UNIVERSIDAD SAN FRANCISCO DE QUITO

Colegio de Ciencias e Ingeniería

Wind Power & Sound Generation: Aero-acoustic Noise Prediction for Wind Turbines

Sterling McBride

Patricio Chiriboga, PhD, Director de Tesis
Ricardo Burdisso, PhD, Co-Director de Tesis

Tesis de grado presentada como requisito
para la obtención del título de Ingeniero Mecánico

Quito, mayo de 2015

UNIVERSIDAD SAN FRANCISCO DE QUITO

COLEGIO DE CIENCIAS E INGENIERÍA

HOJA DE APROBACIÓN DE TESIS

Wind Power & Sound Generation: Aero-acoustic Noise Prediction for Wind Turbines

Sterling McBride

Patricio Chiriboga, Ph.D.,
Director de la tesis ____

David Escudero, Ph.D.,
Miembro del Comité de Tesis ____

Edison Bonifaz, Ph.D.,
Miembro del Comité de Tesis ____

Alfredo Valarezo, Ph.D.,
Director del programa ____

Ximena Córdova, Ph.D.,
Decana de la Escuela de Ingeniería ____
Colegio de Ciencias e Ingeniería

Quito, mayo de 2015

© DERECHOS DE AUTOR

Por medio del presente documento certifico que he leído la Política de Propiedad Intelectual de la Universidad San Francisco de Quito y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo de investigación quedan sujetos a lo dispuesto en la Política.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo de investigación en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Firma: _______________________________________

Nombre: Sterling McBride

C. I.: 1712339884

Lugar y fecha: Quito, mayo de 2015

DEDICATORIA

Para mi familia, por su incondicional apoyo.

AGRADECIMIENTOS

Este proyecto no hubiera sido posible sin el apoyo de Ricardo Burdisso, a quien agradezco por haber sido mi mentor y guía durante el desarrollo del mismo. Las enseñanzas que me ha transmitido, han despertado en mí un espíritu de investigación y de excelencia. Adicionalmente, agradezco a Patricio Chiriboga, ya que gracias a su respaldo y asesoramiento, pude exitosamente llevar a consecución ésta investigación.

RESUMEN

El sonido producido por turbinas de viento se ha convertido en un asunto de preocupación durante los últimos años, especialmente debido a que con el pasar del tiempo se están construyendo turbinas de mayor tamaño, y muchas de ellas se encuentran cerca de lugares poblados. El objetivo principal de este trabajo es crear un modelo que prediga los niveles de presión de sonido [dBa] en una malla esférica alrededor de la turbina. En este caso, se ha utilizado una turbina de modelo Nordtank NTK 500/4, y para obtener los resultados deseados, se han utilizado datos experimentales realizados en un túnel de viento.

Los resultados obtenidos muestran que los niveles de presión de sonido, con ponderación-A tienen valores máximos de 35 [dBa], y mínimos de 0 [dBa]. Los valores más altos se encuentran localizados en los polos de la esfera, en dirección paralela a la rotación de los alabes. Los valores más bajos se encuentran en el centro de la esfera, en dirección perpendicular a la rotación de los alabes. Adicionalmente, se observa que a medida que los alabes giran, el sonido está caracterizado por un comportamiento oscilatorio.

ABSTRACT

The noise produced by wind turbines has become a matter of concern during the last few years, especially since larger wind turbines are being built and some of them near populated areas. The main purpose of this thesis is to create a model that predicts the overall A-weighted sound pressure levels on a spherical grid around a wind turbine. In this case, the wind turbine used for modeling was Nordtank NTK 500/4, and in order to obtain results, experimental wind tunnel airfoil noise data was used.

The obtained results show that the overall A-weighted sound pressure levels on the sphere grid surrounding the wind turbine have maximum values that are around 35 [dBa] and minimum of 0[dBa]. The highest values are located on the poles of the sphere, parallel to the plain of rotation of the blades. The lowest values are located at the center of the sphere, perpendicular to the plain of rotation of the blades. Additionally, it was determined that as the blades of the wind turbine rotate, the noise is characterized by an oscillating behavior.

TABLE OF CONTENTS

INTRODUCTION...17
1.1 Justification 17
1.2 Objectives...18
1.3 Background 19
1.4 Methodology 19
1.5 Literature Review 19
1.5.1 Investigations Focused on the Sound Produced by Wind Turbines. 20
1.5.2 Investigations on the Structural Behavior of Wind Turbines. 22

THEORETICAL FRAMEWORK..24
2.1 Speed of Sound 24
2.2 Spherical Acoustic Waves 25
2.3 Pressure Root Mean Square Values..25
2.4 Sound Pressure Levels...26
2.5 1/3rd Octave Frequency Bands...27
2.6 Overall Sound Pressure Levels...29
2.7 A-Weighting...29
2.8 Directivity 30
2.9 Sound Intensity and Power 30
2.10 Anechoic and Reverberant Chambers...31
2.11 Microphone Phased Arrays 33
2.12 Flow around Wind Turbine Blades..35
2.13 Lift and Drag 37
2.14 Flows Past an Object & Boundary Layer Structures...............................38
2.15 Boundary Layer Thickness...39

WIND TUNNEL AIRFOIL NOISE DATA...41
3.1 Aero-acoustic Test Facilities 41
3.2 Experimental Test Procedure 44
3.3 Data Processing & Results 46
3.3.1 Kevlar Window Associated Sound Pressure Level Corrections............................47
3.3.2 Normalization procedure. 51
3.4 DU96_main.m MATLAB Code Description..60

EXPERIMENTAL RESULTS ANALYSIS AND DISCUSSION 64
4.1 NAFNoise Program Description..64
4.1.1 Airfoil Noise Sources...65
4.1.2 NAFNoise Inputs & Outputs. 69
4.2 Experimental & Modeled data Analysis 73
4.3 NAFNOISE_main.m MATLAB Code Description.....................................81

WIND TURBINE NOISE PREDICTION 83
5.1 Prediction Model Description 83
5.1.1 Incoming Wind Velocity Modeling. 84
5.1.2 Wind Turbine Blade Geometry Modeling. 85
5.1.3 Rotational Positions Modeling...89
5.1.4 Spherical Grid Modeling..91
5.1.5 Noise Source Modeling...92
5.1.6 Overall A-Weighted Sound Pressure Levels Computation. 95

5.2 Noise Prediction Results 96
5.3 WTNOISE_main.m MATLAB Code Description 100

CONCLUSIONS & RECOMMENDATIONS 104

REFERENCES 106

APPENDIX A: DU96.xlsx File 109

APPENDIX B: DU96_main.m 110

APPENDIX C: nafnoise.ipt file 131

APPENDIX D: nafnoise.out File 132

APPENDIX E: NAFNoise_main.m 133

APPENDIX F: WTNOISE_main.m 143

LIST OF TABLES

Table 1. Sound pressure levels for some sources. Retrieved from: Biess, D. & Hansen (2009), Engineering Noise Control, Fourth Edition, CRC Press –USA.......................................27

Table 2. Octave and 1/3rd Octave Frequency Bands. Retrieved from: Biess, D. & Hansen (2009), C., Engineering Noise Control, Fourth Edition, CRC Press -USA. 28

Table 3. Acoustic Coefficient for Acoustic Materials. Retrieved from: Biess, D. & Hansen (2009), Engineering Noise Control, Fourth Edition, CRC Press -USA............................32

Table 4. Target and Actual Flow Velocities for Wind Tunnel Test. 46

Table 5. Curve Fit Coefficients for Each Angle of Attack. 57

Table 6. Experimental and Modeled Overall Sound Pressure levels for a) flow velocity = 34 m/s & AOA=4 deg, b) flow velocity = 44 m/s & AOA=4 deg, c) flow velocity = 54 m/s & AOA=4 deg, d) flow velocity = 64 m/s & AOA=4 deg..75

Table 7. Experimental and Modeled Overall Sound Pressure levels for a) flow velocity = 34 m/s & AOA=8 deg, b) flow velocity = 44 m/s & AOA=8 deg, c) flow velocity = 54 m/s & AOA=8 deg, d) flow velocity = 64 m/s & AOA=8 deg..77

Table 8. Experimental and Modeled Overall Sound Pressure levels for a) flow velocity = 34 m/s & AOA=12 deg, b) flow velocity = 44 m/s & AOA=12 deg, c) flow velocity = 54 m/s & AOA=12 deg, d) flow velocity = 64 m/s & AOA=12 deg. 79

Table 9. Experimental and Modeled Overall Sound Pressure levels for a) flow velocity = 34 m/s & AOA=16 deg, b) flow velocity = 44 m/s & AOA=16 deg, c) flow velocity = 54 m/s & AOA=16 deg, d) flow velocity = 64 m/s & AOA=16 deg. 80

Table 10. Twist and Chord distributions for a Nordtank NTK 500/41 blade. Retrieved From: Hansen, M. (2008). Aerodynamics of Wind Turbines. (pp.58) Earthscan, Sterling, VA-USA. 86

Table 11. Interpolated Twist Distribution for a Nordtank NTK 500/41 blade 87

LIST OF FIGURES

Figure 1. International A, B, and C weighting curves for sound level meters. Retrieved from: Bies, D. & Hansen, C. (2009). Engineering Noise Control: Theory and Practice (4th edition). New York, NY, USA: Taylor and Francis. 29

Figure 2. Anechoic Chamber Walls Assembly. Retrieved from http://www.ilcc.inf.ed.ac.uk/facilities/facilities-images/IMG-2965-1.JPG/image_view_fullscreen. 33

Figure 3. Plane Microphone Phased Array. Retrieved From: http://xn--42cga5ca0b6cdc2bzb3a7ble0ewa3nna3m.com/en/portfolio/microphone-array/. 34

Figure 4. Acoustic Map from Wind Tunnel Test. Retrieved from "Aero-acoustic Testing of Wind Turbine Airfoils", W. Davenport et al., 2010. NREL/SR-500-43471. National Renewable Energy Laboratory. 35

Figure 5. Airfoil with Definition of Flow Angles and Forces. Retrieved from: Gipe, P. (2004), Renewable Power: Renewable Energy for Home, Farm, and Business, Chelsea Green Publishing Company, USA. 36

Figure 6. Change of Flow Velocity Components along an Un-Twisted Wind Turbine Blade. Retrieved From: http://www.learnengineering.org/2013/08/Wind-Turbine-Design.html. 37

Figure 7. Pressure & Shear Stress Distributions. Retrieved from: Munson, B, Okiishi, T., Huebsch, W., Rothmayer, A. (2013). Fundamentals of Fluid Mechanics. P.482. John Wiley & Sons. Hoboken, NJ. 38

Figure 8. Laminar & Turbulent Boundary Layer. Retrieved from: Munson, B, Okiishi, T., Huebsch, W., Rothmayer, A. (2013). Fundamentals of Fluid Mechanics. P.490. John Wiley & Sons. Hoboken, NJ. 39

Figure 9.Boundary Layer Thickness: a) Standard Boundary Layer Thickness, b) Boundary Layer Displacement Thickness. Retrieved from: Munson, B, Okiishi, T., Huebsch, W., Rothmayer, A. (2013). Fundamentals of Fluid Mechanics. P.491. John Wiley & Sons. Hoboken, NJ. 40

Figure 10. General layout of the Virginia Tech Stability Wind Tunnel in Anechoic Configuration. Retrieved from http://www.aoe.vt.edu/research/facilities/stabilitytunnel/acoustics-stabilitytunnel.html..42

Figure 11. Side Cross Section Area of Anechoic Chambers. Retrieved from "Aero-acoustic Testing of Wind Turbine Airfoils", W. Davenport et al., 2010. NREL/SR-500-43471. National Renewable Energy Laboratory. 43

Figure 12. Top Cross Section View of Anechoic System. Retrieved from "Aero-acoustic Testing of Wind Turbine Airfoils", W. Davenport et al., 2010. NREL/SR-500-43471. National Renewable Energy Laboratory. 43

Figure 13. Chord and Span of the DU-96 Airfoil. 44

Figure 14. Wind Tunnel Airfoil Test Configuration (Angle of Attack of Zero Degrees). 45

Figure 15. Attenuation of sound passing through acoustic Kevlar windows as a function of frequency. Retrieved from "Aero-acoustic Testing of Wind Turbine Airfoils", W. Davenport et al., 2010. NREL/SR-500-43471. National Renewable Energy Laboratory.48

Figure 16. Attenuation of sound passing through the shear boundary layer as a function of frequency. Retrieved from "Aero-acoustic Testing of Wind Turbine Airfoils", W. Davenport et al., 2010. NREL/SR-500-43471. National Renewable Energy Laboratory.49

Figure 17. Corrected SPL [dB] vs. Strouhal Number for all Velocities, When the Angle of Attack is a) 4 Degrees, b) 8 Degrees. 54

Figure 18. Corrected SPL [dB] vs. Strouhal Number for all Velocities, When the Angle of Attack is a) 12 Degrees, b) 16 Degrees. 54

Figure 19. Collapsed Curves for Each Angle of Attack, for all Free Stream Velocities. 54

Figure 20. Collapsed Data Curve Fit for an Angle of Attack of 4 Degrees. 55

Figure 21. Collapsed Data Curve Fit for an Angle of Attack of 16 Degrees. 56

Figure 22. Collapsed Data Curve Fit for an Angle of Attack of 8 Degrees. 56

Figure 23. Collapsed Data Curve Fit for an Angle of Attack of 12 Degrees. 57

Figure 24. First Coefficient (a) Fit as a Function of the Angle of Attack. 58

Figure 25. Second Coefficient (b) Fit as a Function of the Angle of Attack. 59

Figure 26. Third Coefficient (c) Fit as a Function of the Angle of Attack. 59

Figure 27. SPL as a Functions of the Strouhal Number for an Angle of Attack of a) 4 Degrees, b) 8 Degrees, c) 12 Degrees, d)16 Degrees. 60

Figure 28. DU96_main.m Code Flow Chart. 62

Figure 29. Turbulent Boundary Layer-Trailing Edge Noise. Retrieved From: Brooks, F., Pope, S., Marcolini, M. (1989). Airfoil Self-Noise and Prediction. NASA Reference Publication 1218. 65

Figure 30. Flow Separation Noise. Retrieved From: Brooks, F., Pope, S., Marcolini, M. (1989). Airfoil Self-Noise and Prediction. NASA Reference Publication 1218. 66

Figure 31. Laminar Boundary Layer-Vortex Shedding Noise. Retrieved From: Brooks, F., Pope, S., Marcolini, M. (1989). Airfoil Self-Noise and Prediction. NASA Reference Publication 1218. 67

Figure 32. Trailing Edge Bluntness-Vortex Shedding Noise. Retrieved From: Brooks, F., Pope, S., Marcolini, M. (1989). Airfoil Self-Noise and Prediction. NASA Reference Publication 1218. 68

Figure 33. Trailing Edge Angle 72

Figure 34. Experimental and Modeled Data for a) flow velocity = 34 m/s & AOA=4 deg, b) flow velocity = 44 m/s & AOA=4 deg, c) flow velocity = 54 m/s & AOA=4 deg, d) flow velocity = 64 m/s & AOA=4 deg. 74

Figure 35. Experimental and Modeled Data for a) flow velocity = 34 m/s & AOA=8 deg, b) flow velocity = 44 m/s & AOA=8 deg, c) flow velocity = 54 m/s & AOA=8 deg, d) flow velocity = 64 m/s & AOA=8 deg. 76

Figure 36. Experimental and Modeled Data for a) flow velocity = 34 m/s & AOA=12 deg, b) flow velocity = 44 m/s & AOA=12 deg, c) flow velocity = 54 m/s & AOA=12 deg, d) flow velocity = 64 m/s & AOA=12 deg. 78

Figure 37. Experimental and Modeled Data for a) flow velocity = 34 m/s & AOA=16 deg, b) flow velocity = 44 m/s & AOA=16 deg, c) flow velocity = 54 m/s & AOA=16 deg, d) flow velocity = 64 m/s & AOA=16 deg. 80

Figure 38. NAFNOISE_main.m Flow chart. 82

Figure 39. Effect of terrain roughness on the wind velocity profile. Retrieved from: Rao,C.S. (2006). Environmental Pollution Control Engineering. New Age International Ltd., New Delhi. 84

Figure 40. United States-Land Based and Offshore Annual Average Wind Speed at 100 m. Retrieved from: http://www.nrel.gov/gis/wind.html 85

Figure 41. Wind Turbine Blade divided into sections. Retrieved From: http://www.becas.dtu.dk/About 86

Figure 42. Airfoil with Definition of Flow Angles and Forces. Retrieved from: Gipe, P. (2004), Renewable Power: Renewable Energy for Home, Farm, and Business, Chelsea Green Publishing Company, USA. 88

Figure 43. Wind Turbine Coordinates. Retrieved From: van Rooij, R. (2001). Terminology, Reference Systems and Conventions. Duwind 2001.004 89

Figure 44. Blades Rotation Given by Azimuth Angle. Retrieved from: Hansen, M. (2008). Aerodynamics of Wind Turbines. (pp.58) Earthscan, Sterling, VA-USA. 90

Figure 45. Spherical Grid. Retrieved From: http://www-personal.umich.edu/~cjablono/dycore_test_suite.html. 91

Figure 46. Spherical Coordinates Definition. Retrieved From: Larson, R. &Edwards, B. (2010). Multivariable Calculus. Cengage Learning, Inc. Belmont, CA, USA.................92

Figure 47. Radiation Pattern of a Dipole at the Trailing Edge of an Airfoil 93

Figure 48. Dipole Unit Vector for each blade section for all azimuth rotation blade positions. 95

Figure 49. Angles of Attack for all Rotational Positions of the Blades..................................97

Figure 50. Apparent or Relative Flow Velocities for all Rotational Positions of the Blades. .98

Figure 51. Predicted OASPL Over Spherical Grid around Turbine (First Position of Blades), with 2501 Grid Points. 99

Figure 52. Predicted OASPL Over Spherical Grid around Turbine (First Position of Blades), with 121 Grid Points. 100

Figure 53. WTNOISE_main.m flowchart...101

CHAPTER I

INTRODUCTION

1.1 Justification

The worldwide wind power capacity has constantly been growing each year during the last two decades, as it is evidenced by the Global Wind Council's, Global Wind Statistics Report from 2014, where it is shown that the global cumulative installed wind capacity has increased from 7,600 MW in 1997 to 369,553 MW in 2014. The main reason for this, are the low costs involved with building and operating wind power facilities, compared to other renewable energy production methods. For example, in the United States of America, where one of the world's largest wind power capacity is installed, the levelized cost of electricity for wind power is one of the lowest. The levelized cost represents the amount of dollars per kilowatt-hour necessary for building and operating a generation plant over a given time cycle (U.S. Energy Information Administration, 2014).

