the model considered the existence of such braking system.

Normal Shutdown DLC 3.1 and 3.2

Normal generator shutdowns are usually applied during experience of extreme wind loading described by the transient situations above causing severe component loading. The load case DLC3.1 analysed generated loads at a NTM wind condition at wind velocity range similar to load case using the same turbulence model.

As for DLC 3.1, the shutdown is consdered to occur in combination with one-year extreme operating gust EOG_1. Considering the fact that this case normally occur at high wind speed, it is required by the standard to evalute loads at a $V_{hub} = V_{max, shutdown}$ = 25 *m/s* which correspond to the rated wind speed at which begins the overheating of the generator. Therefore, it is been used as wind speed for laods evalution in the model.

Emergency or manual shutdown DLC 4.1

During emergency or manual shutdown, brakes are applied to the high speed shaft generally between the gearbox and the generator generating additional stress to rotating components. This situation is considered in the model for loads evalution within the NTM wind condition. Here, the standard gives the choice to manufacturer to choose the wind velocity range at which the turbine should be tested. In the built model wind speed range of cut-in to cut-out, 3 m/s to 25 m/s is considered.

Extreme wind loading during rotor stand-still or rotation DLC 5.1 and DLC5.2

In this design situation, loads applied to turbine components during stand still or idling or spinning are evaluted similar to the previous model. It incloses 2 different load cases:

- DLC 5.1 which combines the stantionary or rotational non-power production status with the existence of Extreme Wind speed Model - EWM described in the standard.
  According to EWM wind condition, load calculation in this case should be examined at a wind speed V_{hub} equal to the 50 year extreme wind speed V_{e50} depending on the reference wind speed V_{ref} and given by the equation:
  $$V_{e50}(z) = 1.4 \cdot V_{ref}(z / z_{hub})^{0.11}, \tag{1.42}$$
  Where 1.4 – is the gust factor at hub height z_{hub}
- DLC 5.2, where loads are evaluated according to NTM wind profile described above.

Parked plus fault conditions DLC 6.1

In this situation, the turbine is parked due to some electrical network fault yet exposed to wind loading that can create fatigue damage considered in the model.

The standards requires the wind condition to be an EWM, load calculation in this case are within a wind speed *V_{hub}* equal to the one year extreme wind speed, V_{e1}, depending on the reference wind speed, V_{ref} , and given by the equation:

$$V_{e1} = 0.75 \cdot V_{e50} , \tag{1.43}$$

Like the SLM, the aeroelastic model also comprises the design load cases which takes into account loads generated during transportation, assembly, maintenance and repair- DLC 7.1. These loads are to be considered by the manufacturer during load determination. However, in our specific case, this was not included into the modelling as it requires datas from these operations.

All the above described design situations characterised by illustrated formulas together with related wind conditions are intergated into a model simulator that will be presented in the following part.

2. WIND TURBINE DESIGN AND ANALYSIS SIMULATION TOOLS

Nowadays, the use of software and computer based tools for design purposes at research and development stage of any engineering concepts as well as during their implementation has become unavoidable. The main aim being to provide to designer with the possibility to model a concept in a virtual or real environment through prototyping, integrating all possible parameters partly or fully characterizing the concept so for it to be close enough to reality. Faults, defects, dysfunctions in concept system and many other undesirable outputs can be foreseen and easily adjusted prior to final release to end users.

2.1. General Overview of Predominant Wind Turbine Simulation Tools

Design and manufacturing of small wind turbines requires the use of multitude computer integrated software and tools. They differ from each other depending on the type of analysis to be conducted and also on turbine components considered.

For structural analysis on robustness and performance of machine components and assemblies, computer-aided engineering (CAE) tools are generally used. They enable conduction of stress analysis using Finite Element Analysis (FEA), thermal and fluid flow analysis with Computational Fluid Dynamics (CFD), Multibody dynamic (MBD) and Optimizations [11]. Furthermore, CAE tools encompass Computer-Aided Design (CAD) software that enables the geometrical representation of machine components and assemblies that can be used as object for the analysis mentioned above. Solidworks by Dassault Systemes known to be one of the best for 3D CAD modelling, AutoCAD, Kompas 3D (software good for managing project of thousands of sub-assemblies, parts, and standard library products), Fusion 360 (3D CAD/CAM tool from AutoDesk) and CATIA are the most popular CAD software used in the wind industry [12].

Analysis of wind turbine structural and aerodynamic response to wind loading generally includes CFD, MBD and FEA analysis. With the increase in necessity of better analysis results with sufficiently high reliability, many sophisticated and highly performant analysis tools have been developed so far. Depending on the type of analysis, here are presented and briefly described some of the most popular simulation software and tools:

- QBlade – an open source HAWT and VAWT wind turbine rotor design and performance simulation software that gives the possibility to users to design custom airfoils and compute their performance directly integrating them into a rotor. It is a recommended software for teaching, as it provides a comprehensive design and simulation capabilities for turbine rotor, exposing all the fundamental relationships of design concepts and turbine performance in a simplified user friendly interface [13].
- Ashes – a software that executes integrated analysis of onshore and offshore wind turbines. It is a suitable tool for students and teachers providing them with an insight on design processes using a user friendly interface, real-time 3D visualization wind and wave loads and the resulting response of the wind turbine, wind turbine model templates with customisable commonly used airfoils data characterizing turbine blades as well as advance integrated analysis for wind loads, sea waves gravity, buoyancy and generator loads assessment [14].
- HAWC2 - (Horizontal Axis Wind turbine simulation Code 2nd generation) is an aeroelastic code intended for calculating wind turbine loading response in time domain.
- FAST – NREL's is a primary CAE tool for simulating the coupled dynamic response of wind turbines that joins aerodynamics models, structural (elastic) dynamics models, control and electrical system (servo) dynamics models and hydrodynamics models for offshore structures to enable coupled nonlinear aero-hydro-servo-elastic simulation in the time domain.

Both FAST and HAWC2 were developed by researchers and developers in the wind industry for the computation of aerodynamic and structural behavour of onshore and offshore wind turbines to wind loading. As noted from definition, FAST and HAWC2 possess similar objectives though developed by distinct parties, Oregon State University and Technical University of Denmark DTU respectively. Both codes were developed to overcome the complexity of the aeroelastic model, incorporating a large variaty of wind field and turbine parameters to be taken into consideration for design purposes. However, this work emphasizes on the use of FAST for development of the aeroelastic model for wind turbine load calculation for quality assurance. A comparative analysis on the choice of FAST over HAWC2 will be developed subsequently following the model implementation.

2.2. NREL FAST Simulation Code

2.2.1. Introduction to FAST

FAST – defined as Fatigue, Aerodynamics, Structures and Turbulence is a comprehensive aeroelastic simulator built to predict both extreme and fatigue loads generated on a two or three-bladed horizontal axis wind turbine during operation lifecycle. It is a code developed and funded by U.S. Department of Energy under the National Renewable Energy Laboratory (NREL). See FAST 7 User's Guide [15] for more detailled information about the Code. Developped since from 2003, it has noticed several improvements to the 3 current versions used nowadays for simulations: FAST Version 7; FAST Version 8 and OpenFAST (the latest).

The code built-in has the capacity to model the dynamic reponse of horizontal axis conventional wind turbines taking into consideration different configurations like rotor-furling, tail-furling, tail aerodynamics and passive/active blade pitch control system, features which are fundamental during analysis of wind turbines response. It was tested and evaluated by the Germanisher Lloyd WindEnergie , see Germanisher Lloyd – Guideline for the Certification of Wind Turbines [16] for more information, who concluded on the code suitability for calculation of onshore wind turbine loads for design and certification purposes.

The FAST model is a combination of modal and multibody dynamics formulation. It relates for two-bladed turbines nine rigid bodies (support platform, nacelle, armature, gears, hub...) and four flexible bodies (tower, two blades, and drive shaft) through 22 degrees of freedom (DOFs). Whereas for three-bladed HAWT, it relates 24 DOFs from which six DOFs related to the translational (surge, sway, and heave), rotational (roll, pitch, and yaw) motions of the support platform relative to the inertia frame, four DOFs accounting for tower motion, one DOF accounting for yawing motion of the nacelle... More information about turbine DOFs considered in the code can be found in FAST version 7 user guide [15]. A combination of the available DOFs and features are integrated in wind turbine analysis model depending on the configuration of the machine.
The described DOFs indicates the detail consideration, into the load analysis, of main components of a machine exposed to the power of the wind. This approach aims to provide a more realistic representation of the turbine in virtual environment with the combination of created wind events, close enough to reality predefined by a set of paramerters and factors.

These parameters and factors describing turbine components and environmental conditions are integrated into FAST in form of modules. Modules are sub-parts of FAST code structure corresponding to different physical domains of coupled aero-hydro-servo-elastic solution, most of which are separated by spatial boundaries hence interconnected.

The figure 2.4 below shows the main difference between the latest versions of the model in terms of built-in modules stating the spatial dynamic boundaries related to each modules. These modules refer to the integration into the code of specific environmental conditions and wind turbine structural properties as indicated by their denominations.

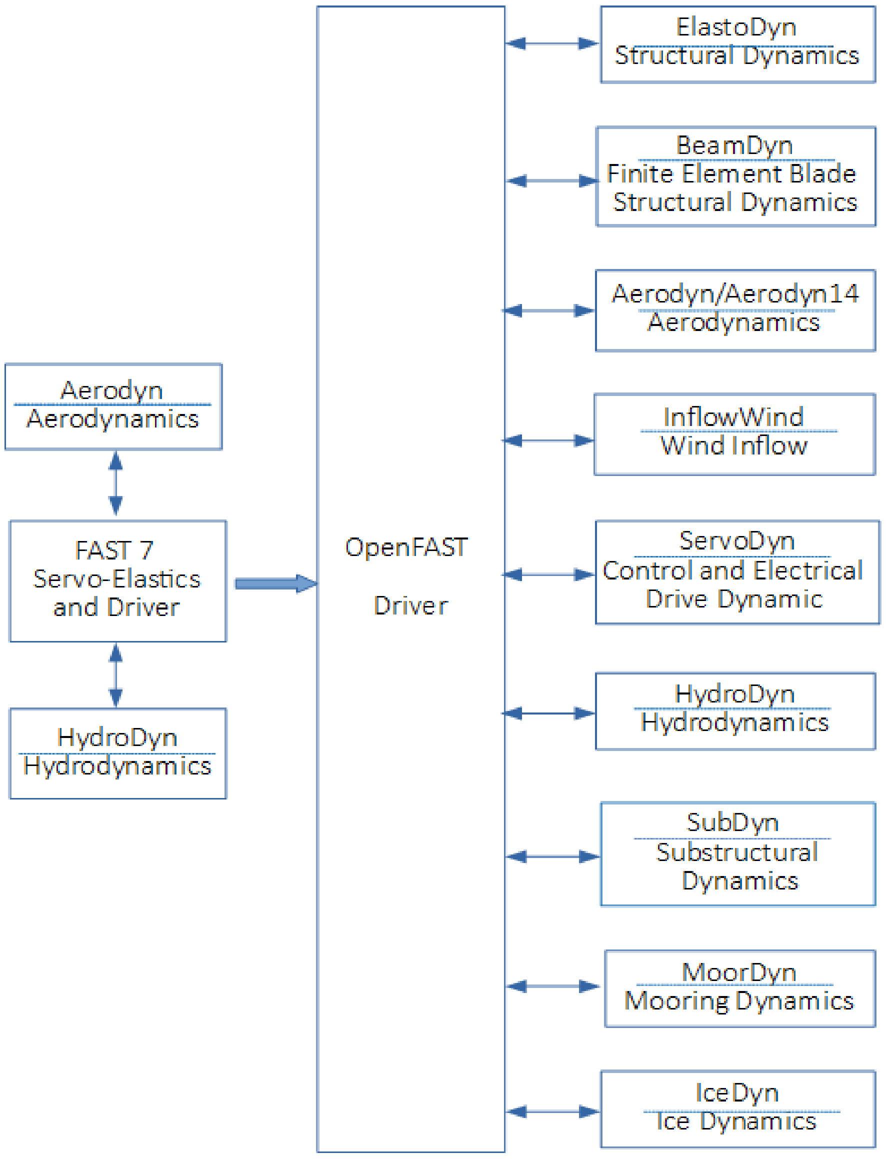


Figure 2.4. Architecture of FAST 7 and OpenFAST

The designation of these modules reveales that FAST is designed to intergrate analysis of not only onshore but also offshore wind turbines taking into account substructural configurations and water conditions with modules like hydrodyn, Subdyn, Icefloe, Moordyn, Icedyn. Each single module is attached to a specific control entity depending on the wind turbine type. Figure 2.5 presents an example of FAST control entities for floating systems.

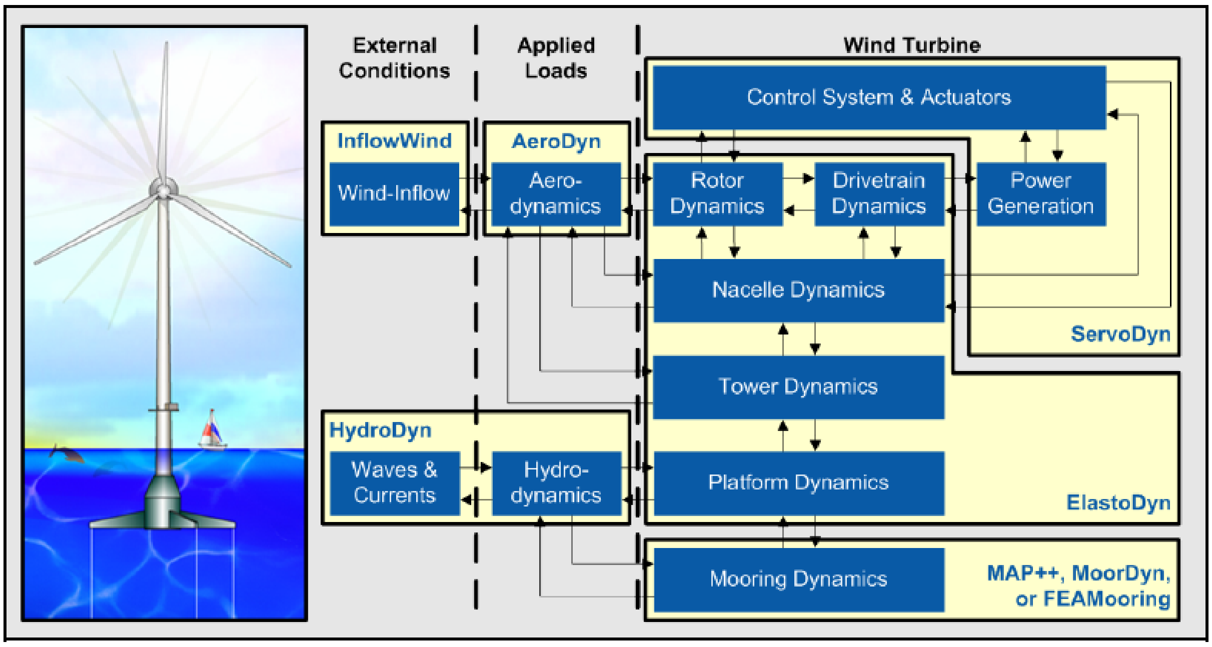


Figure 2.5. FAST control entities for floating wind turbine [7].

25 kW Turbine being an onshore machine, hydrodyn, MoorDyn as well as the platform dynamics in Elastodyn won't be used in the model. The control entities for such system is as represented in figure below.

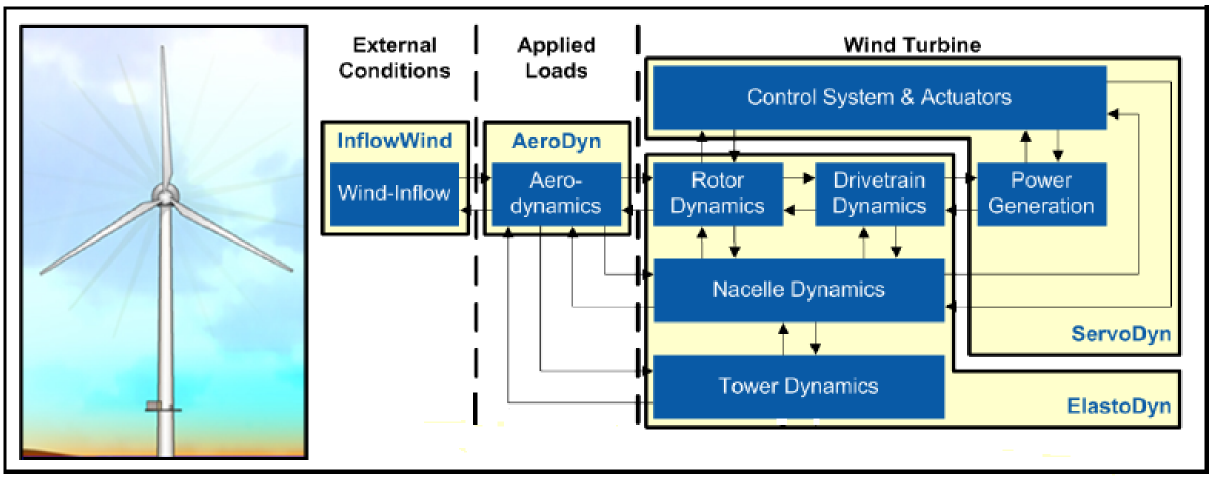


Figure 2.6. FAST control entities for onshore wind turbine [7].

2.2.2. FAST operating mode

FAST operates in a simulation mode defined as the time-matching of non-linear equations of motion. Aerodynamic and structural response of wind turbine to wind conditions are evaluated in time domain. The characteristic of this analysis mode is the output of the simulation which is time-series numerical data points representing the aerodynamic loads as well as loads behind the deflections of structural components of the machine. These outputs are then used for prediction of both fatigue and extreme loads for HATWs.

FAST simulation analysis are run using the open source distributed executable program file from Windows operation system command lines. All modules' paramerters are then introduced into the simulation through inputs files, with each module input files containing information related to each control volumes. The following figure represents the schematic setup of FAST simulation analysis mode of operation.

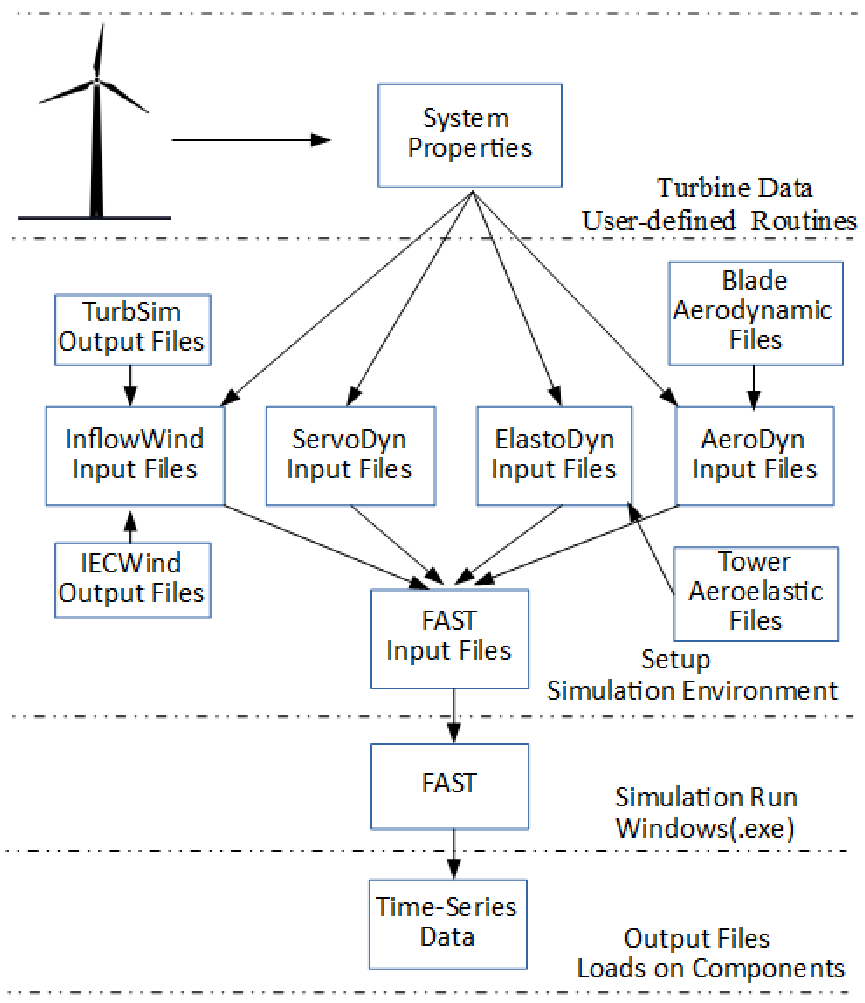


Figure 2.7. FAST mode of Operation

2.2.3. Description of FAST module input files

At system properties level as presented in figure above, are gathered all data related to turbine configuration and wind profile. This data is used for setup of simulation environment according to various modules input files.

InflowWind – here are included wind distribution parameters characterising the wind profile. Its main feature is the wind type or wind condition. FAST proposes through this modules, different wind type, three of which are communly, the steady wind conditions (where only wind speed and reference height to horizontal wind speed values are needed for simulations), the uniform wind (used for analysis of extreme wind conditions requiring the use of a specific wind file generated by IECWind auxilliary to FAST program) and the full-field turbulence wind flow used for analysis of NTM wind condition with the use of specific wind file generated by TurbSim.

IECWind is a utility program used to create wind files for AeroDyn-based programs. It creates wind files that model the extreme conditions (ECD, ECG, EDC, EOG, NWP and EWM) described above, giving the ability to wind turbine designer to run design code simulations of advanced turbine design with simulated transient events in wind propagation, intergrating crucial fluid dynamic features known to unfavorably affect turbine aeroelastic response and loading.

The TurbSim is a stochastic inflow turbulence code developed to provide numarical simulation of a full-field flow containing burst of coherent turbulence reflecting the proper spatiotemporal turbulent velocity field. Its aims is to provide the wind turbine designer with the possibility to run design code simulations of advanced turbine designs with simulated inflow turbulence environments that associate many of the essential fluid dynamic features known to adversely affect turbine aeroelastic response and loading. See Overview of the TurbSim Stochastic Inflow Turbulence Simulator in [17].

The Aeroelastic model in this work includes the latest wind type. For each simulation run, InflowWind modules will require the generation of corresponding wind file from either IECWind or TurbSim. Detail view on InflowWind input file with assisgned feautures for simulation can be seen in Appendix 3.1.

AeroDyn (Aerodynamics) – it takes in features and parameters defining the machine aerodynimics. The main features considered here are environmental condtions (air density, kenematic air velocity, speed of sound, atmospheric pressure values), blade parameters, proberties and airfoils data and tower influence and dynamic. The modules here requires detailled blade data, mainly pre-generated by the designer blade airfloil files containing aerodynamic parameters of each airfloil and blade aerodynamic files containing distributed blade aerodynamic properties. These blade data files are specific to each wind turbine and therefore used in each simulation run as constant parameters. Detail view on Aerodyn input file with assigned features for simulation can be seen in Appendix 3.2.

ElastoDyn – takes in general turbine configuration features and simulation output parameters, the main of which are Turbine DOFs as described in above, turbine configuration (number of blades and rotor dimensions), Blade structural poperties (number of blade nodes and file containing structural properties), rotor properties (rotor teeter spring/damper model, damper position, damping constant

...), drivetrain parameters (gearbox efficiency, gearbox ratio...), tower structural poperties (number of tower nodes and file containing structural properties) and output parameters. The latest gives the possibility to the designer to setup the representation of the simulation output files specifically. Output parameters are nothing else than turbine evaluated parameters (rotor torque, power...) and the calculated loads resulting from simulation generally forces and moments.
The Elastodyn files for both blade and tower are pre-generated file providing the module with a detailled blade structural parameters necesserary for simultion. Detail view on Elastodyn input file with assigned features for Aeroelastic model can be seen in Appendix 3.3.

ServoDyn – is the simulation control module that encompasses all turbine control systems and features essential for high simulation accuracy. It therefore includes features related to Pitch control (blade angle of attack control system, can be unexistant or any user-defined routine), generator and torque control (where parameters like variable-speed control mode, generator efficiency, generator model, generator speed and torque control and many orther generator control parameters are set), simple induction generator features (for control of turbines with simple generator mode type depending also on wether it is a variable-speed or a constant speed), high-speed shaft brake (HSS-brake) (for the control of virtual turbine braking system), tuned mass damper (to initate computation of nacelle and tower mass damping by the device is applicable to the turbine). Detail view on Servodyn input file with assigned features for Aeroelastic model can be seen in Appendix 3.4.

FAST input File and simulation Run

All the aboved presented module with allocated features are called-in in one single input file representing the main FAST input file. This file was built with a specific extension (.fst) indicating it functionality. It contains features related to Simulation control (gives the designer the ability to set-up parameters like total simulation run time and time step of simulation run), features switches and flags (to enable or disable the computation of some modules), input files (where the different generated modules input files are called-in to intergrate the simulation run bringing required parameters values for FAST code execution), output (for setting parameters related to generation the output file like the time step for tabular output in second, time to begin printing in text tabular output file in seconds, format for tabular output file which can be either text file with an exetension of ".out" or a binary file ".outb" and others. Detail view on FAST input with assigned features for the Aeroelastic model can be seen in Appendix 3.7. FAST input file contains also features for control of the Linearization operation mode which is not in our line of concern.

Using available turbine data, the designer sets up the simulation environment by building up the described aboved inputs files in accordance with the simulation to be ran. After building up the final

"FAST.fst" input file, the simulation is ready to be ran from Windows command line through FAST executable program FAST.exe for FAST 7, FAST_x64.exe for FAST 8 or OpenFAST_x64.exe for OpenFAST versions.

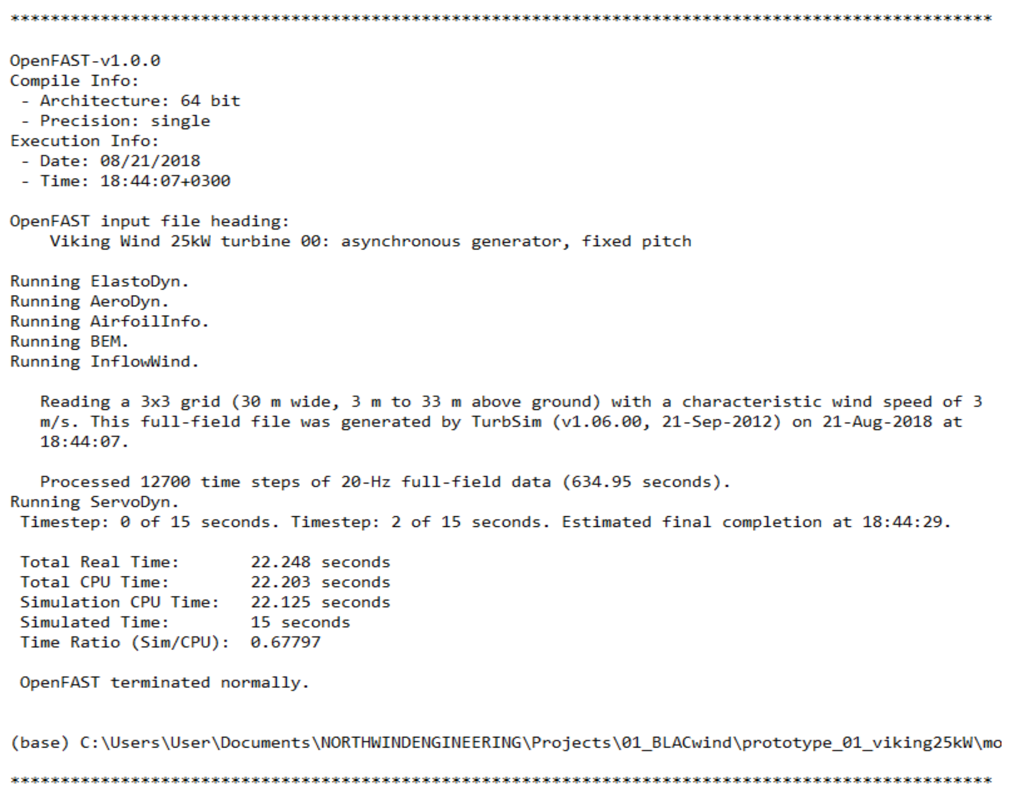
For a proper simulation ran FAST should be install in such a way that it can be ran from any folder in the computer. To run the executable, a command prompt window in opened, navigated to the directory where is saved the executable and the following command-line syntax is inserted:

fast   [<input file>]

where fast – can be either FAST.exe, FAST_x64.exe or OpenFAST_x64.exe,

<input file> – the name of the primary input file name with ".exe" extension. The designer has the choice to decide on the rootname preceding the extension.

At the end of each simulation, FAST prints out some run-time statistics as shown in figure below.



```
**************************************************************************************************

OpenFAST-v1.0.0
Compile Info:
 - Architecture: 64 bit
 - Precision: single
Execution Info:
 - Date: 08/21/2018
 - Time: 18:44:07+0300

OpenFAST input file heading:
    Viking Wind 25kW turbine 00: asynchronous generator, fixed pitch

Running ElastoDyn.
Running AeroDyn.
Running AirfoilInfo.
Running BEM.
Running InflowWind.

   Reading a 3x3 grid (30 m wide, 3 m to 33 m above ground) with a characteristic wind speed of 3
   m/s. This full-field file was generated by TurbSim (v1.06.00, 21-Sep-2012) on 21-Aug-2018 at
   18:44:07.

   Processed 12700 time steps of 20-Hz full-field data (634.95 seconds).
Running ServoDyn.
 Timestep: 0 of 15 seconds. Timestep: 2 of 15 seconds. Estimated final completion at 18:44:29.

 Total Real Time:       22.248 seconds
 Total CPU Time:        22.203 seconds
 Simulation CPU Time:   22.125 seconds
 Simulated Time:        15 seconds
 Time Ratio (Sim/CPU):  0.67797

 OpenFAST terminated normally.


(base) C:\Users\User\Documents\NORTHWINDENGINEERING\Projects\01_BLACwind\prototype_01_viking25kW\mo

**************************************************************************************************
```

Figure 2.8. Example of simulation run display output

The Simulation Time accounts for the amount of time simulated. The Simulation CPU Time accounts for time the computer uses for time-marching part of the simulation. The Simulation Time Ratio is the ratio between the amount of time simulated and the simulation CPU time. The bigger its value, the faster the computer is. In case this value is greater than 1, then FAST can simulate an event in less time than it would take in real life. A value smaller than 1 will mean that the computer might need some performance upgrade.

2.3. Simulation Environments

The SLM and Aeroelastic model for Wind turbine load assessment discussed in this work require specific simulation environment and tools for processing. The SLM, based on simple conservative equations can be build-up in any arithmetical processing tools. Well-known software like Microsoft Excel, Libre Office as well as some programming tools like MatLab, C++ and Python can be used to serve the purpose.

In the wind industry, SLM model is generally built using Spreadsheet for Simple Load Model developed and described in [8]. The latest provides an Excel format (.xls) spreadsheet designed to be adopted to all SWTs following the recommendation prescribed by the standard. However, the thesis investigates on the use of trending programing languages for development of Aeroelastic model for turbine load assessment, emphasising on Python programming package. Reason of this choice will be given further.

Whereas the setup and execution of the Aeroelastic model require generally the integration of both developed code-based simulation tools as FAST and HAWC2 and a simulation environment depending on the voluminous complexity of the model. Both programs are developed to be executable in a Windows or Linux command lines as standard simulation environment. Each single simulation can therefore be directly launch in a command line for load assessment. For multiple simulations run, the need of specific general purposes simulation environment become unavoidable. Developers of both codes considered the use of existing programming languages as simulation environment as the provide the possibility to setup complex schemes enabling and efficient loads computation with the methodology proscribed by the standard.

In the wind industry, MatLab programming language is generally used for its popularity, efficiency, multi-functionality and handiness though cost expensive. Due to the increasing popularity of Python together with its availability as open source programming language, its usability as scientific computing tool for the Aeroelastic model constitutes one of the main objectives in this work.

Python is an open source programming language that provides to users fully supported object-oriented and structured programming. It offers execution of wide range of operation going from simple arithmetical and logical algorithm to complex module integrated scripts using pre-existing standard libraries, logical statements and mathematical expressions. The language was chosen for its simplicity and its handiness compare to other alternative simulator for SLM analysis like Excel and for DLC model analysis like MatLab. One of the predominant advantage of this language is the fact that it can be open in many notebook interface like Jupiter notebook, Sublime IPython notebook, Spyder notebook etc. These notebooks offer different user-definable programming interfaces and functionalities. Spyder notebook was decided as the most suitable programming notebook for the simplicity it offers regarding graphical representation of interface, script management, file management, readability of written codes as well as proper representation of variables and generated coding result.

In addition to the usability study of potential use of Python as scientific computing tool compared to MatLab, the thesis also focuses of the use of freely available tools to build the models in consideration. LibreOffice mentioned is therefore highlighted as alternative to Microsoft Excel. LibreOffice is a freely available office suite that contains programs for word processing, creation and edition of spreadsheets, diagrams, slideshows and drawing. It makes use of the international ISO/IEC standard OpenDocument file format (ODF) as its native file format for documents saving.

Next chapter will be exposing the use of the described above software and tools for load calculation using prerogatives integrating both models as required by the standard together with implementation on a real SWT, parameters and features given.

3. AERODYNAMIC AND STRUCTURAL BEHAVIOUR OF SMALL WIND TURBINES: SIMULATION MODELLING OF A 25 KW WIND TURBINE

3.1. Implementation of Simplified Load Methodology

The model for implementation of SLM for quality assurance through evaluation of aerodynamic and operational loads applied on a SWT was built in Python 2.7 programming language. From the step by step load calculation methodology described in part 1.3.1, a Python script was written to serve the purpose. As the SLM model takes into consideration all varieties of SWTs, to generalize the application of model, the code was written in a way to be used for all type of horizontal axis SWTs, more specifically for two or three-bladed SWTs.