Even though wind power generation has certain economic advantages and is considered a very favorable renewable energy production process, there is a concern over the noise produced by them, and how it affects to humans and the environment. There are research studies that provide information supporting the argument that wind turbine noise has a direct cause on powerful adverse health conditions, whereas there are others that claim that wind turbine noise direct effects are harmless (Ryan, 2014).

According to Pedersen et al. (2004), a wind turbine has high annoyance noise effects on a person who is close to it a considerable amount of time. This is due to both the quality and oscillating levels (beating character) of the sound produced by the aerodynamic movement of air around the wind turbines, as the blades rotate. Another concern with wind turbine noise is concentrated on the effects on human health due to the production of infrasound (sound frequencies lower than 20 Hz), for the reason that studies show that this phenomena causes fatigue, lack of sleep, feeling of apathy, irritability and loss of concentration (Roberts & Roberts, 2009).

Alternatively, some of the negative effects on people's behavior and on their health could be explained by a psychological phenomenon where negative thoughts associated with the wind turbines' noise environmental changes, stimulate negative outcomes. This argument has been supported by the fact that the reported expectations of people has a significant role on their reported symptoms (Ryan, 2014).

Nonetheless of the causes of noise emission related afflictions on people living close to wind turbines, better tools that assist on the research of wind turbine noise and its consequences are needed. This is the reason why the prediction of wind turbine noise is a matter of great importance.

1.2 Objectives

The main purpose of the research is to develop a tool that predicts the propagation of noise produced by wind turbines, through theoretical and experimental procedures, in order to obtain a model that is as accurate as possible with real conditions. The specific objectives include:

- Obtain empirical sound pressure level data, through airfoil wind tunnel experimental procedures, so the noise produced by an airfoil is known.
- Determine how wind tunnel experimental data for an airfoil compares to semi-empirical models developed by other researchers, in order to analyze the behavior of both data sets, and how adequate both fit.
- Establish wind turbine's noise sources and their behavior, by using experimental and theoretical modeling techniques, so a noise propagation model is proposed.

1.3 Background

According to the 2011 World Health Organization Report, traffic noise is the second largest environmental factor affecting human health in the European Union and Norway. Nevertheless, turbine noise is often estimated more annoying that the one produced by transportation systems because of its high variability in both level and quality (Ryan, 2014). Therefore, it has become an imperative need to investigate the sound propagation of the noise produced by wind turbines, by using computerized models.

A tool that predicts the behavior of wind turbine noise, accurately, has not yet been reproduced. Yet, some research models studies have determined certain possible methods that could be useful in determining how noise produced by wind turbines behave. All these resources should be used to reproduce an efficient tool that determines the wind turbine noise propagation.

1.4 Methodology

In order to develop the project, a computational MATLAB code is to be used to reproduce every activity planed. This includes the modeling of noise sources, the determination of influence variables, the introduction of wind tunnel test data, processing experimental data, reproduce semi-empirical data, and finally obtaining results. The results shall be presented, discussed, and conclusion should be withdrawn.

1.5 Literature Review

Currently 65% of electricity production in industrialized countries comes from fossil fuels, which constantly contributes to pollution of the environment (Ryi , Choi, Lee, & Lee , 2014) . Hence the development of renewable energies that are reliable is necessary. Wind turbines have proven to be an innovative and effective solution, however they still need to be improved in terms of their efficiency and production of noise pollution. This means that

Research in areas of propagation and behavior of sound is essential for the optimization of this system of electricity production.

1.5.1 Investigations Focused on the Sound Produced by Wind Turbines.

Sound propagation of a wind turbine can be determined by a prediction obtained from scaled models. Ryi et al. (2013) carried out a process of experimentation to find a methodology that results in the prediction of sound. This was done first, by using an airfoil in a wind tunnel, then using a scale model turbine in the wind tunnel, and finally a full-size turbine test was done. The results obtained determined the errors between 5 and 15 [dBa] on the total sound pressure levels, and for each 1/3rd octave frequency band there were also observed inconsistencies. This shows that the method of weighting should be analyzed thoroughly to determine results consistent with reality.

According to Oerlemans & Schepers (2009), a semi - empirical prediction of the trailing edge of the airfoil of a turbine would be computationally achieved through a code that uses as input the geometry of the turbine and the conditions under which this operates. As a validation method, measurements of directivity and acoustic arrangements are used. The results of such techniques show that there is less than 2 dB difference between the predicted and the real sound pressure levels. It also shows that the average pressure levels are higher downwind of the wind turbine, when the turbine operates under crosswinds.

On the other hand, according to a study conducted by Tadamasa & Zangeneh (2011), in order to determine a method that obtains the contribution of aerodynamic sound on discrete frequency bandwidth of a wind turbine, numerical methods must be applied. This can achieved through a program that solves computational fluid dynamics (CFD), which gives as a result the relevant parameters used in the equations of Ffowcs Williams- Hawkings (FW-H). Thus, sound loading and the overall sound radiated by the turbine can be obtained. This study´s results covers a wide range of working conditions of the turbine, and an experimental validation of the results. Furthermore, it utilizes tools such as CFD in an innovative manner and a code where FW- H can be used to obtain more efficient turbine blade designs.

Undoubtedly, the sound caused by a wind turbine comes from the airflow that makes it rotate, so its operation conditions should be studied. Makarewicz (2013) has determined that the velocity gradient of the wind, under which the wind turbine is exposed, causes a refraction phenomenon that affects levels of average sound pressure over time. Additionally, the sound source of the turbine can be modeled as a series of points around a circle. This only if the effects of directivity of the edge of the airfoil and Doppler amplification are not taken into account. Under these concepts the sound pressure levels around the turbine can be determined, and the existence of a fully sonified zone and shadow zone.

According to Makarewicz & Golebiewski (2014), there is an edge called Partial Ensonified completely separating the sonified area (near the turbine) from the shadow zone (away from the turbine). To obtain an appropriate behavior model of the average A-weighted pressure levels in the time, instead of using point sources, a rotor and a two-dimensional simulation of drive sound source is used. Following these guidelines, a correction of the sound pressure levels in the ensonified region takes into account only the emission corresponding to a certain distance from the blade. In this research, the behavior of the desired pressure levels are obtained successfully, however it is unclear why only the upper tip of the blades is used as the only reference in obtaining the sound pressure levels on the ensonified area.

The shape of the blades of a turbine are directly responsible for the aerodynamic conditions under which it operates. Simulations to determine the interaction between the edge of the airfoil and formed vortices determine the behavior of the sound emitted by the turbine.

Arakawa et al. (2005) developed an aerodynamic computational model in order to find a tool that serves to predict the far-field sound and encourage the use of such research to study the near field, considering reflection phenomena. This research is based on three essential points: a compressible simulation LES (Large Eddy Simulation), a direct simulation of acoustic propagation of hydrodynamic field, and a long field prediction using the method of integration of Ffowcs Williams- Hawkings (FW -H). It also emphasizes that through other more common methods such as the use of a Reynolds Averaged Navier Stokes (RANS) simulation, concrete results that determine best designs of airfoils with better accuracy, may not be obtained.

1.5.2 Investigations on the Structural Behavior of Wind Turbines.

The structure of a wind turbine is constantly vibrating, which is important not only from a structural and materials point of view, but also from the point of view of sound production. All kinds of vibration produces sound, so it is imperative to know the vibrational behavior of turbine blades. The maximum deformation that occurs at the tip of the blades must be studied in conjunction with the vibration modes and natural frequencies obtained after vibration analysis (Kumar Dwivedi , Paliwal , & Patil, 2014) . It is important to note that the natural frequencies of the turbine blades may result in structural deformation and therefore an increase in the loads that these support. This can affect the behavior of the fluid on the turbine and result in turbulent flows over the airfoil, which determines the amount of sound emitted.

According to Mollaselehi et al. (2013) , the sound generated by the structure of the tower of small wind turbines cannot be neglected , because this type of turbines are usually placed in populated areas and at small distances the tower acts as a source of line sound . Operational modal analysis was performed by placing accelerometers, so that the natural frequencies and forms of deflection of the tower can be obtained. When analyzing the spectrum created, it was determined that most of the vibration happens at low frequency bands, especially under 10 Hz. In the same study, a model of a tower was performed to determine the structural fluid-acoustic interaction and it was determined that there is a significant vibrational frequency content about at 48 Hz, despite not being one of the natural frequencies.

The sources of mechanical noise from a wind turbine come from the transmission, generator, electric motors, cooling fans, auxiliary equipment and brakes. All these components are constantly vibrating and usually the sound is shaped as tones even if it has components in the spectrum band. However, in modern turbines mechanical sound is usually not significant in comparison to the aerodynamic sound (Tonin, 2012).

It can be concluded that the investigation on the production and propagation of sound from wind turbines, covers a range of expertise in aerodynamics, vibration and acoustics. Most of these focus on modeling using computational analysis using CFD, FEA analysis and others. Only in a few cases it is observed a computational analysis of discrete sound sources where geometric, dynamic and aerodynamic factors are included.

CHAPTER II

THEORETICAL FRAMEWORK

In this chapter, theoretical concepts that are used throughout the entire thesis are explained. These include basic acoustic definitions, a noise measuring method, aerodynamics of airfoils and wind turbines, and some fluid mechanics principles.

2.1 Speed of Sound

Sound is conducted through a surrounding medium which is generally air, but it can also be any fluid or solid. In fluids, the sound is conducted through longitudinal waves where there’s is a particle motion parallel to the direction of propagation. The propagation speed or speed of sound is dependent of stiffness D and density ρ of the medium, as shown in equation 2.1.

$$c = \sqrt{\frac{D}{\rho}}\ \ [m/s] \tag{2.1}$$

In the case of a fluid, the stiffness is the bulk modulus or the reciprocal of the compressibility. The definition of the compressibility is defined in equation 2.2, where V is a unit volume and $\frac{\delta V}{\delta P}$ is the incremental change in volume associated with the incremental change in static pressure.

$$D_f = -V\left(\frac{\delta V}{\delta P}\right)^{-1} = \rho\left(\frac{\delta P}{\delta \rho}\right)\ [Pa^{-1}] \tag{2.2}$$

In the cases of gases, temperature and density are two important variables because they are associated with variations in pressure. For gases, the equation for adiabatic compression and the equation of state for gases gives the speed of sound, shown in equation 2.3, where γ the ratio of specific heats, T is the temperature in degrees Kelvin, R is the universal constant of gases equivalent to $8.314\, Jmol^{-1}K^{-1}$, and M is the molecular weight.

$$c = \sqrt{\frac{\gamma P}{\rho}} = \sqrt{\frac{\gamma RT}{M}}\,[{}^{m}/_{s}] \tag{2.3}$$

If the sound propagates through air, the speed of sound will be given by equation 2.4, where *T* is the temperature in degrees Celsius (Bies & Hansen, 2009).

$$c = 331 + 0.6T\ [{}^{m}/_{s}] \tag{2.4}$$

2.2 Spherical Acoustic Waves

The propagation of sound of a small source in free space, with no boundaries, is given by spherical wave propagation. A source that produces spherical waves is called a monopole or point source, and is characterized by the fact that the physical dimension of the source is smaller than the wave´s wavelength (Bies & Hansen, 2009). The pressure equation for a spherical wave is given in equation 2.5. P is a constant that can be obtained by using boundary conditions that establish that the pulsating velocity of the source, is equal to the air particles velocity at the source's surface. r is the radial distance from the source [m], *w* is the wave's frequency [rad/s], t is time [s], and $k = \frac{w}{c}$, c is the speed of sound [m/s] (Pierce, 2014).

$$p(r,t) = \frac{P}{r}e^{j(\omega t - kr)} \tag{2.5}$$

The analysis of sound wave propagation is usually complicated. Nevertheless, the propagation properties of spherical and plane waves permit a simplified analysis, because they can be described in terms of one dimension.

2.3 Pressure Root Mean Square Values

The root mean square value of a function is defined in equation 2.6, where T is the time duration over which the average value is taken. When the applied function corresponds to a sound pressure wave equation, the root of the time averaged squared sound pressure is obtained.

$$X_{rms} = \sqrt{\lim_{T\to\infty}\frac{1}{T}\int_0^T x(t)^2\,dt} \tag{2.6}$$

For a sinusoidal function, such as those given by sound pressure wave equations, the root mean square value is equal to the amplitude A of the wave, divided by the square root of two, as shown in equation 2.7 (Bell & Bell, 1994).

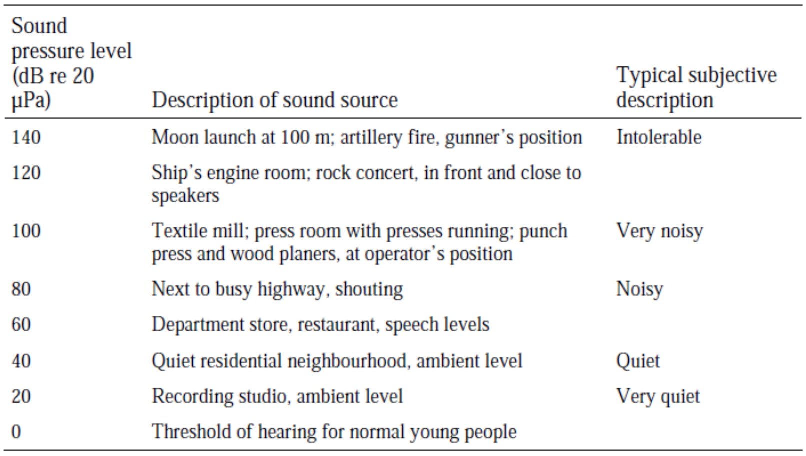
$$P_{rms} = \frac{A}{\sqrt{2}} \tag{2.7}$$

2.4 Sound Pressure Levels

The minimum sound pressure that the human ear is able to detect is 20×10^{-6} [Pa] and the maximum is 60 [Pa]. According to Bies & Hansen (2009), “The incredible dynamic range of the ear suggests that some kind of compressed scale should be used. A scale suitable for expressing the square of the sound pressure in units best matched to subjective response is logarithmic rather than linear.” (p. 38). Consequently, sound pressure levels L_p are determined, as observed in equation 2.8.

$$L_p = 10\,log_{10}\frac{P_{rms}^2}{P_{ref}^2} \tag{2.8}$$

The levels are expressed in terms of the lowest pressure a human ear can detect, which is called P_{ref}. Additionally, a factor of 10 is added so the scale is not too compressed, and the square of the pressure root mean square values are used. The units of sound pressure levels are the Decibels [dB]. Some different noise sources and their corresponding sound pressure levels are presented in Table 1.



Sound pressure level (dB re 20 µPa)	Description of sound source	Typical subjective description
140	Moon launch at 100 m; artillery fire, gunner's position	Intolerable
120	Ship's engine room; rock concert, in front and close to speakers	
100	Textile mill; press room with presses running; punch press and wood planers, at operator's position	Very noisy
80	Next to busy highway, shouting	Noisy
60	Department store, restaurant, speech levels	
40	Quiet residential neighbourhood, ambient level	Quiet
20	Recording studio, ambient level	Very quiet
0	Threshold of hearing for normal young people	

Table 1. Sound pressure levels for some sources. Retrieved from: Biess, D. & Hansen (2009), Engineering Noise Control, Fourth Edition, CRC Press –USA

2.5 One Third Octave Frequency Bands

One third octave frequency bands are used to represent sound spectrums. The reason why this is done, is because these type of bands facilitate the comparison of noise data measurements. Moreover, they have been determined to be the preferred ones for noise applications by The International Standards Organization, because they have the widest band for frequency analysis.

Each octave band is characterized by an upper limit, lower limit and center frequency, from a narrowband spectrum. An octave band center frequency is related to a designated band number, and the center frequency is equal to the square root of the upper and lower frequencies of the band. This is shown in equations 2.9 and 2.10, where BN is the band number, f_c is the center band frequency, f_l is the lower limit band frequency, and finally f_u is the upper limit band frequency.

$$BN = 10\ log_{10} f_c \tag{2.9}$$

$$f_c = \sqrt{f_u f_l} \tag{2.10}$$

A small adjustment to octave frequency bands determines the 1/3rd octave frequency bands, and defines new center, lower limit and upper limit band frequencies. Equation 2.11

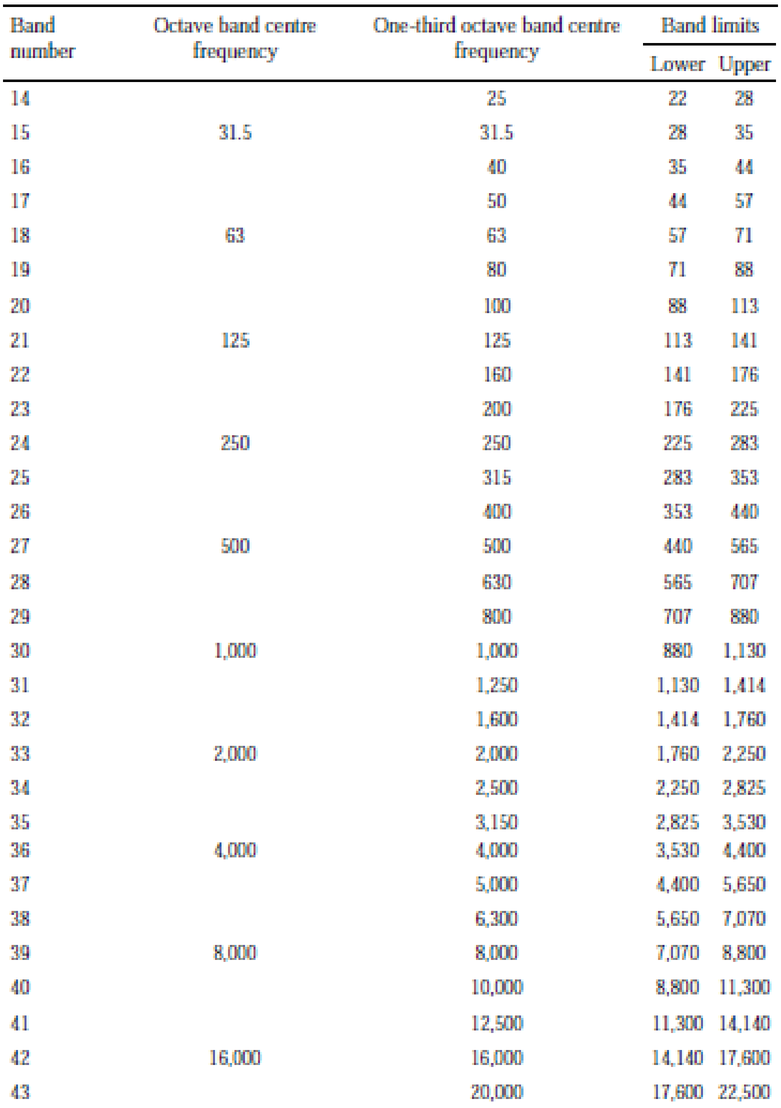
shows this adjustment. For octave frequency bands N=1, and for 1/3rd octave frequency bands, N=3.

$$^{f_u}/_{f_l} = 2^{1/N} \tag{2.11}$$

On the other hand, the band width Δf is given by equation 2.12.

$$\Delta f = f_c \frac{2^{1/N-1}}{2^{1/2N}} \tag{2.12}$$

A list of octave and 1/3rd octave frequency bands is shown in Table 2 (Bies & Hansen, 2009).