In the specific case of this thesis work, models were built and tested for a 25 kW turbine using available turbine rotor, tower and drive train data from Manufacturer. The turbine in consideration is a 3 bladed HAWT with a rated power of 25 kW with a cut-in and cut-out wind speeds of 4,0 m/s and 25,0 m/s respectively. It possesses a rotor diameter of 13 m making a projected area of 132,66 m^2 and a hub height of 18 m above ground level. The blades are 6.3 m long made of fibres glass and equipped with a tip brake mechanism. The rotor stainless steel shaft spins at maximum rotor speed of 65,0 rpm, transmitting its moment through the gearbox to the high speed shaft that turns a douply fed asynchronous induction generator spinning at a maximum rpm of 1525,0. The tower is made of galvanized steel tube, which stands on a concrete foundation stabilized by 8 guyed wires. Below is representative drawing of the actual turbine.

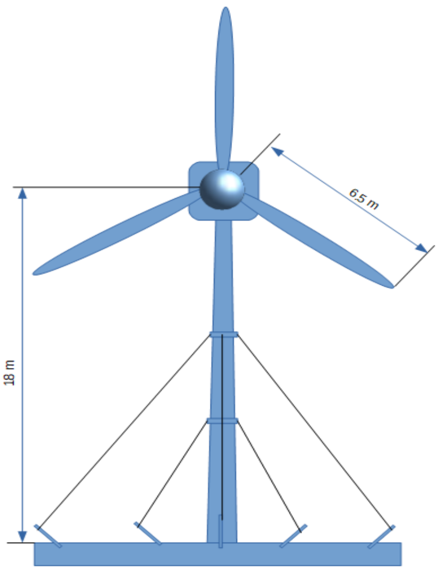


Figure 3.1. Structural representation of the 25 kW wind turbine studied in this work

The code integrates several blocks sequentially inserted, representing parts of the methodology. The first command block comprises of general turbine structural parameters presented above as input data, values of which are from the machine passport provided by manufacturer or designer. These input data inserted into the code as variables, are mainly related to moving components of the machine like rotor and shaft as well as to their inter-connections.

The second block of the script integrates calculated parameters from input data which are generally related to geometry and functionality of turbine components. SLM simulation parameters are inserted in Python in terms of variables. Unlike other programming languages, python possesses no command for variable declaration. They are created the moment a value is assigned to them. These values can be integers, floats or strings representing an arithmetical expression characterising the variable. Every command block in the script is headed by comments informing on the action undertaken in the block. Meanwhile command lines are followed by a comment describing the variable initiated. All these mentioned visual configuration of code ease its readability for a proper understanding and following of the algorithm behind the code. Appendix 1.1 represents a copy of the written code for SLM load evaluation, serving as evidence to the task accomplished.

3.1.1. Computation of loads and equivalent stresses on components

The third block of code represents the computation of fatigue and ultimate loads as required in each load cases. Loads on machine components are expressed in terms of forces, thrusts and moments. Using conservative equations presented in part 1.3.1, joined with the available turbine data, the script here determines the actual value of those loads. The equation being generalised mostly for all types of two or three-bladed SWTs specifically, the logical "if/else" statement is being used for referral the sample turbine. The load calculation part of the script for the 25 kW Turbine is illustrated in Appendix 1.1. Whereas the computed values of loads from simulation are presented in Appendix 1.2.

The fourth command block of the script represent the computation of the equivalent stresses for both analysis types as described in part 1.3.1. Table 3.1 below, displays obtained equivalent stresses on components during normal operation with power production as well as during occurrence of specific transient events. Equivalent stress formulas for circular section geometry in table 1.3 were used, as turbine blade roots and shaft possess a circular cross section geometry.

Table 3.1. Computed equivalent stress on The 25 kW Turbine from SLM python.

Load case	Equivalent stress	Results	Description
A - Fatigue	σ_{eqB}, MPa	9.04	Stress quantified from the peak-to-peak variation in the fatigue load as the standard considers turbine cycling at 0,5 – 1,5 the design speed
	$\sigma_{eq-shaft}, MPa$	48.28	
B - load during yaw	σ_{eqB}, MPa	0.022	Straightforward calculations with no assumption.
	$\sigma_{eq-shaft}, MPa$	38.655	
C – yaw error load on blade	σ_{eqB}, MPa	0.087	Only the bending moment on blades are considered.
D – maximum thrust of shaft	$\sigma_{eq-shaft}, MPa$	1.263	Only the thrust loading on shaft is considered.
E – maximum rpm	σ_{eqB}, MPa	0.075	The centrifugal force at blade root generates stress
	$\sigma_{eq-shaft}, MPa$	15.206	Bending moment of shaft creates highest stress on component at maximum rpm
F – short at load connection	σ_{eqB}, MPa	4.431	Single load for both components are considered with a straightforward calculation using moments created due to shorts.
	$\sigma_{eq-shaft}, MPa$	38.822	
G – shutdown	σ_{eqB}, MPa	8.935	Stress here depends on existence of brake system. “if/else” statement is used in the script to express to condition.
	$\sigma_{eq-shaft}, MPa$	0.006	
H – parked wind loading	σ_{eqB}, MPa	0.135	Load both components are considered.
	$\sigma_{eq-shaft}, MPa$	1.679	

Evidence of the displayed results is table can be seen in figure 3.1 below, which is a screenshot of the all-together quantified parameters in the python SLM model.

3.1.2. Turbine quality assurance – model conclusion

The fifth and last part corresponds to the conclusion on the safety of considered components. In order to conclude on safety of the machine, the calculated equivalent stresses are compared to the ultimate material stress limit. The latest is a partial safety factor dependent variable. The partial safety factors for fatigue and ultimate loads are given in table 3.2 for the SLM and Aeroelastic model and are presented below.

Table 3.2 Partial safety factors for loads [7].

	Fatigue loads, γ_f	Ultimate loads, γ_f
Simplified load model	1.0	3.0
Aeroelastic model	1.0	1.35

The standard implies that material properties should be estimated at 95% of confidence limits, defining an amount of factors to be considered during evaluation of material properties. These factors are as follow:

- full-scale structure material configuration representation;
- test samples manufacturing method representing the full-scale structure;
- fatigue, static and spectrum loading assessment;
- geometry effect on material properties;
- environmental effects on samples like humidity, UV degradation, temperature, corrosion.

Material partial safety factor value to be used depends on material characterization factors mentioned above. So, if during determination of material properties of a component like generator shaft of the specific case of 25 kW Turbine, all the above factors where considered, it means the component is fully characterised. Therefore, partial safety factors for fully and partly characterized materials given in table 3.3 below and represented below will be used. For a component like blade where the material properties evaluation is based just on coupon testing, the minimal characterisation material safety factors are used.

Table 3.3. Partial safety factors for materials [7].

Material characterisation	**Fatigue strength, γ_m**	**Ultimate strength, γ_m**
Full characterisation	1.25	1.1
Minimal characterisation	10	3.0

For a generalization of the python script with the aim of it to be used with wind turbines of different material characterization, the “if/else” statement is used for proper selection of the safety factor.

Fatigue failure are evaluated with the combination of all fatigue load on component. Safety requirement in this case is stated by equation 47 of [1] below:

$$Damage = \sum_i \frac{n_i}{N(\gamma_f \gamma_m s_i)} \leq 1.0 \qquad (3.1)$$

Where n_i – counted number of fatigue cycles in bin *i* of the characteristic load spectrum, including all relevant load cases,

s_i - stress (or strain) level associated with the counted cycles in bin *i*, including the effects of both mean and cyclic range,

N(.) is the number of cycles to failure as a function of the stress (or strain) indicated by the argument (i.e. the characteristic S-N curve) and appropriate safety factor for loads and materials respectively. The value of *N* for blades are taken for fatigue analysis S-N at load case A fatigue curves, $N_B = 9.15e^{15}$. As for shaft, number of cycles of failure is infinite at load case A stress level.

The number of fatigue cycles *n* of turbine is obtained using the following formula:

$$n = \frac{B \cdot n_{design} \cdot T_d}{60}; \tag{3.2}$$

Where T_d – design life time of turbine, *s*.

In general, the fatigue damage limit of main load carrier is equals to 1. Therefore, if the ratio between the number of fatigue cycles and the number of cycles to failure of a component will be lower than 1 then it will be considered to be safe.

The ultimate strength for limit state analysis is expressed by the design requirement is given by Equation 46 in [1]:

$$\sigma_d \leq \frac{f_k}{\gamma_m \cdot \gamma_f} \ . \tag{3.3}$$

Where σ_d – calculated equivalent stress on component,

f_k – is the characteristic material strength,

γ_m – is the partial safety factor for materials,

γ_f – is the partial safety factor for loads.

The 25 kW turbine blades are made of fibre glass reinforced polyester with a characteristic material strength of $f_{k\text{-}blade}$ = 200 *MPa* and generator shaft is made of stainless steel with a characteristic material strength of $f_{k\text{-}shaft}$ = 635 *MPa*. These values are generally provided by component.

Using equations 3.1 and 3.3 together with the partial safety factors and corresponding characteristic material strength values, the ultimate material strength where computed in the script and results presented in the table below.

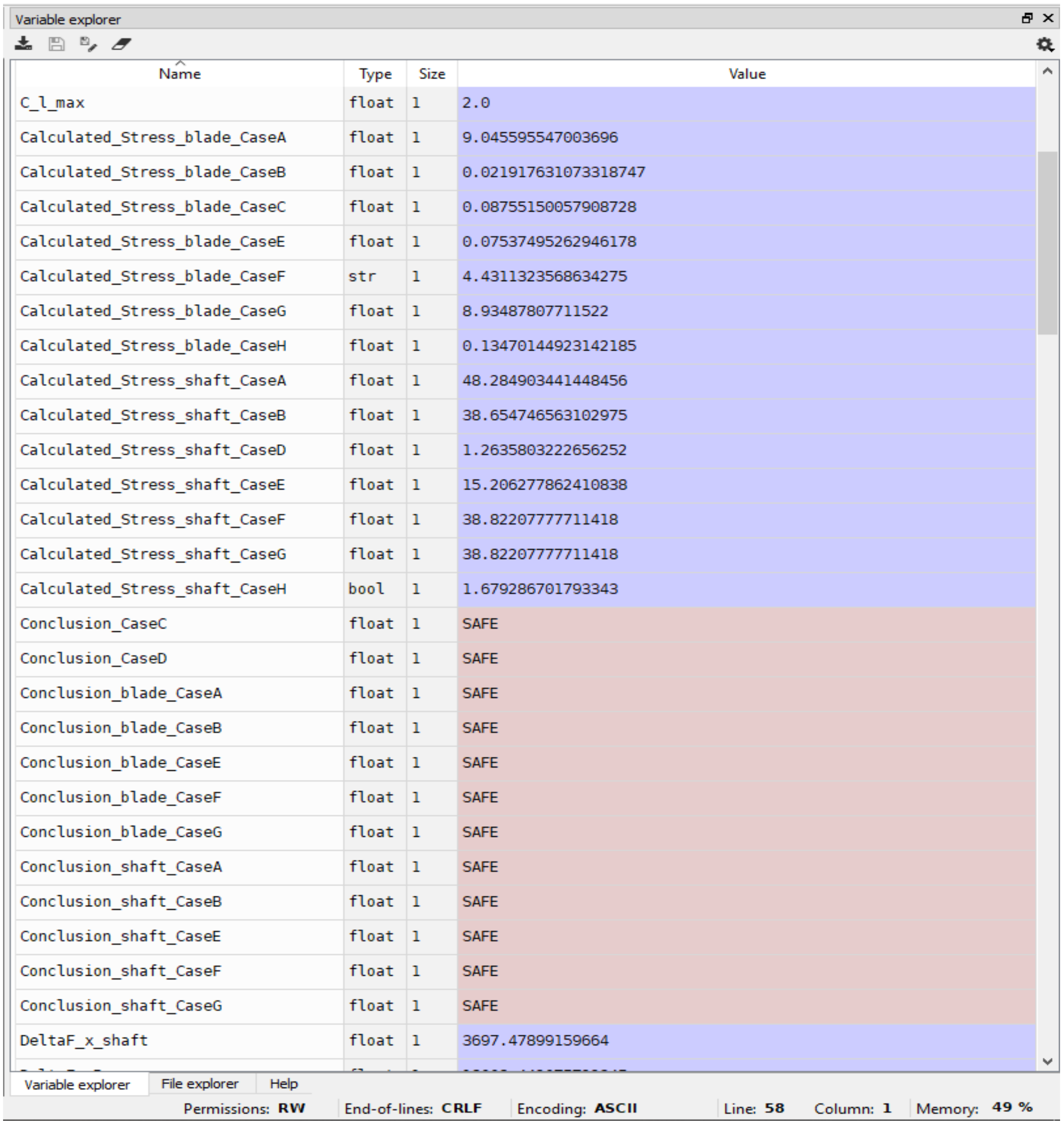
Table 3.4. Calculated result of ultimate material strength for blades and shaft.

Component	Ultimate material strength
Blade	22.2 MPa
Shaft	192.4 MPa

For fatigue limit analysis, if the calculated fatigue load is lower than fatigue damage limit, then the fatigue damage will be set as “infinite life”. This means no matter the load endured by turbine component at this loading condition, its value won’t be high enough to cause component failure during the life time of the machine.

As to other load cases for ultimate analysis, the material stress limit of component which is also identified as ultimate material strength presented in table 3.4, will be compared to component calculated equivalent stress values displayed in table 3.1. Therefore, for a chosen component at a given loading condition, if the calculated equivalent stress is lower than the calculated ultimate material strength, it will be considered as safe, otherwise its failure will be expected at certain point of the turbine lifecycle. The statements “SAFE” and “FAIL” are used in the code to indicate component safety in accordance to comparison results.

In figure 3.1 below, a screenshot from python variable explorer displaying the model result concluding on turbine safety of operation under the specified loading conditions proscribed by the standard. The calculated stress in the figure is nothing else than the equivalent stress presented in table 3.1. It serves as evidence of the SLM model result.



Name	Type	Size	Value
C_l_max	float	1	2.0
Calculated_Stress_blade_CaseA	float	1	9.045595547003696
Calculated_Stress_blade_CaseB	float	1	0.021917631073318747
Calculated_Stress_blade_CaseC	float	1	0.08755150057908728
Calculated_Stress_blade_CaseE	float	1	0.07537495262946178
Calculated_Stress_blade_CaseF	str	1	4.4311323568634275
Calculated_Stress_blade_CaseG	float	1	8.93487807711522
Calculated_Stress_blade_CaseH	float	1	0.13470144923142185
Calculated_Stress_shaft_CaseA	float	1	48.284903441448456
Calculated_Stress_shaft_CaseB	float	1	38.654746563102975
Calculated_Stress_shaft_CaseD	float	1	1.2635803222656252
Calculated_Stress_shaft_CaseE	float	1	15.206277862410838
Calculated_Stress_shaft_CaseF	float	1	38.82207777711418
Calculated_Stress_shaft_CaseG	float	1	38.82207777711418
Calculated_Stress_shaft_CaseH	bool	1	1.679286701793343
Conclusion_CaseC	float	1	SAFE
Conclusion_CaseD	float	1	SAFE
Conclusion_blade_CaseA	float	1	SAFE
Conclusion_blade_CaseB	float	1	SAFE
Conclusion_blade_CaseE	float	1	SAFE
Conclusion_blade_CaseF	float	1	SAFE
Conclusion_blade_CaseG	float	1	SAFE
Conclusion_shaft_CaseA	float	1	SAFE
Conclusion_shaft_CaseB	float	1	SAFE
Conclusion_shaft_CaseE	float	1	SAFE
Conclusion_shaft_CaseF	float	1	SAFE
Conclusion_shaft_CaseG	float	1	SAFE
DeltaF_x_shaft	float	1	3697.47899159664

```
Permissions: RW    End-of-lines: CRLF    Encoding: ASCII    Line: 58    Column: 1    Memory: 49 %
```

Figure 3.1. SLM model quality assurance result for the 25 kW Turbine.

Appendix 1.1 represents a copy of the full python written script for this model. All used initial input data, calculation using the described formulas according to the wind turbine specificity as well as logical and arithmetical expressions are thereby represented. The script was built in a way allowing it use on multiple type of SWT with configuration falling under the standard's requirement.

For a better visualization of the equivalent stress computation from different load cases, a plot was generated in python environment where we have in the x-axis the load cases and on the y-axis corresponding equivalent stress values as illustrated in figure 3.2 below.

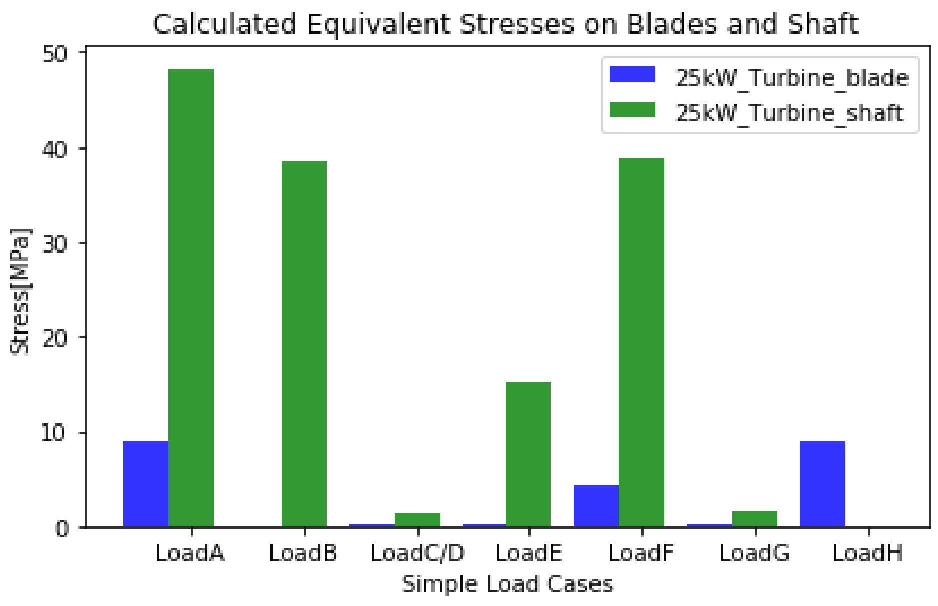


Figure 3.2 Equivalent stresses on main load carriers.

Evidence of written script for equivalent stress plotting can be seen in Appendix 2.

Although the shaft is concluded to be safe in the point of view of design requirement, it is the component endorsing the highest stress during normal power production operation, yaw, maximum rotational speed and short at load connection. Meanwhile blades experience most noticeable stress during normal power production operation, short at load connection and during idling. These conclusion is assumed to be logical as in reality, these events are the cause of most stresses on turbine components leading to their failure in case of non-proper choice of component materials.

3.2. Implementation of Aeroelastic Model in Python 2.7

Loads from the Aeroelastic model are computed using FAST aeroelastic code. Both the model and the code show to be complex entities constituting a huge implementation challenge on a turbine. In this section will be described the setup process of the model by presenting constituents of each simulation, how all simulations are built-up along with their specificity, how are they ran and where resulting output files are the stocked.

3.2.1. Constituents of one simulation process

Each simulation, specific to the wind condition under which load analysis of the turbine are calculated, comprises of parameters specified by both the design load case and the FAST operating mechanism as shown in the diagram below.

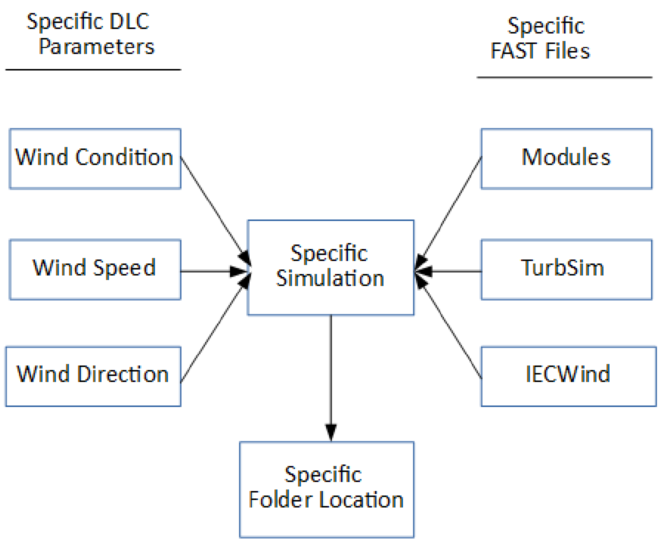


Figure 3.3 Main specificity of each simulation

The design load case describes the simulation conditions which are later used in FAST to run the simulation for load determination and to generate the final output files.

The wind conditions described by the Aeroelastic model are of 2 main types, the normal turbulence model and the extreme wind transient conditions. These 2 types enclose together 7 distinct wind conditions, each of which possesses a wide range of wind velocity going from 3 m/s to 25 m/s with some extreme wind speed reaching 40 m/s. The model also requires the turbine to be tested in different yaw error (in terms of angle) as in real life. The following directions are therefore assumed:

- 0°, 10°, 350° for the NTM;
- Generally, 0° for most Extreme Conditions;
- 0° - 330° with a step of 30° for EWM.

For a better management of files created and those generated during the simulation process, each result files are saved in a specific allocated folder so to be easily located for use in the future for other purposes.

Therefore, wind loadings are to be evaluated in each of the mentioned above conditions taking into consideration file management as well. This constitutes the specificity of each simulation and raises the question of difficulty to manually build the model that shows to be complex and time consuming making a part of the aim of this thesis.

Let’s consider as example the first simulation set up in this model. The following steps and data were required:

- File saving location;
- Wind condition (type) – NTM
- Wind speed = 3 m/s
- Wind direction = 0°
- Random seed for TurbSim wind file generation = 13428
- TurbSim output wind file generated using random seed and wind speed
- InflowWind input file generated using TurbSim output file
- AeroDyn file generated accounting blade aerodynamic standard file
- ElastoDyn file generated taking in tower aero-elastic standard file
- Servodyn file generated
- FAST input file generated calling in all available module generated input files
- Time to start simulation = 20 s
- Time to end simulation = 620 s
- Rotor speed = 65 rmp
- Time to deploy HSS-brake = 0
- Brake torque = 250 N/m
- Time to turn generator on = 0 s
- Time to turn generator off = 9999.9 s

All this set-up operation should be processed before running the simulation. Generally, this can require couple of hour not taking into consideration the time spent for model study and simulator operating mode understanding. Meanwhile, in accordance to requirements of the standard, a total of 345 distinct simulations are needed so to cover all the spectrum of design load cases for the 25 kW Turbine. Therefore, manually setting up all these simulations and running them one after the order is a studious and time consuming task. To ease this process, the auto-generation process was developed.

3.2.2. Auto-generation process of all required simulation for the DLCs

Given the complexity of the setup process of a single simulation, the question of how to develop a more efficient modelling process enabling a faster setup of the Aeroelastic model for later execution by the simulator is hereby raised. Here comes the need of general-purposes programming languages reputed of having the ability to manage complex algorithm in order to complete specific tasks in form of codes, scripts or commands assigned by the user. Python 2.7 was chosen as suitable programming language for this purpose.

In order to model a repetitive setup process for each simulation sequentially, a so called Auto-Generation process was developed using python 2.7 in Anaconda Spyder environment. The process comprises of 4 mains steps described below:

- Creation of main location folders in the computer drive to save input and output files;
- Setup of Master files for all module input files;
- Creation and setup of an auto-generation spreadsheet for simulation parameters or variables;
- Python scripting for execution of auto-generation process and simulations combined.

The first step of the process is creating a directory in the drive where folders necessary for saving the automatically generated files are located. For the 25 kW Turbine, one directory was created which contains 3 sub-directories (blades permanent folder, tower permanent folder and Simulation folder). The diagram below shows the arrangement of folders in directory. The blade folder contains blade airfoil files, aerodynamic files and aero-elastic file. The tower folder contains tower's aerodynamic file and aero-elastic file.

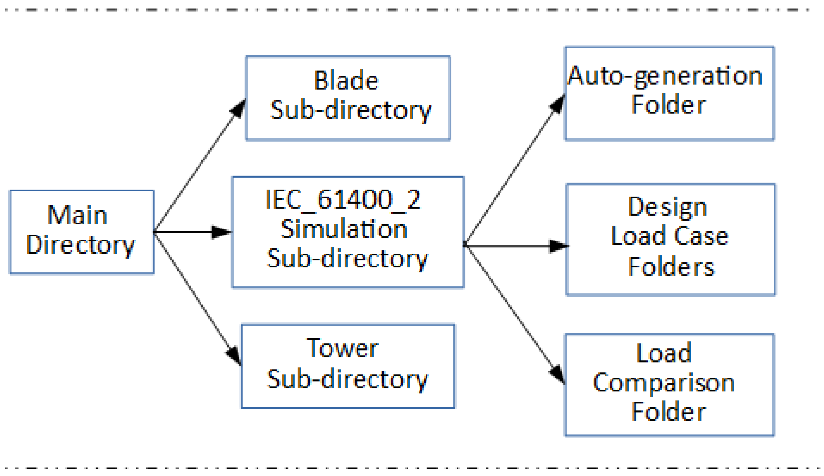


Figure 3.4 Aeroelastic model directory scheme

In the auto-generation folder are located all master files. These files are modules input files in which the parameter values to be modified according to each simulation are temporally set to a "Place Holder". This place holder will later be automatically change to the real value taken from the spreadsheet and characterizing the parameter. Below are the following master files included in the model, the rootname before the extension ".dat" of which can be user-defined:

- AeroDyn14_DLC2_Master.dat – to generate all aerodynamic input files for DLC 2 simulations;

- AeroDyn15_Master.dat – to generate all aerodynamic input files for all other DLC simulations;
- ElastoDyn_Master.dat – to generate all ElastoDyn input files for all DLC simulations;
- ServoDyn_Master.dat – to generate all ServoDyn input files for all DLC simulations;
- InflowWind_Master.dat – to generate all InflowWind input files for all DLC simulations;
- IEC_Master.dat – to generate all IECWind input files for all DLC simulations;
- DLC2_Turbsim_Master.dat – to generate all TurbSim input files for DLC 2 simulations;
- TurbSim_Master.dat – to generate all TurbSim input files for all other DLC simulations;
- fst_DLC2_Master.fst – to generate all FAST input files for DLC 2 simulations;
- fst_Master.fst – to generate all FAST input files for all other DLC simulations.

Auto-generation folder contains also all python scripts files. It can be called the brain folder of the whole simulation.

Master files shown in Appendix 3 are simply generated by the modification of sample input files provided by FAST, TurbSim and IECWind programs in which the values to be modified in accordance to each specific simulation are replaced by place holders. Those sample files are renamed to Master Files after modification and saved in the auto-generation folder.

In DLC folders are stocked all automatically generated specific simulation files from master files as described above as well as all output files after the simulation run. There are 14 different folders corresponding to all design load cases with their names indicating the wind conditions: DLC11_NTM, DLC12_ECD, DLC13_EOG50, DLC14_EDC50, DLC15_ECG, DLC21_NWP, DLC22_NTM, DLC23_EOG1, DLC31_NTM, DLC32_EOG1, DLC41_NTM, DLC51_EWM, DLC52_NTM and DLC61_EWM.

The auto-generation spreadsheet for the Aeroelastic model was created from the LibreOffice package. The generated spreadsheet for the 25 kW Turbine Aeroelastic model named "AutogenerationSpreadsheet.ods", comprises of rows representing each simulation and columns exhibiting parameters or variables specific to each simulation.

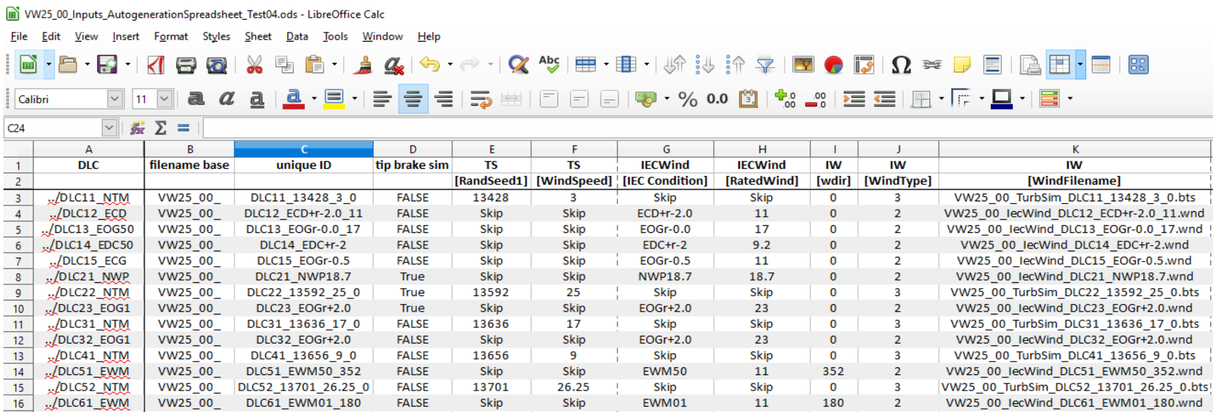
The two first rows are the most important for the python code as they serve as orientation on where in the spreadsheet the values should be taken to create needed files. The first row carries description of each simulation parameter and FAST modules whereas the second carries place holders. These place holders indicate names of the parameters and are written in squared brackets enabling is readability by Python. Here are the main place holders in the spreadsheet:

- [RandSeed1] – TurbSim random seed,
- [WindSpeed] – wind speed,
- [IEC Condition] – to indicate the extreme condition for IECWind wind file generation,

- [RatedWind] – rated wind speed used by IECWind for wind file generation,
- [wdir] – wind direction,
- [WindType] – to indicates if it is a binary TurbSim full-field or a steady extreme wind type,
- [WindFilename] – to call in generated wind files by TurbSim or IECWind into InflowWind input file,
- [InflowWind] – to call in InflowWind input file into FAST input file,
- [AeroDyn] – to call in aerodynamic input file into FAST input file,
- [ServoDyn] – to call-in ServoDyn input file into FAST input file,
- [ElastoDyn] – to call-in ElastoDyn input file into FAST input file,
- [Tstart] – time to start writing simulation output in output file
- [Tmax] – time to stop simulation, maximum simulation time.

Therefore, during auto-generation process, the script will tell python whenever it meets any of these place holders in master files, it should replace it by the corresponding value from rows in the spreadsheet. The row represents a single simulation.

On the column side of the spreadsheet, the first column indicates the location in which files should be saved, the second and third form a unique identification to each file created, the rest of columns contain values corresponding to each parameters. The figure below shoes representation of the spreadsheet.


VW25_00_Inputs_AutogenerationSpreadsheet_Test04.ods - LibreOffice Calc
File Edit View Insert Format Styles Sheet Data Tools Window Help
Calibri
11
C24

	A	B	C	D	E	F	G	H	I	J	K
1	DLC	filename base	unique ID	tip brake sim	TS	TS	IECWind	IECWind	IW	IW	IW
2					[RandSeed1]	[WindSpeed]	[IEC Condition]	[RatedWind]	[wdir]	[WindType]	[WindFilename]
3	../DLC11_NTM	VW25_00_	DLC11_13428_3_0	FALSE	13428	3	Skip	Skip	0	3	VW25_00_TurbSim_DLC11_13428_3_0.bts
4	../DLC12_ECD	VW25_00_	DLC12_ECD+r-2.0_11	FALSE	Skip	Skip	ECD+r-2.0	11	0	2	VW25_00_IecWind_DLC12_ECD+r-2.0_11.wnd
5	../DLC13_EOG50	VW25_00_	DLC13_EOGr-0.0_17	FALSE	Skip	Skip	EOGr-0.0	17	0	2	VW25_00_IecWind_DLC13_EOGr-0.0_17.wnd
6	../DLC14_EDC50	VW25_00_	DLC14_EDC+r-2	FALSE	Skip	Skip	EDC+r-2	9.2	0	2	VW25_00_IecWind_DLC14_EDC+r-2.wnd
7	../DLC15_ECG	VW25_00_	DLC15_EOGr-0.5	FALSE	Skip	Skip	EOGr-0.5	11	0	2	VW25_00_IecWind_DLC15_EOGr-0.5.wnd
8	../DLC21_NWP	VW25_00_	DLC21_NWP18.7	True	Skip	Skip	NWP18.7	18.7	0	2	VW25_00_IecWind_DLC21_NWP18.7.wnd
9	../DLC22_NTM	VW25_00_	DLC22_13592_25_0	True	13592	25	Skip	Skip	0	3	VW25_00_TurbSim_DLC22_13592_25_0.bts
10	../DLC23_EOG1	VW25_00_	DLC23_EOGr+2.0	True	Skip	Skip	EOGr+2.0	23	0	2	VW25_00_IecWind_DLC23_EOGr+2.0.wnd
11	../DLC31_NTM	VW25_00_	DLC31_13636_17_0	FALSE	13636	17	Skip	Skip	0	3	VW25_00_TurbSim_DLC31_13636_17_0.bts
12	../DLC32_EOG1	VW25_00_	DLC32_EOGr+2.0	FALSE	Skip	Skip	EOGr+2.0	23	0	2	VW25_00_IecWind_DLC32_EOGr+2.0.wnd
13	../DLC41_NTM	VW25_00_	DLC41_13656_9_0	FALSE	13656	9	Skip	Skip	0	3	VW25_00_TurbSim_DLC41_13656_9_0.bts
14	../DLC51_EWM	VW25_00_	DLC51_EWM50_352	FALSE	Skip	Skip	EWM50	11	352	2	VW25_00_IecWind_DLC51_EWM50_352.wnd
15	../DLC52_NTM	VW25_00_	DLC52_13701_26.25_0	FALSE	13701	26.25	Skip	Skip	0	3	VW25_00_TurbSim_DLC52_13701_26.25_0.bts
16	../DLC61_EWM	VW25_00_	DLC61_EWM01_180	FALSE	Skip	Skip	EWM01	11	180	2	VW25_00_IecWind_DLC61_EWM01_180.wnd

Figure 3.5 The 25 kW Turbine Auto-generation Spreadsheet short representation.

As noticeable in the figure ahead, DLC 2 in which during occurrence of fault, the tip-brakes are deployed for system protection. This requires a special simulation setup involving the tip-brake feature contained only in FAST 7 version. Therefore, a flag in the spreadsheet is used in order to tell python which version of FAST code is to be executed for the simulation.

Other 2 auxiliary programs TurbSim and IECWind described above are run to generate wind files for simulations with normal turbulence model and extreme transient respectively. Therefore, in the spreadsheet a “Skip” flag is used to tell python not to create files and run simulation whenever its meets this statement during the looping process.