Band number	Octave band centre frequency	One-third octave band centre frequency	Band limits	
			Lower	Upper
14		25	22	28
15	31.5	31.5	28	35
16		40	35	44
17		50	44	57
18	63	63	57	71
19		80	71	88
20		100	88	113
21	125	125	113	141
22		160	141	176
23		200	176	225
24	250	250	225	283
25		315	283	353
26		400	353	440
27	500	500	440	565
28		630	565	707
29		800	707	880
30	1,000	1,000	880	1,130
31		1,250	1,130	1,414
32		1,600	1,414	1,760
33	2,000	2,000	1,760	2,250
34		2,500	2,250	2,825
35		3,150	2,825	3,530
36	4,000	4,000	3,530	4,400
37		5,000	4,400	5,650
38		6,300	5,650	7,070
39	8,000	8,000	7,070	8,800
40		10,000	8,800	11,300
41		12,500	11,300	14,140
42	16,000	16,000	14,140	17,600
43		20,000	17,600	22,500

Table 2. Octave and 1/3rd Octave Frequency Bands. Retrieved from: Biess, D. & Hansen (2009), C., Engineering Noise Control, Fourth Edition, CRC Press -USA.

2.6 Overall Sound Pressure Levels

The overall or total sound pressure levels are determined by the sum of incoherent sounds, which corresponds to noise produced by different sources or bands with random phases. The computation is done by using equation 2.13 (Bies & Hansen, 2009).

$$OSPL = L_{pt} = 10\, log_{10}\left(10^{L_1/10} + 10^{L_2/10} + \cdots + 10^{L_N/10}\right) \qquad (2.13)$$

2.7 A-Weighting

The apparent loudness sensed by the ear varies with frequency and sound pressure. Measuring instruments permit allowances that take into account the ears behavior by providing weighting methods, recommended by standard organizations. A, B, and C networks are the most utilized, but the A-weighting curve has higher approximation accuracy of the ear response to low level sound. Figure 1 shows the correction that must be added to a reading, for a determined 1/3rd octave frequency band.

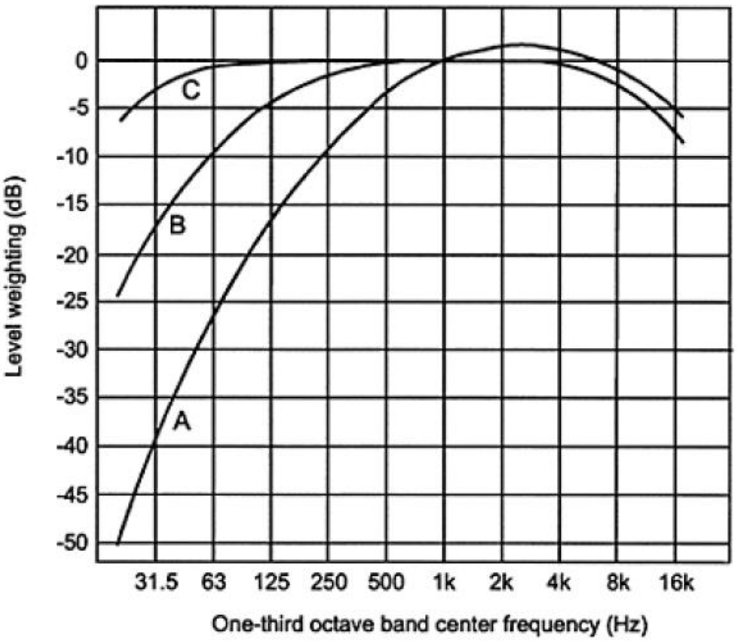


Figure 1. International A, B, and C weighting curves for sound level meters. Retrieved from: Bies, D. & Hansen, C. (2009). Engineering Noise Control: Theory and Practice (4th edition). New York, NY, USA: Taylor and Francis.

The expression to determine the A-weighting values is given by equation 2.14a and 2.14b.

$$R_A(f) = \frac{12200^2 \times f^4}{(f^2+20.6^2)\sqrt{(f^2+107.7^2)(f^2+737.9^2)}(f^2+12200^2)} \qquad (2.14a)$$

$$A \;= 2.0 + 20\ log_{10}\big(R_A(f)\big) \quad [dB] \qquad (2.14b)$$

2.8 Directivity

The noise radiation produced by a source is dependent on the type of source, and is usually directional, meaning that it is larger in some directions. In the case of far filed radiation, the directional behavior of sound may be quantified by a dimensionless directivity factor, which is shown in equation 2.15. Here, $< I > = \,^{W}\!/_{4\pi r^2} \left[\frac{Watt}{m^2}\right]$ is the mean intensity averaged over a spherical surface, and $I_\theta \left[\frac{Watt}{m^2}\right]$ is the intensity in the desired direction as a function of two angles: (θ, ψ).

$$D_\theta = \frac{I_\theta}{<I>} \qquad (2.15)$$

Alternatively, it can also be expressed in logarithmic scale with the directivity index, as shown in equation 2.16 (Bies & Hansen, 2009).

$$DI = 10\ log_{10}\ D_\theta \qquad (2.16)$$

2.9 Sound Intensity and Power

The sound intensity of a source a vector obtained from the product of the sound pressure and the component of particle velocity in the direction of the intensity vector, as shown in equation 2.17 The intensity represents the rate at which work is done on a conducting medium by an advancing sound wave, or the rate of the power transmitted through a surface normal to the intensity vector.

$$\overline{I(r)} = < p(\bar{r},t)\bar{u}(\bar{r},t) > = \lim_{T\to\infty} \frac{1}{T}\int_0^T p(\bar{r},t)\bar{u}(\bar{r},t)\ dt\ \left[{}^{Watt}\!/_{m^2}\right] \qquad (2.17)$$

When obtaining the resulting expressions for intensity, for the cases of plane waves and spherical waves, two components are obtained. The first one is the propagated active intensity and the second one is the reactive intensity, which is a measure of the energy stored in the field during each cycle, but is not transmitted.

Given that the intensity of a source measures the power passing through unit area of an acoustic medium, the power is obtained as shown in equation 2.18, where $\bar{n}$ is the unit vector normal to the surface S. In the case where a noise source produces uniform spherical waves, a spherical surface is convenient and the power is determined as shown in equation 2.19 (Bies & Hansen, 2009).

$$W = \int \bar{I} \cdot \bar{n}\, dS \qquad (2.18)$$

$$W = 4\pi r^2\ I\ [Watt] \qquad (2.19)$$

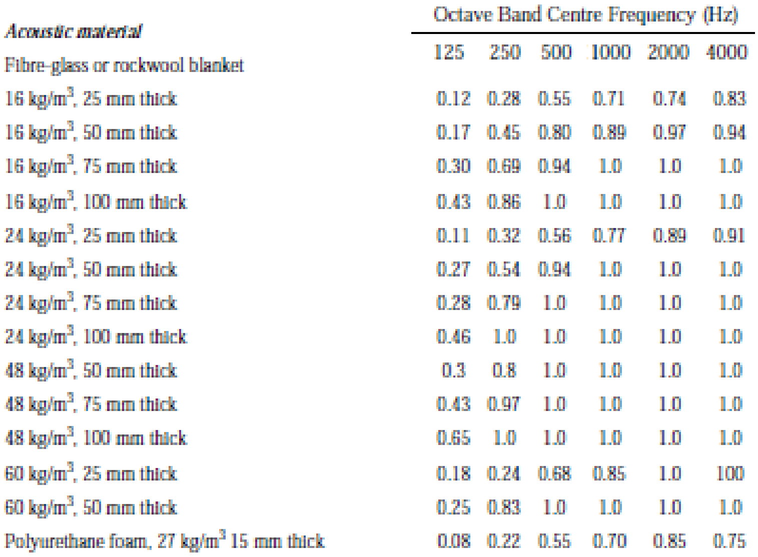
2.10 Anechoic and Reverberant Chambers

In order to reproduce a free field in which the radiation of sound is studied, an anechoic room should be used. Within this type of chamber, all the acoustic energy that strikes the walls is absorbed (Barron, 2003). This means that the energy from the source is directly transmitted to a receiver, without wall reflections (direct field). Anechoic rooms permit the extraction of precise noise data and noise directivity information, however they are limited by their dimensions and therefore the possibility to reach the far field, for some type of noise sources (Bies & Hansen, 2009).

The radiation of sound in a free field is divided into three regions. The first one corresponds to the hydrodynamic near filed, which is the region immediately adjacent to the source's vibrating surface. In this case, the fluid motion does not have a relation with sound propagation. The second region is known as the geometric near field and is located adjacent to the hydrodynamic field. This region is characterized by the interference of sound waves that come from different parts of the source, and the pressure and the particle velocities of the resulting combination of waves are not in phase. The final region is known as the far field and extends beyond the geometric field. Here, the sound pressure levels decrease by 6 [dB] as the distance from the source is doubled. In order to achieve the far field region, three conditions must be met. These are shown in equation 2.20.

$$\gamma \gg {}^{1}\!/_{k}\,, \qquad \gamma \gg 2 \qquad \& \qquad \gamma \gg k \tag{2.20}$$

Where, $\gamma = {}^{2r}\!/_{l}$, $k = {}^{\pi l}\!/_{\lambda}$, λ is the wavelength of the radiated sound, r is the distance of the source to the measurement position [m], and l is the characteristic source dimension [m] (Bies & Hansen, 2009).



Acoustic material	Octave Band Centre Frequency (Hz)					
Fibre-glass or rockwool blanket	125	250	500	1000	2000	4000
16 kg/m^3, 25 mm thick	0.12	0.28	0.55	0.71	0.74	0.83
16 kg/m^3, 50 mm thick	0.17	0.45	0.80	0.89	0.97	0.94
16 kg/m^3, 75 mm thick	0.30	0.69	0.94	1.0	1.0	1.0
16 kg/m^3, 100 mm thick	0.43	0.86	1.0	1.0	1.0	1.0
24 kg/m^3, 25 mm thick	0.11	0.32	0.56	0.77	0.89	0.91
24 kg/m^3, 50 mm thick	0.27	0.54	0.94	1.0	1.0	1.0
24 kg/m^3, 75 mm thick	0.28	0.79	1.0	1.0	1.0	1.0
24 kg/m^3, 100 mm thick	0.46	1.0	1.0	1.0	1.0	1.0
48 kg/m^3, 50 mm thick	0.3	0.8	1.0	1.0	1.0	1.0
48 kg/m^3, 75 mm thick	0.43	0.97	1.0	1.0	1.0	1.0
48 kg/m^3, 100 mm thick	0.65	1.0	1.0	1.0	1.0	1.0
60 kg/m^3, 25 mm thick	0.18	0.24	0.68	0.85	1.0	100
60 kg/m^3, 50 mm thick	0.25	0.83	1.0	1.0	1.0	1.0
Polyurethane foam, 27 kg/m^3 15 mm thick	0.08	0.22	0.55	0.70	0.85	0.75

Table 3. Acoustic Coefficient for Acoustic Materials. Retrieved from: Biess, D. & Hansen (2009), Engineering Noise Control, Fourth Edition, CRC Press -USA

All the walls of an anechoic room must be covered with sound absorbing material such as sheets of glass wool or plastic foam. These materials have high sound absorption coefficients, as it is observed in Table 3. Additionally, they are usually cut into wedges and assembled as observed in Figure 1 walls (Kleiner & Tichy, 2014). The purpose of this is to achieve full absorption of the sound waves, because if any of them enters into the cavity between the wedges it will be reflected back and forth until it is fully absorbed into the walls (Randall, 2005).

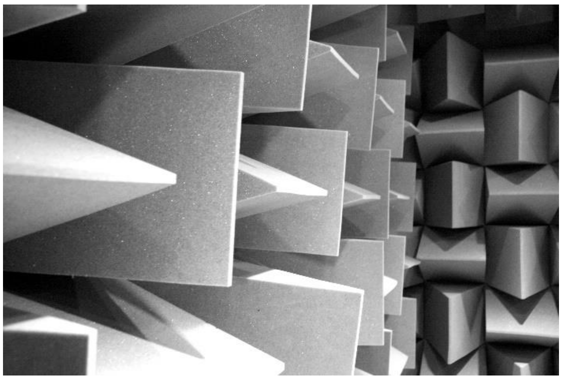


Figure 2. Anechoic Chamber Walls Assembly. Retrieved from http://www.ilcc.inf.ed.ac.uk/facilities/facilities-images/IMG-2965-1.JPG/image_view_fullscreen.

On the other hand, a reverberant chamber is a room in which reflected sound waves from the room's surfaces superimpose the source original waves. The sound field generated by the reflections is called the reverberant field. All the surfaces in this type of room have a low absorption coefficient, and the energy field within it is uniform (Barron, 2003).

2.11 Microphone Phased Arrays

An array of microphones located at different positions (Microphone Phased Array), provides a solution for acoustical applications such as eliminating reverberation noise from enclosed spaces, localizing noise sources, and sound field reconstruction. Microphone arrays can be constructed in different geometries. A plane geometry array such as the one shown in Figure 3 is one of the most popular, and it is principally used for reconstruction of complex radiating sources.

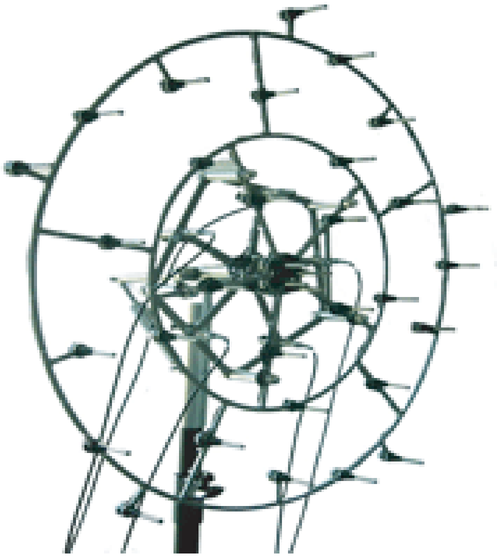


Figure 3. Plane Microphone Phased Array. Retrieved From: http://xn--42cga5ca0b6cdc2bzb3a7ble0ewa3nna3m.com/en/portfolio/microphone-array/.

In order to determine any sound field, a procedure called beamforming is utilized. It consists on maximizing the total summed output of the microphones for sound coming from a specific direction, and minimizing it for sound coming from different directions (Bader, 2014).

The beamforming procedure starts by creating a plane grid of points on any area that is being tested. Later one point on the grid is chosen, and the distance from it to each microphone is determined. Because each distance is different, if there is a noise source at the chosen point, the source's sound waves should arrive to each microphone with a different phase. These phases can be calculated and added to the signals from each microphone, so the summed output of all the microphones is maximized. In the case were there is not a noise source at the chosen grid point, at the moment of adding the phases to the signals, the output should result in lower values.

This method is then repeated for all the points on the grid, and after analyzing the data, a noise map can be produced for the entire chosen grid (Bies & Hansen, 2009). Figure 4 shows how a noise map obtained from a wind tunnel airfoil test looks. Each frequency band used in the analysis should have its own map.

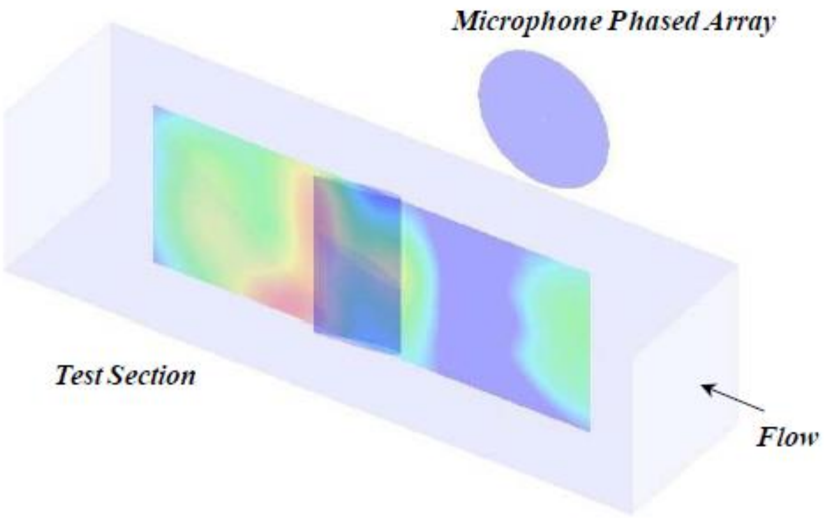


Figure 4. Acoustic Map from Wind Tunnel Test. Retrieved from "Aero-acoustic Testing of Wind Turbine Airfoils", W. Davenport et al., 2010. NREL/SR-500-43471. National Renewable Energy Laboratory.

2.12 Flow around Wind Turbine Blades

A common approach used to study the aerodynamic properties of a wind turbine blade and its geometry, is with a two-dimensional blade element (aerodynamic airfoil) taken at a radial distance *r* from the rotor axis of the wind turbine. This approach permits the analysis of upwind flow conditions and aerodynamic reacting forces (Hau, 2013).

Figure 5 shows a blade element (airfoil) from a wind turbine blade. In it, the airfoil's upstream velocity vectors are shown. They are: the flow velocity $\overline{W}$ perceived by the blade, the wind velocity $\overline{V}$ that flows towards the turbine, and the rotational flow speed $\overline{U}$ of the blade.

A wind turbine airfoil is designed with a geometric shape that diverts the incoming flow towards the rotor plane. Therefore a reaction force $\overline{F}$ and its two components, the Lift $\overline{L}$

and the Drag $\overline{D}$ are produced. The lift component produced on the airfoil is what makes the entire blade rotate. The surface of the airfoil were there are relatively low flow velocities and high pressure is called the pressure side, and the surface were there are high flow velocities and low pressure is called the suction side.

The angle formed between the direction of the perceived flow velocity and the chord line of the airfoil is the angle of attack α. The angle β formed between the rotor plane and the chord line, is the sum of the local fixed blade twist angle and the adjustable blade pitch angle (Bowlder & Leventhal, 2011).

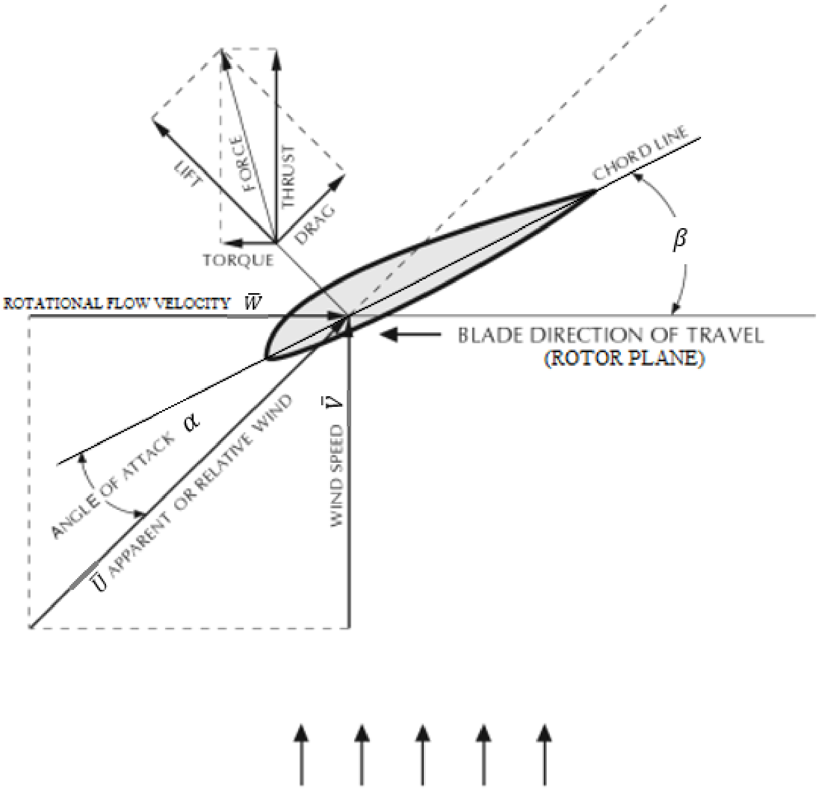

LIFT
FORCE
THRUST
DRAG
TORQUE
CHORD LINE
β
ROTATIONAL FLOW VELOCITY $\overline{W}$
BLADE DIRECTION OF TRAVEL
(ROTOR PLANE)
α
ANGLE OF ATTACK
$\overline{U}$ APPARENT OR RELATIVE WIND
WIND SPEED $\overline{V}$

Figure 5. Airfoil with Definition of Flow Angles and Forces. Retrieved from: Gipe, P. (2004), Renewable Power: Renewable Energy for Home, Farm, and Business, Chelsea Green Publishing Company, USA.

As shown in Figure 6, the rotational velocity $\overline{U}$ must increase from the base to the tip of the blade, and the wind velocity $\overline{V}$ does not vary significantly. This causes an increase of the flow perceived velocity $\overline{W}$, and a change on its direction, from the base to the tip of the blade. These changes in the flow velocity $\overline{W}$ also cause variations on the local angles of attack of each blade element, by making them decrease from the base to the tip of the blade. This is only applies for the case for a flat blade.

The lift to drag ratio is dependent of the angle of attack, and wind turbine designers use high ratios to obtain higher performance. Therefore, a constant angle of attack that optimizes the design is always chosen along the blade for each element contained in it. This means that the blade design should have its elements gradually twisted from base to tip, in order to guarantee a constant angle of attack. Additionally, turbines usually are equipped with blades that can rotate a determined angle (pitch angle), so that the optimum angle of attack is maintained for varying wind conditions (Gipe, 2004).

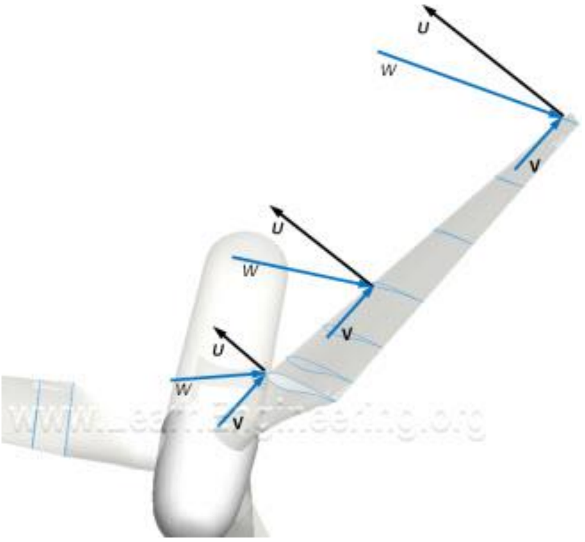


Figure 6. Change of Flow Velocity Components along an Un-Twisted Wind Turbine Blade. Retrieved From: http://www.learnengineering.org/2013/08/Wind-Turbine-Design.html.

2.13 Lift and Drag

The interactions of an object with the fluid in which it is immersed are determined by the reacting forces on the body. These forces are caused by the wall shear stress τ_w that is generated because of viscous effects, and the normal stress that is determined by the pressure P, as observed in Figure 7.

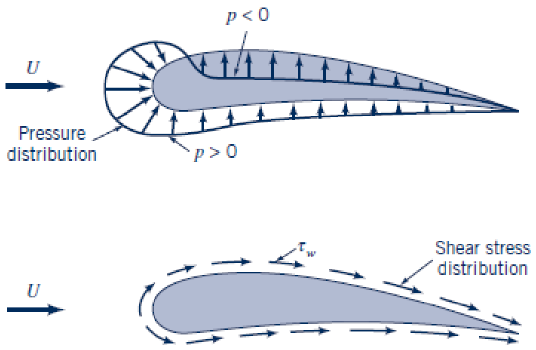


Figure 7. Pressure & Shear Stress Distributions. Retrieved from: Munson, B, Okiishi, T., Huebsch, W., Rothmayer, A. (2013). Fundamentals of Fluid Mechanics. P.482. John Wiley & Sons. Hoboken, NJ.

In the case of an airfoil, the integrated effects of both the shear stress distribution and the pressure distribution, for the total body area, determine the lift and drag forces. They can be obtained as shown in equation 2.21a & 2.21b (Munson et al, 2013).

$$Drag = \int dF_x = \int p\, cos\theta\, dA + \int \tau_w \sin\theta\ \ dA \text{ [N]} \qquad (2.21a)$$

$$Lift = \int dF_y = \int p \sin\theta\, dA + \int \tau_w \cos\theta\ \ dA \text{ [N]} \qquad (2.21b)$$

2.14 Flows Past an Object & Boundary Layer Structures

The most important dimensionless parameter for typical external flows is the Reynolds number, which is given by $Re = \frac{\rho U l}{\mu}$, where l is the characteristic length of the object [m], U is the flow velocity[m/s], ρ is the density $\left[{}^{Kg}/_{m^3}\right]$, and μ is the viscosity of the fluid [Pa s]. For a flow past an object, as the Reynolds number increases, the flow is dominated by inertial forces and the viscous effects become negligible, except in a thin region close to the object's surface called the boundary layer.

The behavior of a fluid particle along the flow past an object, retains its original shape as it flows in the uniform flow outside of the boundary layer. This is observed in Figure 8.

When a particle enters the boundary layer, it gets distorted because of the velocity gradient present within the layer.

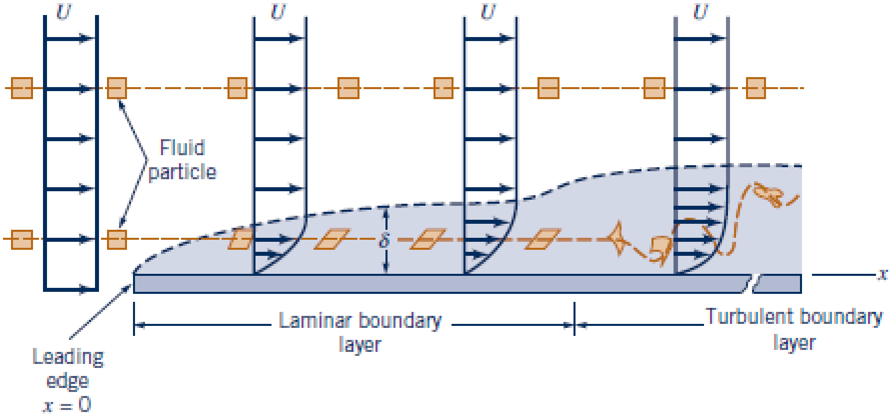


Figure 8. Laminar & Turbulent Boundary Layer. Retrieved from: Munson, B, Okiishi, T., Huebsch, W., Rothmayer, A. (2013). Fundamentals of Fluid Mechanics. P.490. John Wiley & Sons. Hoboken, NJ.

At some point of the boundary layer, there is a transition from laminar to a turbulent flow, where the particles become greatly distorted. The values of the Reynolds number at the transition zone is a function of various parameters such as roughness of the surface, curvature and disturbances outside the boundary layer, and for a flat plate the typical values go from 2×10^5 and 3×10^6. The structure of a turbulent boundary layer is characterized by the random unsteady velocities of the flow at any given location, and flatter velocity profile with larger velocity gradients at the wall (Munson et al, 2013).

2.15 Boundary Layer Thickness

At the boundary layer, the velocity profile changes from the flow velocity U, which is the velocity outside the boundary layer, to zero at the surface of the object on which the flow passes through. The boundary layer thickness is defined as the distance between the plate and an upstream velocity value. Typically, the velocity value is 99% of the velocity U, as shown in Figure 9a.

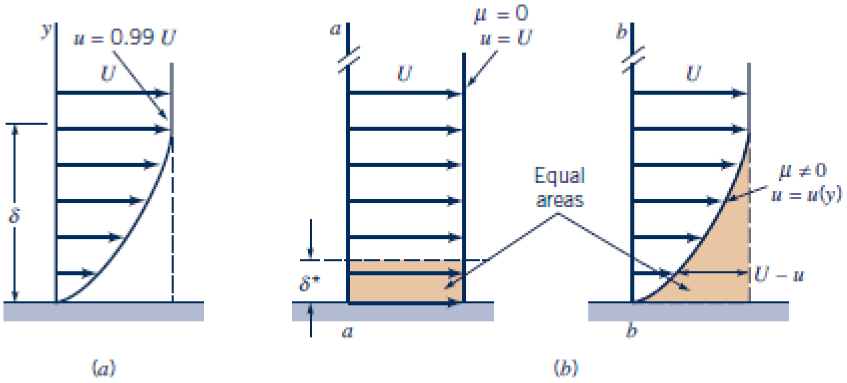


Figure 9.Boundary Layer Thickness: a) Standard Boundary Layer Thickness, b) Boundary Layer Displacement Thickness. Retrieved from: Munson, B, Okiishi, T., Huebsch, W., Rothmayer, A. (2013). Fundamentals of Fluid Mechanics. P.491. John Wiley & Sons. Hoboken, NJ.

In order to remove the arbitrary percentage used for the definition of boundary layer thickness, the boundary layer displacement thickness is determined, as shown in Figure 9b. According to Munson et al (2013), "The displacement thickness represents the amount that the thickness of the body must be increased so that the fictitious uniform inviscid flow has the same mass flowrate properties as the actual viscous flow." The boundary displacement thickness is determined by equation 2.22 (Munson et al, 2013).

$$\delta^* = \int_0^\infty \left(1 - \frac{u}{U}\right) dy \qquad (2.22)$$

CHAPTER III

WIND TUNNEL AIRFOIL NOISE DATA

In order to determine the sound power of the noise produced by a wind turbine airfoil, experimental wind tunnel noise data such as the sound pressure level spectrum, is to be obtained. In this case, the data was gathered from experimental testing performed on a DU-96 airfoil at the Virginia Polytechnic Institute and State University Stability Wind Tunnel. This type of airfoil was chosen because it was developed, designed and extensively tested at Delft University, The Netherlands, specifically for wind turbine applications (Timmer & van Rooij, 2001). All the measurements were conducted by Virginia Tech personnel, and the sound pressure levels for 1/3rd octave frequency bands, for different flow velocities, angles of attack and trip conditions, were provided.

The objective of this chapter is to explain how the experimental wind tunnel test on the airfoil was performed. Additionally, it demonstrates how the obtained noise data must be processed, so it is useful on wind turbine noise prediction computations, which will be performed in Chapter 5. Therefore, this chapter consists of four sections, the first one describes the facilities in which the test was conducted, the second one explains how it was done, the third one explains how and why the resulting data has been processed, and finally, the last section explains the programmed MATLAB code used for the data processing.

3.1 Aero-acoustic Test Facilities

A scheme of the Virginia Tech Stability Wind Tunnel that was used to obtain the experimental noise data is observed in Figure 10. The facility is equipped with a 0.45-MW variable speed motor that achieves rotation velocities close to 600 rpm. The motor is connected to a 4.3 meter propeller that allows maximum flow speeds of 75 m/s and a Reynolds number of 5×10^6. In addition, the closed loop shaped tunnel is equipped with an arrangement of seven screens that serve for turbulence reduction purposes. They are located in a downstream direction from an air exchanger tower, which has access to the atmosphere, and serves to stabilize the internal temperature of the tunnel.

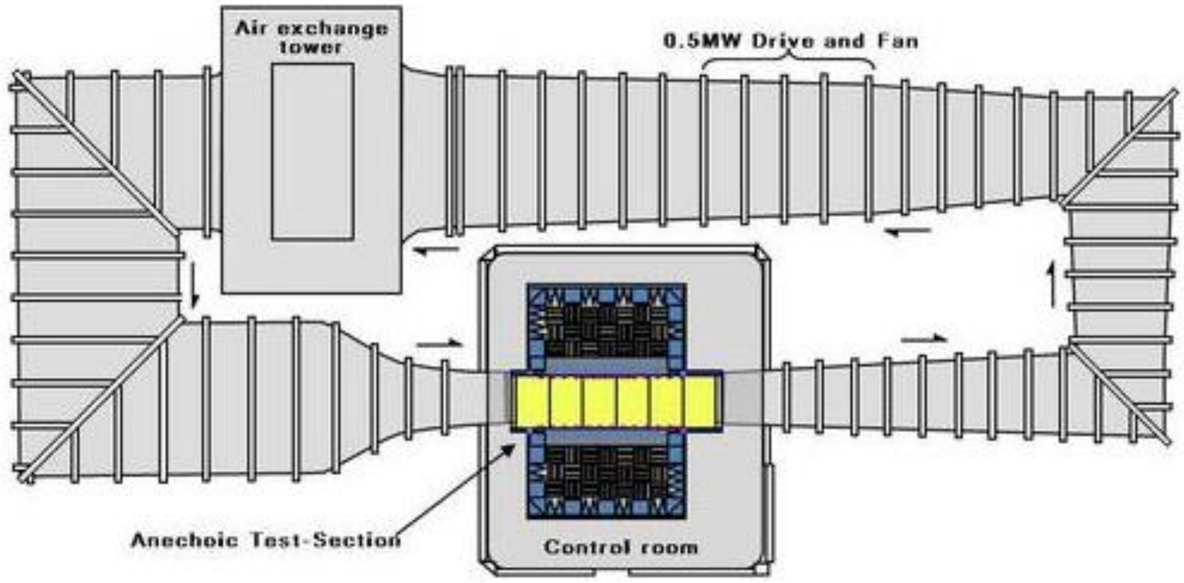


Figure 10. General layout of the Virginia Tech Stability Wind Tunnel in Anechoic Configuration. Retrieved from http://www.aoe.vt.edu/research/facilities/stabilitytunnel/acoustics-stabilitytunnel.html.

The test section of the tunnel has 7.3 meters in length and a cross sectional area shaped as a square with an edge of 1.83 meters. This section is removable and can be configured according to aero-acoustic test requirements. It is also hermetically sealed, because the pressure inside is lower than the atmospheric pressure, and equal to the static pressure of the flow that passes through it. Upstream of the test section, there is a contraction nozzle that reduces turbulence and accelerates the speed of the flow. On the other hand, in a downstream direction, there is a diffuser with vortex generators at all the walls, which serve to delay local flow separation and prevent the surge of instabilities within the tunnel (Davenport et al., 2010).

Figure 11 shows a side cross sectional view from the test section of the wind tunnel. As observed, it is equipped with acoustically treated walls at the ceiling and floor, and at the sides there are two anechoic chambers. An explanation on how an anechoic chamber works is provided in the theoretical framework section of this thesis. In Figure 12, a top cross sectional view of the test section is observed. It shows that between the anechoic chambers and the test section, there are Kevlar cloth windows that serve to contain the flow, but let the sound pass through with minimum losses (Davenport et al., 2010).

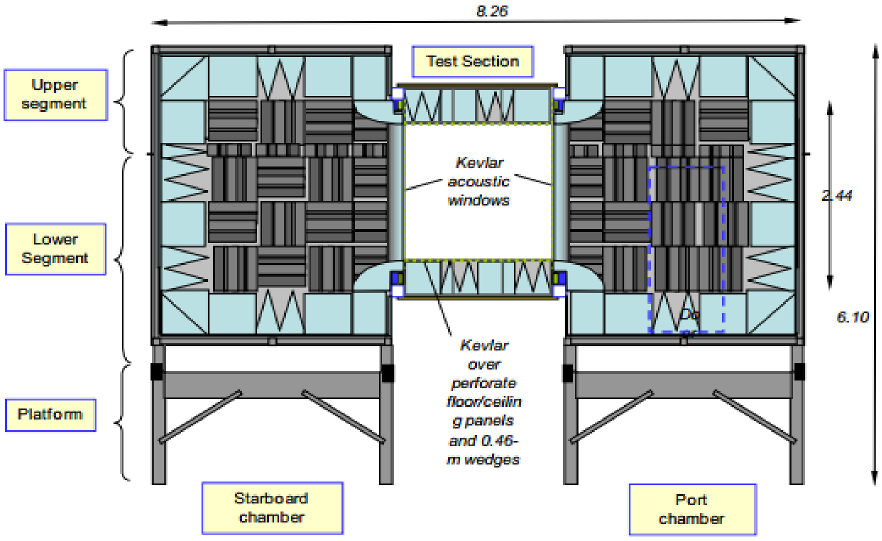


Figure 11. Side Cross Section Area of Anechoic Chambers. Retrieved from "Aero-acoustic Testing of Wind Turbine Airfoils", W. Davenport et al., 2010. NREL/SR-500-43471. National Renewable Energy Laboratory.