3.2.3. Python scripting for execution of auto-generation process and simulations

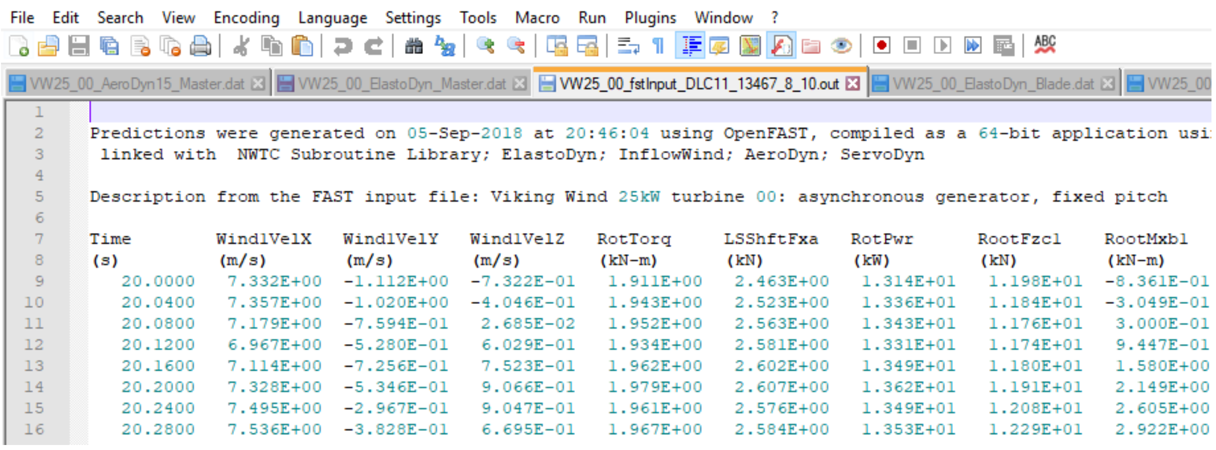
The python script was written to enable a fast and coherent reading of the auto-generation spreadsheet, to generate input files, save them into assigned location, run all need programs and generate output files as result of simulations. Appendix 4 presents a copy of python script for simulation process, meanwhile described below. The script encloses 4 main parts:

- In the first part of the script, python variables calling-in the master files and the auto-generation spreadsheet are created. Through this action, python locates those files in the Auto-generation folder and assigns them specific names
- The second part opens the master files, reads the contain and saves them into its memory as template files. These template files contain the same data as in the master files but on in a python readable format.
- The third part of the script is specific to reading a ODS file type like the auto-generation file. It requires the use of ODS library freely available for users that gives the possibility to python to read ODS file format. By calling the “ReadODS” function, auto-generation spreadsheet saved previously as a variable is opened, read according to rows and columns and saved into python memory. It also resizes the spreadsheet to contain only desired data by creating delimitation in rows and columns and assigning to them specific variable names that will later be used during looping iteration through the spreadsheet. Rows 3 to 347 represent each simulation, row 1 is the files ID representing the modules and flags and row 2 the place holders.
- The fourth and fifth parts correspond to the looping. The script tells python for each simulation, to first locate from the main working directory (Auto-generation Folder) the DLC folder where to save files that will be generated, next to go to corresponding row, create file name with unique IDs, check which version of FAST to run depending on the “True” or “False” tip-brake flag, check TurbSim or IECWind flags (if value in cell exist, it creates input files from master files, saves them in its memory, runs TurbSim or IECWind executables from command line using created input files and save the resulted output files in the corresponding folder, while if values in cell is “Skip”, it skips all operations related to the program and moves forward), it further moves to columns corresponding to modules, creates and saves input files in respective directories. After all the files created and save, Python checks if tip-brake flag was set to “True”, runs FAST7_x64 executable program from command line, generate final output files and save them in the folder assigned to FAST input and output files. Otherwise, it runs OpenFAST_x64. Operation executed, it jumps to

next row and repeats the same action until it goes through all the 345 rows (simulation) printing out all simulation summary presented in figure 2.8 into a text file. This will later be used to identify simulation that failed to run in case of a bug during the process.

3.2.4. FAST output files

A FAST output file is a tabular time-matching file containing a description of the simulation ran, columns representing computed parameters and rows representing obtained results after an interval of time. This interval of time called time step, is the time after which FAST prints out new computed result for each single parameter. Time step in the model is generally equals to 0.04 s. Figure 3.6 shows a representation of the output files.


File
Edit
Search
View
Encoding
Language
Settings
Tools
Macro
Run
Plugins
Window
?
VW25_00_AeroDyn15_Master.dat
VW25_00_ElastoDyn_Master.dat
VW25_00_fstInput_DLC11_13467_8_10.out
VW25_00_ElastoDyn_Blade.dat
VW25_00

```

Predictions were generated on 05-Sep-2018 at 20:46:04 using OpenFAST, compiled as a 64-bit application usi
 linked with  NWTC Subroutine Library; ElastoDyn; InflowWind; AeroDyn; ServoDyn

Description from the FAST input file: Viking Wind 25kW turbine 00: asynchronous generator, fixed pitch

Time         Wind1VelX    Wind1VelY    Wind1VelZ    RotTorq      LSShftFxa    RotPwr       RootFzc1     RootMxb1
(s)          (m/s)        (m/s)        (m/s)        (kN-m)       (kN)         (kW)         (kN)         (kN-m)
   20.0000    7.332E+00   -1.112E+00   -7.322E-01    1.911E+00    2.463E+00    1.314E+01    1.198E+01   -8.361E-01
   20.0400    7.357E+00   -1.020E+00   -4.046E-01    1.943E+00    2.523E+00    1.336E+01    1.184E+01   -3.049E-01
   20.0800    7.179E+00   -7.594E-01    2.685E-02    1.952E+00    2.563E+00    1.343E+01    1.176E+01    3.000E-01
   20.1200    6.967E+00   -5.280E-01    6.029E-01    1.934E+00    2.581E+00    1.331E+01    1.174E+01    9.447E-01
   20.1600    7.114E+00   -7.256E-01    7.523E-01    1.962E+00    2.602E+00    1.349E+01    1.180E+01    1.580E+00
   20.2000    7.328E+00   -5.346E-01    9.066E-01    1.979E+00    2.607E+00    1.362E+01    1.191E+01    2.149E+00
   20.2400    7.495E+00   -2.967E-01    9.047E-01    1.961E+00    2.576E+00    1.349E+01    1.208E+01    2.605E+00
   20.2800    7.536E+00   -3.828E-01    6.695E-01    1.967E+00    2.584E+00    1.353E+01    1.229E+01    2.922E+00
```

Figure 3.6 FAST Output file representation.

FAST gives a wide variety of turbine loads and parameters generally called output parameters or output channels that can be computed from the code and can be user-defined to suite Aeroelastic model needs. However, the time and wind speed parameters are standard, therefore cannot be modified. These parameters are all inserted into the FAST via ServoDyn control module.

For the 25 kW Turbine Aeroelastic model, the following output parameters were chosen:

- *RotTorq* – Rotor torque;
- *LSShftFxa*, *RotThrust* – Low-speed shaft thrust force (this is constant along the shaft and is equivalent to the rotor thrust force). Directed along the xa- and xs-axes (*kN*);
- *RotPwr* – Rotor power (this is equivalent to the low-speed shaft power);
- *RootFzc1* – Blade 1 axial force at the blade root directed along the zc1- and zb1-axes (*kN*);
- *RootMxb1* – Blade 1 edgewise moment (i.e., the moment caused by edgewise forces) at the blade root. About the xb1-axis (*kN·m*);
- *LSShftMxa* – Low-speed shaft torque (this is constant along the shaft and is equivalent to the rotor torque). About the xa- and xs-axes (*kN·m*);

- *TwrBsMxt* – Tower base roll (or side-to-side) moment (i.e., the moment caused by side-to-side forces). About the xt-axis (*kN·m*);
- *YawBrFzn* – Tower-top / yaw bearing axial force. Directed along the zn (*kN*);
- *YawBrMxp* – Nonrotating tower-top / yaw bearing roll moment. About the xp-axis (kN·m);
- *RootFxb1* – Blade 1 flapwise shear force at the blade root. Directed along the xb1-axis (kN);
- *RootMzc1* – Blade 1 pitching moment at the blade root. About the zc1- and zb1-axes (kN·m).

The other output parameters can be seen in ElastoDyn input master file in Appendix 3.3.

Simulation output files as presented in figure 3.6 all-together constitute a massive set of billions of data points useless to designers without any additional data treatment applied. Therefore, in order to make use of the obtained data points, the so called Post-Processing needs to be implemented. Post-processing in this case will be defined as the action taken to transform a bunch of data points to a graphically readable object characterizing the result of the calculated parameters describing the turbine.

3.2.5. FAST output files post-processing

Output results from simulations contain extremely large amount of numbers. The post-processing will be the extraction of maximum, mean and minimum values from all output channels of all simulations of the corresponding DLC. This statistical computation is a studious and very time consuming task to process manually as it involves the treatment of more than 25 output parameters in each of the 345 output files located in different folders. The post-processing not only includes finding of peak values from the output channels but also graphically representing them so to be easily readable by the designer. This process is of a high complexity to be built manually for each simulation. Therefore, the need of integrating Python scripting arises again.

For the 25 kW Turbine Aeroelastic model, 3 distinct python scripts were written for the statistical computation:

- The first script finds maximum, mean and minimum values from all output channels of the same type in all output files of a particular DLC, plots the result on a graph where on y-axis are the *max*, *mean*, *min* values of output parameters against the wind speed on x-axis. See figure below.

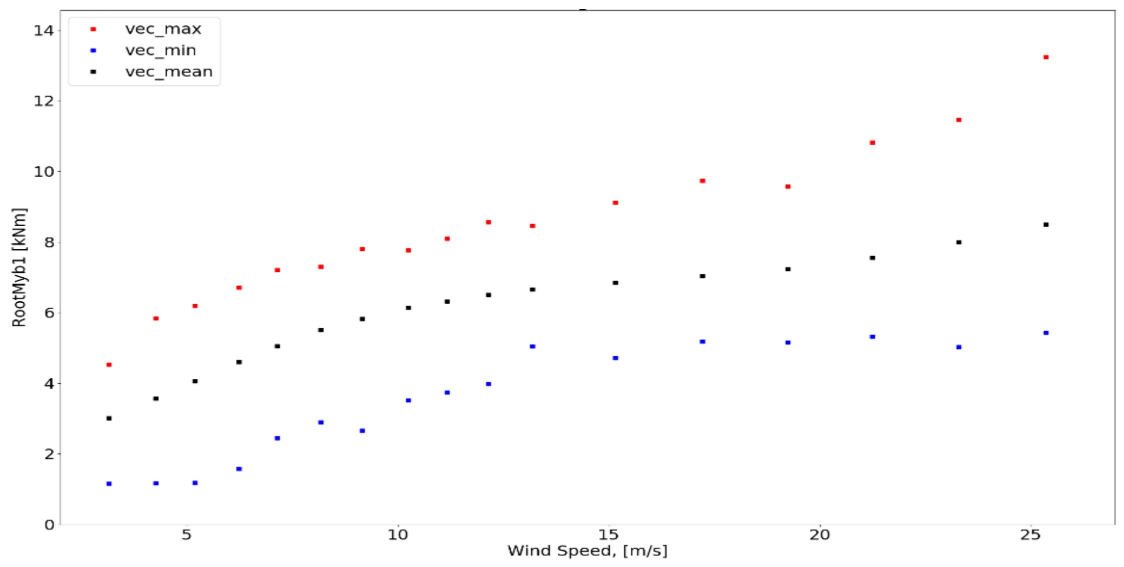


Figure 3.7 Blade flapwise bending moments on y axis for DLC 1.1.

Evidence of the written script can be found in Appendix 5.

- The second script finds *max*, *mean* and *min* values in the combination of all simulations output channels of the same type from all DLCs and creates plots representing on the y-axis output channels values against the wind speed. This simply means that python goes into all 345 output files collects data of same load channel, combines them and finds peak values at each wind speed. This gives the possibility to identify the maximum generated load with corresponding wind speed at each output channel and from all DLCs. See figure below.

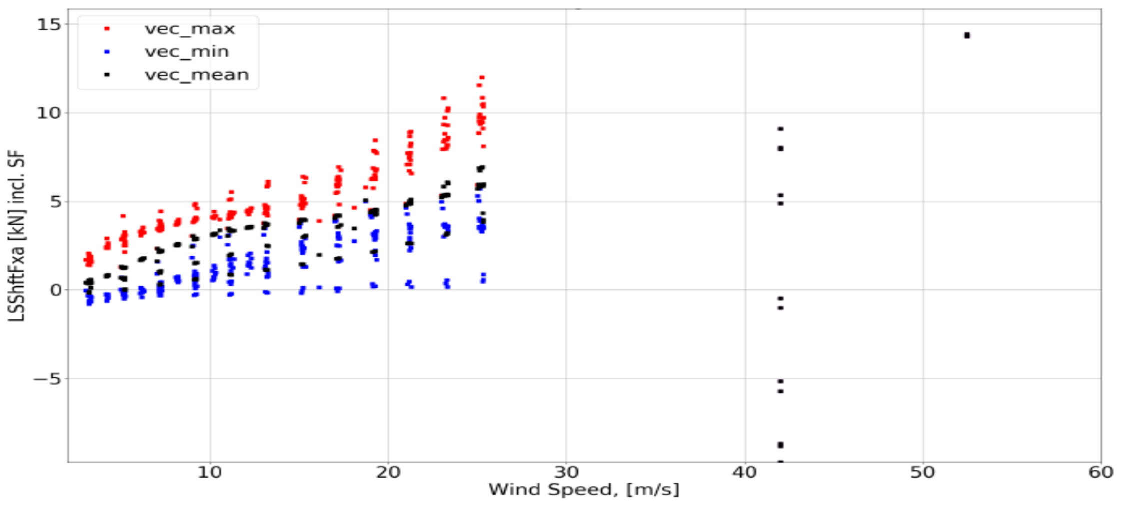


Figure 3.8 Low speed thrust force on shaft from all DLCs

Evidence of the written script can be found in Appendix 6.

- The third script combines obtained values for each output parameters of the same type finds the first 10 maximums and minimum values indicating from which DLC they apply and later bar-plotting them. The bar plot here represents loads from output parameters on the y-axis and on the x-axis the simulation causing such loading. See figure below.

Evidence of the written script can be found in Appendix 7.

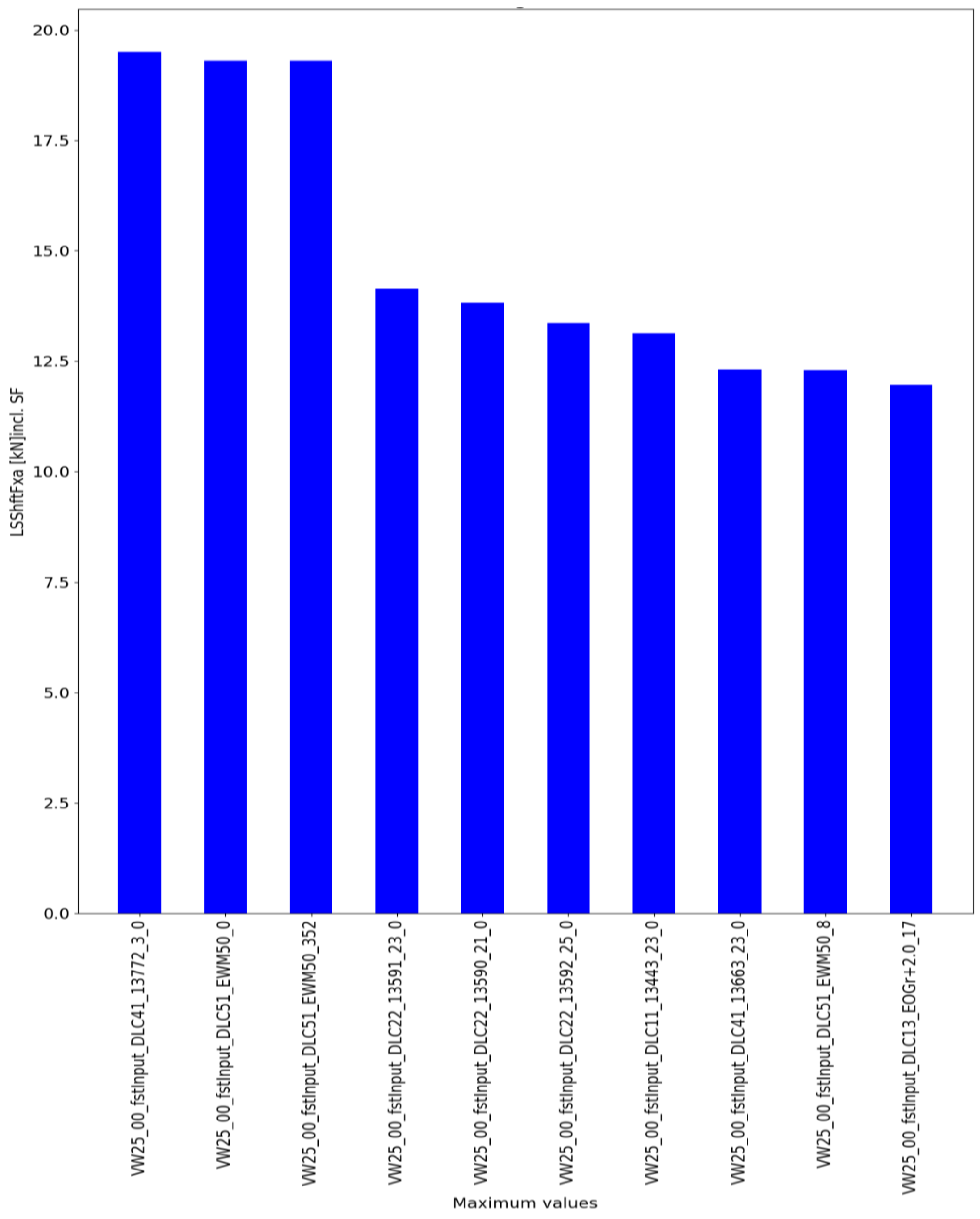
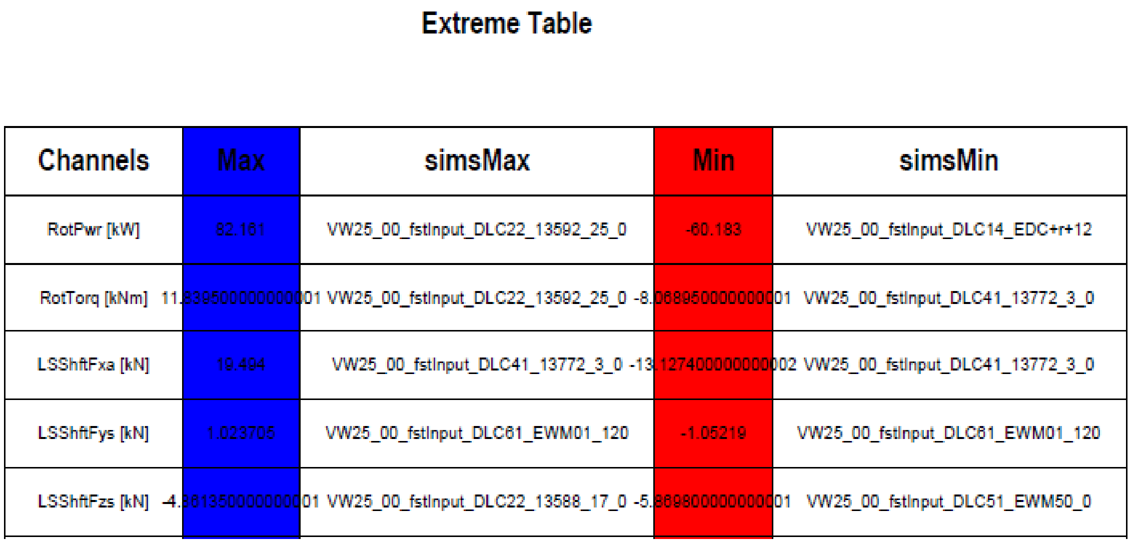


Figure 3.9 Low-speed shaft thrust force 10 maximum values from all simulations.

The last script not only bar-plot the 10 maximum values in of one output parameter but also multiply the values by the safety factor for Aeroelastic model, *SF = 1.35*. Therefore, the plotted values correspond to the final simulation output characterising each loads apply to the turbine and can be compare to results from other parallel simulation tools like HAWC2 as well as to other simulation methods like SLM in occurrence. In order to make this comparison possible, the overall maximum values of each loads needs to be identified. The script also fulfils this task by identifying these extreme values, creating a table and saving them into a generated a pdf files. See figure 3.10 for a screenshot of generated table containing results of the 25 kW turbine Aeroelastic model. The full table is represented in Appendix 5.5.



Extreme Table

Channels	Max	simsMax	Min	simsMin
RotPwr [kW]	82.161	VW25_00_fstInput_DLC22_13592_25_0	-60.183	VW25_00_fstInput_DLC14_EDC+r+12
RotTorq [kNm]	11.839500000000001	VW25_00_fstInput_DLC22_13592_25_0	-8.068950000000001	VW25_00_fstInput_DLC41_13772_3_0
LSShftFxa [kN]	19.494	VW25_00_fstInput_DLC41_13772_3_0	-13.127400000000002	VW25_00_fstInput_DLC41_13772_3_0
LSShftFys [kN]	1.023705	VW25_00_fstInput_DLC61_EWM01_120	-1.05219	VW25_00_fstInput_DLC61_EWM01_120
LSShftFzs [kN]	-4.861350000000001	VW25_00_fstInput_DLC22_13588_17_0	-5.869800000000001	VW25_00_fstInput_DLC51_EWM50_0

Figure 3.10 The 25 kW Turbine Aeroelastic model result.

The data from the table in appendix 5.5 shows that axial force at the blade root (RootFzc1) generates the highest loads on turbine blades during power production plus occurrence of fault. This result seems logical as for this case the rotor goes in to over-speed, inducing tip brakes deployment for system protection generating severe loads at blade roots. The table conclusion can be verified in figure 3.10 Similarly, tower base pitching (or fore-aft) moment (TwrBsMyt), the moment caused by fore-aft forces generates the highest moment load of turbine tower during manual or emergency shutdown generally being and instantaneous turbine switch-off while still in operation due to some severe wind condition or for maintenance.

The table can later be used as reference results for the Aeroelastic model. Nevertheless, it is highly valuable during turbine check on quality assurance for certification as it presents only the necessary peak values that can be generated for each considered load on machine components. The model therefore includes a final script that generates a PDF report file for statistic computation of design load cases simulation. The report file contains in each page plots generated by the second and third script for each output parameter as well as the generated extreme table. A page from the PDF report file is presented in Appendix 5. for reference. This PDF is to be attached to the turbine passport during Certification check, making it relevant importance.

An additional evidence demonstrating a successful conduction of the model will be the comparison between graphical representation of some transient extreme condition described in subpart 1.3.2 and represented in [7] and some graphs plotted from the obtained simulation results. Figure 3.11 below illustrate wind gusting at EOG event successfully modelled in figure 3.12 by FAST.

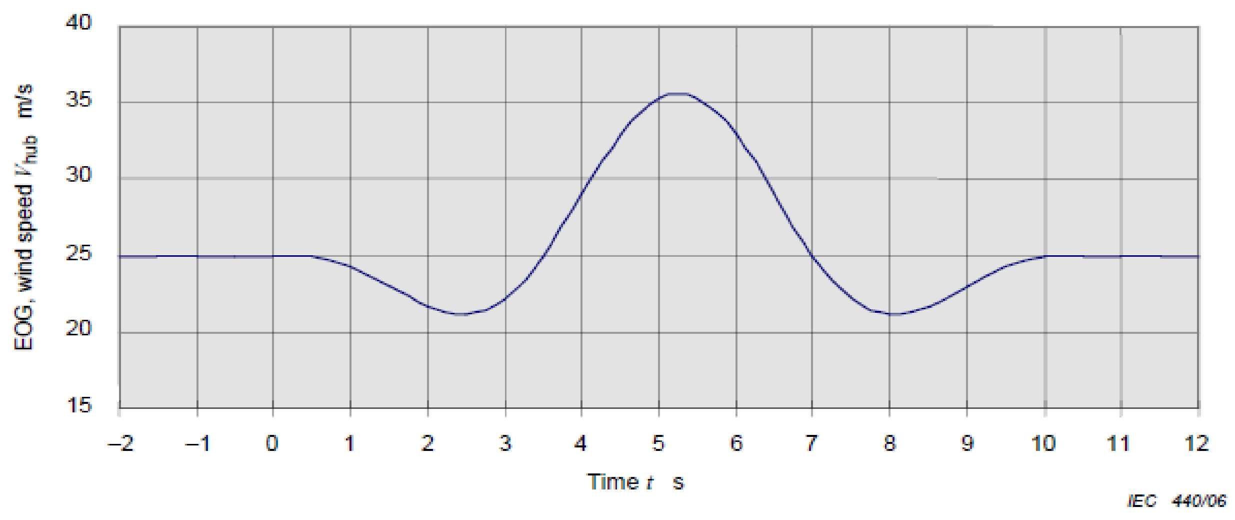


Figure 3.11 Example of extreme operating gust (EOG) at N=1 and V_{hub} = 25 m/s [7]

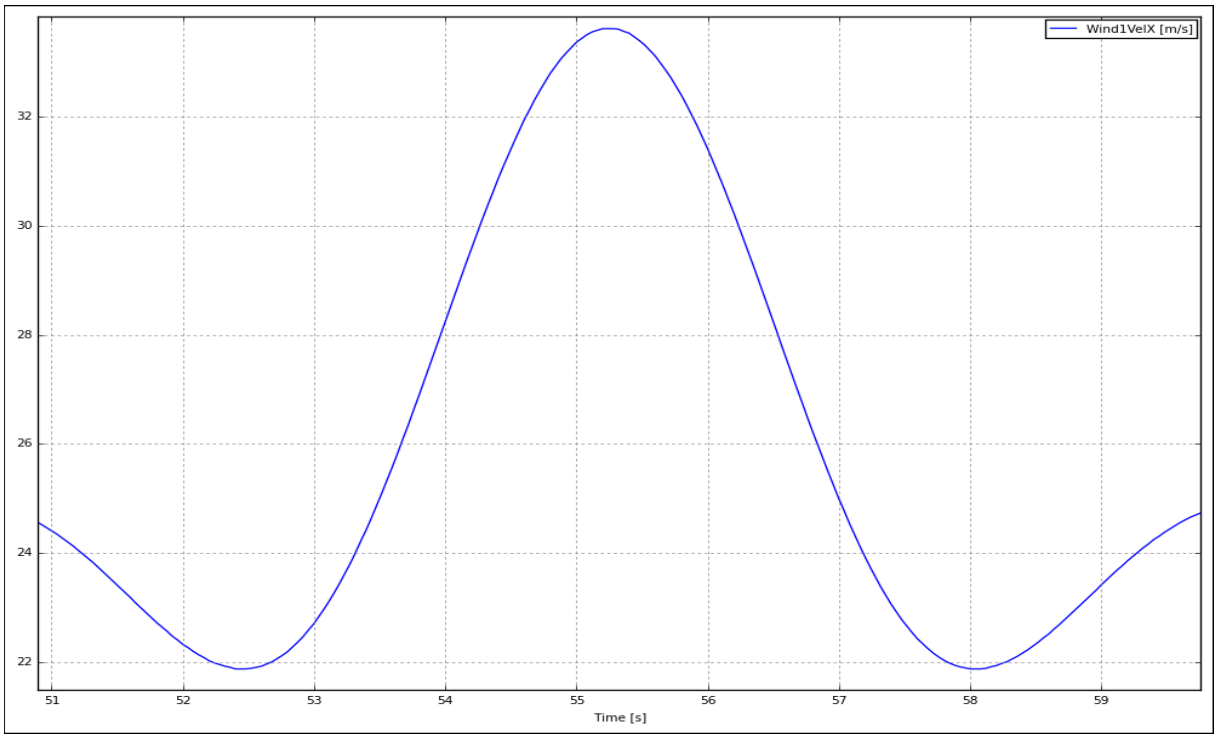


Figure 3.12 Sample 25 kW Turbine DLC 3.2 normal shutdown EOG at N=1 and V_{hub} = 25 m/s

As noticeable, both graphs are similarly characterizing the EOG at normal shutdown 1-year occurrence condition. This similarity demonstrates that the model built in python was successfully able to simulate the extreme condition. This should be sufficient but not enough to conclude on the effectiveness of the model in load calculation in this case.

Figure 3.13 represent an illustration of wind gusting during an extreme gust (ECG) event represented in the standard. Meanwhile the following figure 3.14 is a modelled wind behaviour obtained from the aeroelastic model.

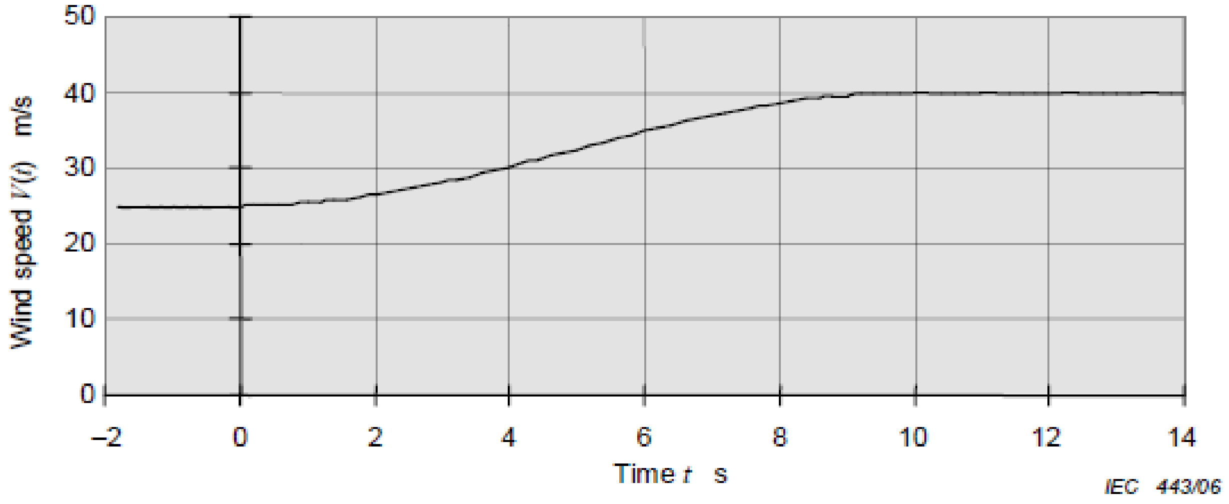


Figure 3.13 Example of an ECG at V_{hub} = 25 m/s [7]

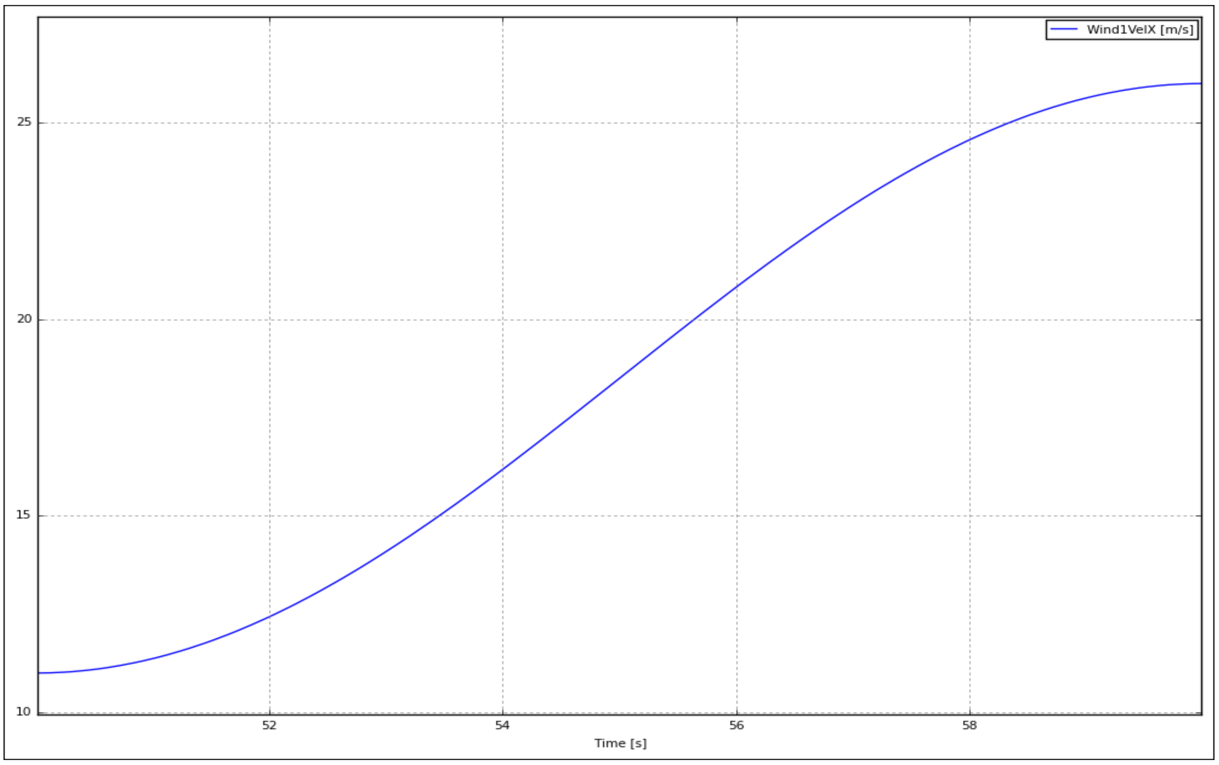


Figure 3.14 Sample 25 kW Turbine DLC 1.5 ECG at V_{hub} = 10.5 m/s

The figure 3.14 clearly shows a sudden gusting with a magnitude of 15 m/s which similarly characterise the event as recommended by the standard. This similarity should sufficient to prove a positive result of the whole model.

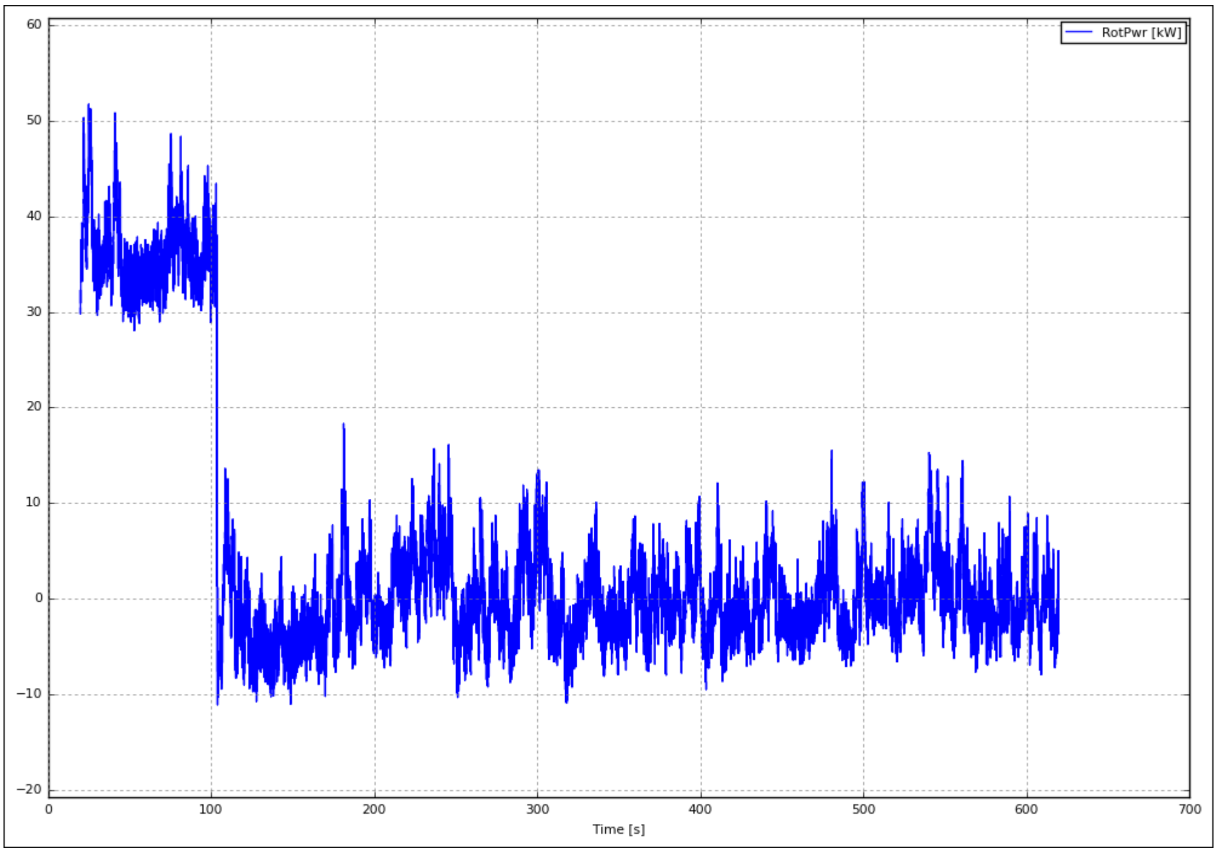


Figure 3.15 DLC 2.1 NTM at 23 m/s showing sudden drop of rotor power to 0 kW.

The figure 3.15 here clearly illustrates a sudden decrease in power production from around 35 to 0 kW caused by the deployment of tip-brake due to system fault.

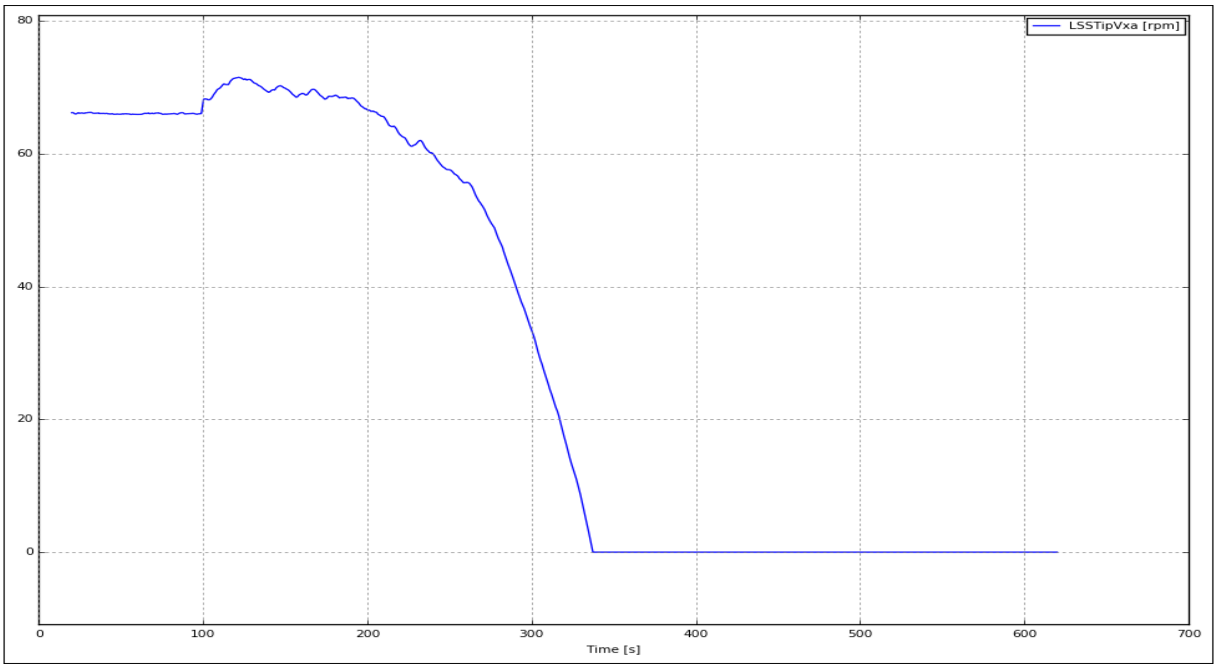


Figure 3.16 DLC 4.1 NTM emergency or manual shutdown at 25 m/s.

The figure 3.16 clearly represents a drop in rotor speed from 65 to 0 rpm characterizing a shutdown case.

The figure 3.17 below is an illustration of the axial force at blade root decrease due to an emergency shutdown situation. Its value normally leads towards zero.

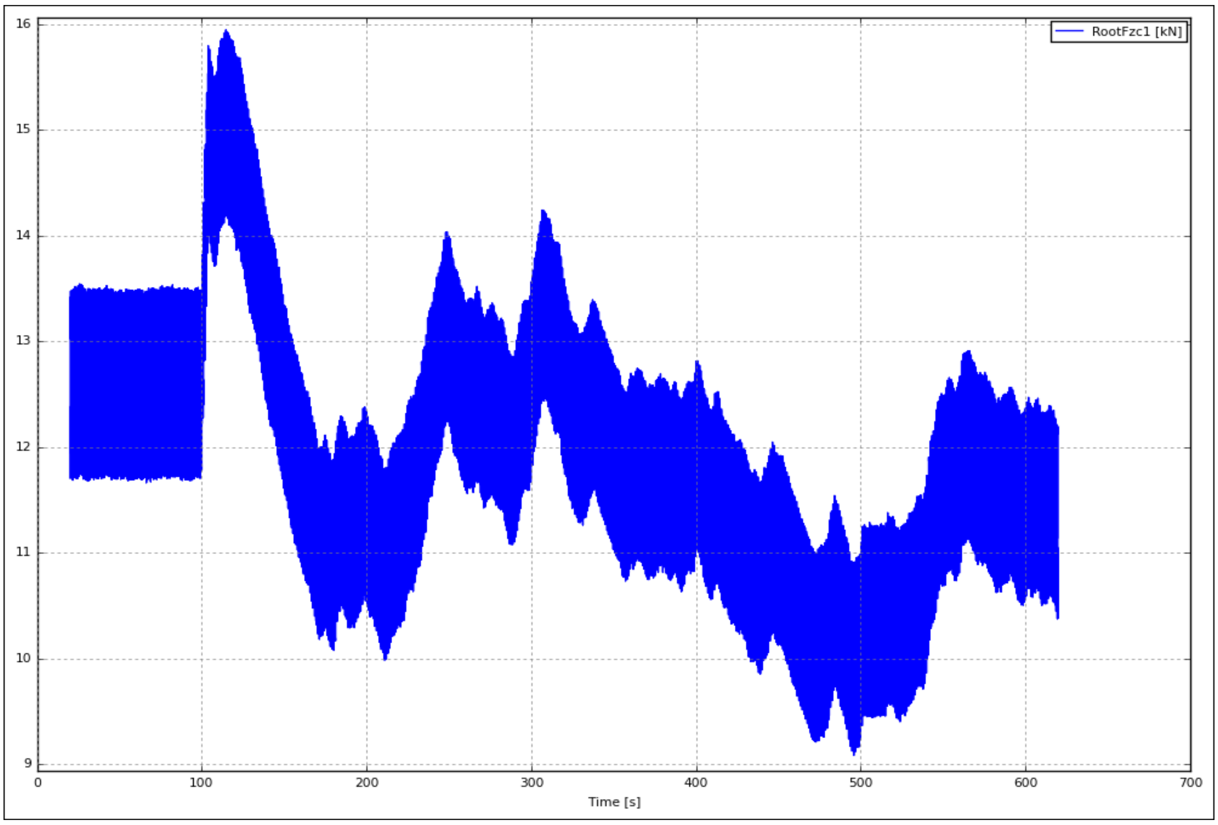


Figure 3.17 DLC 2.2 NTM emergency or manual shutdown case at 23 m/s, axial force on blade root.

All above presented graphs are plotted from DLC simulation output files. If the model is able to simulate the conditions as described in the standard, the loads obtain obtained from the model should reflect turbine response to wind loading and serve as evidence of the proper functionality of the model.

4. COMPARISON BETWEEN SIMPLE LOAD METHODOLOGY AND AEROELASTIC MODEL

In SLM, load calculation is strictly based on conservative equations describing turbine dynamic. The loads obtained are roughly structural and aerodynamic response of the turbine to the average wind speed loading with a slight consideration of external conditions. This limits the model efficiency as machine are generally exposed to relatively high wind speeds of different turbulence intensities, with different directions and recurrent occurrence of extreme transient events. Subsequently, these loads are generally high because of the uncertainty in the estimation method. Due to these uncertainties, the standard requires the loads to be multiplied by a safety factor of 3 as displayed in table 3.2. However, SLM can serve as baseline in turbine loads calculation, providing the manufacturer with an understanding of the stress level to expect on machine components with regards to their ultimate material strength limit. A knowledge of the expected loads on components gives the possibility to proper choice of material enabling assembly of a safer to operation SWT.

The Aeroelastic model DLC on the other hand is a higher fidelity model that better represents the machine and leads to more accurate results. It integrates all possible known situations with existing non-negligible probability of occurrence, for loads identification with more accuracy though very complex and time consuming to build. Due to its higher level of accuracy and the reduced number of uncertainty, loads evaluated from the model are generally lower in comparison to the SLM, safety factor included. The stated reasons explain the lower safety factor 1.35 presented in Table 3.2. Additionally, the model integrates as well all turbine components alongside with their DOF, attached parameters and features, building in its body a complete virtual representation of the turbine giving a push up to its accuracy. It is therefore more realistic.

A comparison of the loads obtained after evaluation of loads on the sample turbine using both models is represented in the table below. A proper comparison will be between maximum values obtained of each wind loading of the same type, moments and forces, from both model.

Table 4.1. Model results comparison

Model results comparison							
Load Type	**Units**	**SLM**	**Aeroelastic Model**	**Percentage Difference SLM / Aeroelastic**	**SLM**	**Aeroelastic Model**	**Percentage Difference SLM / Aeroelastic**
Safety Factor		**Included**			**Excluded**		
Centrifugal Force at the Blade Root	kN	54	21.56	-60.07%	18	14.37	-20.17%
Blade Root Edgewise Bending Moment	kN·m	15.33	6.94	-54.73%	5.11	4.62	-9.59%
Blade Root Flapwise Bending Moment	kN·m	61.71	19.57	-68.29%	20.57	13.05	-36.56%
Maximum Thrust on Shaft	kN	56.97	19.49	-65.79%	18.99	12.99	-31.60%
Shaft Bending Moment	kN·m	28.83	14.70	-49.01%	9.61	9.8	1.98%
Yaw bearing axial force	kN	51.66	19.87	-61.54%	17.22	13.25	-23.05%
Yaw bearing roll moment	kN·m	56.97	11.98	-78.97%	18.99	7.98	-58%
Tower Bending Moment	kN·m	56.97	356.40	299.43%	18.99	237.15	218.16%
Thrust force on Tower	kN	108.63	19.87	-81.7%	36.21	12.81	-23.40%

As noticed in 4.1, loads from SLM are much higher in most cases including and excluding the applied safety factor. This is explained by the high level of uncertainty and low accuracy of the model compared to the Aeroelastic model. Though the quality assurance result (figure 3.1), evaluated using ultimate material stress limits and calculated equivalent stresses, concluded on component safety with SLM method. This shows as well that turbine quality assurance is highly dependent of ultimate material strength. Therefore, in case turbine is revealed to be unsafe using SLM method, a more detail analysis need to be considered using the Aeroelastic model for load calculation, further concluding on component safety by calculating equivalent stresses on components using formulas in table 1.3 and comparing obtained results with ultimate material strength. In conclusion, the Aeroelastic model should be used for reduction of the load obtained from SLM using the high accuracy it offers alongside with its design methodology being more realistic.

5. COMPARISON BETWEEN FAST AND ALTERNATIVE CODE HAWC2

Upon research, no other time marching aeroelastic wind turbine simulation codes were found other than HAWC2 (Horizontal Axis Wind turbine simulation Code 2nd generation). Subsequently, it is the only existing alternative to FAST. It was developed and distributed by the Aeroelastic Design Research Group at DTU Wind Energy in Technical University of Denmark [18]. It has been utilized in several research projects as well as industrial applications.

Similar to FAST, HAWC2 is used for design and verification purposes. It integrates the following properties and features for simulations of wind turbines structural and aerodynamic response in time domain:

- Onshore 1, 2, 3 or more bladed wind turbines;
- Vertical Axis Wind Turbines;
- Pitch and (active) stall controlled wind turbines;
- Guyed support structures;
- Offshore wind turbines;
- Multiple rotor in one simulation;
- Multibody formulation capable of handling multiple degree of freedom like blade torsion;
- Detail aerodynamic BEM model that includes dynamic stall models, skew inflow model, shear effect on the induction, dynamic inflow model and near wake model;
- Hydrodynamic model;
- Water kinematic;
- Wind, turbulence and wake models;
- Default controller provided with a pitch-regulated variable speed controller;
- Eigenvalue analysis at standstill.

The HAWC2 software package comprises an executable file along with all necessary files to run the code. Hence, HAWC2 isn't an open source package. It is available in 3 form of licenses:

- Commercial multiple user license;
- Research single user license for students;
- Research multiple user license for institution.

Table 5.1 below presents the cost of each licenses and support system attached them.

Table 5.1. Cost of HAWC2 licenses [19]

Licenses	Price: first year/annual license fee	E-learning Accounts	Annual support hours
Commercial (multiple user)	40000/25000 Euro	15	40
SME	15000/12500 Euro	15	40
Institutions	5000 Euro	10	20
Student	1000 Euro	1	5

Where SME stands for micro, small and medium-sized enterprises according to European standards, which employ less than 250 individuals with an annual turnover not outpacing 50 million Euro.
Cost wise, HACW2 is an expensive tool not easily affordable neither for small wind turbine manufacturers nor for student interested in mastering the tool for their future career in the wind turbine design analysis. Despite its affordability, HAWC2 is an excellent wind turbine simulation tools that is being used by many companies and institutions and stand-alone researchers around the world providing them technical support program and access to all updates and sub models.

NREL FAST and DTU HAWC2 possess similar objectives, give the possibility to turbine designer to evaluate and foresee all possible loads that will endure wind turbines during their availability period. However, NREL FAST compare to its concurrent, is a completely free availibility tool for download in NREL webpage, where are posted all updates in the code, new versions, all sub-modules updates and new features attached to the code. There, can also be downloaded all available users' guides necessary for understanding the code and its functionality. Additionally, it provides a support Forum platform where users can submit their questions, comments, queries and receive in return support not only from FAST code main developers (Bonnie Jonkman, Jason Jonkman, Katherine Dykes and Matthew Lackner) but also from other users having better commands of the program. See [20] for NREL FAST support platform. The code existence as free resource can be explained by the fact that its development is funded by U.S. Department of Energy.

The above elaborated comparison exhibits the choice of FAST as simulator for studies of structural and aerodynamic response of 25 kW wind turbine considered serving as object in this work. Its existence as free resource represents as well a predominant advantage over its opponent. Small wind turbine manufacturers often face financial difficulties in wind industry, so having to release additional investment to purchase licenses for programs like HAWC2 for their turbine verification purposes will unprofitable for their business. Therefore, the use of FAST combine with Python programming language can reduce expenses on turbine design and manufacturing.

Additionally, the availability of FAST as open source code is also beneficial for student seeking additional knowledge in simulation of wind turbine dynamic. Personally, as a student with low revenue, I had the opportunity to use the code for my studies which would not have been possible if I were to use HAWC2 for this purpose. Thanks to the program, I have a better understanding of operation mode of wind turbine aerodynamic simulators in general and FAST in particular, making me a good fit for a future career in this field.

6. COMPARISON OF PYTHON TO MATLAB

Both NREL FAST and HAWC2 are generally ran in Windows or Linux operating system command lines or through MatLab computing tool. The latest serving as simulation environment commonly used by companies and institutions. It is available as downloadable licenses in MathWorks portal. The table below shows the approximate cost of different available MatLab licenses which can vary according to regions and countries. See MatLab store for evidence of displayed prices in [21].

Table 6.1. Approximated Cost of MatLab licenses

Licenses	**Annual / perpetual cost, Euro**	**Description**
Standard	800 / 2000	For end user individual license and organisations
Academic	250 / 500	For faculties, staffs, or researchers at an educational institution
Student	35	Restricted to if user falls in previous categories
Student suite	69	
Home	119	For personal use

These prices are for the regular software package and will increase with the use of optimization toolbox and add-on products like Simulink, Curve Fitting Toolbox, Control System Toolbox, Image Processing Toolbox, DSP System Toolbox, Instrument Control Toolbox, Parallel Computing Toolbox, Optimization Toolbox, Signal Processing Toolbox, Symbolic Math Toolbox, Statistics and Machine Learning Toolbox and many others each of which generally cost additional 20 Euro.

In parallel, Python is currently a worldwide competing programming language that has proved to provide many similar features compared to MatLab but at an absolutely zero-cost price including all available toolboxes and libraries. It is ranked 1st in PYPL popularity of programming language, see [22], with a share rate of 26.42% and an increasing trend of +5.2% in the last 5 years of existence. Meanwhile, according to the same popularity classification MatLab occupies the 10th position with only 1.98% share as programming language. In conclusion, Python availability as free resource represents the predominant advantage over MatLab. This give reasons to its used as computational tool for the implementation of both models in this work. However, it should be pointed that the complexity of building the Aeroelastic model is similar in both Python and MatLab.

I do believe many wind energy companies will switch to Python in the nearest future due to this revealed popularity as programming environment for simulations of structural and aerodynamic wind turbines response to wind loading.

SUMMARY

In line with addressing the usability of open source computational tools for wind turbine load calculation following IEC-61400-2 standard requirements, an overview on the aerodynamic and structural dynamic of HAWT was first presented emphasizing on the interaction between the environmental conditions and turbine components which is at the essence of the rotation of the rotor induced by the generated aerodynamic thrust, lift and drag forces. These forces were demonstrated to create fatigue and ultimate loads on exposed components during turbine lifespan. With the used of the two load calculation methodologies proscribed by the standard, both the Simplified Load Methodology and Aeroelastic model were built and implemented for a 25 kW wind turbine using freely available simulation tools in comparison to license-based tools.

This thesis essentially addressed the use of Python programming language for building simulation models and optimization of simulation processes alternatively to MatLab package regularly used for similar purposes. Both SLM and Aeroelastic models were built in Python programming environment for load calculation. However, the Aeroelastic model exhibited a particularity characterized by the integration of NREL FAST simulation code making the model more complex to build. Furthermore, the scripts were generalized so to be implemented on variety of horizontal axis SWTs, this increasing the suitability of the models.

A comparison of simulation results from both models revealed that values of similar loads types obtained from the SLM model were up to 62.62% (SF included) and 25.27% (SF excluded) in average higher than their pairs from the Aeroelastic model. This found explanation on the low accuracy of the first model as it bases solely on conservative mathematical equations to describe turbine behaviour, the second being more accurate for its integration of all possible environmental characteristics coupled with turbine parameters and features.

Python as trending programming language, demonstrated its efficiency not only in the execution of wind turbine aerodynamic simulations but also in optimization of the whole process. For the Aeroelastic model, Python revealed to be substantially valuable in the automation processing of the 345 simulations executed alongside with the post-processing statistical computation of obtained results. It essentially helps in reduction of time required for processing at absolutely zero-cost. Meanwhile, processing such analysis using MatLab package would have required the purchase of corresponding licenses bearing in mind additional cost for acquiring additional libraries or modules if needed.

The resulting time series plots, fig. 3.12 and 3.14 of wind behaviour from the Aeroelastic model using NREL FAST show a clear similarity with the considered transient extreme event described in the standard, fig. 3.11 and fig. 3.13. This is a relevant proof of the effectiveness of FAST code in properly representing the model as proscribed by the standard, leading to the conclusion on the correctness of the loads calculated. However, both FAST and HAWC2 are known to perform coupled aerodynamic modelling. The choice of the first code over its concurrent was therefore purely cost-wise.

To summarize, the results obtained from both models efficiently demonstrate the possibility of usage of open source computer-aided tools for conduction of wind project related to analysis of aerodynamic and structural dynamic response of wind turbines to wind loadings at absolutely zero investment cost. The work fulfilled can be beneficial to tree main parties:

- Small wind turbine manufacturers desiring to pass their turbine verification and certification process. They can use both built models for load determination, necessary for the quality assurance verification as required by the standard, saving substantial amount of money in comparison to the use of MatLab and HACW2. Considering the use of the latest tools instead, the manufacturer will have to disburse approximately 15800 Euros annually for both licenses.
- Educational institutions desiring to provide students with detailed knowledge about the dynamics of wind turbines together with deep insight in Python programming language.
- Students and sole individuals willing to get connected to the software part of the wind industry without any financing investment. This will particularly be beneficial to students willing to take a journey in the field of wind energy loads assessment for turbine safety qualification.

This project on its own required considerable amount of time to build till the final point, as it integrating tonnes of information to process, acknowledge and implement. Its constitutes of several parts mainly simulation related files and diverse python scripts with diverse executable programs as presented in appendixes below. All these materials together can be hugely confusing to individuals desiring to use the models for their turbine certification purposes or perhaps for learning purposes merely. Therefore, the question of the possibility of development a so called ONE-CLICK operation arises. Meaning, the development of a computer-based software that integrates all programs used in these models, in which the user will just be required to insert his turbine's parameters and wind field characteristics, with a single click obtains from the software all loads requested. I believe such project would be very innovative, creative and extremely beneficial for the wind energy industry.

REFERENCE

1. Kelly R., Rana A., Hannah E. M., Laura E. W., Adam Br. HIGHLIGHTS of the REN21 Renewables 2018 Global Status Report in perspective.
2. Haliade-X Offshore Wind Turbine Platform. [WWW] https://www.ge.com/renewableenergy/wind-energy/offshore-wind/haliade-x-offshore-turbine , (02.05.2019)
3. Manwell J. F. and McGowan J. G. Wind Energy Explained: Theory, Design and Application - Edition 2.
4. Vitale A.J. and Rossi A.P. Software Tool for Horizontal-Axis Wind Turbine Simulation.
5. Dr. Meyers J. M., Dr. Fletcher D. G. and Dubief Dr. Y. Lift and Drag on an Airfoil.
6. Chawin C. and Warawut T. Determination of Wind Turbine Blade Flapwise Bending Dynamics.
7. CENELEC European Committee for Electrotechnical Standardization. IEC-61400-2:2014 Standard - Wind turbines - Part 2: Small wind turbines.
8. David W. Small Wind Turbines Analysis, Design, and Application.
9. David S. Wind Energy Handbook.
10. Zhiyu J., Torgeir M., Zhen G. A Comparative Study of Shutdown Procedures on the Dynamic Responses of Wind Turbines.
11. What is CAE | Computer-Aided Engineering? [WWW] https://www.simscale.com/docs/content/simwiki/general/whatiscae.html , (04.05.2019).
12. The Most Popular CAD Software. [WWW] https://tutorial45.com/the-most-popular-cad-software/ , (04.05.2019).
13. QBlade Wind Turbine Design and Simulation. [Online] http://www.q-blade.org/#second-screen , (04.05.2019).
14. What is Ashes? [WWW] https://www.simis.io/#Products_Ashes_Features , (04.05.2019).
15. Jason M. J., Marshall L. B. FAST 7 User’s Guide.
16. Germanisher Lloyd WindEnergie. Guideline for the Certification of Wind Turbines.
17. Kelley N.D. and Jonkman B.J. Overview of the TurbSim Stochastic Inflow Turbulence Simulator Version 1.21 (Revised February 1, 2007).
18. Welcome to HAWC2. [Online] http://www.hawc2.dk/hawc2-info, (05/05/2019).
19. HACW2 Price information. [Online] http://www.hawc2.dk/hawc2-info/price-information, (05/05/2019)
20. NWTC NREL's National Wind Technology Centre. [WWW] https://wind.nrel.gov/forum/wind/ , (04.05.2019).

21. MathWorks: Pricing and Licensing. [WWW] https://se.mathworks.com/pricing-licensing.html?prodcode=ML&intendeduse=comm , (04.05.2019)
22. PYPL Popularity of Programming Language. [WWW] http://pypl.github.io/PYPL.html , (04.05.2019).