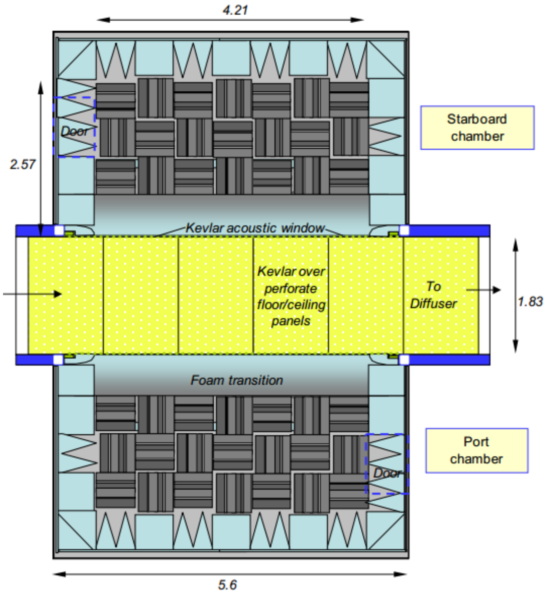


Figure 12. Top Cross Section View of Anechoic System. Retrieved from "Aero-acoustic Testing of Wind Turbine Airfoils", W. Davenport et al., 2010. NREL/SR-500-43471. National Renewable Energy Laboratory.

3.2 Experimental Test Procedure

In order to obtain the airfoil’s sound pressure levels at its trailing edge (where most of the noise is observed), the DU-96 airfoil was placed within the test area section of the stability wind tunnel, which was configured in an aero-acoustic mode. The used profile consisted of a chord distance of 0.9 meters and a span of 6 feet or 1.83 meters, as represented in Figure 13.

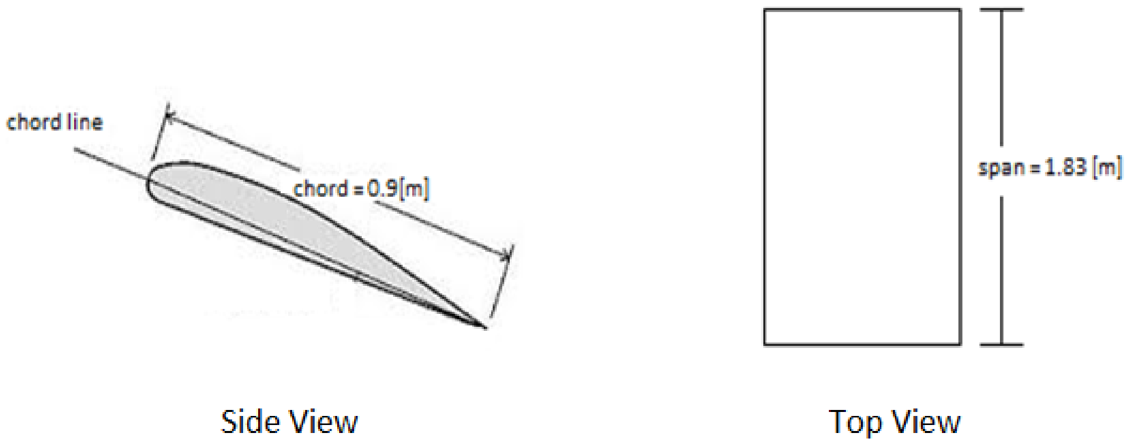


Figure 13. Chord and Span of the DU-96 Airfoil.

Figure 14 shows the test configuration that was used. It consisted of a microphone phased array that had its centroid located at a distance of 1.6 meters from the leading edge. It was placed on the pressure side of the airfoil, in such a manner that the array’s plane was parallel to the chord line when the angle of attack had a value of zero (i.e. when chord line is parallel to the direction of the incoming flow). It has been assumed that as the angle of attack is altered and the airfoil rotates, the change in the distance of 1.6 meters is negligible.

The microphone array was therefore outside of the test section of the wind tunnel and inside of one of the anechoic chambers, which means that one of the Kevlar cloth windows was located between the airfoil and the microphones. The definition of airfoil flow angles such as the angle of attack, as well as an explanation on how a microphone phased array is used to obtain sound measurements, is provided in the theoretical framework section of this thesis.

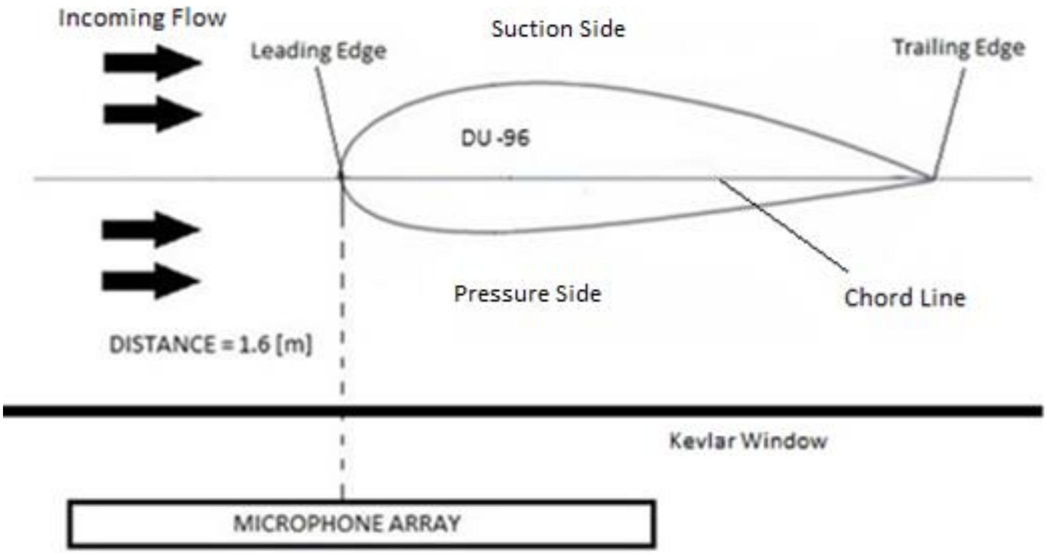


Figure 14. Wind Tunnel Airfoil Test Configuration (Angle of Attack of Zero Degrees).

The experimental test was conducted with four different uniform flow velocities of 34 [m/s], 44 [m/s], 54 [m/s] and 64 [m/s], each with four different angles of attack of 4^{o},8^{o}, $12^{o}$and 16^{o}. These flow velocities where chosen because they fall within typical relative flow speeds experienced by functional wind turbine blades. On the other hand, the angles of attack where chosen based on distinctive ranges that prevent the airfoil to stall.

The actual experimental velocities reached and their respective Mach number, for each angle of attack, are shown in Table 4. It is observed that the Mach number for all the cases is within a range of 0.09 to 0.19. This falls within a low Mach number range, which is in accordance to real wind turbine blade functioning conditions. Wind turbines are usually designed in this way, so that non-linearities like shock waves are prevented (Wagner, Bareib & Guidati, 1996).

Additionally, the airfoil was tested with and without a trip, for each flow velocity and angle of attack. A trip is a small surface disturbance that is used to create a physically realistic model, because it accounts for surface imperfections or debris stuck to a wind turbine blade, while it operates (Moriarty & Migliore, 2003). According to Wagner, Bareib and Guidati (1996), it also prevents laminar vortex shedding noise because it induces a transition from laminar to turbulent flow. Vortex shedding is a phenomena that happens when laminar flow regions extend to the trailing edge, and instabilities such as separation bubbles that propagate along the chord, trigger a raise on the produced noise levels.

	Flow Target Velocity of 34[m/s]		**Flow Target Velocity of 44[m/s]**		**Flow Target Velocity of 54[m/s]**		**Flow Target Velocity of 64[m/s]**	
Angle of Attack	Actual Flow Velocity	Actual Mach Number	Actual Flow Velocity	Actual Mach Number	Actual Flow Velocity	Actual Mach Number	Actual Flow Velocity	Actual Mach Number
4^o	34.124	0.0992	44.036	0.1275	54.024	0.1562	64.029	0.1842
8^o	34.001	0.0988	44.251	0.1281	54.039	0.1562	64.283	0.1849
12^o	33.998	0.0988	43.993	0.1274	53.937	0.1554	63.850	0.1835
16^o	34.004	0.0988	44.156	0.1278	54.623	0.1578	63.940	0.1837

Table 4. Target and Actual Flow Velocities for Wind Tunnel Test.

3.3 Data Processing & Results

The data provided by Virginia Tech was composed of the sound pressure levels as a function of frequency in 1/3rd octave bands, for each angle of attack, flow test velocity, and tripped condition (the definition of 1/3rd octave frequency bands is contained in the theoretical framework of this thesis). It was all presented in an excel *.xlsx* file. Additionally, it contained the uncorrected data and the corresponding correction due to the losses associated with the Kevlar cloth window, from the wind tunnel test section

This section of the thesis is divided into 2 sub-sections. First there is brief explanation on how to correct the sound pressure level losses in dB associated with the Kevlar windows. The second sub-section explains how all the data for every different case (different flow velocities and angles of attack) is normalized to a specific value of flow velocity, airfoil span and distance between the airfoil and the microphone array. The objective of this procedure is to obtain a straightforward normalized equation for the sound pressure levels [dB], produced by an airfoil. It should result to be only as a function of the dimensionless Strouhal number and the angle of attack of the airfoil. This is because in this way the equation can be easily plugged into wind turbine noise prediction computations, as it will be seen in Chapter 5.

3.3.1 Kevlar Window Associated Sound Pressure Level Corrections.

Given that between the airfoil and the microphone phased array there is a Kevlar window, the measured sound pressure levels do not represent the real airfoil noise. This is because when the sound waves pass through the Kevlar, there are sound pressure level losses. Furthermore, there are losses caused by the shear boundary layer at the window’s surfaces. This means that the true sound pressure levels should be expressed as shown in equation 3.1, where SPL_{true} are the corrected sound pressure level values, $SPL_{measured}$ are the sound pressure level measured values, ΔK is the Kevlar window correction, and ΔB is the shear boundary layer correction (Davenport et al., 2010).

$$Lp_{true} = Lp_{measured} + \Delta K + \Delta B \tag{3.1}$$

Within the data provided by Virginia Tech, these corrections were already applied. In this section, a step by step explanation is provided on how a correction equation was obtained and then implemented.

For frequencies less than 5000 [Hz], the value of ΔK is insignificant and therefore equal to zero. Since the higher frequency band used in this analysis was up to 7100 [Hz], the value of ΔK will have a maximum value of 1 [dB]. This analysis was conducted on the Virginia Tech wind tunnel and was provided by Davenport et al., as shown in Figure 15.

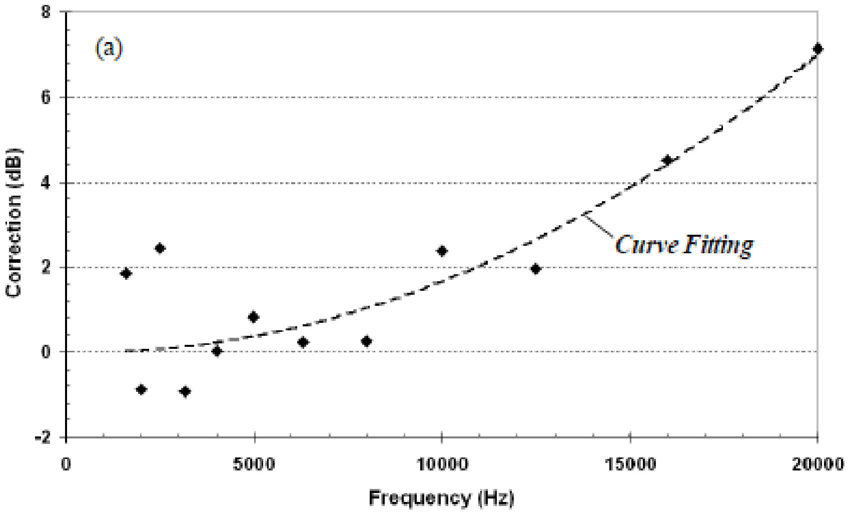


Figure 15. Attenuation of sound passing through acoustic Kevlar windows as a function of frequency. Retrieved from "Aero-acoustic Testing of Wind Turbine Airfoils", W. Davenport et al., 2010. NREL/SR-500-43471. National Renewable Energy Laboratory.

On the other hand, in order to obtain the value of ΔB. Davenport et al. determined that there was a difference of 2 [dB] when the free stream velocity within the tunnel changed from 58 [m/s] to 41 [m/s], independently from the frequency. This is shown in Figure 16, where the speed of sound used was 341.67 [m/s].

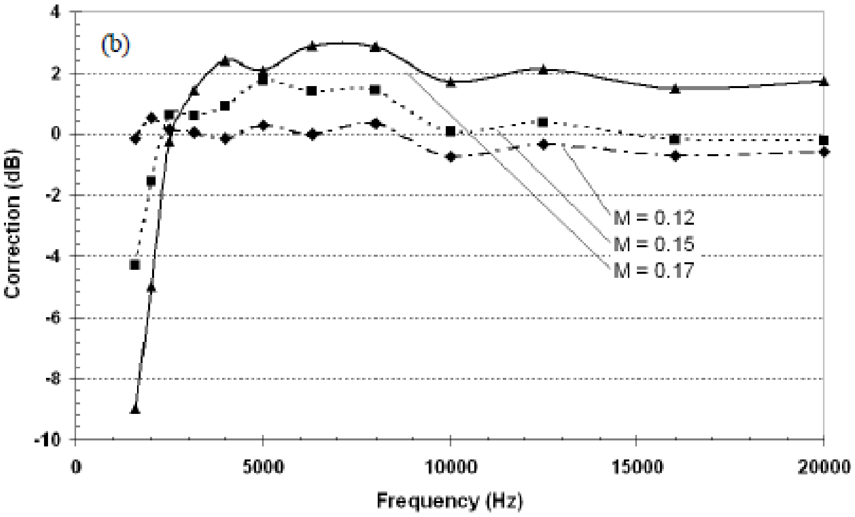


Figure 16. Attenuation of sound passing through the shear boundary layer as a function of frequency. Retrieved from "Aero-acoustic Testing of Wind Turbine Airfoils", W. Davenport et al., 2010. NREL/SR-500-43471. National Renewable Energy Laboratory.

In order to obtain the value of ΔB, first the sound pressure levels for the two velocities were defined as shown in equation 3.2 and equation 3.3.

$$Lp_{58\,m/s} = 10\ log_{10}\left(\frac{P^2_{rms-58\,m/s}}{P^2_{ref}}\right) \tag{3.2}$$

$$Lp_{41\,m/s} = 10\ log_{10}\left(\frac{P^2_{rms-41\,m/s}}{P^2_{ref}}\right) \tag{3.3}$$

Where P_{ref} is the reference pressure, and P_{rms} refer to the pressure root mean square value. Further explanation on how these acoustic equations are obtained and what P_{ref} and P_{rms} mean, is found in the theoretical framework section of this thesis.

Next, the expression for the sound pressure level difference between $SPL_{58\,m/s}$ and $SPL_{41\,m/s}$ was obtained. This was done by subtracting equation 3.3 from equation 3.2, and determining the expression that is shown in equation 3.4, after various simplifications.

$$Lp_{58\,m/s} - Lp_{41\,m/s} = 10\ log_{10}\left(\frac{P^2_{rms-58\,m/s}}{P^2_{ref}}\right) - 10\ log_{10}\left(\frac{P^2_{rms-41\,m/s}}{P^2_{ref}}\right)$$

$$Lp_{58\,m/s} - Lp_{41\,m/s} = 10\ \times \left[log_{10}\left(\frac{P^2_{rms-58\,m/s}}{P^2_{ref}}\right) - log_{10}\left(\frac{P^2_{rms-41\,m/s}}{P^2_{ref}}\right)\right]$$

$$Lp_{58\,m/s} - Lp_{41\,m/s} = 10 \times \left[log_{10} \left(\frac{P^2_{rms-58\,m/s}}{P^2_{ref}} \Bigg/ \frac{P^2_{rms-41\,m/s}}{P^2_{ref}} \right) \right]$$

$$Lp_{58\,m/s} - Lp_{41\,m/s} = 10 \times \left[log_{10} \left(\frac{P^2_{rms-58\,m/s}}{P^2_{rms-41\,m/s}} \right) \right] \tag{3.4}$$

After this, a power law was defined as a proportional relation between P^2_{rms} and any free stream flow velocity U_∞. This is shown in equation 3.5, where C is a constant and the power S is unknown.

$$P^2_{rms} \propto U^S_\infty$$

$$P^2_{rms} = C \times U^S_\infty \tag{3.5}$$

Equation 3.5 was then replaced into equation 3.4, for each of the two free stream flow velocities and simplified into equation 3.6. In this way the value of S was obtained, since it was the only unknown.

$$Lp_{58\,m/s} - Lp_{41\,m/s} = 10 \times \left[log_{10} \left(\frac{C \times 58^S}{C \times 41^S} \right) \right]$$

$$Lp_{58\,m/s} - Lp_{41\frac{m}{s}} = 2[dB] = S \times 10 \times \left[log_{10} \left(\frac{58}{41} \right) \right] \tag{3.6}$$

$$S = 1.3$$

This means that the value of ΔB was given by the expression on equation 3.7. In this case U_{test} is the free stream flow velocity used for each test performed on the wind tunnel.

$$\Delta B = S \times 10 \times \left[log_{10} \left(\frac{U_{test}}{41} \right) \right] = 1.3 \times 10 \times \left[log_{10} \left(\frac{U_{test}}{41} \right) \right] \tag{3.7}$$

All the data provided by Virginia Tech has been corrected using these methods, and therefore more precise sound pressure levels were able to be used in further normalization procedures.

3.3.2 Normalization procedure.

The data provided by Virginia Tech contains the SPL [dB] for 1/3rd octave frequency bands, for each of the four angles of attack tested and each flow velocity. Even though, the data set contains tripped and un-tripped experiment sets, only the tripped ones were chosen for further analysis. This is because of their resemblance to real turbine working conditions and favorable noise reduction circumstances. In addition, only the Kevlar corrected values were used.

This section of the thesis first explains the procedure to perform four normalizations on the sound pressure levels for each frequency band of all the chosen data (Strouhal number scaling, airfoil span normalization, airfoil-microphones distance normalization & free stream velocity normalization), and the reason why this is done. Additionally, the results obtained from applying this process are presented in various figures. Next, this section illustrates how regression methods applied to the normalized data, are used to determine an equation for the sound pressure levels [dB] as a function of the angle of attack and the Strouhal number. This equation is very important because it provides a simplistic empirical airfoil sound pressure level computation method. Furthermore, its versatility will be shown later in Chapter 5, were it will be plugged into wind noise turbine prediction calculations.

The first step to process the selected data, was to scale the frequency domain of all the data with the use of the Strouhal number, shown in equation 3.8. This dimensionless parameter multiplies the airfoil’s chord [m] and the frequency [Hz], and then divides it by the free stream velocity $[m/s]$ used in the experiment.

$$St = \frac{f[Hz]\times Chord[m]}{U_{\infty}[m/s]} \tag{3.8}$$

The objective of scaling the frequency is to normalize the frequency axis, according to the free stream velocity of each of the performed tests and the chord of the airfoil.