APPENDIX 1 SLM

A 1.1. Simple Load Model Script

```
#Created on Thu Mar 29 14:18:43 2018
#Michael Keumatio Lontsie
from pylab import *
import matplotlib.pyplot as plt

#Adaptation from Simple Load Model Spreadsheet (SLM)
# 25kW turbine
#Turbine data
Air_Density = 1.2250
g = 9.8100 #Gravitational acceleration
V_ref = 37.5 # Reference wind speed
V_ave = 7.5 # Average wind speed
B = 3.0 # Number of Blades
A_proj_B = 2.5 # Total Platform Area of the Blades
A_proj_T = 6.8 # Total Tower Projection Area
C_d = 1.5 #Drag Coefficient of the Blades
C_l_max = 2.0 # Maximum Lift Coefficent of the Blades
C_T = 0.5 # Thrust Coefficient
C_f = 1.5# Force Coefficient
n_max = 70.0 # Maximum Rotor Speed
n_design = 65.0 # Design Rotor Speed
m_B = 82.0 # Single Blade Mass
R_cog = 2.37 # Distance from Blade centre of gravity to rotor axis
L_rb = 0.45 # Distance between the rotor centre and Firt Bearing
L_rt = 0.89 # Distance between the rotor centre and the yaw axis
Gear = 23.25 # Gearbox Ratio (enter 1.0 for no gearbox)
High_Speed_Gearbox_Brake = True # Enter "T" if brake is on high speed side of the gearbox, otherwise "F"
M_brake = 250.0 # Brake Torque (enter 0.0 for no brake)
P_design = 22.0 # Design Power
G = 2 # Short Circuit Torque Factor
Blade_Stationary_during_Parking = True # Type "T" if blades are stationary during parking, otherwise "F"
Hub_Radius = 0.3
Hub_Mass = 102.0
Pitch_Bearing_Mass = 54.0

# Parameters Calculated from Input Data
R = 6.2 + Hub_Radius # Blade Tip Radius
I_B = R_cog**2*m_B*3 # Second Moment of Inertia for each Blade
m_r = 3*m_B+Hub_Mass # Rotor Mass ( All the blades plus Hub)
V_design = 1.4*V_ave # Design Wind Speed
V_e50 = 1.4*V_ref # 50 years Extreme Wind Speed
Lamda_e50 = n_max*pi*R/(30*V_e50) # 50 years Extreme Tip Speed Ratio
Lamda_design = R*pi*n_design/(30*V_design) # Design Tip Ratio
Nu = 0.85 # Drive_Train_Efficiency
Q_design = 30000*P_design/(n_design*pi*Nu) #Design Torgue
A_proj = pi*R**2 # Projected Area (Turbine swept area)
Omega_n_design = n_design*pi/30 # Design Rotational Speed of the Rotor
Omega_n_max = pi*n_max/30 # Max Possible Rotor Speed
Omega_yaw_max = 0.036 # Maximum yaw rate
e_r = 0.124 # Eccentricity of the Rotor Center of Mass
if High_Speed_Gearbox_Brake == True:
    Eff_M_brake = Gear*M_brake
else:
    Eff_M_brake = M_brake

# Loads from SLM
# Load Cas A- Fatigue Loads onBlades and Rotor Shaft
# Blade Loads
DeltaF_zB = 2*m_B*R_cog*Omega_n_design**2 # Centrifugal Force at the Blade Root (z-axis)
DeltaM_xB = Q_design/B+2*m_B*g*R_cog # Lead-lag Root Bending Moment (x-axis)
DeltaM_yB = Lamda_design*Q_design/B #Flapwise Root Bending Moment (y-axis)

# Shaft Loads
```

```
DeltaF_x_shaft = 1.5*(Lamda_design*Q_design)/R # Thrust on shaft (x-axis)
DeltaM_x_shaft = Q_design+2*m_r*g*e_r # Shaft Moment about x-axis
DeltaM_shaft = 2*m_r*g*L_rb+R*DeltaF_x_shaft/6.0 #Shaft Moment
# Load Case B- Blade and Rotor Shaft Loads during Yaw
M_yB_CaseB =
m_B*Omega_yaw_max**2*L_rt*R_cog+2*Omega_yaw_max*I_B*Omega_n_design+R*DeltaF_x_shaft/9
# Flapwise Root Bending Moment (y-axis)
if B == 2:
   M_shaft_CaseB = 4*Omega_yaw_max*Omega_n_design*I_B+m_r*g*L_rb*+R*DeltaF_x_shaft/6
else:
   M_shaft_CaseB = B*Omega_yaw_max*Omega_n_design*I_B+m_r*g*L_rb+R*DeltaF_x_shaft/6 # Bending
Moment of the shaft
# Load Case C- Yaw Error Load on Blades
M_yB_CaseC =
0.125*Air_Density*A_proj_B*C_l_max*R**3*Omega_n_design**2*(1+4/(3*Lamda_design)+1/Lamda_design**2)
# Flapwise Root bending moment on the blade
# Load Case D- Maximum Thrust on Shaft
F_x_shaft_CaseD = C_T*3.125*Air_Density*V_ave**2*pi*R**2 # Maximum Thrust on Shaft

# Load Case E- Maximum Rotational Speed
F_zB = m_B*Omega_n_max**2*R_cog #Centrifugal Force at the Blade Root (z-axis)
M_shaft_CaseE = m_r*g*L_rb+m_r*e_r*Omega_n_max**2*L_rb # Bending Moment on the Shaft

# Load Case F- Short at Load Connection
M_x_shaft_CaseF = G*Q_design # Bending Moment on the Shaft
M_xB_CaseF = M_x_shaft_CaseF/B # Lead-lag Root Bending Moment (x-axis)

# Load Case G- Shutdown Braking
if M_brake>0:
   M_x_shaft_CaseG = Eff_M_brake+Q_design
else:
   M_x_shaft_CaseG = "n/a" # Bending Moment on Shaft
if M_x_shaft_CaseG == "n/a":
   M_xB_CaseG = "n/a"
else:
   M_xB_CaseG = M_x_shaft_CaseG/B+m_B*g*R_cog # Lead-lag Root Bending Moment (x-axis)

# Load Case H- Parked Wind Loads during idlling
if Blade_Stationary_during_Parking == True:
   M_yB_CaseH = C_d*0.25*Air_Density*V_e50**2*A_proj_B*R
else:
   M_yB_CaseH = C_l_max*Air_Density*V_e50**2*A_proj_B*R # Flapwise Root Bending Moment (y-axis)
if Blade_Stationary_during_Parking == True:
   F_x_shaft_CaseH = 0.5*B*Air_Density*C_d*V_e50**2*A_proj_B
else:
   F_x_shaft_CaseH = 0.17*B*Lamda_e50**2*Air_Density*V_e50**2 # Maximum Thrust on Shaft
if Blade_Stationary_during_Parking == True:
   M_T = 0.5*B*Air_Density*C_d*V_e50**2*A_proj_B # Maximum Tower Bending Moment
else:
   M_T = 0.17*B*Lamda_e50**2*Air_Density*V_e50**2

F_T = 0.5*Air_Density*C_f*V_e50**2*A_proj_T # Maximum Thrust on Shaft

#   Additional Data
Diameter_shaft = 0.12
A_shaft = pi*Diameter_shaft**2/4 # cross sectional area of the shaft
I_x_shaft = pi*Diameter_shaft**4/64.0 #The second moment of Inertia of the shaft
W_shaft = 2*I_x_shaft/Diameter_shaft # Section modulus of the shaft
A_B = pi*(0.42/2)**2 # Cross sectionarea of the Blade root
I_xxB = pi*(0.42**4)/64.0 # Ixx for the Blade
c_xB = 2.77-0.1 # x- distance from the blade centroid to the maximuim stress point
I_yyB = I_xxB # Iyy for the Blade
c_yB = 0.01 # Y- distance from Blade centroid to the maximum stress point
W_xB = I_xxB/c_xB # Blade x- section modulus
W_yB = I_yyB/c_yB # Blade y_section modulus
f_kB = 200.0 # Ultimate Material strength for the Blades
f_k_shaft = 635.0 # Ultimate material strength for the Shaft
```

```
# Additional Data for Fatigue calculation
Yrs = 20.0 # Years
T_d = Yrs*365.25*24*60.0*60.0 # Design life of the Turbine
n_i = (n_design*B*T_d)/60.0 # Number of Fatigue Cycles
N_shaft = 1e10 # Number of Cycles of Failure as a Function of Stress (Shaft)
N_blade = 1.23e13 # Number of Cycles of Failure as a Function of Stress (Blade)
W_B = 2*I_xxB/0.42 # Ixx,Iyy assume circular x-sect w/root specs

# Calculation of the Equivalent Stresses
#Load Case A- Fatigue Loads on Blades and Rotor Shaft
Eq_Stress_blade_CaseA = ((DeltaF_zB/A_B)+(DeltaM_xB/W_xB)+(DeltaM_yB/W_yB))/1000000
Eq_Stress_shaft_CaseA =
math.sqrt(((DeltaF_x_shaft/A_shaft)+DeltaM_shaft/W_shaft)**2+0.75*(DeltaM_x_shaft/W_shaft)**2)/1000000

# Load Case B- Blade ond Rotor Shaft Load during Yaw
Eq_Stress_blade_CaseB = (M_yB_CaseB/W_yB)/1000000.0
Eq_Stress_shaft_CaseB = (M_shaft_CaseB/W_shaft)/1000000.0

#Load Case C- Yaw Error Load on Blades
Eq_Stress_blade_CaseC = (M_yB_CaseC/W_yB)/1000000.0

# Load Case D - Maximum Thrust on Shaft
Max_Thrust_Shaft = (F_x_shaft_CaseD/A_shaft)/1000000.0

# Load Case E- Maximum Rotational Speed
Max_Rotational_Speed_Blade = (F_zB/A_B)/1000000.0
Max_Rotational_Speed_Shaft = (M_shaft_CaseE/W_shaft)/1000000.0

# Load Case F- Short at Load Connection
Eq_Stress_Short_Load_Connect_blade = (M_xB_CaseF/W_xB)/1000000.0
Eq_Stress_Short_Load_Connect_shaft = (0.5*math.sqrt(3)*M_x_shaft_CaseF/W_shaft)/1000000.0

# Load Case G- Shutdown Braking
if M_x_shaft_CaseG == "n/a":
   Eq_Stress_Shortdown_Braking_blade = "n/a"
else: Eq_Stress_Shortdown_Braking_blade = (M_xB_CaseG/W_xB)/1000000.0
if M_xB_CaseG == "n/a":
   Eq_Stress_Shortdown_Braking_shaft = "n/a"
else: Eq_Stress_Shortdown_Braking_shaft = math.sqrt(0.75*M_x_shaft_CaseG/W_shaft)/1000000.0

# Load Case H- Parked Wind Load during Idling
Eq_Stress_Parked_blade = (M_yB_CaseH/W_yB)/1000000.0
Eq_Stress_Parked_shaft = (F_x_shaft_CaseH/A_shaft)/1000000.0

# Load Case A- Fatique Loads on Blades and Rotor Shaft
Eq_Stress_blade = ((DeltaF_zB/A_B)+math.sqrt(DeltaM_xB**2+DeltaM_yB**2)/W_B)/1000000.0

# Partial Safety Factors (PSF) for SLM
g_ff = 1.0 # Partial Safety Factor for Fatigue Loads
g_fu = 3.0 # Partial Safety Factor for Ultimate Loading

# Calculation of the material PSF
Full_charact_blade_Material = False # Otherwise type True
gamma_mB_f = 10.0 # Blade Fatigue Strength Partial Safety Factor
gamma_muB_u = 3.0 # Blade Ultimate Strength Partial Safety Factor
Full_charact_shaft_Material = 1.0 # Otherwise type 0
if Full_charact_shaft_Material == 0:
   gamma_mShaft_f = 10.0
else: gamma_mShaft_f = 1.25 # Shaft Fatigue Strength Partial Safety Factor
if Full_charact_shaft_Material == 0:
  gamma_mShaft_u = 3.0
else: gamma_mShaft_u = 1.1 # Shaft Ultimate Strength Partial Safety Factor

# Material Strengths ( with Factors of Safety)
Ultimate_Strength_Blade_Material = f_kB/gamma_muB_u/g_fu
Ultimate_Strength_Shaft_Material = f_k_shaft/gamma_mShaft_u/g_fu
```

```
# Calculation of Fatigue
s_iB = Eq_Stress_blade_CaseA # Blade Stress Level
Associated_s_iB = s_iB*gamma_mB_f # Associated Blade Stress Level
N_B = 9.81e15 # Number of Cycles for Failure at this Stress
s_i_shaft = Eq_Stress_shaft_CaseA # Shaft Stress Level
Associated_s_i_shaft = s_i_shaft*gamma_mShaft_f # Associated Shaft Stress Level
N_shaft = "inf" # Number of Cycles to Failure at this Stress

# SIMPLE LOAD MODEL RESULTS
#Load Case A- Fatigue Loads on Blades and Rotor Shaft
Fatigue_damage_limit_blade = 1.0
if Ultimate_Strength_Blade_Material == "inf":
    Fatigue_damage_blade = "infinite life"
else: Fatigue_damage_blade = n_i/N_B
if logical_or(Fatigue_damage_blade < 1, Fatigue_damage_blade == "infinite life"):
    Conclusion_blade_CaseA = "SAFE"
else: Conclusion_blade_CaseA = "FAIL"
Fatigue_damage_limit_shaft = 1.0
if N_shaft == "inf":
    Fatigue_damage_shaft = "infinite life"
else: Fatigue_damage_shaft = n_i/N_shaft
if logical_or(Fatigue_damage_shaft < 1, Fatigue_damage_shaft == "infinite life"):
    Conclusion_shaft_CaseA = "SAFE"
else: Conclusion_shaft_CaseA = "FAIL"

# Load Case B- Blade ond Rotor Shaft Load during Yaw
Material_stress_limit_blade = Ultimate_Strength_Blade_Material
Calculated_Stress_blade_CaseB = Eq_Stress_blade_CaseB
if Material_stress_limit_blade > Calculated_Stress_blade_CaseB:
   Conclusion_blade_CaseB = "SAFE"
else: Conclusion_blade_CaseB = "FAIL"
Material_stress_limit_shaft = Ultimate_Strength_Shaft_Material
Calculated_Stress_shaft_CaseB = Eq_Stress_shaft_CaseB
if Material_stress_limit_shaft > Calculated_Stress_shaft_CaseB:
   Conclusion_shaft_CaseB = "SAFE"
else: Conclusion_shaft_CaseB = "FAIL"

#Load Case C- Yaw Error Load on Blades
Material_stress_limit_blade = Ultimate_Strength_Blade_Material
Calculated_Stress_blade_CaseC = Eq_Stress_blade_CaseC
if Material_stress_limit_blade > Calculated_Stress_blade_CaseC:
    Conclusion_CaseC = "SAFE"
else: Conclusion_CaseC = "FAIL"

# Load Case D - Maximum Thrust on Shaft
Material_stress_limit_shaft = Ultimate_Strength_Shaft_Material
Calculated_Stress_shaft_CaseD = Max_Thrust_Shaft
if Material_stress_limit_shaft > Calculated_Stress_shaft_CaseD:
    Conclusion_CaseD = "SAFE"
else: Conclusion_CaseD = "FAIL"

# Load Case E- Maximum Rotational Speed
Material_stress_limit_blade = Ultimate_Strength_Blade_Material
Calculated_Stress_blade_CaseE = Max_Rotational_Speed_Blade
if Material_stress_limit_blade > Calculated_Stress_blade_CaseE:
   Conclusion_blade_CaseE = "SAFE"
else: Conclusion_blade_CaseB = "FAIL"
Material_stress_limit_shaft = Ultimate_Strength_Shaft_Material
Calculated_Stress_shaft_CaseE = Max_Rotational_Speed_Shaft
if Material_stress_limit_shaft > Calculated_Stress_shaft_CaseE:
   Conclusion_shaft_CaseE = "SAFE"
else: Conclusion_shaft_CaseE = "FAIL"

# Load Case F- Short at Load Connection
Material_stress_limit_blade = Ultimate_Strength_Blade_Material
Calculated_Stress_blade_CaseF = Eq_Stress_Short_Load_Connect_blade
```

```
if Material_stress_limit_blade > Calculated_Stress_blade_CaseF:
  Conclusion_blade_CaseF = "SAFE"
else: Conclusion_blade_CaseF = "FAIL"
Material_stress_limit_shaft = Ultimate_Strength_Shaft_Material
Calculated_Stress_shaft_CaseF = Eq_Stress_Short_Load_Connect_shaft
if Material_stress_limit_shaft > Calculated_Stress_shaft_CaseF:
  Conclusion_shaft_CaseF = "SAFE"
else: Conclusion_shaft_CaseF = "FAIL"

# Load Case G- Shortdown Braking
Material_stress_limit_blade = Ultimate_Strength_Blade_Material
Calculated_Stress_blade_CaseG = Eq_Stress_Shortdown_Braking_blade
if Material_stress_limit_blade > Calculated_Stress_blade_CaseG:
  Conclusion_blade_CaseG = "SAFE"
else: Conclusion_blade_CaseG = "FAIL"
Material_stress_limit_shaft = Ultimate_Strength_Shaft_Material
Calculated_Stress_shaft_CaseG = Eq_Stress_Short_Load_Connect_shaft
if Material_stress_limit_shaft > Calculated_Stress_shaft_CaseG:
  Conclusion_shaft_CaseG = "SAFE"
else: Conclusion_shaft_CaseG = "FAIL"

# Load Case H- Parked Wind Load during Idling
Material_stress_limit_blade = Ultimate_Strength_Blade_Material
Calculated_Stress_blade_CaseH = Eq_Stress_Parked_blade
if Material_stress_limit_blade > Calculated_Stress_blade_CaseH:
  Conclusion_blade_CaseG = "SAFE"
else: Conclusion_blade_CaseG = "FAIL"
Material_stress_limit_shaft = Ultimate_Strength_Shaft_Material
Calculated_Stress_shaft_CaseH = Eq_Stress_Parked_shaft
if Material_stress_limit_shaft > Calculated_Stress_shaft_CaseH:
  Conclusion_shaft_CaseG = "SAFE"
else: Conclusion_shaft_CaseG = "FAIL"

Calculated_Stress_blade_CaseA = Eq_Stress_blade_CaseA
Calculated_Stress_shaft_CaseA = Eq_Stress_shaft_CaseA

Factor = Calculated_Stress_shaft_CaseB/Material_stress_limit_shaft
shaft_factor = Factor**(1/3)

def y1(Eq_Stress_blade_CaseA, Eq_Stress_blade_CaseB, Eq_Stress_blade_CaseC,
Max_Rotational_Speed_Blade, Eq_Stress_Short_Load_Connect_blade, Eq_Stress_Parked_blade,
Eq_Stress_Shortdown_Braking_blade):
   print y1
   return;
def ys1(Eq_Stress_shaft_CaseA, Eq_Stress_shaft_CaseB, Max_Thrust_Shaft, Max_Rotational_Speed_Shaft,
Eq_Stress_Short_Load_Connect_shaft, Eq_Stress_Parked_shaft, Eq_Stress_Shortdown_Braking_shaft):
   print ys1
   return;
```

A 1.2. Loads results calculated using SLM

Table 1. Obtained loads evaluation results from SLM model

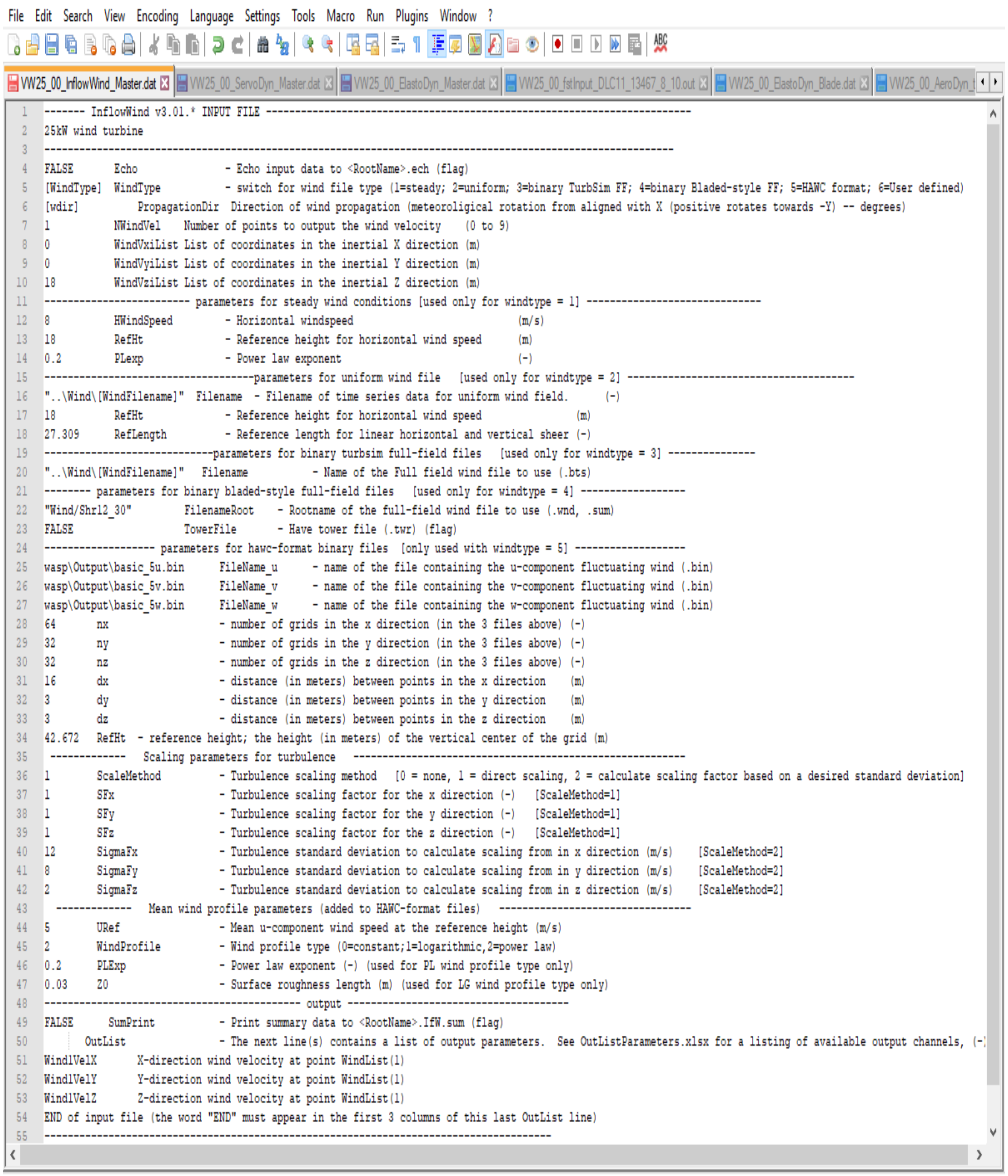
Load cases	Parts	Variables	Results	Description
Fatigue				
Load case A	Blade	$\Delta F_{zB}, N$	18008.44	Centrifugal Force at the Blade Root (z-axis)
		$\Delta M_{xB}, Nm$	5080.43	Edgewise Root Bending Moment (x-axis)
		$\Delta M_{yB}, Nm$	5340.80	Flapwise Root Bending Moment (y-axis)
	Shaft	$\Delta F_{x-shaft}, N$	3697.48	Thrust on shaft (x-axis)
		$\Delta M_{x-shaft}, Nm$	174649.08	Shaft Moment about x-axis
		$\Delta M_{shaft}, Nm$	7078.09	Shaft Moment
Ultimate				
Load case B Yawing	Blade	M_{yB}, Nm	3347.81	Flapwise Root Bending Moment (y-axis)
	Shaft	M_{shaft}, Nm	6557.62	Bending Moment of the shaft
Load Case C Yaw Error	Blade	M_{yB}, Nm	13373.06	Flapwise Root bending moment on the blade
Load Case D Maximum Thrust	Shaft	$F_{x-shaft}, N$	14290.76	Maximum Thrust on Shaft
Load Case E Maximum Rotational Speed	Blade	F_{zB}, N	10442.77	Centrifugal Force at the Blade Root (z-axis)
	Shaft	M_{shaft}, Nm	2579.68	Bending Moment on the Shaft
Load Case F Short at Load Connection	Shaft	$M_{x-shaft}, Nm$	7604.87	Bending Moment on the Shaft
	Blade	M_{xB}, Nm	2534.96	Edgewise Root Bending Moment (x-axis)
Load Case G Shutdown Braking	Shaft	$M_{x-shaft}, Nm$	9614.93	Bending Moment on Shaft
	Blade	M_{xB}, Nm	5111.45	Edgewise Root Bending Moment (x-axis)
Load Case H Parked Wind Loads during idling	Blade	M_{yB}, Nm	20574.98	Flapwise Root Bending Moment (y-axis)
	Shaft	$F_{x-shaft}, N$	18992.29	Maximum Thrust on Shaft
	Tower	M_T, Nm	18992.29	Maximum Tower Bending Moment
		F_T, N	36211.46	Thrust Force on Tower

APPENDIX 2 SLM equivalent stress plotting

```
#from pylab import *
import numpy as np
import matplotlib.pyplot as plt

# import them
import Python_SLM_VW25kW
import Python_SLM_VW25kW_BlackWind
from Python_SLM_VW25kW import y1
from Python_SLM_VW25kW import ys1

y1 = (Eq_Stress_blade_CaseA, Eq_Stress_blade_CaseB, Eq_Stress_blade_CaseC,
Max_Rotational_Speed_Blade, Eq_Stress_Short_Load_Connect_blade, Eq_Stress_Parked_blade,
Eq_Stress_Shortdown_Braking_blade)
#from Python_SLM_VW25kW_BlackWind import y2
#y2 = (Eq_Stress_blade_CaseA2, Eq_Stress_blade_CaseB2, Eq_Stress_blade_CaseC2,
Max_Rotational_Speed_Blade2, Eq_Stress_Short_Load_Connect_blade2, Eq_Stress_Parked_blade2,
Eq_Stress_Shortdown_Braking_blade2)

ys1 = (Eq_Stress_shaft_CaseA, Eq_Stress_shaft_CaseB, Max_Thrust_Shaft, Max_Rotational_Speed_Shaft,
Eq_Stress_Short_Load_Connect_shaft, Eq_Stress_Parked_shaft, Eq_Stress_Shortdown_Braking_shaft)
n_groups = 7

# create plot
fig, ax = plt.subplots()
index = np.arange(n_groups)
bar_width = 0.4
opacity = 0.8

rects1 = plt.bar(index, y1, bar_width, alpha=opacity, color='b', label='25kWTurbine_blade')
rects2 = plt.bar(index + bar_width, ys1, bar_width, alpha=opacity, color='g', label='25kWTurbine_shaft')
#rects2 = plt.bar(index + bar_width, y2, bar_width, alpha=opacity, color='g', label='25kWTurbine+BlackWind')

plt.xlabel('Simple Load Cases')
plt.ylabel('Stress[MPa]')
plt.title('Calculated Equivalent Stresses on Blades and Shaft')
plt.xticks(index + bar_width, ('LoadA', 'LoadB', 'LoadC/D', 'LoadE', 'LoadF', 'LoadG', 'LoadH'))
plt.legend()

plt.tight_layout()
plt.show()
```

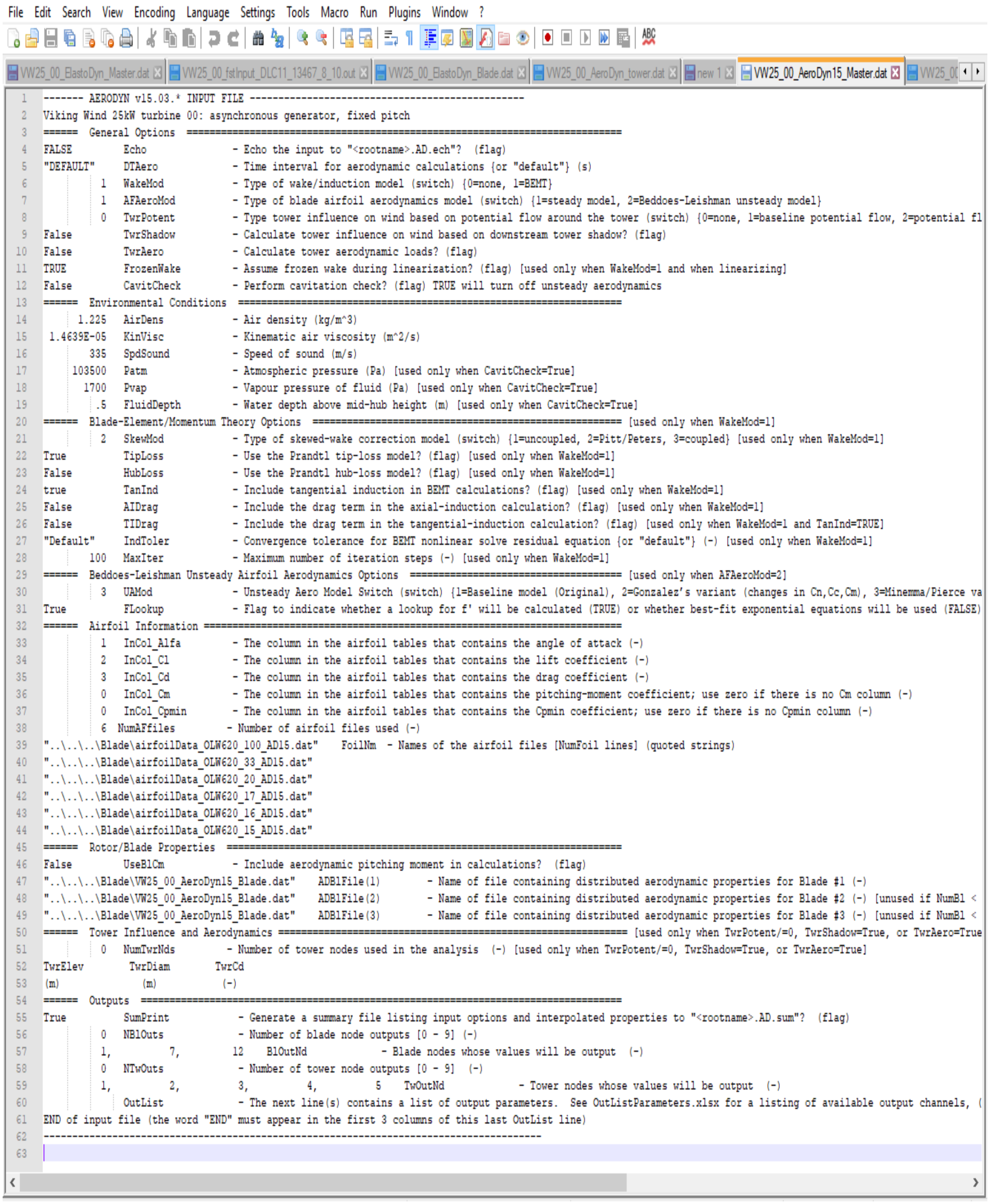
APPENDIX 3 FAST Master Files

A 3.1 InflowWind input master file


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
VW25_00_InflowWind_Master.dat
VW25_00_ServoDyn_Master.dat
VW25_00_ElastoDyn_Master.dat
VW25_00_fstInput_DLC11_13467_8_10.out
VW25_00_ElastoDyn_Blade.dat
VW25_00_AeroDyn_

```
------- InflowWind v3.01.* INPUT FILE -------------------------------------------------------------------------
25kW wind turbine
---------------------------------------------------------------------------------------------------
FALSE       Echo              - Echo input data to <RootName>.ech (flag)
[WindType]  WindType          - switch for wind file type (1=steady; 2=uniform; 3=binary TurbSim FF; 4=binary Bladed-style FF; 5=HAWC format; 6=User defined)
[wdir]          PropagationDir  Direction of wind propagation (meteoroligical rotation from aligned with X (positive rotates towards -Y) -- degrees)
1           NWindVel    Number of points to output the wind velocity    (0 to 9)
0           WindVxiList List of coordinates in the inertial X direction (m)
0           WindVyiList List of coordinates in the inertial Y direction (m)
18          WindVziList List of coordinates in the inertial Z direction (m)
------------------------- parameters for steady wind conditions [used only for windtype = 1] ------------------------------
8           HWindSpeed        - Horizontal windspeed                            (m/s)
18          RefHt             - Reference height for horizontal wind speed      (m)
0.2         PLexp             - Power law exponent                              (-)
-----------------------------------parameters for uniform wind file   [used only for windtype = 2] ---------------------------------------
"..\Wind\[WindFilename]"  Filename  - Filename of time series data for uniform wind field.      (-)
18          RefHt             - Reference height for horizontal wind speed                (m)
27.309      RefLength         - Reference length for linear horizontal and vertical sheer (-)
-----------------------------parameters for binary turbsim full-field files   [used only for windtype = 3] ---------------
"..\Wind\[WindFilename]"   Filename           - Name of the Full field wind file to use (.bts)
-------- parameters for binary bladed-style full-field files   [used only for windtype = 4] -----------------
"Wind/Shr12_30"         FilenameRoot    - Rootname of the full-field wind file to use (.wnd, .sum)
FALSE                   TowerFile       - Have tower file (.twr) (flag)
------------------- parameters for hawc-format binary files  [only used with windtype = 5] ------------------
wasp\Output\basic_5u.bin      FileName_u      - name of the file containing the u-component fluctuating wind (.bin)
wasp\Output\basic_5v.bin      FileName_v      - name of the file containing the v-component fluctuating wind (.bin)
wasp\Output\basic_5w.bin      FileName_w      - name of the file containing the w-component fluctuating wind (.bin)
64       nx                   - number of grids in the x direction (in the 3 files above) (-)
32       ny                   - number of grids in the y direction (in the 3 files above) (-)
32       nz                   - number of grids in the z direction (in the 3 files above) (-)
16       dx                   - distance (in meters) between points in the x direction    (m)
3        dy                   - distance (in meters) between points in the y direction    (m)
3        dz                   - distance (in meters) between points in the z direction    (m)
42.672   RefHt  - reference height; the height (in meters) of the vertical center of the grid (m)
 -------------   Scaling parameters for turbulence   ---------------------------------------------------------
1        ScaleMethod          - Turbulence scaling method   [0 = none, 1 = direct scaling, 2 = calculate scaling factor based on a desired standard deviation]
1        SFx                  - Turbulence scaling factor for the x direction (-)   [ScaleMethod=1]
1        SFy                  - Turbulence scaling factor for the y direction (-)   [ScaleMethod=1]
1        SFz                  - Turbulence scaling factor for the z direction (-)   [ScaleMethod=1]
12       SigmaFx              - Turbulence standard deviation to calculate scaling from in x direction (m/s)    [ScaleMethod=2]
8        SigmaFy              - Turbulence standard deviation to calculate scaling from in y direction (m/s)    [ScaleMethod=2]
2        SigmaFz              - Turbulence standard deviation to calculate scaling from in z direction (m/s)    [ScaleMethod=2]
  -------------   Mean wind profile parameters (added to HAWC-format files)   --------------------------------
5        URef                 - Mean u-component wind speed at the reference height (m/s)
2        WindProfile          - Wind profile type (0=constant;1=logarithmic,2=power law)
0.2      PLExp                - Power law exponent (-) (used for PL wind profile type only)
0.03     Z0                   - Surface roughness length (m) (used for LG wind profile type only)
--------------------------------------------- output ---------------------------------------
FALSE      SumPrint           - Print summary data to <RootName>.IfW.sum (flag)
       OutList                - The next line(s) contains a list of output parameters.  See OutListParameters.xlsx for a listing of available output channels, (-)
Wind1VelX      X-direction wind velocity at point WindList(1)
Wind1VelY      Y-direction wind velocity at point WindList(1)
Wind1VelZ      Z-direction wind velocity at point WindList(1)
END of input file (the word "END" must appear in the first 3 columns of this last OutList line)
---------------------------------------------------------------------------------------
```