The next step was to proceed with the airfoil span normalization, the airfoil-microphones distance normalization (i.e. distance between the chord line of the airfoil and the plane of the microphone array), and the free stream flow velocity normalization. This was done for the sound pressure level data corresponding to all frequency bands, and each of the four angles of attack used for the experimental tests done by Virginia Tech. In order to do this, a power law was defined as $P_{rms}^2 \propto Span$, $P_{rms}^2 \propto Distance_{airfoil-microphones}^2$ and $P_{rms}^2 \propto U_\infty^5$. Equation 3.9 shows the proportionality between P_{rms}^2 and the airfoil span, equation 3.10 shows it between P_{rms}^2 and the airfoil-microphones distance, and equation 3.11 shows it between $P_{rms}^2$and the free stream flow velocity. In each case K is a random constant.

$$P_{rms}^2 = K \times Span \tag{3.9}$$

$$P_{rms}^2 = K \times Distance_{airfoil-microphones}^2 \tag{3.10}$$

$$P_{rms}^2 = K \times U_\infty^5 \tag{3.11}$$

In order to obtain the span normalized sound pressure levels, they must be defined as it is in equation 3.12, and for the measured sound pressure levels in equation 3.13.

$$L_{p-span\,normalized} = 10\,log_{10}\left(\frac{P_{rms-span\,normalized}^2}{P_{ref}^2}\right) \tag{3.12}$$

$$L_{p-measured} = 10\,log_{10}\left(\frac{P_{rms-measured}^2}{P_{ref}^2}\right) \tag{3.13}$$

Because the values of both $P_{rms-span\,normalized}^2$ and $P_{rms-measured}^2$ are not known, the difference between $L_{p-span\,normalized}$ and $L_{p-measured}$ is implemented using the expressions on equations 3.13 and 3.12. The result is shown in equation 3.14, which is then simplified into equation 3.15. Finally the proportionality expression shown in equation 3.9 is applied and equation 3.16 is obtained, which is the final equation that normalizes all the data, for all angles of attack. The span normalization value used was of 1 [m].

$$Lp_{span\,normalized} - Lp_{measured} = 10 \times log_{10}\left[\left(\frac{\frac{P^2_{rms-span\,normalized}}{P^2_{ref}}}{\frac{P^2_{rms-measured}}{P^2_{ref}}}\right)\right] \tag{3.14}$$

$$Lp_{span\,normalized} - Lp_{measured} = 10 \times log_{10}\left[\frac{P^2_{rms-span\,normalized}}{P^2_{rms-measured}}\right] \tag{3.15}$$

$$Lp_{span\,normalized} = Lp_{measured} + 10 \times log_{10}\left[\frac{Span_{normalization\,value}}{Span_{\,airfoil}}\right] \tag{3.16}$$

A similar process was used to perform the normalization on the microphone-airfoil distance, as well as the free stream velocities. This was done using the proportionality expressions shown in equations 3.10 and 3.11, and the final expressions are shown in equations 3.17 and 3.18. The used airfoil-microphones distance normalization value was 1 [m] and the velocity normalization value was 1 [m/s].

$$Lp_{normalized\,airfoil-microphone\,distance} = Lp_{measured} + 20 \times \left[log_{10}\left(\frac{Distance_{normalization\,value}}{Distance_{test}}\right)\right] \tag{3.17}$$

$$Lp_{normalized\,velocity} = Lp_{measured} + 50 \times \left[log_{10}\left(\frac{U_{normalization\,value}}{U_{test}}\right)\right] \tag{3.18}$$

The purpose of properly normalizing the data is to be able to collapse the sound pressure level values from the four different flow test velocities in one curve, for each angle of attack. All this as a function of the Strouhal number. The values of the powers (from each power-law shown in equations 3.9 through 3.11) used for each normalization determined how well the data collapsed, so they were carefully selected, in order to guarantee accurate curves. All the collapsed curves for each experimental angle of attack were computed within the MATLAB code *DU96_main.m*. The results for a 4 degree angle of attack, and for an 8 degree angle of attack are shown in Figure 17. The results for a 12 degree angle of attack, and for a 16 degree angle of attack are shown in Figure 18, and finally all the curves are shown in a 3D plot (Angle of attack vs. Strouhal number vs. SPL [dB]) in Figure 19.

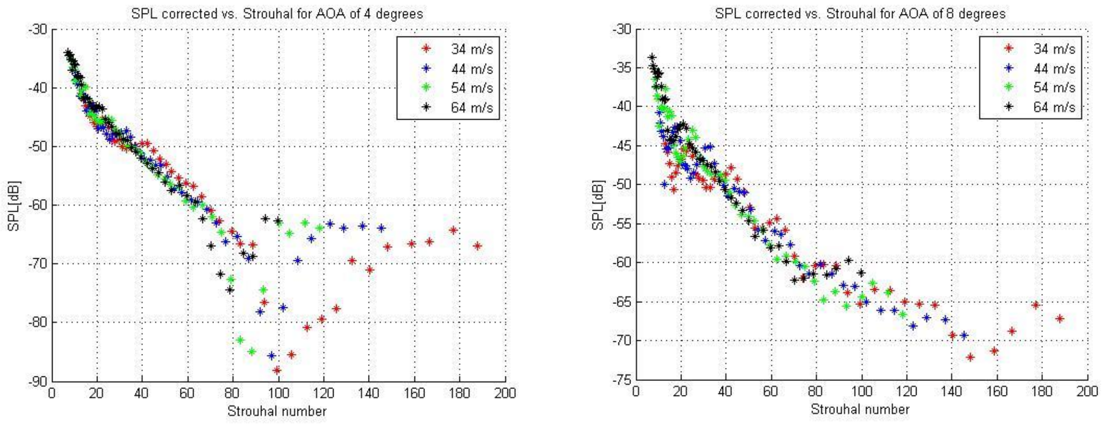


Figure 17. Corrected SPL [dB] vs. Strouhal Number for all Velocities, When the Angle of Attack is a) 4 Degrees, b) 8 Degrees.

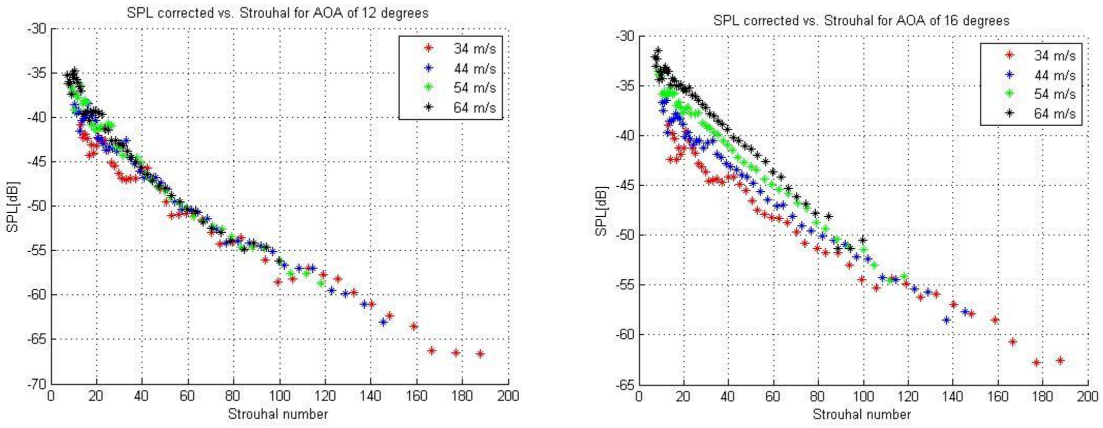


Figure 18. Corrected SPL [dB] vs. Strouhal Number for all Velocities, When the Angle of Attack is a) 12 Degrees, b) 16 Degrees.

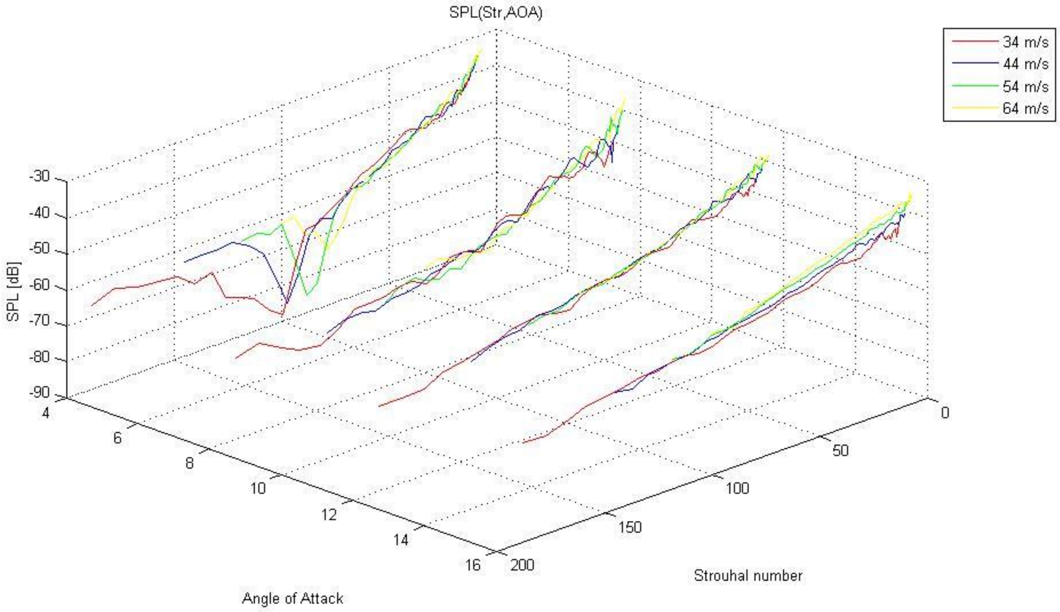


Figure 19. Collapsed Curves for Each Angle of Attack, for all Free Stream Velocities.

After the data was properly normalized and collapsed, it was proceeded to obtain an equation for the sound pressure levels produced by the airfoil as a function of the angle of attack and Strouhal number. To do so, the initial step was to fit each of the collapsed curves for each of the four experimental angles of attack into a polynomial of the form shown in equation 18, where St is the Strouhal number, SPL are the sound pressure levels and ***a***, ***b*** and ***c*** are constant coefficients.

$$Lp\ [dB] = aSt^{b} + c \tag{3.19}$$

The regressions were performed on MATLAB, as part of *DU96_main.m.*The results obtained for the fitted curve and its residuals for the angle of attack of 4 degrees is shown in Figure 20, for the angle of attack of 8 degrees in Figure 21, for the angle of attack of 12 degrees in Figure 22, and finally for the angle of attack of 16 degrees in Figure 23.

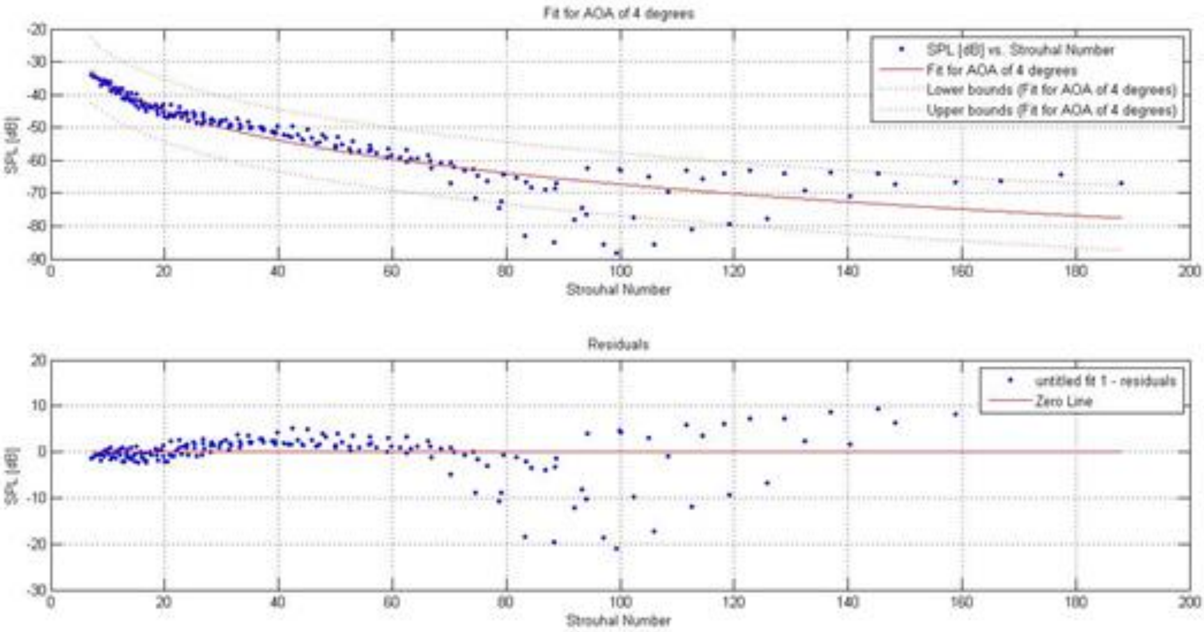


Figure 20. Collapsed Data Curve Fit for an Angle of Attack of 4 Degrees.

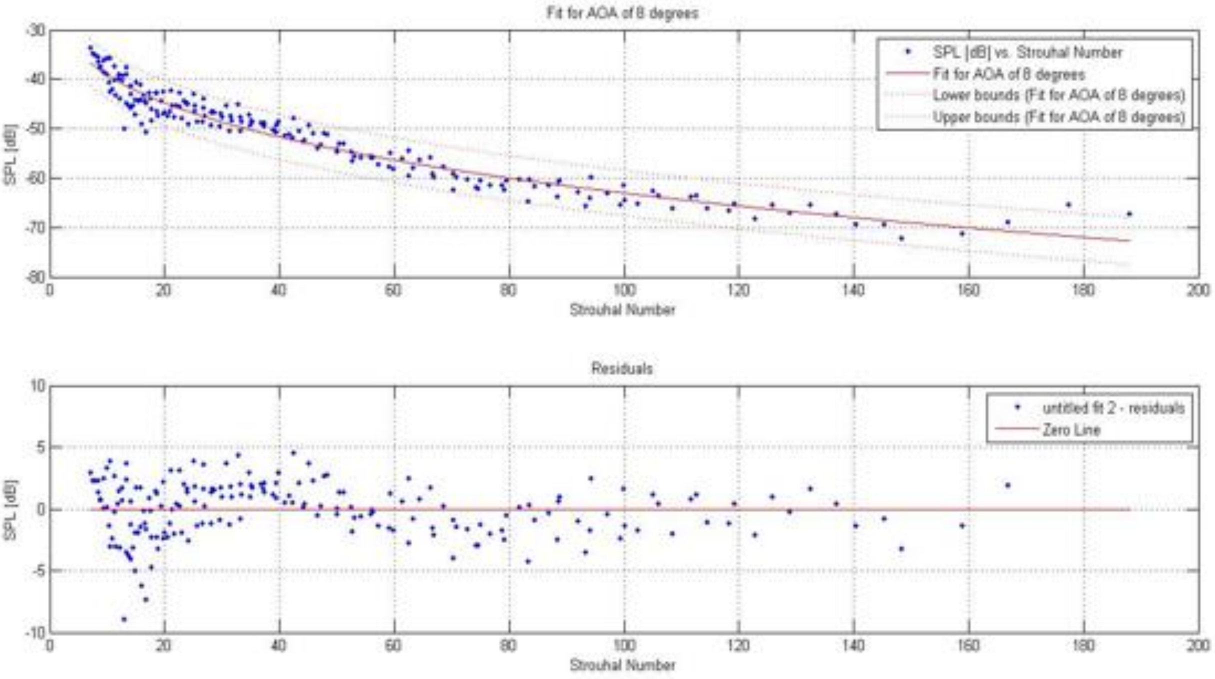


Figure 21. Collapsed Data Curve Fit for an Angle of Attack of 8 Degrees.

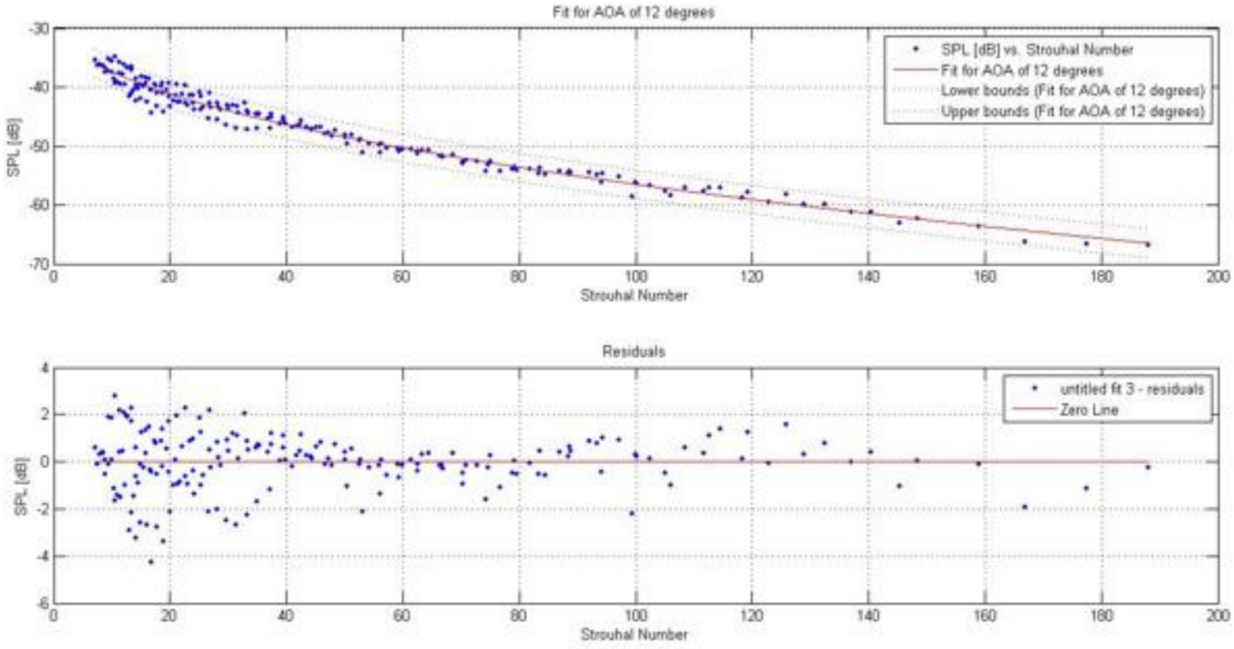


Figure 22. Collapsed Data Curve Fit for an Angle of Attack of 12 Degrees.