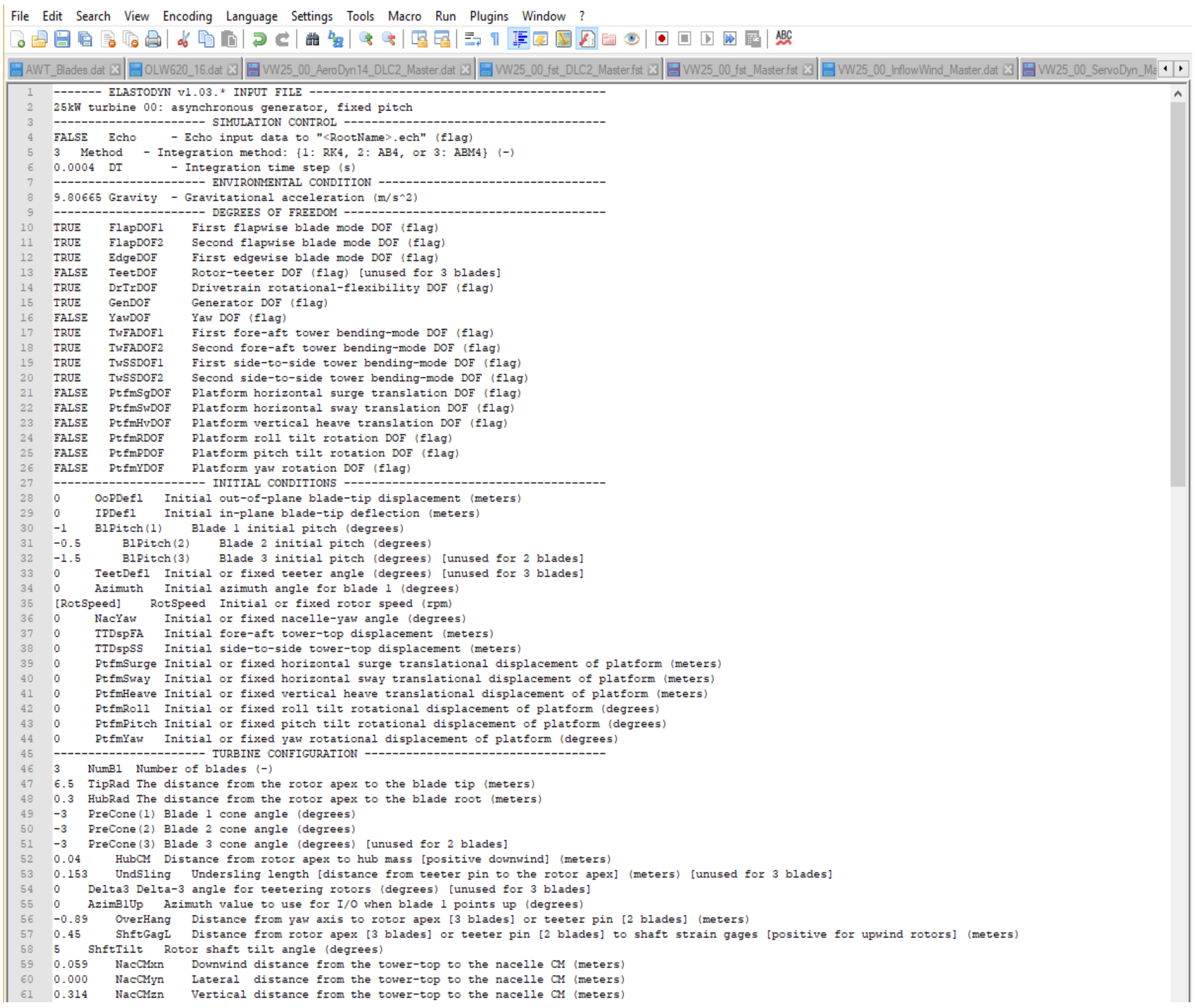
A 3.2 AeroDyn input master file



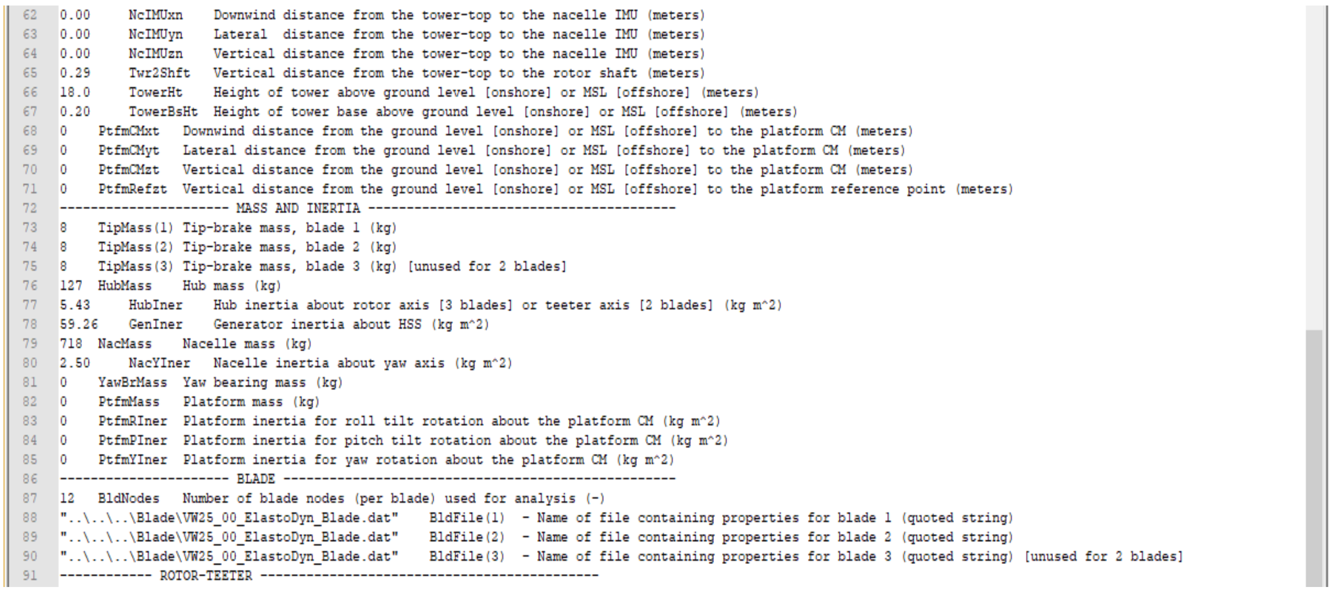
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
VW25_00_ElastoDyn_Master.dat
VW25_00_fstInput_DLC11_13467_8_10.out
VW25_00_ElastoDyn_Blade.dat
VW25_00_AeroDyn_tower.dat
new 1
VW25_00_AeroDyn15_Master.dat
VW25_0

```
------- AERODYN v15.03.* INPUT FILE ------------------------------------------------
Viking Wind 25kW turbine 00: asynchronous generator, fixed pitch
======  General Options  ============================================================================
FALSE          Echo                 - Echo the input to "<rootname>.AD.ech"?  (flag)
"DEFAULT"      DTAero               - Time interval for aerodynamic calculations {or "default"} (s)
          1    WakeMod              - Type of wake/induction model (switch) {0=none, 1=BEMT}
          1    AFAeroMod            - Type of blade airfoil aerodynamics model (switch) {1=steady model, 2=Beddoes-Leishman unsteady model}
          0    TwrPotent            - Type tower influence on wind based on potential flow around the tower (switch) {0=none, 1=baseline potential flow, 2=potential fl
False          TwrShadow            - Calculate tower influence on wind based on downstream tower shadow? (flag)
False          TwrAero              - Calculate tower aerodynamic loads? (flag)
TRUE           FrozenWake           - Assume frozen wake during linearization? (flag) [used only when WakeMod=1 and when linearizing]
False          CavitCheck           - Perform cavitation check? (flag) TRUE will turn off unsteady aerodynamics
======  Environmental Conditions  ===================================================================
      1.225    AirDens              - Air density (kg/m^3)
  1.4639E-05   KinVisc              - Kinematic air viscosity (m^2/s)
        335    SpdSound             - Speed of sound (m/s)
     103500    Patm                 - Atmospheric pressure (Pa) [used only when CavitCheck=True]
       1700    Pvap                 - Vapour pressure of fluid (Pa) [used only when CavitCheck=True]
         .5    FluidDepth           - Water depth above mid-hub height (m) [used only when CavitCheck=True]
======  Blade-Element/Momentum Theory Options  ====================================================== [used only when WakeMod=1]
          2    SkewMod              - Type of skewed-wake correction model (switch) {1=uncoupled, 2=Pitt/Peters, 3=coupled} [used only when WakeMod=1]
True           TipLoss              - Use the Prandtl tip-loss model? (flag) [used only when WakeMod=1]
False          HubLoss              - Use the Prandtl hub-loss model? (flag) [used only when WakeMod=1]
true           TanInd               - Include tangential induction in BEMT calculations? (flag) [used only when WakeMod=1]
False          AIDrag               - Include the drag term in the axial-induction calculation? (flag) [used only when WakeMod=1]
False          TIDrag               - Include the drag term in the tangential-induction calculation? (flag) [used only when WakeMod=1 and TanInd=TRUE]
"Default"      IndToler             - Convergence tolerance for BEMT nonlinear solve residual equation {or "default"} (-) [used only when WakeMod=1]
        100    MaxIter              - Maximum number of iteration steps (-) [used only when WakeMod=1]
======  Beddoes-Leishman Unsteady Airfoil Aerodynamics Options  ===================================== [used only when AFAeroMod=2]
          3    UAMod                - Unsteady Aero Model Switch (switch) {1=Baseline model (Original), 2=Gonzalez’s variant (changes in Cn,Cc,Cm), 3=Minemma/Pierce va
True           FLookup              - Flag to indicate whether a lookup for f' will be calculated (TRUE) or whether best-fit exponential equations will be used (FALSE)
======  Airfoil Information =========================================================================
          1    InCol_Alfa           - The column in the airfoil tables that contains the angle of attack (-)
          2    InCol_Cl             - The column in the airfoil tables that contains the lift coefficient (-)
          3    InCol_Cd             - The column in the airfoil tables that contains the drag coefficient (-)
          0    InCol_Cm             - The column in the airfoil tables that contains the pitching-moment coefficient; use zero if there is no Cm column (-)
          0    InCol_Cpmin          - The column in the airfoil tables that contains the Cpmin coefficient; use zero if there is no Cpmin column (-)
          6  NumAFfiles            - Number of airfoil files used (-)
"..\..\..\Blade\airfoilData_OLW620_100_AD15.dat"    FoilNm  - Names of the airfoil files [NumFoil lines] (quoted strings)
"..\..\..\Blade\airfoilData_OLW620_33_AD15.dat"
"..\..\..\Blade\airfoilData_OLW620_20_AD15.dat"
"..\..\..\Blade\airfoilData_OLW620_17_AD15.dat"
"..\..\..\Blade\airfoilData_OLW620_16_AD15.dat"
"..\..\..\Blade\airfoilData_OLW620_15_AD15.dat"
======  Rotor/Blade Properties  =====================================================================
False          UseBlCm              - Include aerodynamic pitching moment in calculations?  (flag)
"..\..\..\Blade\VW25_00_AeroDyn15_Blade.dat"    ADBlFile(1)        - Name of file containing distributed aerodynamic properties for Blade #1 (-)
"..\..\..\Blade\VW25_00_AeroDyn15_Blade.dat"    ADBlFile(2)        - Name of file containing distributed aerodynamic properties for Blade #2 (-) [unused if NumBl <
"..\..\..\Blade\VW25_00_AeroDyn15_Blade.dat"    ADBlFile(3)        - Name of file containing distributed aerodynamic properties for Blade #3 (-) [unused if NumBl <
======  Tower Influence and Aerodynamics ============================================================= [used only when TwrPotent/=0, TwrShadow=True, or TwrAero=True
          0    NumTwrNds            - Number of tower nodes used in the analysis  (-) [used only when TwrPotent/=0, TwrShadow=True, or TwrAero=True]
TwrElev        TwrDiam        TwrCd
(m)              (m)            (-)
======  Outputs  ====================================================================================
True           SumPrint             - Generate a summary file listing input options and interpolated properties to "<rootname>.AD.sum"?  (flag)
          0    NBlOuts              - Number of blade node outputs [0 - 9] (-)
          1,           7,          12    BlOutNd              - Blade nodes whose values will be output  (-)
          0    NTwOuts              - Number of tower node outputs [0 - 9]  (-)
          1,           2,           3,           4,           5    TwOutNd              - Tower nodes whose values will be output  (-)
               OutList              - The next line(s) contains a list of output parameters.  See OutListParameters.xlsx for a listing of available output channels, (
END of input file (the word "END" must appear in the first 3 columns of this last OutList line)
---------------------------------------------------------------------------------------

```

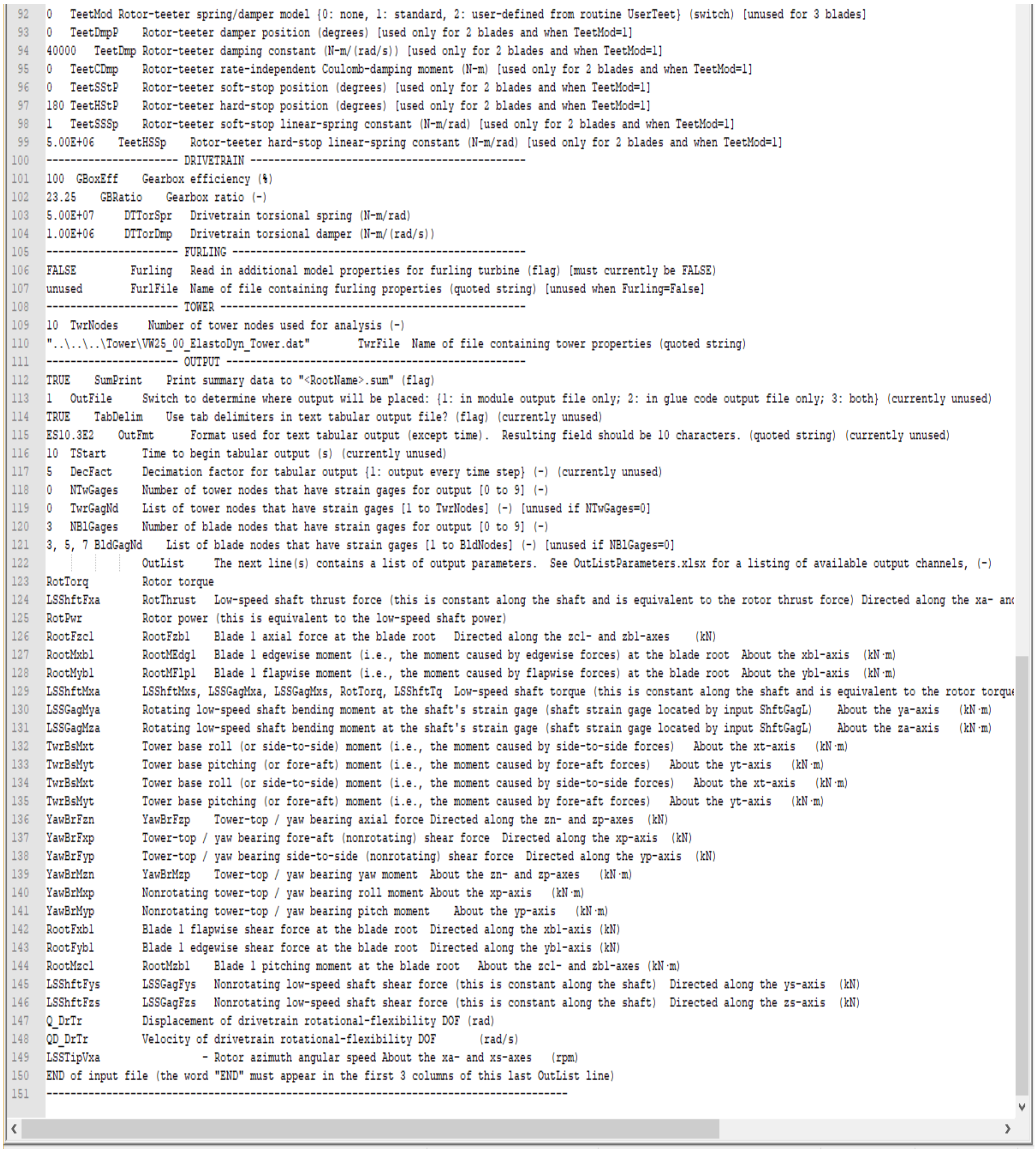
A 3.3 ElastoDyn input master file


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
AWT_Blades.dat
OLW620_16.dat
VW25_00_AeroDyn14_DLC2_Master.dat
VW25_00_fst_DLC2_Master.fst
VW25_00_fst_Master.fst
VW25_00_InflowWind_Master.dat
VW25_00_ServoDyn_Ma

```
------- ELASTODYN v1.03.* INPUT FILE -------------------------------------------
25kW turbine 00: asynchronous generator, fixed pitch
---------------------- SIMULATION CONTROL --------------------------------------
FALSE   Echo       - Echo input data to "<RootName>.ech" (flag)
3   Method    - Integration method: {1: RK4, 2: AB4, or 3: ABM4} (-)
0.0004  DT         - Integration time step (s)
---------------------- ENVIRONMENTAL CONDITION ---------------------------------
9.80665 Gravity  - Gravitational acceleration (m/s^2)
---------------------- DEGREES OF FREEDOM --------------------------------------
TRUE    FlapDOF1    First flapwise blade mode DOF (flag)
TRUE    FlapDOF2    Second flapwise blade mode DOF (flag)
TRUE    EdgeDOF     First edgewise blade mode DOF (flag)
FALSE   TeetDOF     Rotor-teeter DOF (flag) [unused for 3 blades]
TRUE    DrTrDOF     Drivetrain rotational-flexibility DOF (flag)
TRUE    GenDOF      Generator DOF (flag)
FALSE   YawDOF      Yaw DOF (flag)
TRUE    TwFADOF1    First fore-aft tower bending-mode DOF (flag)
TRUE    TwFADOF2    Second fore-aft tower bending-mode DOF (flag)
TRUE    TwSSDOF1    First side-to-side tower bending-mode DOF (flag)
TRUE    TwSSDOF2    Second side-to-side tower bending-mode DOF (flag)
FALSE   PtfmSgDOF   Platform horizontal surge translation DOF (flag)
FALSE   PtfmSwDOF   Platform horizontal sway translation DOF (flag)
FALSE   PtfmHvDOF   Platform vertical heave translation DOF (flag)
FALSE   PtfmRDOF    Platform roll tilt rotation DOF (flag)
FALSE   PtfmPDOF    Platform pitch tilt rotation DOF (flag)
FALSE   PtfmYDOF    Platform yaw rotation DOF (flag)
---------------------- INITIAL CONDITIONS --------------------------------------
0      OoPDefl   Initial out-of-plane blade-tip displacement (meters)
0      IPDefl    Initial in-plane blade-tip deflection (meters)
-1     BlPitch(1)    Blade 1 initial pitch (degrees)
-0.5       BlPitch(2)    Blade 2 initial pitch (degrees)
-1.5       BlPitch(3)    Blade 3 initial pitch (degrees) [unused for 2 blades]
0      TeetDefl  Initial or fixed teeter angle (degrees) [unused for 3 blades]
0      Azimuth   Initial azimuth angle for blade 1 (degrees)
[RotSpeed]    RotSpeed  Initial or fixed rotor speed (rpm)
0      NacYaw    Initial or fixed nacelle-yaw angle (degrees)
0      TTDspFA   Initial fore-aft tower-top displacement (meters)
0      TTDspSS   Initial side-to-side tower-top displacement (meters)
0      PtfmSurge Initial or fixed horizontal surge translational displacement of platform (meters)
0      PtfmSway  Initial or fixed horizontal sway translational displacement of platform (meters)
0      PtfmHeave Initial or fixed vertical heave translational displacement of platform (meters)
0      PtfmRoll  Initial or fixed roll tilt rotational displacement of platform (degrees)
0      PtfmPitch Initial or fixed pitch tilt rotational displacement of platform (degrees)
0      PtfmYaw   Initial or fixed yaw rotational displacement of platform (degrees)
---------------------- TURBINE CONFIGURATION -----------------------------------
3    NumBl  Number of blades (-)
6.5  TipRad The distance from the rotor apex to the blade tip (meters)
0.3  HubRad The distance from the rotor apex to the blade root (meters)
-3   PreCone(1) Blade 1 cone angle (degrees)
-3   PreCone(2) Blade 2 cone angle (degrees)
-3   PreCone(3) Blade 3 cone angle (degrees) [unused for 2 blades]
0.04     HubCM  Distance from rotor apex to hub mass [positive downwind] (meters)
0.153    UndSling   Undersling length [distance from teeter pin to the rotor apex] (meters) [unused for 3 blades]
0    Delta3 Delta-3 angle for teetering rotors (degrees) [unused for 3 blades]
0    AzimB1Up   Azimuth value to use for I/O when blade 1 points up (degrees)
-0.89    OverHang   Distance from yaw axis to rotor apex [3 blades] or teeter pin [2 blades] (meters)
0.45     ShftGagL   Distance from rotor apex [3 blades] or teeter pin [2 blades] to shaft strain gages [positive for upwind rotors] (meters)
5    ShftTilt   Rotor shaft tilt angle (degrees)
0.059    NacCMxn    Downwind distance from the tower-top to the nacelle CM (meters)
0.000    NacCMyn    Lateral  distance from the tower-top to the nacelle CM (meters)
0.314    NacCMzn    Vertical distance from the tower-top to the nacelle CM (meters)
```

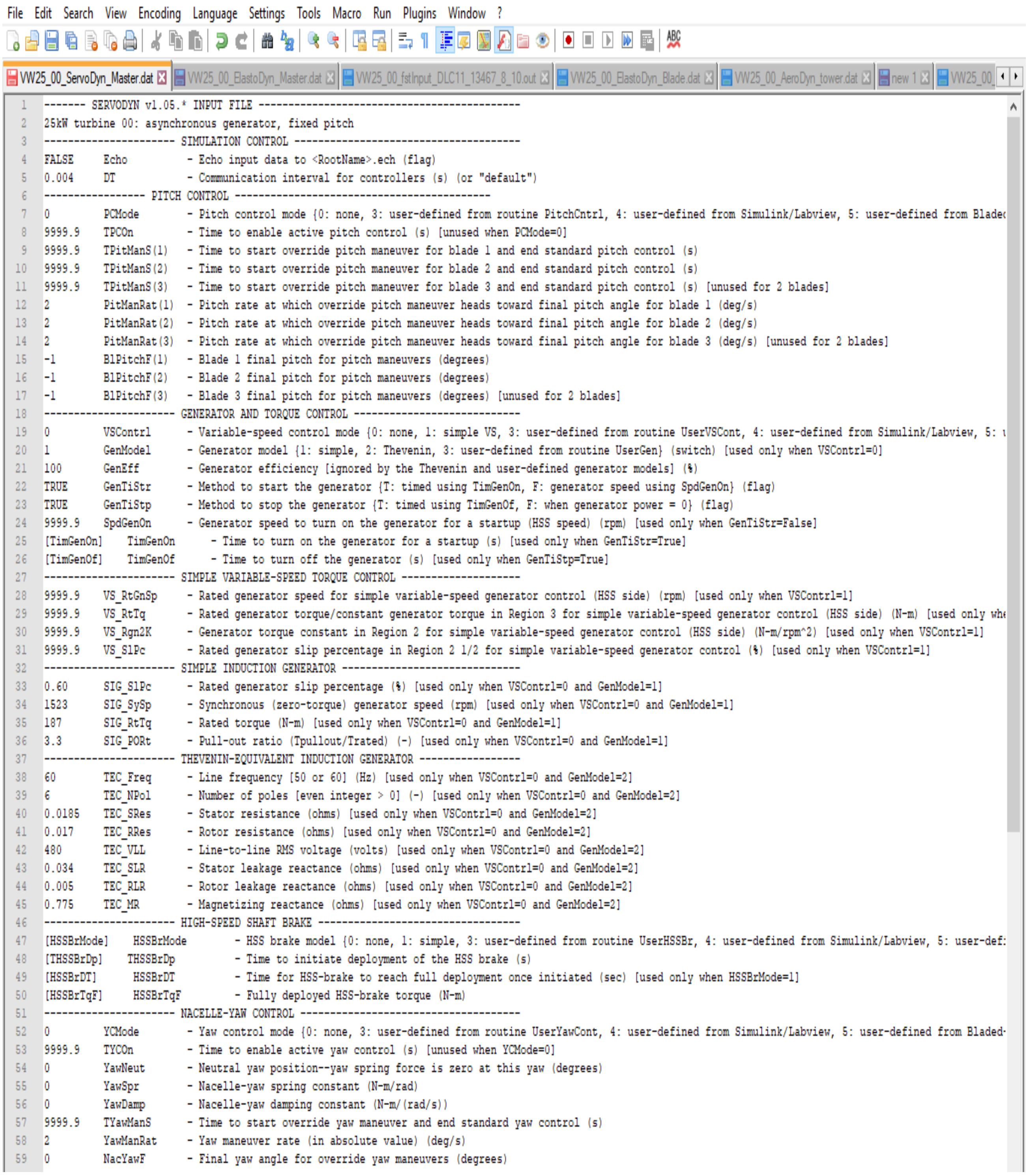


```
0.00     NcIMUxn    Downwind distance from the tower-top to the nacelle IMU (meters)
0.00     NcIMUyn    Lateral  distance from the tower-top to the nacelle IMU (meters)
0.00     NcIMUzn    Vertical distance from the tower-top to the nacelle IMU (meters)
0.29     Twr2Shft   Vertical distance from the tower-top to the rotor shaft (meters)
18.0     TowerHt    Height of tower above ground level [onshore] or MSL [offshore] (meters)
0.20     TowerBsHt  Height of tower base above ground level [onshore] or MSL [offshore] (meters)
0    PtfmCMxt   Downwind distance from the ground level [onshore] or MSL [offshore] to the platform CM (meters)
0    PtfmCMyt   Lateral distance from the ground level [onshore] or MSL [offshore] to the platform CM (meters)
0    PtfmCMzt   Vertical distance from the ground level [onshore] or MSL [offshore] to the platform CM (meters)
0    PtfmRefzt  Vertical distance from the ground level [onshore] or MSL [offshore] to the platform reference point (meters)
---------------------- MASS AND INERTIA ----------------------------------------
8    TipMass(1) Tip-brake mass, blade 1 (kg)
8    TipMass(2) Tip-brake mass, blade 2 (kg)
8    TipMass(3) Tip-brake mass, blade 3 (kg) [unused for 2 blades]
127  HubMass    Hub mass (kg)
5.43     HubIner    Hub inertia about rotor axis [3 blades] or teeter axis [2 blades] (kg m^2)
59.26    GenIner    Generator inertia about HSS (kg m^2)
718  NacMass    Nacelle mass (kg)
2.50     NacYIner   Nacelle inertia about yaw axis (kg m^2)
0    YawBrMass  Yaw bearing mass (kg)
0    PtfmMass   Platform mass (kg)
0    PtfmRIner  Platform inertia for roll tilt rotation about the platform CM (kg m^2)
0    PtfmPIner  Platform inertia for pitch tilt rotation about the platform CM (kg m^2)
0    PtfmYIner  Platform inertia for yaw rotation about the platform CM (kg m^2)
---------------------- BLADE ---------------------------------------------------
12   BldNodes   Number of blade nodes (per blade) used for analysis (-)
"..\..\..\Blade\VW25_00_ElastoDyn_Blade.dat"    BldFile(1)  - Name of file containing properties for blade 1 (quoted string)
"..\..\..\Blade\VW25_00_ElastoDyn_Blade.dat"    BldFile(2)  - Name of file containing properties for blade 2 (quoted string)
"..\..\..\Blade\VW25_00_ElastoDyn_Blade.dat"    BldFile(3)  - Name of file containing properties for blade 3 (quoted string) [unused for 2 blades]
------------ ROTOR-TEETER ------------------------------------------
```

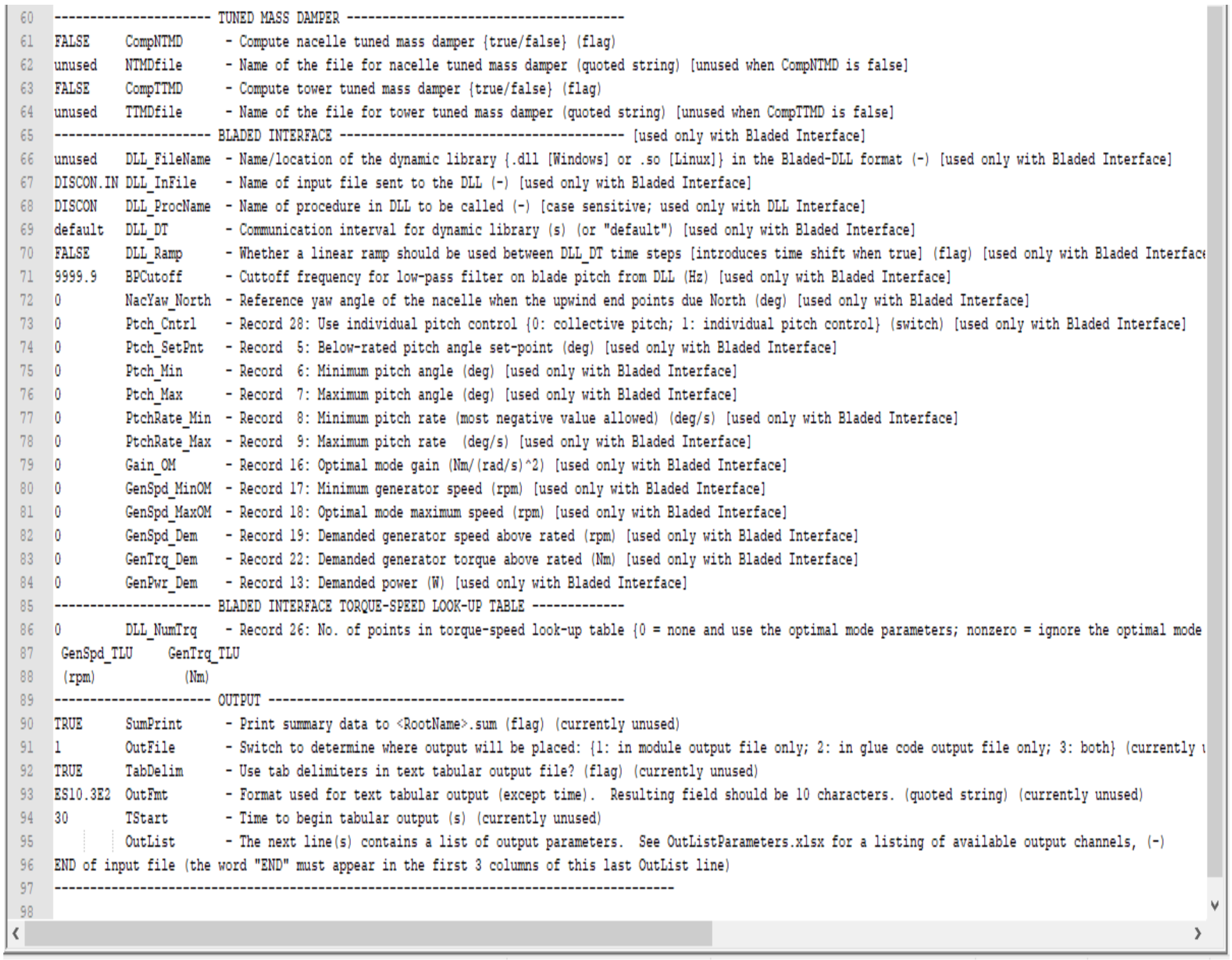


```
0   TeetMod Rotor-teeter spring/damper model {0: none, 1: standard, 2: user-defined from routine UserTeet} (switch) [unused for 3 blades]
0   TeetDmpP    Rotor-teeter damper position (degrees) [used only for 2 blades and when TeetMod=1]
40000   TeetDmp Rotor-teeter damping constant (N-m/(rad/s)) [used only for 2 blades and when TeetMod=1]
0   TeetCDmp    Rotor-teeter rate-independent Coulomb-damping moment (N-m) [used only for 2 blades and when TeetMod=1]
0   TeetSStP    Rotor-teeter soft-stop position (degrees) [used only for 2 blades and when TeetMod=1]
180 TeetHStP    Rotor-teeter hard-stop position (degrees) [used only for 2 blades and when TeetMod=1]
1   TeetSSSp    Rotor-teeter soft-stop linear-spring constant (N-m/rad) [used only for 2 blades and when TeetMod=1]
5.00E+06    TeetHSSp    Rotor-teeter hard-stop linear-spring constant (N-m/rad) [used only for 2 blades and when TeetMod=1]
---------------------- DRIVETRAIN ----------------------------------------------
100  GBoxEff    Gearbox efficiency (%)
23.25    GBRatio    Gearbox ratio (-)
5.00E+07    DTTorSpr   Drivetrain torsional spring (N-m/rad)
1.00E+06    DTTorDmp   Drivetrain torsional damper (N-m/(rad/s))
---------------------- FURLING -------------------------------------------------
FALSE        Furling   Read in additional model properties for furling turbine (flag) [must currently be FALSE)
unused       FurlFile  Name of file containing furling properties (quoted string) [unused when Furling=False]
---------------------- TOWER ---------------------------------------------------
10  TwrNodes     Number of tower nodes used for analysis (-)
"..\..\..\Tower\VW25_00_ElastoDyn_Tower.dat"        TwrFile  Name of file containing tower properties (quoted string)
---------------------- OUTPUT --------------------------------------------------
TRUE    SumPrint    Print summary data to "<RootName>.sum" (flag)
1   OutFile     Switch to determine where output will be placed: {1: in module output file only; 2: in glue code output file only; 3: both} (currently unused)
TRUE    TabDelim    Use tab delimiters in text tabular output file? (flag) (currently unused)
ES10.3E2    OutFmt      Format used for text tabular output (except time).  Resulting field should be 10 characters. (quoted string) (currently unused)
10  TStart      Time to begin tabular output (s) (currently unused)
5   DecFact     Decimation factor for tabular output {1: output every time step} (-) (currently unused)
0   NTwGages    Number of tower nodes that have strain gages for output [0 to 9] (-)
0   TwrGagNd    List of tower nodes that have strain gages [1 to TwrNodes] (-) [unused if NTwGages=0]
3   NBlGages    Number of blade nodes that have strain gages for output [0 to 9] (-)
3, 5, 7 BldGagNd    List of blade nodes that have strain gages [1 to BldNodes] (-) [unused if NBlGages=0]
                OutList     The next line(s) contains a list of output parameters.  See OutListParameters.xlsx for a listing of available output channels, (-)
RotTorq         Rotor torque
LSShftFxa       RotThrust   Low-speed shaft thrust force (this is constant along the shaft and is equivalent to the rotor thrust force) Directed along the xa- anc
RotPwr          Rotor power (this is equivalent to the low-speed shaft power)
RootFzc1        RootFzb1    Blade 1 axial force at the blade root   Directed along the zc1- and zb1-axes    (kN)
RootMxb1        RootMEdg1   Blade 1 edgewise moment (i.e., the moment caused by edgewise forces) at the blade root  About the xb1-axis  (kN·m)
RootMyb1        RootMFlp1   Blade 1 flapwise moment (i.e., the moment caused by flapwise forces) at the blade root  About the yb1-axis  (kN·m)
LSShftMxa       LSShftMxs, LSSGagMxa, LSSGagMxs, RotTorq, LSShftTq  Low-speed shaft torque (this is constant along the shaft and is equivalent to the rotor torque
LSSGagMya       Rotating low-speed shaft bending moment at the shaft's strain gage (shaft strain gage located by input ShftGagL)   About the ya-axis   (kN·m)
LSSGagMza       Rotating low-speed shaft bending moment at the shaft's strain gage (shaft strain gage located by input ShftGagL)   About the za-axis   (kN·m)
TwrBsMxt        Tower base roll (or side-to-side) moment (i.e., the moment caused by side-to-side forces)   About the xt-axis   (kN·m)
TwrBsMyt        Tower base pitching (or fore-aft) moment (i.e., the moment caused by fore-aft forces)   About the yt-axis   (kN·m)
TwrBsMxt        Tower base roll (or side-to-side) moment (i.e., the moment caused by side-to-side forces)   About the xt-axis   (kN·m)
TwrBsMyt        Tower base pitching (or fore-aft) moment (i.e., the moment caused by fore-aft forces)   About the yt-axis   (kN·m)
YawBrFzn        YawBrFzp    Tower-top / yaw bearing axial force Directed along the zn- and zp-axes  (kN)
YawBrFxp        Tower-top / yaw bearing fore-aft (nonrotating) shear force  Directed along the xp-axis  (kN)
YawBrFyp        Tower-top / yaw bearing side-to-side (nonrotating) shear force  Directed along the yp-axis  (kN)
YawBrMzn        YawBrMzp    Tower-top / yaw bearing yaw moment  About the zn- and zp-axes   (kN·m)
YawBrMxp        Nonrotating tower-top / yaw bearing roll moment About the xp-axis   (kN·m)
YawBrMyp        Nonrotating tower-top / yaw bearing pitch moment    About the yp-axis   (kN·m)
RootFxb1        Blade 1 flapwise shear force at the blade root  Directed along the xb1-axis (kN)
RootFyb1        Blade 1 edgewise shear force at the blade root  Directed along the yb1-axis (kN)
RootMzc1        RootMzb1    Blade 1 pitching moment at the blade root   About the zc1- and zb1-axes (kN·m)
LSShftFys       LSSGagFys   Nonrotating low-speed shaft shear force (this is constant along the shaft)  Directed along the ys-axis  (kN)
LSShftFzs       LSSGagFzs   Nonrotating low-speed shaft shear force (this is constant along the shaft)  Directed along the zs-axis  (kN)
Q_DrTr          Displacement of drivetrain rotational-flexibility DOF (rad)
QD_DrTr         Velocity of drivetrain rotational-flexibility DOF       (rad/s)
LSSTipVxa                   - Rotor azimuth angular speed About the xa- and xs-axes    (rpm)
END of input file (the word "END" must appear in the first 3 columns of this last OutList line)
---------------------------------------------------------------------------------------
```