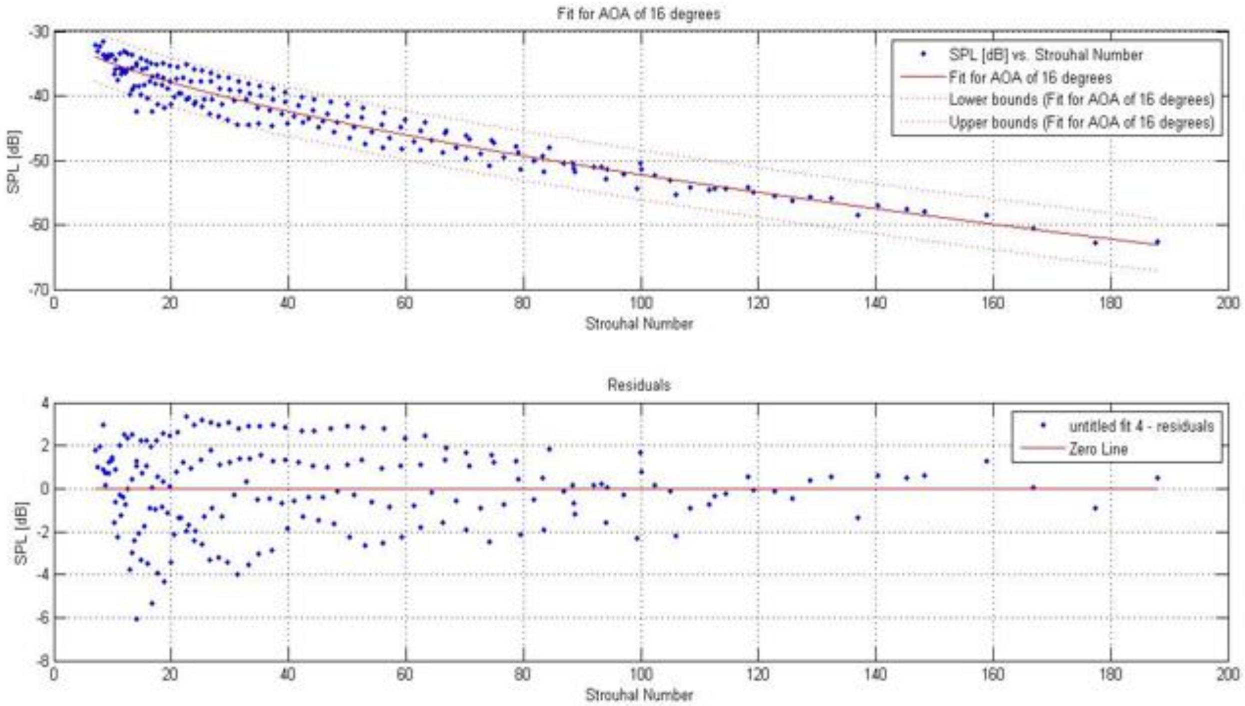


Figure 23. Collapsed Data Curve Fit for an Angle of Attack of 16 Degrees.

The type of function used for the regression is the one which most adequately fits the data because it provides the lowest residuals on all the collapsed curves. In order to create this regressions, the MATLAB curve fit toolbox was used. The obtained coefficients, ***a***, ***b*** and ***c*** from equation 3.19, for each angle of attack, are presented on Table 5.

		Angle of Attack [deg.]			
		4	8	12	16
Coefficients	a	-60.9403	-14.3460	-3.3064	-1.3070
	b	0.1364	0.2752	0.4705	0.6197
	c	46.9239	-12.0984	-27.6123	-29.5599

Table 5. Curve Fit Coefficients for Each Angle of Attack.

From the data provided in Table 5, it is seen that each coefficient can be fitted as a function of the angle of attack. This means that the final sound pressure level equation would be a function of the Strouhal number and the angle of attack, as observed on equation 3.20 where AOA is the angle of attack, St is the Strouhal number.

$$Lp = \ a(AOA)St^{b(AOA)} + c(AOA) \tag{3.20}$$

Therefore, each coefficient was fitted to a polynomial, using the MATLAB curve fit toolbox. The regression and its residuals resulted for coefficient ***a*** is shown in Figure 24, for coefficient ***b*** in Figure 25, and for coefficient ***c*** in Figure 26. The resulting sound pressure level expression is shown in equation 3.21.

$$Lp = \ \left[(-823.5139 \times AOA^{-1.8263} + 4.5162) \times St^{\left(0.0301 \times AOA^{1.0974} - 0.0056\right)}\right] + (958.8100 \times AOA^{-1.7456} - 38.2905) \tag{3.21}$$

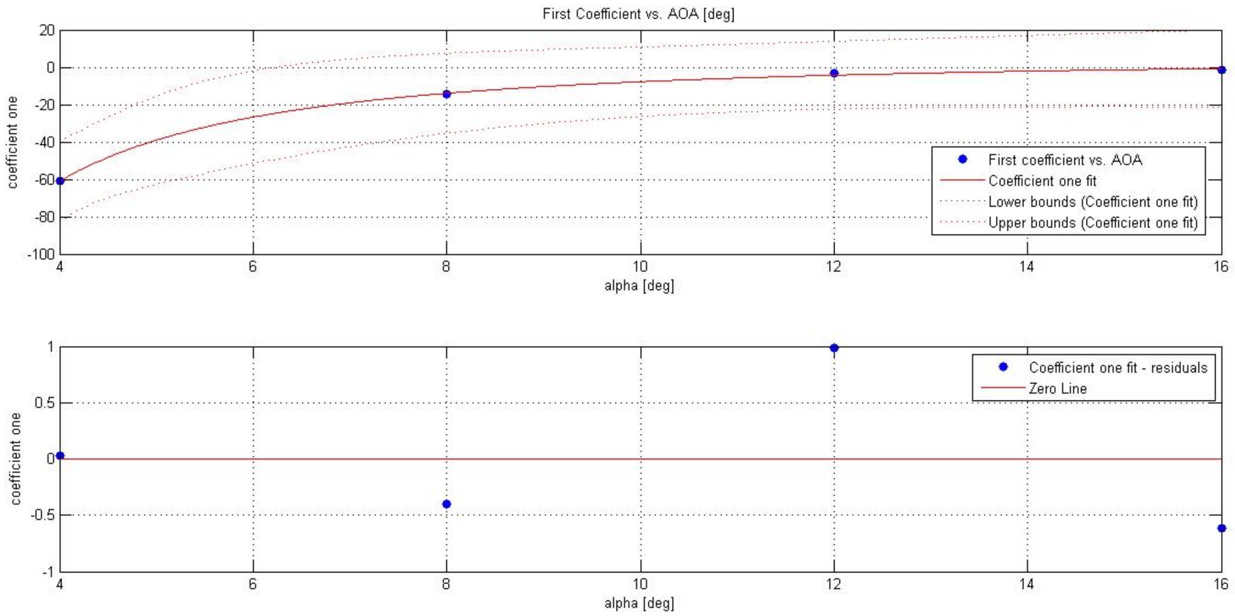


Figure 24. First Coefficient (a) Fit as a Function of the Angle of Attack.

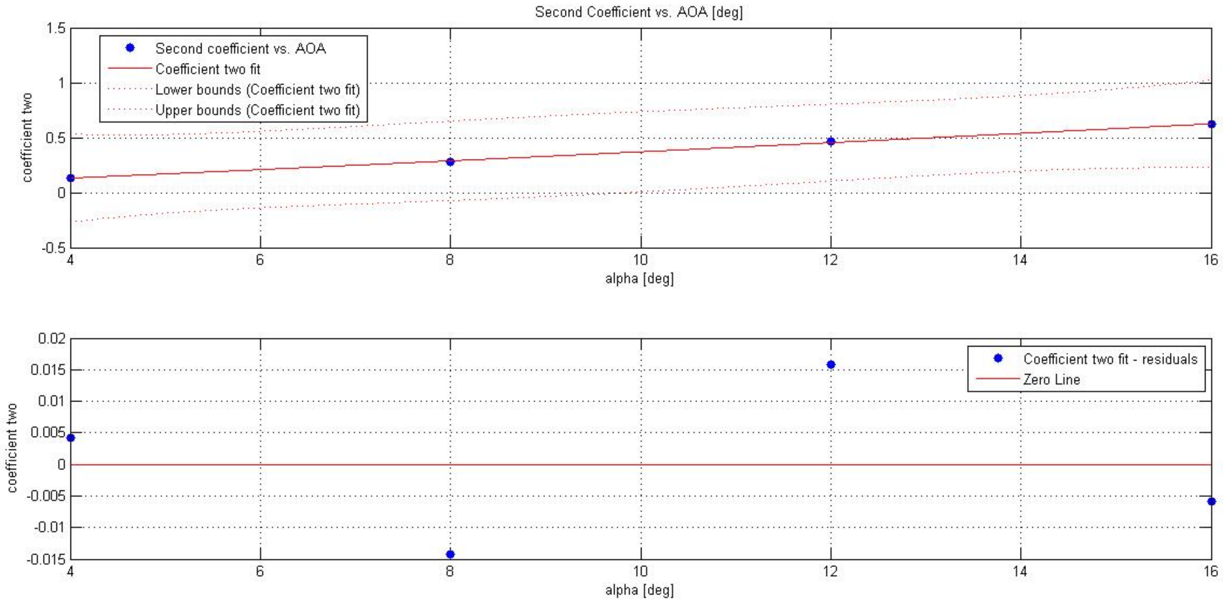


Figure 25. Second Coefficient (b) Fit as a Function of the Angle of Attack.

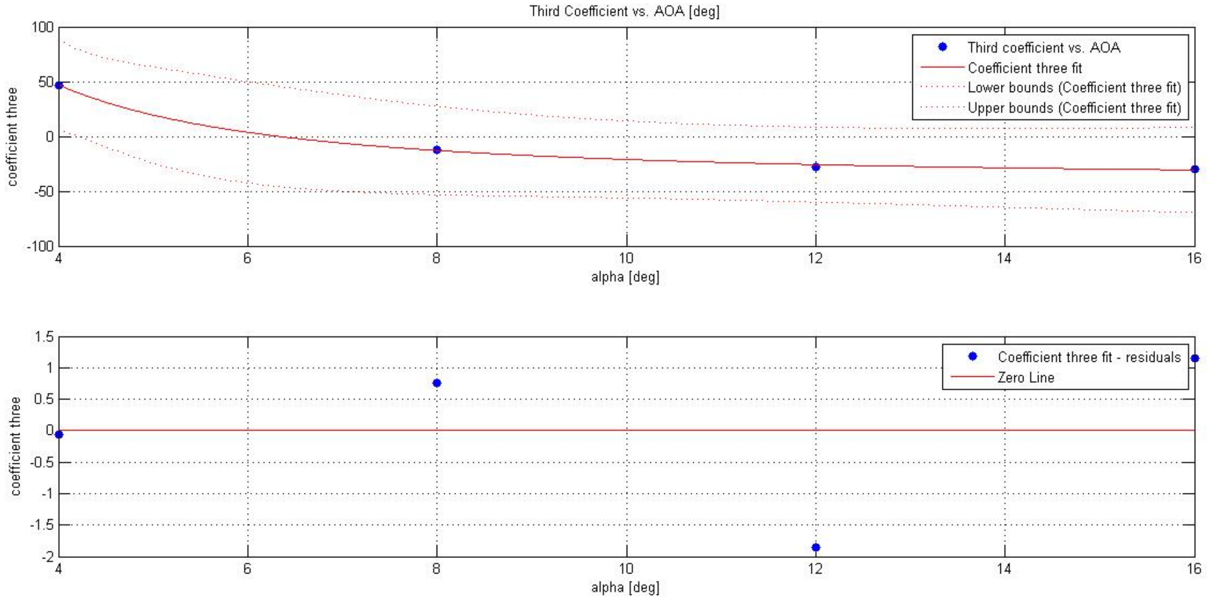
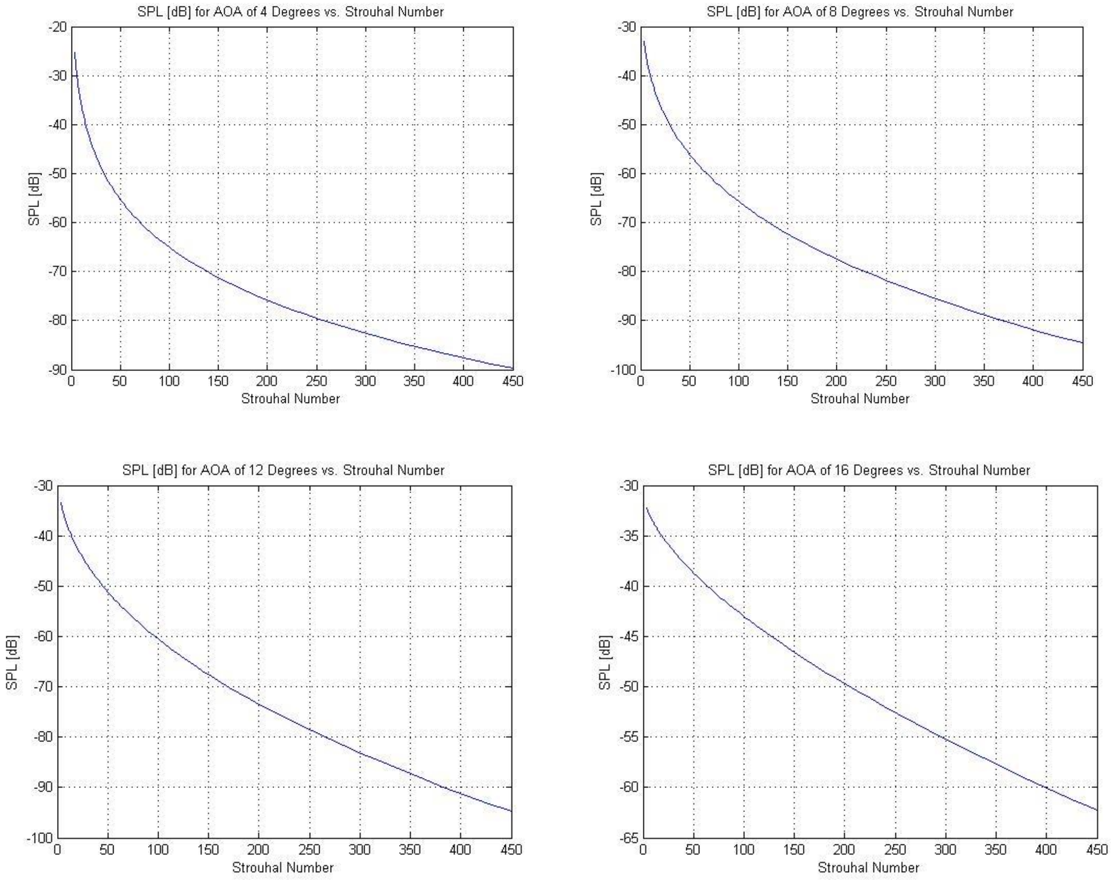


Figure 26. Third Coefficient (c) Fit as a Function of the Angle of Attack.

Finally, equation 3.21 was plotted for angles of attack of 4, 8, 12 and 16 degrees. The normalized sound pressure levels as a function of the Strouhal number for an angle of attack of 4, 8, 12 and 16 degrees are shown in Figure 27. It is noted that the obtained sound pressure levels in the curves are negative for all the Strouhal number domain. This is because of the normalizing for 1 [m] span and airfoil-microphones distance, as well as the free stream velocity of 1[m/s] that was previously applied to the experimental data.

Equation 3.21 will later be used in Chapter 5 and it will be a crucial part in the computations of wind turbine noise predictions. This is why it was essential to use data from thorough experimental wind tunnel tests, as well as proper data processing methods.


SPL [dB] for AOA of 4 Degrees vs. Strouhal Number

SPL [dB] for AOA of 8 Degrees vs. Strouhal Number

SPL [dB] for AOA of 12 Degrees vs. Strouhal Number

SPL [dB] for AOA of 16 Degrees vs. Strouhal Number

Figure 27. SPL as a Functions of the Strouhal Number for an Angle of Attack of a) 4 Degrees, b) 8 Degrees, c) 12 Degrees, d)16 Degrees.

3.4 DU96_main.m MATLAB Code Description

The routine *DU96_main.m* is contained in the appendix of this thesis. This code runs a total of 11 sub-routines that perform different tasks, as it is observed in the flow chart on

Figure 28. The first sub-routine is *DU96_input.m*, and its main function is to define values for reference and geometric variables. These are: reference free stream velocity, reference airfoil span, reference airfoil-microphones distance, airfoil chord, airfoil span and airfoil-microphones distance. It also extracts and saves the experimental data from an Excel *.xlsx* file, into separate arrays. The Excel file *DU96.xlsx* is found in the appendix of this thesis, and contains the wind tunnel SPL for each 1/3rd octave frequency band, angle of attack and free stream flow velocity.

Once all the needed inputs are available to the main routine, it runs *DU96_Strouhal.m*, which computes four vectors containing the values of the Strouhal numbers for each flow velocity, and all frequency bands. Later, the sub-routine *DU96_Uref_Correction.m* applies the free stream flow velocity normalization to all the sound pressure level experimental data. On the other hand, *DU96_Span_Corrections.m,* performs the reference span normalization and *DU96_distacnce_Corrections.m* scales the airfoil-microphone distance. All the corrections performed in these sub-routines is saved in separate arrays.

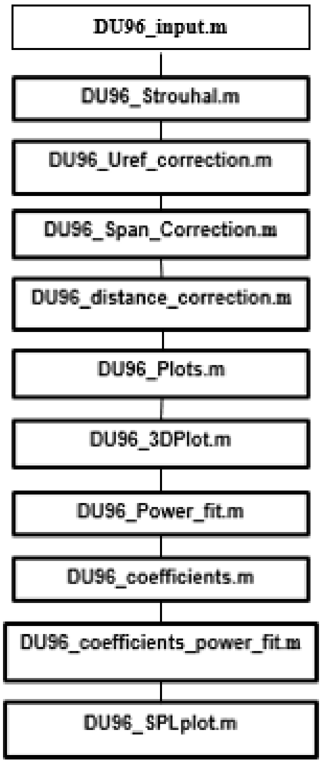


Figure 28. DU96_main.m Code Flow Chart.

The sub-routine DU96_Plots.m, plots all the corrected SPL that were previously obtained as a function of the Strouhal number. This is done for each angle of attack, and all the free stream velocities. Therefore, four plots are generated and they show how well the data has collapsed for each angle of attack; they are shown in Figures 17 and 18. DU96_3DPlot.m also plots the collapsed data, but it does so in a 3D visualization with the sound pressure levels as a function of the Strouhal number and the angle of attack, as shown in Figure 19.