A 3.4 ServoDyn input master file

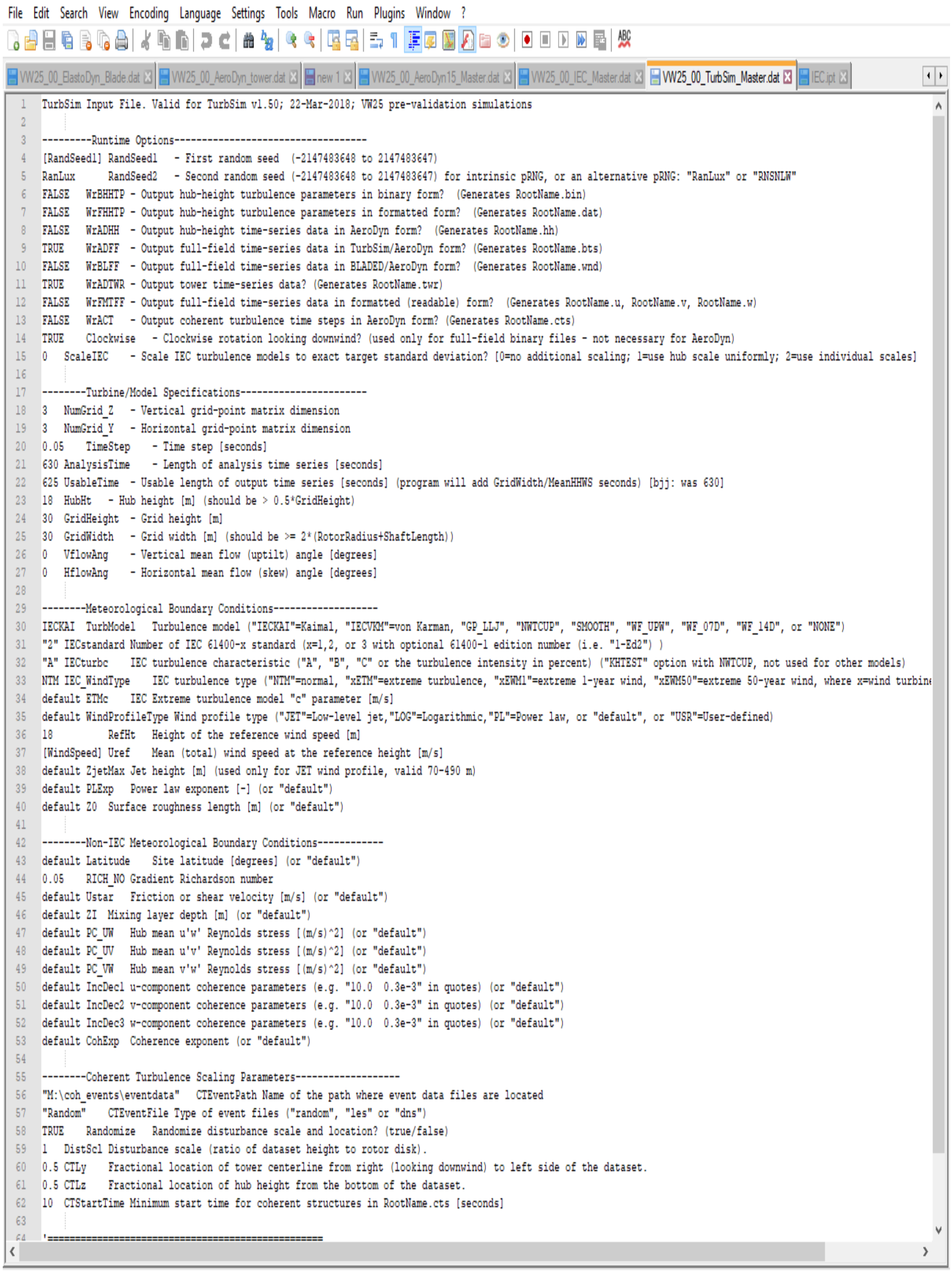

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
VW25_00_ServoDyn_Master.dat
VW25_00_ElastoDyn_Master.dat
VW25_00_fstInput_DLC11_13467_8_10.out
VW25_00_ElastoDyn_Blade.dat
VW25_00_AeroDyn_tower.dat
new 1
VW25_00_

```
------- SERVODYN v1.05.* INPUT FILE --------------------------------------------
25kW turbine 00: asynchronous generator, fixed pitch
---------------------- SIMULATION CONTROL --------------------------------------
FALSE      Echo           - Echo input data to <RootName>.ech (flag)
0.004      DT             - Communication interval for controllers (s) (or "default")
----------------- PITCH CONTROL ------------------------------------------------
0          PCMode         - Pitch control mode {0: none, 3: user-defined from routine PitchCntrl, 4: user-defined from Simulink/Labview, 5: user-defined from Bladec
9999.9     TPCOn          - Time to enable active pitch control (s) [unused when PCMode=0]
9999.9     TPitManS(1)    - Time to start override pitch maneuver for blade 1 and end standard pitch control (s)
9999.9     TPitManS(2)    - Time to start override pitch maneuver for blade 2 and end standard pitch control (s)
9999.9     TPitManS(3)    - Time to start override pitch maneuver for blade 3 and end standard pitch control (s) [unused for 2 blades]
2          PitManRat(1)   - Pitch rate at which override pitch maneuver heads toward final pitch angle for blade 1 (deg/s)
2          PitManRat(2)   - Pitch rate at which override pitch maneuver heads toward final pitch angle for blade 2 (deg/s)
2          PitManRat(3)   - Pitch rate at which override pitch maneuver heads toward final pitch angle for blade 3 (deg/s) [unused for 2 blades]
-1         BlPitchF(1)    - Blade 1 final pitch for pitch maneuvers (degrees)
-1         BlPitchF(2)    - Blade 2 final pitch for pitch maneuvers (degrees)
-1         BlPitchF(3)    - Blade 3 final pitch for pitch maneuvers (degrees) [unused for 2 blades]
---------------------- GENERATOR AND TORQUE CONTROL ----------------------------
0          VSContrl       - Variable-speed control mode {0: none, 1: simple VS, 3: user-defined from routine UserVSCont, 4: user-defined from Simulink/Labview, 5:
1          GenModel       - Generator model {1: simple, 2: Thevenin, 3: user-defined from routine UserGen} (switch) [used only when VSContrl=0]
100        GenEff         - Generator efficiency [ignored by the Thevenin and user-defined generator models] (%)
TRUE       GenTiStr       - Method to start the generator {T: timed using TimGenOn, F: generator speed using SpdGenOn} (flag)
TRUE       GenTiStp       - Method to stop the generator {T: timed using TimGenOf, F: when generator power = 0} (flag)
9999.9     SpdGenOn       - Generator speed to turn on the generator for a startup (HSS speed) (rpm) [used only when GenTiStr=False]
[TimGenOn]     TimGenOn       - Time to turn on the generator for a startup (s) [used only when GenTiStr=True]
[TimGenOf]     TimGenOf       - Time to turn off the generator (s) [used only when GenTiStp=True]
---------------------- SIMPLE VARIABLE-SPEED TORQUE CONTROL --------------------
9999.9     VS_RtGnSp      - Rated generator speed for simple variable-speed generator control (HSS side) (rpm) [used only when VSContrl=1]
9999.9     VS_RtTq        - Rated generator torque/constant generator torque in Region 3 for simple variable-speed generator control (HSS side) (N-m) [used only whe
9999.9     VS_Rgn2K       - Generator torque constant in Region 2 for simple variable-speed generator control (HSS side) (N-m/rpm^2) [used only when VSContrl=1]
9999.9     VS_SlPc        - Rated generator slip percentage in Region 2 1/2 for simple variable-speed generator control (%) [used only when VSContrl=1]
---------------------- SIMPLE INDUCTION GENERATOR ------------------------------
0.60       SIG_SlPc       - Rated generator slip percentage (%) [used only when VSContrl=0 and GenModel=1]
1523       SIG_SySp       - Synchronous (zero-torque) generator speed (rpm) [used only when VSContrl=0 and GenModel=1]
187        SIG_RtTq       - Rated torque (N-m) [used only when VSContrl=0 and GenModel=1]
3.3        SIG_PORt       - Pull-out ratio (Tpullout/Trated) (-) [used only when VSContrl=0 and GenModel=1]
---------------------- THEVENIN-EQUIVALENT INDUCTION GENERATOR -----------------
60         TEC_Freq       - Line frequency [50 or 60] (Hz) [used only when VSContrl=0 and GenModel=2]
6          TEC_NPol       - Number of poles [even integer > 0] (-) [used only when VSContrl=0 and GenModel=2]
0.0185     TEC_SRes       - Stator resistance (ohms) [used only when VSContrl=0 and GenModel=2]
0.017      TEC_RRes       - Rotor resistance (ohms) [used only when VSContrl=0 and GenModel=2]
480        TEC_VLL        - Line-to-line RMS voltage (volts) [used only when VSContrl=0 and GenModel=2]
0.034      TEC_SLR        - Stator leakage reactance (ohms) [used only when VSContrl=0 and GenModel=2]
0.005      TEC_RLR        - Rotor leakage reactance (ohms) [used only when VSContrl=0 and GenModel=2]
0.775      TEC_MR         - Magnetizing reactance (ohms) [used only when VSContrl=0 and GenModel=2]
---------------------- HIGH-SPEED SHAFT BRAKE ----------------------------------
[HSSBrMode]    HSSBrMode          - HSS brake model {0: none, 1: simple, 3: user-defined from routine UserHSSBr, 4: user-defined from Simulink/Labview, 5: user-def
[THSSBrDp]    THSSBrDp           - Time to initiate deployment of the HSS brake (s)
[HSSBrDT]      HSSBrDT            - Time for HSS-brake to reach full deployment once initiated (sec) [used only when HSSBrMode=1]
[HSSBrTqF]     HSSBrTqF           - Fully deployed HSS-brake torque (N-m)
---------------------- NACELLE-YAW CONTROL -------------------------------------
0          YCMode         - Yaw control mode {0: none, 3: user-defined from routine UserYawCont, 4: user-defined from Simulink/Labview, 5: user-defined from Bladed
9999.9     TYCOn          - Time to enable active yaw control (s) [unused when YCMode=0]
0          YawNeut        - Neutral yaw position--yaw spring force is zero at this yaw (degrees)
0          YawSpr         - Nacelle-yaw spring constant (N-m/rad)
0          YawDamp        - Nacelle-yaw damping constant (N-m/(rad/s))
9999.9     TYawManS       - Time to start override yaw maneuver and end standard yaw control (s)
2          YawManRat      - Yaw maneuver rate (in absolute value) (deg/s)
0          NacYawF        - Final yaw angle for override yaw maneuvers (degrees)
```



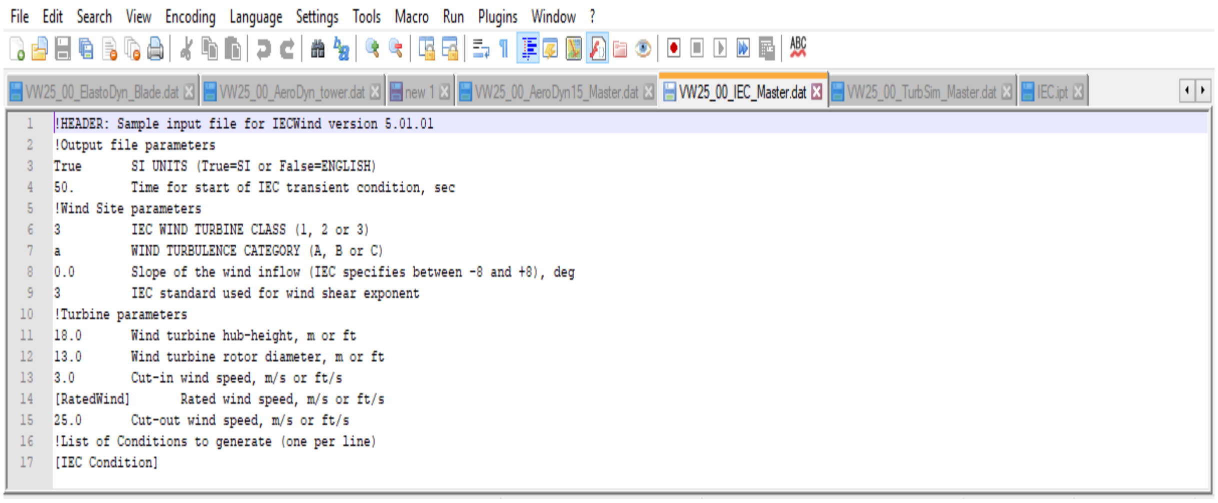
```
--------------------- TUNED MASS DAMPER ---------------------------------------
FALSE     CompNTMD      - Compute nacelle tuned mass damper {true/false} (flag)
unused    NTMDfile      - Name of the file for nacelle tuned mass damper (quoted string) [unused when CompNTMD is false]
FALSE     CompTTMD      - Compute tower tuned mass damper {true/false} (flag)
unused    TTMDfile      - Name of the file for tower tuned mass damper (quoted string) [unused when CompTTMD is false]
--------------------- BLADED INTERFACE ---------------------------------------- [used only with Bladed Interface]
unused    DLL_FileName  - Name/location of the dynamic library {.dll [Windows] or .so [Linux]} in the Bladed-DLL format (-) [used only with Bladed Interface]
DISCON.IN DLL_InFile    - Name of input file sent to the DLL (-) [used only with Bladed Interface]
DISCON    DLL_ProcName  - Name of procedure in DLL to be called (-) [case sensitive; used only with DLL Interface]
default   DLL_DT        - Communication interval for dynamic library (s) (or "default") [used only with Bladed Interface]
FALSE     DLL_Ramp      - Whether a linear ramp should be used between DLL_DT time steps [introduces time shift when true] (flag) [used only with Bladed Interface
9999.9    BPCutoff      - Cuttoff frequency for low-pass filter on blade pitch from DLL (Hz) [used only with Bladed Interface]
0         NacYaw_North  - Reference yaw angle of the nacelle when the upwind end points due North (deg) [used only with Bladed Interface]
0         Ptch_Cntrl    - Record 28: Use individual pitch control {0: collective pitch; 1: individual pitch control} (switch) [used only with Bladed Interface]
0         Ptch_SetPnt   - Record  5: Below-rated pitch angle set-point (deg) [used only with Bladed Interface]
0         Ptch_Min      - Record  6: Minimum pitch angle (deg) [used only with Bladed Interface]
0         Ptch_Max      - Record  7: Maximum pitch angle (deg) [used only with Bladed Interface]
0         PtchRate_Min  - Record  8: Minimum pitch rate (most negative value allowed) (deg/s) [used only with Bladed Interface]
0         PtchRate_Max  - Record  9: Maximum pitch rate  (deg/s) [used only with Bladed Interface]
0         Gain_OM       - Record 16: Optimal mode gain (Nm/(rad/s)^2) [used only with Bladed Interface]
0         GenSpd_MinOM  - Record 17: Minimum generator speed (rpm) [used only with Bladed Interface]
0         GenSpd_MaxOM  - Record 18: Optimal mode maximum speed (rpm) [used only with Bladed Interface]
0         GenSpd_Dem    - Record 19: Demanded generator speed above rated (rpm) [used only with Bladed Interface]
0         GenTrq_Dem    - Record 22: Demanded generator torque above rated (Nm) [used only with Bladed Interface]
0         GenPwr_Dem    - Record 13: Demanded power (W) [used only with Bladed Interface]
--------------------- BLADED INTERFACE TORQUE-SPEED LOOK-UP TABLE -------------
0         DLL_NumTrq    - Record 26: No. of points in torque-speed look-up table {0 = none and use the optimal mode parameters; nonzero = ignore the optimal mode
 GenSpd_TLU     GenTrq_TLU
 (rpm)           (Nm)
--------------------- OUTPUT --------------------------------------------------
TRUE      SumPrint      - Print summary data to <RootName>.sum (flag) (currently unused)
1         OutFile       - Switch to determine where output will be placed: {1: in module output file only; 2: in glue code output file only; 3: both} (currently u
TRUE      TabDelim      - Use tab delimiters in text tabular output file? (flag) (currently unused)
ES10.3E2  OutFmt        - Format used for text tabular output (except time).  Resulting field should be 10 characters. (quoted string) (currently unused)
30        TStart        - Time to begin tabular output (s) (currently unused)
          OutList       - The next line(s) contains a list of output parameters.  See OutListParameters.xlsx for a listing of available output channels, (-)
END of input file (the word "END" must appear in the first 3 columns of this last OutList line)
---------------------------------------------------------------------------------------
```

A 3.5 TurbSim input master file


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
VW25_00_ElastoDyn_Blade.dat
VW25_00_AeroDyn_tower.dat
new 1
VW25_00_AeroDyn15_Master.dat
VW25_00_IEC_Master.dat
VW25_00_TurbSim_Master.dat
IEC.ipt

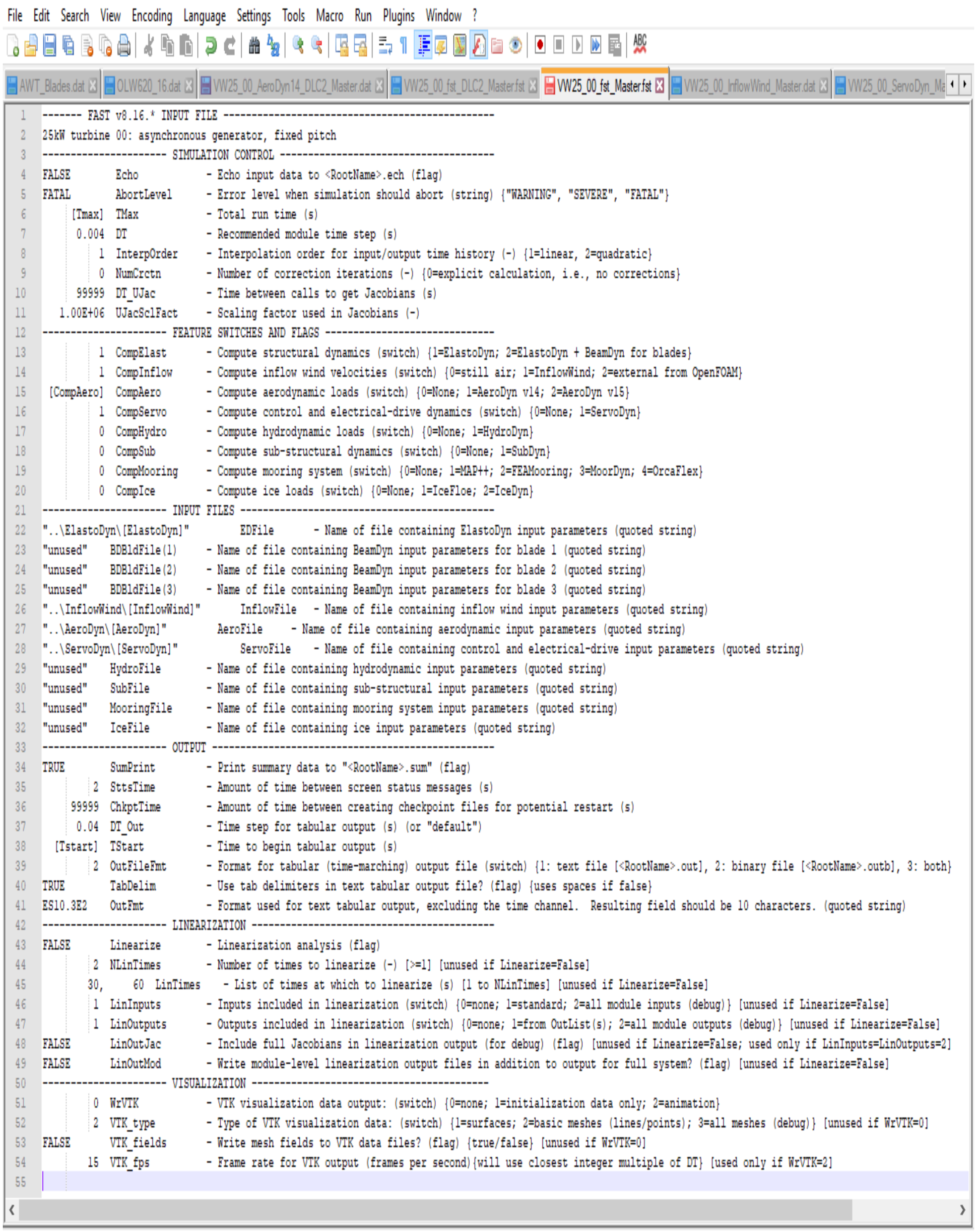
```
TurbSim Input File. Valid for TurbSim v1.50; 22-Mar-2018; VW25 pre-validation simulations

---------Runtime Options-----------------------------------
[RandSeed1] RandSeed1   - First random seed  (-2147483648 to 2147483647)
RanLux      RandSeed2   - Second random seed (-2147483648 to 2147483647) for intrinsic pRNG, or an alternative pRNG: "RanLux" or "RNSNLW"
FALSE   WrBHHTP - Output hub-height turbulence parameters in binary form?  (Generates RootName.bin)
FALSE   WrFHHTP - Output hub-height turbulence parameters in formatted form?  (Generates RootName.dat)
FALSE   WrADHH  - Output hub-height time-series data in AeroDyn form?  (Generates RootName.hh)
TRUE    WrADFF  - Output full-field time-series data in TurbSim/AeroDyn form? (Generates RootName.bts)
FALSE   WrBLFF  - Output full-field time-series data in BLADED/AeroDyn form?  (Generates RootName.wnd)
TRUE    WrADTWR - Output tower time-series data? (Generates RootName.twr)
FALSE   WrFMTFF - Output full-field time-series data in formatted (readable) form?  (Generates RootName.u, RootName.v, RootName.w)
FALSE   WrACT   - Output coherent turbulence time steps in AeroDyn form? (Generates RootName.cts)
TRUE    Clockwise   - Clockwise rotation looking downwind? (used only for full-field binary files - not necessary for AeroDyn)
0   ScaleIEC    - Scale IEC turbulence models to exact target standard deviation? [0=no additional scaling; 1=use hub scale uniformly; 2=use individual scales]

--------Turbine/Model Specifications-----------------------
3   NumGrid_Z   - Vertical grid-point matrix dimension
3   NumGrid_Y   - Horizontal grid-point matrix dimension
0.05    TimeStep    - Time step [seconds]
630 AnalysisTime    - Length of analysis time series [seconds]
625 UsableTime  - Usable length of output time series [seconds] (program will add GridWidth/MeanHHWS seconds) [bjj: was 630]
18  HubHt   - Hub height [m] (should be > 0.5*GridHeight)
30  GridHeight  - Grid height [m]
30  GridWidth   - Grid width [m] (should be >= 2*(RotorRadius+ShaftLength))
0   VflowAng    - Vertical mean flow (uptilt) angle [degrees]
0   HflowAng    - Horizontal mean flow (skew) angle [degrees]

--------Meteorological Boundary Conditions-------------------
IECKAI  TurbModel   Turbulence model ("IECKAI"=Kaimal, "IECVKM"=von Karman, "GP_LLJ", "NWTCUP", "SMOOTH", "WF_UPW", "WF_07D", "WF_14D", or "NONE")
"2" IECstandard Number of IEC 61400-x standard (x=1,2, or 3 with optional 61400-1 edition number (i.e. "1-Ed2") )
"A" IECturbc    IEC turbulence characteristic ("A", "B", "C" or the turbulence intensity in percent) ("KHTEST" option with NWTCUP, not used for other models)
NTM IEC_WindType    IEC turbulence type ("NTM"=normal, "xETM"=extreme turbulence, "xEWM1"=extreme 1-year wind, "xEWM50"=extreme 50-year wind, where x=wind turbine
default ETMc    IEC Extreme turbulence model "c" parameter [m/s]
default WindProfileType Wind profile type ("JET"=Low-level jet,"LOG"=Logarithmic,"PL"=Power law, or "default", or "USR"=User-defined)
18          RefHt   Height of the reference wind speed [m]
[WindSpeed] Uref    Mean (total) wind speed at the reference height [m/s]
default ZjetMax Jet height [m] (used only for JET wind profile, valid 70-490 m)
default PLExp   Power law exponent [-] (or "default")
default Z0  Surface roughness length [m] (or "default")

--------Non-IEC Meteorological Boundary Conditions------------
default Latitude    Site latitude [degrees] (or "default")
0.05    RICH_NO Gradient Richardson number
default Ustar   Friction or shear velocity [m/s] (or "default")
default ZI  Mixing layer depth [m] (or "default")
default PC_UW   Hub mean u'w' Reynolds stress [(m/s)^2] (or "default")
default PC_UV   Hub mean u'v' Reynolds stress [(m/s)^2] (or "default")
default PC_VW   Hub mean v'w' Reynolds stress [(m/s)^2] (or "default")
default IncDec1 u-component coherence parameters (e.g. "10.0  0.3e-3" in quotes) (or "default")
default IncDec2 v-component coherence parameters (e.g. "10.0  0.3e-3" in quotes) (or "default")
default IncDec3 w-component coherence parameters (e.g. "10.0  0.3e-3" in quotes) (or "default")
default CohExp  Coherence exponent (or "default")

--------Coherent Turbulence Scaling Parameters-------------------
"M:\coh_events\eventdata"   CTEventPath Name of the path where event data files are located
"Random"    CTEventFile Type of event files ("random", "les" or "dns")
TRUE    Randomize   Randomize disturbance scale and location? (true/false)
1   DistScl Disturbance scale (ratio of dataset height to rotor disk).
0.5 CTLy    Fractional location of tower centerline from right (looking downwind) to left side of the dataset.
0.5 CTLz    Fractional location of hub height from the bottom of the dataset.
10  CTStartTime Minimum start time for coherent structures in RootName.cts [seconds]

'==================================================
```

A 3.6 IECWind input master file


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
VW25_00_ElastoDyn_Blade.dat
VW25_00_AeroDyn_tower.dat
new 1
VW25_00_AeroDyn15_Master.dat
VW25_00_IEC_Master.dat
VW25_00_TurbSim_Master.dat
IEC.ipt

```
!HEADER: Sample input file for IECWind version 5.01.01
!Output file parameters
True        SI UNITS (True=SI or False=ENGLISH)
50.         Time for start of IEC transient condition, sec
!Wind Site parameters
3           IEC WIND TURBINE CLASS (1, 2 or 3)
a           WIND TURBULENCE CATEGORY (A, B or C)
0.0         Slope of the wind inflow (IEC specifies between -8 and +8), deg
3           IEC standard used for wind shear exponent
!Turbine parameters
18.0        Wind turbine hub-height, m or ft
13.0        Wind turbine rotor diameter, m or ft
3.0         Cut-in wind speed, m/s or ft/s
[RatedWind]        Rated wind speed, m/s or ft/s
25.0        Cut-out wind speed, m/s or ft/s
!List of Conditions to generate (one per line)
[IEC Condition]
```

A 3.7 FAST input master file


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
AWT_Blades.dat
OLW620_16.dat
VW25_00_AeroDyn14_DLC2_Master.dat
VW25_00_fst_DLC2_Master.fst
VW25_00_fst_Master.fst
VW25_00_InflowWind_Master.dat
VW25_00_ServoDyn_Ma

```
------- FAST v8.16.* INPUT FILE ------------------------------------------------
25kW turbine 00: asynchronous generator, fixed pitch
--------------------- SIMULATION CONTROL --------------------------------------
FALSE         Echo            - Echo input data to <RootName>.ech (flag)
FATAL         AbortLevel      - Error level when simulation should abort (string) {"WARNING", "SEVERE", "FATAL"}
     [Tmax]   TMax            - Total run time (s)
      0.004   DT              - Recommended module time step (s)
          1   InterpOrder     - Interpolation order for input/output time history (-) {1=linear, 2=quadratic}
          0   NumCrctn        - Number of correction iterations (-) {0=explicit calculation, i.e., no corrections}
      99999   DT_UJac         - Time between calls to get Jacobians (s)
   1.00E+06   UJacSclFact     - Scaling factor used in Jacobians (-)
--------------------- FEATURE SWITCHES AND FLAGS ------------------------------
          1   CompElast       - Compute structural dynamics (switch) {1=ElastoDyn; 2=ElastoDyn + BeamDyn for blades}
          1   CompInflow      - Compute inflow wind velocities (switch) {0=still air; 1=InflowWind; 2=external from OpenFOAM}
 [CompAero]   CompAero        - Compute aerodynamic loads (switch) {0=None; 1=AeroDyn v14; 2=AeroDyn v15}
          1   CompServo       - Compute control and electrical-drive dynamics (switch) {0=None; 1=ServoDyn}
          0   CompHydro       - Compute hydrodynamic loads (switch) {0=None; 1=HydroDyn}
          0   CompSub         - Compute sub-structural dynamics (switch) {0=None; 1=SubDyn}
          0   CompMooring     - Compute mooring system (switch) {0=None; 1=MAP++; 2=FEAMooring; 3=MoorDyn; 4=OrcaFlex}
          0   CompIce         - Compute ice loads (switch) {0=None; 1=IceFloe; 2=IceDyn}
--------------------- INPUT FILES ---------------------------------------------
"..\ElastoDyn\[ElastoDyn]"        EDFile       - Name of file containing ElastoDyn input parameters (quoted string)
"unused"    BDBldFile(1)      - Name of file containing BeamDyn input parameters for blade 1 (quoted string)
"unused"    BDBldFile(2)      - Name of file containing BeamDyn input parameters for blade 2 (quoted string)
"unused"    BDBldFile(3)      - Name of file containing BeamDyn input parameters for blade 3 (quoted string)
"..\InflowWind\[InflowWind]"      InflowFile   - Name of file containing inflow wind input parameters (quoted string)
"..\AeroDyn\[AeroDyn]"        AeroFile     - Name of file containing aerodynamic input parameters (quoted string)
"..\ServoDyn\[ServoDyn]"          ServoFile    - Name of file containing control and electrical-drive input parameters (quoted string)
"unused"    HydroFile         - Name of file containing hydrodynamic input parameters (quoted string)
"unused"    SubFile           - Name of file containing sub-structural input parameters (quoted string)
"unused"    MooringFile       - Name of file containing mooring system input parameters (quoted string)
"unused"    IceFile           - Name of file containing ice input parameters (quoted string)
--------------------- OUTPUT --------------------------------------------------
TRUE        SumPrint          - Print summary data to "<RootName>.sum" (flag)
         2  SttsTime          - Amount of time between screen status messages (s)
     99999  ChkptTime         - Amount of time between creating checkpoint files for potential restart (s)
      0.04  DT_Out            - Time step for tabular output (s) (or "default")
  [Tstart]  TStart            - Time to begin tabular output (s)
         2  OutFileFmt        - Format for tabular (time-marching) output file (switch) {1: text file [<RootName>.out], 2: binary file [<RootName>.outb], 3: both}
TRUE        TabDelim          - Use tab delimiters in text tabular output file? (flag) {uses spaces if false}
ES10.3E2    OutFmt            - Format used for text tabular output, excluding the time channel.  Resulting field should be 10 characters. (quoted string)
--------------------- LINEARIZATION -------------------------------------------
FALSE       Linearize         - Linearization analysis (flag)
         2  NLinTimes         - Number of times to linearize (-) [>=1] [unused if Linearize=False]
        30,     60  LinTimes    - List of times at which to linearize (s) [1 to NLinTimes] [unused if Linearize=False]
         1  LinInputs         - Inputs included in linearization (switch) {0=none; 1=standard; 2=all module inputs (debug)} [unused if Linearize=False]
         1  LinOutputs        - Outputs included in linearization (switch) {0=none; 1=from OutList(s); 2=all module outputs (debug)} [unused if Linearize=False]
FALSE       LinOutJac         - Include full Jacobians in linearization output (for debug) (flag) [unused if Linearize=False; used only if LinInputs=LinOutputs=2]
FALSE       LinOutMod         - Write module-level linearization output files in addition to output for full system? (flag) [unused if Linearize=False]
--------------------- VISUALIZATION -------------------------------------------
         0  WrVTK             - VTK visualization data output: (switch) {0=none; 1=initialization data only; 2=animation}
         2  VTK_type          - Type of VTK visualization data: (switch) {1=surfaces; 2=basic meshes (lines/points); 3=all meshes (debug)} [unused if WrVTK=0]
FALSE       VTK_fields        - Write mesh fields to VTK data files? (flag) {true/false} [unused if WrVTK=0]
        15  VTK_fps           - Frame rate for VTK output (frames per second){will use closest integer multiple of DT} [used only if WrVTK=2]

```

APPENDIX 4 Python Auto-generation Script

```
#Created on Thu May 15 12:30:29 2018
#Michael Keumatio Lontsie

from pylab import *
from ReadODS import *
from execute import *
import sys
import subprocess
import os
import ECD_ECG

#Set the masterfiles as variables
#The python script needs to be set to the directory where the files are saved

TurbSimMasterFile = "VW25_00_TurbSim_Master.dat"
InflowWindMasterFile = "VW25_00_InflowWind_Master.dat"
fstMasterFile = "VW25_00_fst_Master.fst"
fstMasterFile_DLC2 = "VW25_00_fst_DLC2_Master.fst"
ServoDyn = "VW25_00_ServoDyn_Master.dat"
ElastoDyn = "VW25_00_ElastoDyn_Master.dat"
AeroDyn14 = "VW25_00_AeroDyn14_Master.dat"
AeroDyn14_DLC2 ="VW25_00_AeroDyn14_DLC2_Master.dat"
AeroDyn15 = "VW25_00_AeroDyn15_Master.dat"
IECWind = "VW25_00_IEC_Master.dat"

VW25_00_AutoGSpeadsheet = "VW25_00_Inputs_AutogenerationSpreadsheet_Test04.ods"

runTurbSim = True
runOpenFAST = True
runIEC = True

# Open the masterfiles
f = open(TurbSimMasterFile,'r')
TurbSimTemplate = f.read()
f.close()

f = open(InflowWindMasterFile,'r')
InflowWindTemplate = f.read()
f.close()

f = open(fstMasterFile,'r')
fstTemplate = f.read()
f.close()

f = open(fstMasterFile_DLC2,'r')
fstTemplate_DLC2 = f.read()
f.close()

f = open(ServoDyn,'r')
ServoDynTemplate = f.read()
f.close()

f = open(ElastoDyn,'r')
ElastoDynTemplate = f.read()
f.close()

f = open(AeroDyn14,'r')
AeroDyn14Template = f.read()
f.close()

f = open(AeroDyn14_DLC2,'r')
AeroDyn14Template_DLC2 = f.read()
f.close()

f = open(AeroDyn15,'r')
```

```
AeroDyn15Template = f.read()
f.close()

f = open(IECWind,'r')
IECWindTemplate = f.read()
f.close()

# This commands open and read odf files
# "conda install -c conda-forge odfpy"- to install the odfpy package in anaconda
# This script also requires the ODSReader.py and ReadODS.py files

a = np.array(ReadODS(VW25_00_AutoGSpeadsheet, "Sheet1", cutEmpty=True), dtype='str')
elems = a.nonzero()
noRows= max(elems[0])
noColumns= max(elems[1])
ExchangeSheetData = a[:noRows+1, :noColumns+1]

noSims = noRows-2
FileID = ExchangeSheetData[0,:]
PlaceHolder = ExchangeSheetData[1,:]

# To generate each input files and looping
for i in range(0,noSims+1) :

   dlcFolder=ExchangeSheetData[i+2,0]
   filenameBase=ExchangeSheetData[i+2,1]
   uniqueId=ExchangeSheetData[i+2,2]
   tipBrakeFlag=ExchangeSheetData[i+2,3]

   TurbSimTemplatetemp = TurbSimTemplate
   InflowWindTemplatetemp = InflowWindTemplate
   if tipBrakeFlag=="True":
      fstTemplatetemp = fstTemplate_DLC2
      AeroDyn14Templatetemp = AeroDyn14Template_DLC2
   else:
      fstTemplatetemp = fstTemplate
      AeroDyn14Templatetemp = AeroDyn14Template
   ServoDynTemplatetemp = ServoDynTemplate
   ElastoDynTemplatetemp = ElastoDynTemplate
   AeroDyn15Templatetemp = AeroDyn15Template
   IECWindTemplatetemp = IECWindTemplate

   Skip_TurbSim = False
   Skip_IECWind = False
   ecgCase=False

   for j in range(0, noColumns+1):
      if FileID[j] == "TS":
         if ExchangeSheetData[i+2,j] == "Skip":
            Skip_TurbSim = True
         else:
            TurbSimTemplatetemp = TurbSimTemplatetemp.replace(PlaceHolder[j],ExchangeSheetData[i+2,j])
      elif FileID[j] == "IW":
         InflowWindTemplatetemp = InflowWindTemplatetemp.replace(PlaceHolder[j],ExchangeSheetData[i+2,j])
      elif FileID[j] == "FST":
         fstTemplatetemp = fstTemplatetemp.replace(PlaceHolder[j],ExchangeSheetData[i+2,j])
      elif FileID[j] == "SD":
         ServoDynTemplatetemp = ServoDynTemplatetemp.replace(PlaceHolder[j],ExchangeSheetData[i+2,j])
      elif FileID[j] == "ED":
         ElastoDynTemplatetemp = ElastoDynTemplatetemp.replace(PlaceHolder[j],ExchangeSheetData[i+2,j])
#       elif FileID[j] == "AD14":
#          AeroDyn14Templatetemp = AeroDyn14Templatetemp.replace(PlaceHolder[j],ExchangeSheetData[i+2,j])
#       elif FileID[j] == "AD15":
#          AeroDyn15Templatetemp = AeroDyn15Templatetemp.replace(PlaceHolder[j],ExchangeSheetData[i+2,j])
      elif FileID[j] == "IECWind":
         if ExchangeSheetData[i+2,j] == "Skip":
            Skip_IECWind = True
            else:
                # check if uniqueId contains ECG, if so run IECWind for ECD case
                # then alter resulting .wnd file for ECG conditions
                if "ECG" in ExchangeSheetData[i+2,j]:
                    ecgCase=True
                    ExchangeSheetData[i+2,j]=ExchangeSheetData[i+2,j].replace("ECG","ECD")
                IECWindTemplatetemp = IECWindTemplatetemp.replace(PlaceHolder[j],ExchangeSheetData[i+2,j])
        elif FileID[j] == "AD14F7":
            if logical_not(ExchangeSheetData[i+2,j] =="Skip"):
                AeroDyn14Templatetemp =
AeroDyn14Templatetemp.replace(PlaceHolder[j],ExchangeSheetData[i+2,j])

############To generate each input file with specific name and save in attached directories##############

    if Skip_TurbSim == False:
        TurbSimfilename = ''.join(np.array([filenameBase,'TurbSim_',uniqueId,'.dat']))
        TurbSimfilepath = ''.join(np.array([dlcFolder,'/Wind/', TurbSimfilename]))
        with open(TurbSimfilepath, "w") as f:
            f.write(TurbSimTemplatetemp)
            f.close()
        if runTurbSim == True:
            # To run TurbSim64.exe and generate TurbSim.bts files
            args = ['TurbSim64', TurbSimfilepath]
            subprocess.call(args, shell=True)

    if Skip_IECWind == False:
        IecWindInputFileName = "IEC.ipt"
        IECWindInputFilepath = ''.join(np.array([dlcFolder,'/Wind/', IecWindInputFileName]))
        with open(IECWindInputFilepath, "w") as f:
            f.write(IECWindTemplatetemp)
            f.close()
        if runIEC == True:
            windDir=''.join(np.array([dlcFolder,'/Wind']))
            wd = os.getcwd()
            os.chdir(windDir)
            p=subprocess.Popen("IECWind", shell=True)
            p_status = p.wait()
            with open(IecWindInputFileName, 'r') as f:
                read_data = f.readlines()
                f.close()
            IECWindfilename_std = ''.join(np.array([read_data[-1],".wnd"]))
            if ecgCase:
                # convert from ECD to ECG
                ECD_ECG.ECD_ECG(IECWindfilename_std) # do whatever is in ECD_ECG.py
            IECWindfilename = ''.join(np.array([filenameBase,'IecWind_',uniqueId,'.wnd']))
            args= ["ren", IECWindfilename_std, IECWindfilename]
            p=subprocess.Popen(args, shell=True)
            os.chdir(wd)

    AeroDyn14filename = ''.join(np.array(['VW25_00_AeroDyn14.dat']))
    AeroDyn14filepath = ''.join(np.array([dlcFolder,'/AeroDyn/', AeroDyn14filename]))
    with open(AeroDyn14filepath, "w") as f:
        f.write(AeroDyn14Templatetemp)
        f.close()

    if tipBrakeFlag!="True":
        Inflowfilename = ''.join(np.array([filenameBase,'InflowWind_',uniqueId,'.dat']))
        Inflowfilepath = ''.join(np.array([dlcFolder,'/InflowWind/', Inflowfilename]))
        with open(Inflowfilepath, "w") as f:
            f.write(InflowWindTemplatetemp)
            f.close()

        ElastoDynfilename = ''.join(np.array([filenameBase,'ElastoDyn_',uniqueId,'.dat']))
        ElastoDynfilepath = ''.join(np.array([dlcFolder,'/ElastoDyn/', ElastoDynfilename]))
        with open(ElastoDynfilepath, "w") as f:
            f.write(ElastoDynTemplatetemp)
            f.close()
        ServoDynfilename = ''.join(np.array([filenameBase,'ServoDyn_',uniqueId,'.dat']))
        ServoDynfilepath = ''.join(np.array([dlcFolder,'/ServoDyn/', ServoDynfilename]))
        with open(ServoDynfilepath, "w") as f:
            f.write(ServoDynTemplatetemp)
            f.close()
        AeroDyn15filename = ''.join(np.array(['VW25_00_AeroDyn15.dat']))
        AeroDyn15filepath = ''.join(np.array([dlcFolder,'/AeroDyn/', AeroDyn15filename]))
        with open(AeroDyn15filepath, "w") as f:
            f.write(AeroDyn15Templatetemp)
            f.close()

    fstfilename = ''.join(np.array([filenameBase,'fstInput_',uniqueId,'.fst']))
    FASTfilepath = ''.join(np.array([dlcFolder,'/fst/', fstfilename]))
    with open(FASTfilepath, "w") as f:
        f.write(fstTemplatetemp)
        f.close()

#    sys.stdout=open("25kWTurbine_FAST_ConsoleOut.txt","w")
    if runOpenFAST == True:
 #       sys.stdout=open("../25kWTurbine_FAST_ConsoleOut.txt","a")
        if tipBrakeFlag=="True":
            args = ['FAST7_x64', FASTfilepath]
        else:
            args = ['OpenFAST_x64', FASTfilepath]
        #subprocess.call(args, shell=True)
        execute(args)

#sys.stdout.close()
```

APPENDIX 5 Statistical computation Python script

A 5.1 Statistical computation Python script 1

```
# -*- coding: utf-8 -*-
"""
Created on Mon Aug  4 16:25:08 2018

@author: Michael Keumatio Lontsie
"""
import glob, os
import numpy as np
import matplotlib.pyplot as plt
from wetb.dlc import high_level
from wetb.fast.fast_io import load_output
from wetb.fatigue_tools.fatigue import eq_load, rainflow_astm, rainflow_windap, cycle_matrix

# Specific of DLCs
my_path = os.getcwd()

for dir in os.listdir(my_path):
    #print(dir)
    if dir.startswith('DLC') and os.path.isdir(os.path.join(my_path, dir)):
         if dir.startswith('DLC31_NTM'):
            continue
        elif dir.startswith('DLC52'):
            continue
        for f in os.listdir(my_path +'/'+ dir):
            #print(dir + ':' + f)
            if f.startswith('f'):
                os.chdir(my_path+'/'+ dir +'/'+ f)

                #os.chdir('../fst')
                turbineModel=25kWTurbine'
                statName= dir
# initialization of variables
                vec = np.zeros([600001,1])
                vec_max = np.zeros([600001,31])
                vec_min = np.zeros([600001,31])
                vec_mean = np.zeros([600001,31])
                vec_abs =  np.zeros([600001,31])
                windspeed = np.zeros([600001,1])
                windspeed1 = np.zeros([600001,1])
                i = 0
                for file in glob.glob('*.out'):
                    data, info  = load_output(file)
                    #short_Eq = eq_load(data[:,info['attribute_names'].index('RootMyb1')], m=12,
                     #             neq=1e7, rainflow_func=rainflow_astm)
                    #vec[i] = short_Eq[0]
# Statistics computation (including mean, max, min values)
                    vec_max1 =  [np.max(data[:,[k]])  for k in range(0,len(data[1,:]))]
                    vec_min1 =  [np.min(data[:,[k]])  for k in range(0,len(data[1,:]))]
                    vec_mean1 = [np.mean(data[:,[k]])  for k in range(0,len(data[1,:]))]
                    vec_abs1 =  [np.max(np.abs([vec_max1[k],vec_min1[k]])) for k in range(0,len(data[1,:]))]
                    windspeed[i] = np.mean(np.sqrt(data[:,[1]]**2 + data[:,[2]]**2 + data[:,[3]]**2))
                    vec_max[i] = vec_max1[:]
                    vec_min[i] = vec_min1[:]
                    vec_mean[i] = vec_mean1[:]
                    vec_abs[i] = vec_abs1[:]
                #    windspeed1[i] = np.sqrt(vec_mean1[1]**2 + vec_mean1[2]**2 + vec_mean1[3]**2)
                #    windspeed1[i] = vec_mean1[1]
                    i += 1
                #    print("\nEquivalent load for the Blade Root Flapwise Moment of the DTU 10MW (FAST):\n")
                #    print(short_Eq)
                #    print("\n")
# plot figures:
        plt.close("all")
        plt.rcParams['font.size'] = 23
        idxFig=[4,5,6,7,8,9,10,11,12,13,14,17,18,19,20,21,22,23,24,25,26,27]
        axisName=['RotTorq [kNm]','LSShftFxa [kN]','RotPwr [kW]','RootFzc1 [kN]','RootMxb1 [kNm]','RootMyb1
[kNm]','LSShftMxa [kNm]','LSSGagMya [kNm]','LSSGagMza [kNm]','TwrBsMxt [kNm]','TwrBsMyt
[kNm]','YawBrFzn [kN]','YawBrFxp [kN]','YawBrFyp [kN]','YawBrMzn [kNm]','YawBrMxp [kNm]','YawBrMyp
[kNm]','RootFxb1 [kN]','RootFyb1 [kN]','RootMzc1 [kNm]','LSShftFys [kN]','LSShftFzs [kN]','Q_DrTr
[rad]','QD_DrTr [rad/s]']
        for i in range(0,len(idxFig)):
          fig = plt.figure(figsize=(25,15))
          plt.plot(windspeed,vec_max[:,idxFig[i]],'rs')
          plt.plot(windspeed,vec_min[:,idxFig[i]],'bs')
          plt.plot(windspeed,vec_mean[:,idxFig[i]],'ks')
          if dir.startswith('DLC6'):
            plt.axis([40, 50, np.min(vec_min[:,idxFig[i]]), 1.1*np.max(vec_max[:,idxFig[i]])])
          elif dir.startswith('DLC51'):
            plt.axis([40, 60, np.min(vec_min[:,idxFig[i]]), 1.1*np.max(vec_max[:,idxFig[i]])])
          else:
            plt.axis([2, 27, np.min(vec_min[:,idxFig[i]]), 1.1*np.max(vec_max[:,idxFig[i]])])
          #    plt.grid(True)
          plt.ylabel(axisName[i])
          plt.xlabel('Wind Speed, [m/s]')
          plt.title(turbineModel)
          plt.legend(('vec_max', 'vec_min', 'vec_mean'),
                loc='best', shadow=None)
          plt.show()
          fig.savefig(my_path + '/'+ dir + '/'+ axisName[i]+ '.png', dpi=fig.dpi)
        # save variables
        #os.chdir(my_path)
        os.chdir('../../Load_Comparison')
        vec_abs_s = [np.max(vec_abs[:,[k]])  for k in range(0,len(vec_abs[1,:]))]
        np.savetxt(statName + '_max.txt', vec_abs_s, delimiter=',')
```

A 5.2 Statistical computation Python script 2

```
"""
Created on Wed Aug 01 10:32:53 2018

@author: Michael Keumatio Lontsie
"""

import glob, os
import numpy as np
import matplotlib.pyplot as plt
import heapq
import pandas as pd
from wetb.dlc import high_level
from wetb.fast.fast_io import load_output
from wetb.fatigue_tools.fatigue import eq_load, rainflow_astm, rainflow_windap, cycle_matrix



my_path = os.getcwd()
#Defining  max, min, mean vectors variables
vec_max=np.zeros([1,31])
vec_min=np.zeros([1,31])
vec_mean=np.zeros([1,31])
windspeed=np.zeros([1,1])
unicID=np.array('-', dtype=str)

# Specific of DLCs
#Looping through the DLCs, the fst folders and the output files in them
for dir in os.listdir(my_path):
    if dir.startswith('DLC') and os.path.isdir(os.path.join(my_path, dir)):
        if dir.startswith('DLC31_NTM'):
            continue
        elif dir.startswith('DLC52'):
            continue
        for f in os.listdir(my_path +'/'+ dir):
            #print(dir + ':' + f)

            if f.startswith('f'):
                os.chdir(my_path+'/'+ dir +'/'+ f)
                turbineModel='25kWTurbine'
                statName= dir
                # initialization of variables
                vec = np.zeros([600001,1])
#                 vec_max = np.zeros([351,19])
#                 vec_min = np.zeros([351,19])
#                 vec_mean = np.zeros([351,19])
                vec_abs =  np.zeros([600001,31])
                #windspeed = np.zeros([351,1])
                windspeed1 = np.zeros([600001,1])

                i = 0
                for file in glob.glob('*.outb'):
                    data, info  = load_output(file)
                    # Statistics computation (including mean, max, min values)
                    vec_max1 =  [np.max(data[:,[k]])  for k in range(0,len(data[1,:]))]
                    vec_min1 =  [np.min(data[:,[k]])  for k in range(0,len(data[1,:]))]
                    vec_mean1 = [np.mean(data[:,[k]])  for k in range(0,len(data[1,:]))]
                    vec_abs1 =  [np.max(np.abs([vec_max1[k],vec_min1[k]])) for k in range(0,len(data[1,:]))]
                    windspeed = np.vstack([windspeed,np.mean(np.sqrt(data[:,[1]]**2 + data[:,[2]]**2 + data[:,[3]]**2))])
                    #concatenating (stacking) max, min, mean saved values from each i files after each loop
                    vec_max = np.vstack([vec_max,vec_max1[:]])
                    vec_min = np.vstack([vec_min,vec_min1[:]])
                    vec_mean = np.vstack([vec_mean,vec_mean1[:]])
                    Vvec_max = 1.35*vec_max
                    Vvec_min = 1.35*vec_min
                    Vvec_mean = 1.35*vec_mean
                    unicID = np.vstack([unicID, file[:-4]])
                    vec_abs[i] = vec_abs1[:]
              # save variables
                os.chdir('../../Load_Comparison')
                vec_abs_s = [np.max(vec_abs[:,[k]])  for k in range(0,len(vec_abs[1,:]))]
                np.savetxt(statName + '_max.txt', vec_abs_s, delimiter=',')
# plot figures:
plt.close("all")
plt.rcParams['font.size'] = 28
idxFig=[4,5,6,7,8,9,10,11,12,13,14,17,18,19,20,21,22,23,24,25,26,27]
axisName=['RotTorq [kNm]','LSShftFxa [kN]','RotPwr [kW]','RootFzc1 [kN]','RootMxb1 [kNm]','RootMyb1
[kNm]','LSShftMxa [kNm]','LSSGagMya [kNm]','LSSGagMza [kNm]','TwrBsMxt [kNm]','TwrBsMyt
[kNm]','YawBrFzn [kN]','YawBrFxp [kN]','YawBrFyp [kN]','YawBrMzn [kNm]','YawBrMxp [kNm]','YawBrMyp
[kNm]','RootFxb1 [kN]','RootFyb1 [kN]','RootMzc1 [kNm]','LSShftFys [kN]','LSShftFzs [kN]','Q_DrTr
[rad]','QD_DrTr [rad/s]']
for i in range(0,len(idxFig)):
    fig = plt.figure(figsize=(25,15))
    plt.plot(windspeed,vec_max[:,idxFig[i]],'rs')
    plt.plot(windspeed,vec_min[:,idxFig[i]],'bs')
    plt.plot(windspeed,vec_mean[:,idxFig[i]],'ks')
    plt.axis([2, 60, np.min(vec_min[:,idxFig[i]]), 1.1*np.max(vec_max[:,idxFig[i]])])
    plt.grid(True)
    plt.ylabel(axisName[i]+' incl. SF')
    plt.xlabel('Wind Speed, [m/s]')
    plt.title(turbineModel)
    plt.legend(('vec_max', 'vec_min', 'vec_mean'),
           loc='best', shadow=None)
    plt.show()
    fig.savefig(my_path + '/Load_Comparison/'+ axisName[i]+ '.png', dpi=fig.dpi)
```

A 5.3 Statistical computation Python script 3

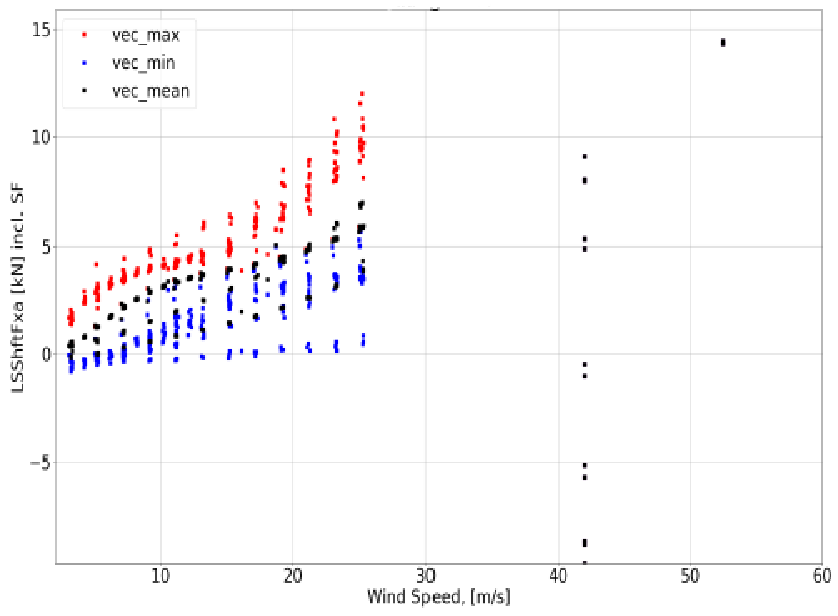
```
"""
Created on Thu Aug 02 11:04:41 2018

@author: Michael Keumatio Lontsie
"""
import glob, os
import numpy as np
import matplotlib.pyplot as plt
from wetb.dlc import high_level
from wetb.fast.fast_io import load_output
from wetb.fatigue_tools.fatigue import eq_load, rainflow_astm, rainflow_windap, cycle_matrix
import heapq
import pandas as pd
import numpy as np
from fpdf import FPDF
import bottleneck as bn

my_path = os.getcwd()
#Defining  max, min, mean vectors variables
vec_max=np.zeros([1,31])
vec_min=np.zeros([1,31])
vec_mean=np.zeros([1,31])
windspeed=np.zeros([1,1])
unicID=np.array('-', dtype=str)
Vec_10min=np.zeros([1,10])
Vec_10max=np.zeros([1,10])
# Specific of DLCs
#Looping through the DLCs, the fst folders and the output files in them
for dir in os.listdir(my_path):
    if dir.startswith('DLC') and os.path.isdir(os.path.join(my_path, dir)):
        if dir.startswith('DLC31_NTM'):
            continue
        elif dir.startswith('DLC52'):
            continue
        for f in os.listdir(my_path +'/'+ dir):
            #print(dir + ':' + f)
            if f.startswith('f'):
                os.chdir(my_path+'/'+ dir +'/'+ f)
                turbineModel='25kWTurbine'
                statName= dir
                # initialization of variables
                vec = np.zeros([600001,1])
                vec_abs =  np.zeros([600001,31])
                #windspeed = np.zeros([351,1])
                windspeed1 = np.zeros([600001,1])

                i = 0
                for file in glob.glob('*.out'):
                    data, info  = load_output(file)
# Statistics computation (including mean, max, min values)
                    vec_max1 =  [np.max(data[:,[k]])  for k in range(0,len(data[1,:]))]
                    vec_min1 =  [np.min(data[:,[k]])  for k in range(0,len(data[1,:]))]
                    vec_mean1 = [np.mean(data[:,[k]])  for k in range(0,len(data[1,:]))]
                    vec_abs1 =  [np.max(np.abs([vec_max1[k],vec_min1[k]])) for k in range(0,len(data[1,:]))]
                    unicID = np.vstack([unicID, file[:-4]])
                    windspeed = np.vstack([windspeed,np.mean(np.sqrt(data[:,[1]]**2 + data[:,[2]]**2 + data[:,[3]]**2))])
#concatenating (stacking) max, min, mean saved values from each i files after each loop
                    vec_max = np.vstack([vec_max[:],vec_max1[:]])
                    vec_min = np.vstack([vec_min[:],vec_min1[:]])
                    vec_mean = np.vstack([vec_mean[:],vec_mean1[:]])
                    vec_abs[i] = vec_abs1[:]
# to remove the first "0" row
Vec_max = 1.35*vec_max[1:]
Vec_min = 1.35*vec_min[1:]
Vec_mean = 1.35*vec_mean[1:]
```

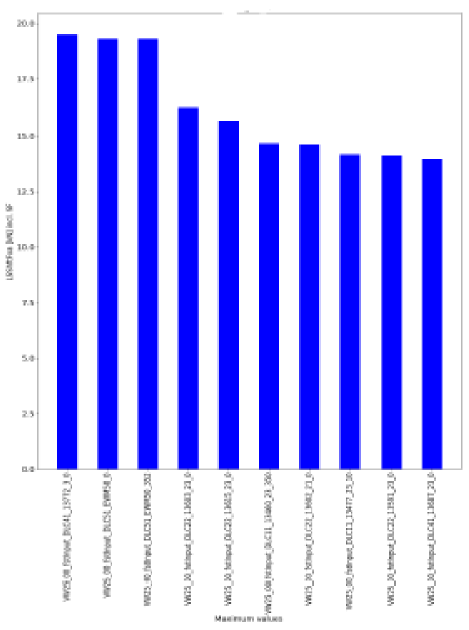
```
# plot figures:
#plt.close("all")
plt.rcParams['font.size'] = 16
#idxFig=[4,5,6,7,8,9,10,11,12,13,14,17,18,19,20,21,22,23,24,25,26,27]
#axisName=['RotTorq [kNm]','LSShftFxa [kN]','RotPwr [kW]','RootFzc1 [kN]','RootMxb1 [kNm]','RootMyb1
[kNm]','LSShftMxa [kNm]','LSSGagMya [kNm]','LSSGagMza [kNm]','TwrBsMxt [kNm]','TwrBsMyt
[kNm]','YawBrFzn [kN]','YawBrFxp [kN]','YawBrFyp [kN]','YawBrMzn [kNm]','YawBrMxp [kNm]','YawBrMyp
[kNm]','RootFxb1 [kN]','RootFyb1 [kN]','RootMzc1 [kNm]','LSShftFys [kN]','LSShftFzs [kN]','Q_DrTr
[rad]','QD_DrTr [rad/s]']
idxFig=[6,4,5,26,27,10,11,12,23,24,7,8,9,25,18,19,17,20,21,22,13,14]
axisName=['RotPwr [kW]','RotTorq [kNm]','LSShftFxa [kN]','LSShftFys [kN]','LSShftFzs [kN]','LSShftMxa
[kNm]','LSSGagMya [kNm]','LSSGagMza [kNm]','RootFxb1 [kN]','RootFyb1 [kN]','RootFzc1 [kN]','RootMxb1
[kNm]','RootMyb1 [kNm]','RootMzc1 [kNm]','YawBrFxp [kN]','YawBrFyp [kN]','YawBrFzn [kN]','YawBrMzn
[kNm]','YawBrMxp [kNm]','YawBrMyp [kNm]','TwrBsMxt [kNm]','TwrBsMyt [kNm]','Q_DrTr [rad]','QD_DrTr [rad/s]']
for i in range(0,len(idxFig)):
    #plots 10max
    vec_max_i = Vec_max[:,idxFig[i]]
    vec_10max_ind = heapq.nlargest(10, range(len(vec_max_i)), vec_max_i.take)
    vec_10max = Vec_max[vec_10max_ind,idxFig[i]]

    fig = plt.figure(figsize=(15,25))
    N = len(vec_10max)
    x = range(N)
    width = 0.5

    plt.bar(x,vec_10max, width, color="blue")
    plt.ylabel(axisName[i]+'incl. SF')
    plt.xlabel('Maximum values')
    # Create the bars names
    plt.xticks(x, unicID[vec_10max_ind,0], color='black', rotation=90)
    # Custom the subplot layout
    plt.subplots_adjust(bottom=0.4, top=0.99)
    plt.title(turbineModel)
    fig.savefig(my_path + '/Load_Comparison/'+ axisName[i]+ 'Max.png', dpi=fig.dpi)

    #plots 10min
    vec_min_i = Vec_min[:,idxFig[i]]
    vec_10min_ind = bn.argpartition(vec_min_i, 10)[:10]
    vec_10min = bn.partition(vec_min_i, 10)[:10]
    vvec_10min = np.sort(vec_10min)
#    k = 10
#    vec_10min_ind = np.argpartition(vec_min_i, -k)
#    vec_10min = vec_min_i[vec_10min_ind[:k]]
    fig = plt.figure(figsize=(15,25))
    M = len(vec_10min)
    y = range(M)
    width = 0.5
    plt.bar(y,vvec_10min, width, color="blue")
    plt.ylim(plt.ylim()[::1])
    plt.xlim(plt.xlim()[::1])
    plt.ylabel(axisName[i]+' incl. SF')
    plt.xlabel('Minimum values')
    # Create the bars names
    plt.xticks(y, unicID[vec_10min_ind,0], color='black', rotation=90)
    plt.title(turbineModel)
    # Custom the subplot layout
    plt.subplots_adjust(bottom=0.4, top=0.99)
    fig.savefig(my_path + '/Load_Comparison/'+ axisName[i]+ 'Min.png', dpi=fig.dpi)

###########################################################################################
#PDF file for the extreme table
###########################################################################################
#    Vec_10min = np.hstack([Vec_10min[1:],vec_10min[:]])
#    Vec_10max = np.hstack([Vec_10max[1:],vec_10max[:]])
VVec_max= Vec_max[:,idxFig]
VVec_min= Vec_min[:,idxFig]
```

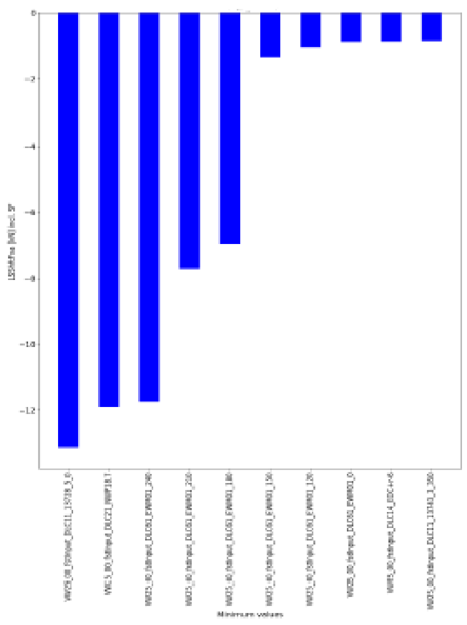
```
#VVec_max = Vec_max[0:,4:]
#VVec_min = Vec_min[0:,4:]
Bmax = np.max(VVec_max, axis=0)
Bmax_ind = np.argmax(VVec_max, axis=0)
BBmax = Bmax.tolist()
unicIDMax = unicID[Bmax_ind,0]
UnicIDMax = unicIDMax.tolist()

Bmin = np.min(VVec_min, axis=0)
Bmin_ind = np.argmax(VVec_min, axis=0)
BBmin = Bmin.tolist()
unicIDMin = unicID[Bmin_ind,0]
UnicIDMin = unicIDMin.tolist()
#axisName1=['RotTorq [kNm]','LSShftFxa [kN]','RotPwr [kW]','RootFzc1 [kN]','RootMxb1 [kNm]','RootMyb1
[kNm]','LSShftMxa [kNm]','LSSGagMya [kNm]','LSSGagMza [kNm]','TwrBsMxt [kNm]','TwrBsMyt
[kNm]','YawBrFzn [kN]','YawBrFxp [kN]','YawBrFyp [kN]','YawBrMzn [kNm]','YawBrMxp [kNm]','YawBrMyp
[kNm]','RootFxb1 [kN]','RootFyb1 [kN]','RootMzc1 [kNm]','LSShftFys [kN]','LSShftFzs [kN]']
axisName1=['RotPwr [kW]','RotTorq [kNm]','LSShftFxa [kN]','LSShftFys [kN]','LSShftFzs [kN]','LSShftMxa
[kNm]','LSSGagMya [kNm]','LSSGagMza [kNm]','RootFxb1 [kN]','RootFyb1 [kN]','RootFzc1 [kN]','RootMxb1
[kNm]','RootMyb1 [kNm]','RootMzc1 [kNm]','YawBrFxp [kN]','YawBrFyp [kN]','YawBrFzn [kN]','YawBrMzn
[kNm]','YawBrMxp [kNm]','YawBrMyp [kNm]','TwrBsMxt [kNm]','TwrBsMyt [kNm]']

dataset =
pd.DataFrame({'Channels':axisName1,'Max':BBmax,'SimsMax':UnicIDMax,'Min':BBmin,'SimsMin':UnicIDMin})
print(dataset)
dataset.to_csv(r'../../Load_Comparison/ExtremeTable.txt', header=True, index=None, mode='a', sep='\t')
## Creating a dataframe and saving as output.xlsx in current directory
#writer = pd.ExcelWriter('../../Load_Comparison/ExtremeTable.xlsx')
#dataset.to_excel(writer,'Sheet1')
#writer.save()

#read in the .xlsx file just created
#df_2 = pd.read_excel('../../Load_Comparison/ExtremeTable.xlsx')

#creating a pdf in called test.pdf in the current directory
pdf = FPDF()
pdf.add_page()
pdf.set_xy(0, 0)
pdf.set_font('arial', 'B', 12)
#pdf.set_text_color(0,0,255)
pdf.cell(60)
pdf.cell(70, 30, 'Extreme Table', 0, 2, 'C')
pdf.cell(-50)
pdf.cell(30, 10, 'Channels', 1, 0, 'C')
pdf.set_fill_color(0,0,255)        #to color the cells in Red,Green,Blue (RGB) order
pdf.cell(20, 10, 'Max', 1, 0, 'C', fill=True)
pdf.cell(60, 10, 'simsMax', 1, 0, 'C')
pdf.set_fill_color(255,0,0)
pdf.cell(20, 10, 'Min', 1, 0, 'C', fill=True)
pdf.cell(60, 10, 'simsMin', 1, 2, 'C')
pdf.cell(-130)
pdf.set_font('arial', '', 8)
for i in range(0, len(dataset)):
    col_channels = str(dataset.Channels.loc[i])
    col_Max = str(dataset.Max.loc[i])
    col_simsMax = str(dataset.SimsMax.loc[i])
    col_Min = str(dataset.Min.loc[i])
    col_simsMin = str(dataset.SimsMin.loc[i])
    pdf.cell(30,10, '%s' % (col_channels), 1, 0, 'C')
    pdf.set_fill_color(0,0,255)
    pdf.cell(20,10, '%s' % (col_Max), 1, 0, 'C', fill=True)
    pdf.cell(60,10, '%s' % (col_simsMax), 1, 0, 'C')
    pdf.set_fill_color(255,0,0)
    pdf.cell(20,10, '%s' % (col_Min), 1, 0, 'C', fill=True)
    pdf.cell(60,10, '%s' % (col_simsMin), 1, 2, 'C')
    pdf.cell(-130)
pdf.output('../../Load_Comparison/ExtremeTable.pdf', 'F')
```

A 5.4 DLC Aeroelastic model sample page in PDF report File.

LSShftFxa (Low-speed shaft thrust force)







A 5.5 DLC Aeroelastic model results – Extreme Table.

Extreme Table

Channels	Max	simsMax	Min	simsMin
RotPwr [kW]	82.161	VW25_00_fstInput_DLC22_13592_25_0	-60.183	VW25_00_fstInput_DLC14_EDC+r+12
RotTorq [kNm]	11.839500000000001	VW25_00_fstInput_DLC22_13592_25_0	-8.068950000000001	VW25_00_fstInput_DLC41_13772_3_0
LSShftFxa [kN]	19.494	VW25_00_fstInput_DLC41_13772_3_0	-13.127400000000002	VW25_00_fstInput_DLC41_13772_3_0
LSShftFys [kN]	1.023705	VW25_00_fstInput_DLC61_EWM01_120	-1.05219	VW25_00_fstInput_DLC61_EWM01_120
LSShftFzs [kN]	-4.861350000000001	VW25_00_fstInput_DLC22_13588_17_0	-5.869800000000001	VW25_00_fstInput_DLC51_EWM50_0
LSShftMxa [kNm]	11.839500000000001	VW25_00_fstInput_DLC22_13592_25_0	-8.068950000000001	VW25_00_fstInput_DLC41_13772_3_0
LSSGagMya [kNm]	14.283000000000001	VW25_00_fstInput_DLC12_ECD+r-2.0_11	-13.487600000000003	VW25_00_fstInput_DLC61_EWM01_30
LSSGagMza [kNm]	14.701500000000001	VW25_00_fstInput_DLC12_ECD+r-2.0_11	-12.6711	VW25_00_fstInput_DLC61_EWM01_60
RootFxb1 [kN]	6.712200000000001	VW25_00_fstInput_DLC51_EWM50_352	-4.75875	VW25_00_fstInput_DLC51_EWM50_352
RootFyb1 [kN]	1.681850000000003	VW25_00_fstInput_DLC22_13581_3_0	-2.7405	VW25_00_fstInput_DLC61_EWM01_120
RootFzc1 [kN]	21.559500000000003	VW25_00_fstInput_DLC22_13591_23_0	-1.47285	VW25_00_fstInput_DLC14_EDC+r+12
RootMxb1 [kNm]	6.9363	VW25_00_fstInput_DLC22_13591_23_0	-5.59575	VW25_00_fstInput_DLC51_EWM50_0
RootMyb1 [kNm]	19.575000000000003	VW25_00_fstInput_DLC51_EWM50_352	-13.675500000000001	VW25_00_fstInput_DLC51_EWM50_352
RootMzc1 [kNm]	0.0878445	VW25_00_fstInput_DLC22_13591_23_0	-0.080433	VW25_00_fstInput_DLC61_EWM01_180
YawBrFxp [kN]	19.872000000000003	VW25_00_fstInput_DLC41_13772_3_0	-12.59145	VW25_00_fstInput_DLC41_13772_3_0
YawBrFyp [kN]	1.02384	VW25_00_fstInput_DLC61_EWM01_120	-1.052325	VW25_00_fstInput_DLC61_EWM01_120
YawBrFzn [kN]	-13.022100000000002	VW25_00_fstInput_DLC41_13772_3_0	-16.2135	VW25_00_fstInput_DLC41_13772_3_0
YawBrMzn [kNm]	5.54985	VW25_00_fstInput_DLC22_13591_23_0	-11.734200000000001	VW25_00_fstInput_DLC61_EWM01_60
YawBrMxp [kNm]	11.981250000000001	VW25_00_fstInput_DLC22_13592_25_0	-7.768550000000005	VW25_00_fstInput_DLC51_EWM50_0
YawBrMyp [kNm]	5.54985	VW25_00_fstInput_DLC22_13591_23_0	-15.741000000000001	VW25_00_fstInput_DLC61_EWM01_30
TwrBsMxt [kNm]	27.189000000000004	VW25_00_fstInput_DLC22_13591_23_0	-22.3425	VW25_00_fstInput_DLC61_EWM01_30
TwrBsMyt [kNm]	356.40000000000003	VW25_00_fstInput_DLC41_13772_3_0	-233.41500000000002	VW25_00_fstInput_DLC41_13772_3_0