MATLAB provides a curve fitting tool that generates the code for the given data and provided regression. This tool was used to create the sub-routine DU96_Power_fit.m, which fits a curve for the collapsed data, for the four experimental angles of attack of 4, 8, 12 and 16 degrees. This code also provides four plots of the fitted curves with their corresponding residuals, as seen in Figures 20 through 23. In addition, the code was edited so that each coefficient **a**, **b** and **c**, shown in equation 3.19, are saved with appropriate variable names.

Given that each coefficient from equation 3.19 was determined to be a function of the angle of attack, a plot of each one was generated in order to observe the trend of the resulting curves. To do so, the sub-routine *DU96_coefficients.m* was incorporated in the code. Moreover, *DU96_coefficients_power_fit.m* determines the curve fit, so each coefficient from the previous regression is structured as a function of the angle of attack, as seen in equation 3.20 and Figures 24 through 26. This code was also generated with the MATLAB curve fitting tool, and was properly edited.

The last sub-routine run by *DU96_main.m*, is *DU96_SPLplot.m*. The purpose of this sub-routine is to plot the sound pressure levels as a function of the Strouhal number for four different angles of attack, using equation 3.21. In this way, the normalized data for an angle of attack of 4, 8, 12 and 16 degrees can be easily observed. The final results are shown in Figure 27.

CHAPTER IV

EXPERIMENTAL RESULTS ANALYSIS AND DISCUSSION

The experimental results from wind tunnel test procedures, used to obtain airfoil noise data, are essential to the investigation of all the implicated noise sources. Given the many uses of airfoil profiles, including those in renewable energy applications, theoretical approaches on modeling the noise produced by an airfoil, are in need. Moriarty (2005), developed a program called *NAFNoise* that stands for *NREL AirFoil Noise,* and predicts the noise produced by an airfoil. This program uses various semi-empirical sub-routines for different types of noise sources, where, models based on other author's work are integrated. This program has been developed specifically for wind turbine applications, because it is designed so the output data is used for discretized individual blade segments from a wind turbine (Moriarty & Migliore, 2003).

The main purpose of this chapter is to use the model developed by Moriarty and Migliore, to simulate the airfoil wind tunnel experimental test performed at the Virginia Tech Polytechnic Institute and State University Stability Wind Tunnel, shown in Chapter 3. Then the results from the experimental and modeled data are analyzed and compared, by using a MATLAB code that permits running the NAFNoise program multiple times for all the different combinations of test flow velocities and angles of attack used on the test.

4.1 NAFNoise Program Description

The NAFNoise program predicts the noise produced by any airfoil profile from five different independent noise sources, while it is operating. In this section, the five sources used by the software, as well as its required inputs, will be explained. The inputs must be carefully selected, so the experimental wind tunnel test performed at Virginia Tech on the DU-96 airfoil, is coherently modeled with NAFNoise.

4.1.1 Airfoil Noise Sources.

The first four noise sources were determined by Brooks, Pope and Marcolini. They developed semi-empirical formulations for each one of them, in their publication (Brooks, Pope & Marcolini, 1989). The last noise source was developed by Lowson (1993), and Amiet (1975). Each of this noise sources are explained in the following paragraphs.

a) Turbulent Boundary Layer-Trailing Edge Noise (TBL-TE)

The first noise source is determined by the interaction between the turbulent boundary layer and the trailing edge of the airfoil, as shown in Figure 29.

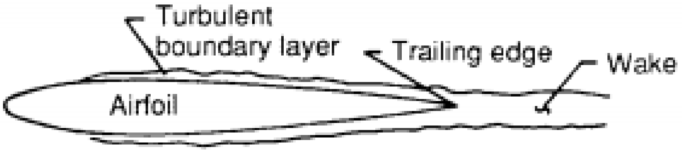


Figure 29. Turbulent Boundary Layer-Trailing Edge Noise. Retrieved From: Brooks, F., Pope, S., Marcolini, M. (1989). Airfoil Self-Noise and Prediction. NASA Reference Publication 1218.

This type of noise source is very common at high Reynolds numbers, and it can originate in both the suction and pressure side of an airfoil. The noise produced for the pressure side is shown in equation 4.1, where $\delta^*$is the boundary layer thickness [m], M is the Mach number, L is the span of the airfoil [m], D_h is the noise directivity (a unit-less value that corrects the sound pressure levels as it sound propagates in specific directions), r_e is the distance to the observer, St is the Strouhal number, $St_1 = 0.02M^6$, $K_1(Re_c)$ is an empirical function obtained by experimentation that depends on the Reynolds number based on the chord Re_c, $\Delta K_1(\alpha, Re_{\delta^*})$ is an empirical function obtained by experimentation that depends on the Reynolds number based on boundary layer thickness δ^* and the angle of attack, and finally A is an empirical spectral shape obtained from experimentation. In the case of the suction side, a similar equation is used (Moriarty & Migliore, 2003). Explanation on variables such as the Directivity of the noise sources, and boundary layer properties, are provided on the theoretical framework of this thesis.

$$L_{p(TBL-TE)} = 10\ log_{10}\left[\frac{\delta^{*}M^{5}LD_{h}}{r_{e}^{2}}\right] + A\left(\frac{St}{St_{1}}\right) + (K_{1} - 3) + \Delta K_{1}\quad [\text{dB}] \tag{4.1}$$

b) Separated Flow Noise (SF)

As the angle of attack of an airfoil is larger, the lift to drag coefficient gets smaller. This is because the size of the turbulent boundary layer, on the suction side becomes very large in size, as observed in Figure 30. This causes stalling, and large amounts of noise. The equation for the noise produced in this case is similar to equation 4.1, the difference is that it contains different empirical functions that are only a function of the angle of attack (Moriarty & Migliore, 2003).

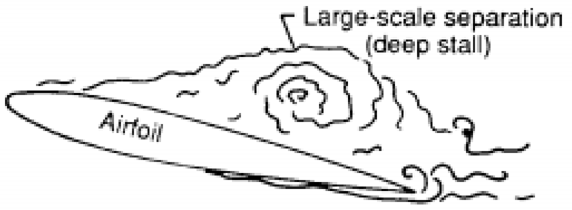


Figure 30. Flow Separation Noise. Retrieved From: Brooks, F., Pope, S., Marcolini, M. (1989). Airfoil Self-Noise and Prediction. NASA Reference Publication 1218.

c) Laminar Boundary Layer-Vortex Shedding Noise (LBL-VS)

In this case, the noise is generated when there is a feedback loop, caused by the interaction of the laminar boundary layer and the vortices that are being shed at the trailing edge of the airfoil, as shown in Figure 31. This type of noise source has more probability of being present at the pressure side of an airfoil, and is tonal in nature because of the amplification caused by the feedback loop (Moriarty & Migliore, 2003).

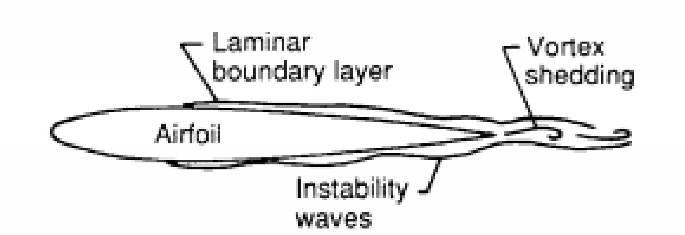


Figure 31. Laminar Boundary Layer-Vortex Shedding Noise. Retrieved From: Brooks, F., Pope, S., Marcolini, M. (1989). Airfoil Self-Noise and Prediction. NASA Reference Publication 1218.

According to Raju (2011), vortex shedding is an unsteady flow that takes place at specific flow velocities and is caused when air flows past a blunt structure. They consist of alternating low pressure instabilities. At very low Reynolds numbers, the fluid has a creeping motion behavior, where there is no separation of the fluid. As the Reynolds number gets larger, a stable vortex is formed downstream of the blunt object. The oscillatory wake starts to have higher amplitudes and to show different flow patterns when the Reynolds number gets even larger. This happens until a phenomena called the Von Kármán vortex street takes place, where, the vortices alternate and detach downstream of the flow. For very large Reynolds numbers, the laminar boundary layer becomes turbulent, therefore, the laminar vortices become turbulent (Granger, 1995). This means that this type of laminar noise is not significant in wind turbines, because most of them operate while having very large Reynolds numbers across their blades.

The noise produced by this source is determined by equation 4.2, where $\delta^*$is the boundary [m], M is the Mach number, L is the span of the airfoil [m], D_h is the noise directivity, r_e is the distance to the observer, $\propto$ is the angle of attack, G_1, G_2, G_3, are empirical relation functions based on St', which is the Strouhal number based on the boundary layer thickness, St'_{peak} is the strouhal number based on Re_c, and Re' is a Reynolds number at the based on the angle of attack (Moriarty & Migliore, 2003) .

$$L_{p(LBL-VS)} = 10\ log_{10}\left[\frac{\delta^* M^5 L D_h}{r_e^2}\right] + G_1\left(\frac{St'}{St'_{peak}}\right) + G_2\left(\frac{Re_c}{Re'}\right) + G_3\ \ [\mathrm{dB}] \qquad (4.2)$$

d) Trailing Edge Bluntness-Vortex Shedding Noise (TEB-VS)

The geometry of the trailing edge of an airfoil determines the amount of noise produced by an airfoil. In the case, if the airfoil trailing edge thickness is very large compared to the boundary layer thickness, the bluntness vortex shedding noise will be dominating. This case is shown in Figure 32.

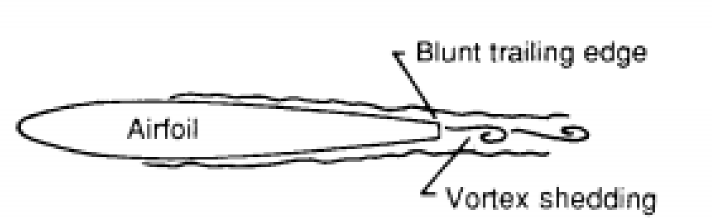


Figure 32. Trailing Edge Bluntness-Vortex Shedding Noise. Retrieved From: Brooks, F., Pope, S., Marcolini, M. (1989). Airfoil Self-Noise and Prediction. NASA Reference Publication 1218.

This type of noise source is determined by equation 4.3, where $\delta^*$is the boundary [m], M is the Mach number, L is the span of the airfoil [m], D_h is the noise directivity, r_e is the distance to the observer, $\propto$ is the angle of attack,G_4,G_5, are empirical relation functions, δ^*_{avg} is the average boundary layer thickness, h is the trailing edge thickness, ψ is the trailing edge angle (between the two surfaces of the trailing edge), and St'' and St''_{peak} are the Strouhal numbers based on h and h/δ^*_{avg} , respectively (Moriarty & Migliore, 2003) .

$$L_{p(TEB-VS)} = 10\ log_{10}\left[\frac{\delta^* M^5 L D_h}{r_e^2}\right] + G_4\left(\frac{h}{\delta^*_{avg}}, \psi\right) + G_5\left(\frac{h}{\delta^*_{avg}}, \psi, \frac{St''}{St''_{peak}}\right) + G_3(\propto)\ \ [\mathrm{dB}] \qquad (4.3)$$

e) Turbulent Inflow Noise

This type of noise source is determined by the interaction of turbulent incoming flow and the leading edge of an airfoil. It becomes of great importance, in the case where the turbulent eddies from the incoming flow are significantly larger than the leading edge radius of the airfoil. This noise source is determined by equation 4.4.

$$L_{p(Inflow)} = L^{H}_{p(Inflow)} + 10\ log_{10}\left[\frac{LFC}{1+LFC}\right]$$

(4.4)

Here, $L^{H}_{p(Inflow)} = 10\ log_{10}\left[\frac{\rho_0^2 c_0^2 lL}{2r_e^2}\ M^3 u^2 I^2 \frac{k^3}{(1+k^2)^{7/3}} D_L\right] + 58.4$, where $\rho_0$is the air density, $c_0$is the speed of sound, l is the turbulence scale (parameter used according to International Electrotechnical Commission-IEC), u is the wind mean speed, I is percentage of wind turbulence, D_L is the directivity, M is the Mach number, and K is the local wave number that is given by $K = {}^{\pi f c}/_{U}$ (f is the frequency of interest, c is the local airfoil chord, and U is the local flow velocity over the airfoil).

On the other hand, $LFC = 10S^2MK^2\beta^{-2}$, where $S^2 = \left(\frac{2\pi K}{\beta^2} + \left(1 + 2.4\frac{K}{\beta^2}\right)^{-1}\right)^{-1}$, and $\beta^2 = 1 - M^2$ (Moriarty & Migliore, 2003) .

4.1.2 NAFNoise Inputs & Outputs.

The NAFNoise software is available for download from the National Renewable Energy Laboratory, on the National Wind Technology Center Information Portal. The software contains an executable file called NAFNoise.exe, and in order to run it, an input file called nafnoise.ipt is required. The input file needs to be modified according to the shape of the airfoil, and the required flow conditions. In this case, these conditions were determined by the fact that the objective of using this program was to model the experiment performed at the wind tunnel at Virginia Tech, on a DU-96 airfoil. An example of the input file nafnoise.ipt is included in the appendix of this thesis.

In the input file nafnoise.ipt, the first three lines are comments. After this, the first section of the file contains atmospheric constants. The speed of sound of the air was chosen to be 337.7559 [m/s], which is approximately the equivalent value for 10 degrees Celsius. This temperature was chosen because it is the one expected during experimental wind tunnel tests. For the same temperature, the kinematic viscosity and density were chosen to be 1.4529 E-5 [$^{m^2}/_{s}$] and 1.225 $\left[^{Kg}/_{m^3}\right]$, respectively.

The next section of the input file correspond to the settings that determine which noise sources will be computed and how. In this case, all the variables were chosen so that the developed noise model created with NAFnoise.exe simulates the same conditions as the experimental wind tunnel test from Chapter 3. The first setting is the tripping condition. Because all of the DU-96 airfoil experimental data chosen was for a tripped condition, a heavy trip was selected (0 is equivalent to no trip, 1 is equivalent to heavy trip, and 2 is equivalent to light trip). The difference between the light and heavy trip, is that for the second one a larger boundary layer thickness is produced (Moriarty, 2005).

The second setting corresponds to the method used to determine the boundary layer thickness. One of them is proposed by Brooks, Pope and Marcolini (1989), and the other is determined by a set of routines that pertain to a program called Xfoil, which was developed at MIT for the analysis of subsonic isolated airfoils (Moriarty, 2005). Because for a heavy trip only the Brooks, Pope and Marcolini method is permitted by the program, this one was chosen (1is equivalent to BPM & 2 is equivalent to Xfoil).

The next setting is the parameter that determines the method for the calculation of the turbulent boundary layer-trailing edge noise. The two methods that can be used are one proposed by Brooks, Pope and Marcolini (1989), and other proposed by Moriarty, Guidati and Migliore (2005). In this case the second one was chosen because as stated by the authors, this method has shown more accurate results than those from Brooks, Pope & Marcolini (0 equivalent to no noise of this source, 1 equivalent to BPM, 2 equivalent to MGM).

The final three settings of this section are set to a value of zero (in all cases 0 is equivalent to the absence of the feature), because in the wind tunnel experiment there was not a turbulent inflow, the airfoil did not have trailing edge bluntness, and there was not laminar boundary layer-vortex shedding noise, because it is prevented with the trip in the airfoil.

The subsequent section of the input file consists on defining the airfoil properties. The first line corresponds to the airfoil chord and the second one to the airfoil span. The third and fourth lines correspond to the income flow velocity and angle of attack, respectively. These two variables have to be changed for all the four experimental flow velocities and angles of attack, and each time this is done the NAFNoise.exe program needs to be executed again. Finally, the trailing edge thickness of the airfoil was set to zero, and the trailing edge angle between the upper and bottom surface was calculated.

In order to calculate the trailing edge angle, the coordinates of the airfoil profile were required. These were obtained from *The Institute of Aeronautics and Astronautics* (AIAA) web site. The airfoil coordinates consist of a file containing two columns corresponding to the chord normalized x-values, and y-values of the airfoil profile. From it, three points were chosen, the first one corresponding to the tip trailing edge point, the second one to the point on the pressure side surface of the airfoil which is the closest to the trailing edge tip, and the third to the point on the suction side surface of the airfoil which is the closest to the trailing edge tip.

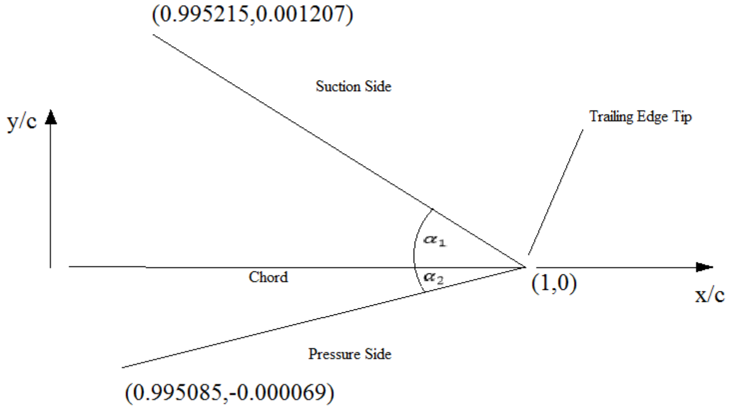


Figure 33. Trailing Edge Angle

Thus, as observed in Figure 33, two angles are formed, and the trailing edge angle α is obtained as the sum of α_1 and α_2, as shown in equation 4.5.

$$\alpha = \alpha_1 + \alpha_2 = tan^{-1}\left(\frac{0.001207}{0.004785}\right) + tan^{-1}\left(\frac{0.000069}{0.004915}\right) = 14.1696^0 \tag{4.5}$$

The next section of the input file has four lines, the first and second one determine the upper and lower trip chord normalized locations. These values were not known for the experimental data shown in Chapter 3, but it has been assumed a value of 0.1 at the pressure and suction side, because a trip is typically located close to the leading edge of the airfoil, in order to induce a transition from laminar to turbulent flow and a thicker boundary layer. The next line of this section is to determine if the program will be using a NACA airfoil profile or not. In this case, since this is not true because a DU-96 profile is used, .FALSE. is written in this line. The final line of this section corresponds to the name of the file containing the coordinates of the airfoil. In this case DU96.dat was used.

The subsequent section of the input file is ignored, because no turbulent inflow has been used, thus all of the inputs concerning this feature have been set to zero. The final section of the input file corresponds to the position of the viewer. In this case this is the position of the microphone array in the wind tunnel experiment. This was at 1.6 meters, with a -90 degrees for the angle relative to the span and -90 degrees for the angle relative to the chord line.

The output file that NAFNoise.exe creates is called nafnoise.out. In it, noise data is presented for 1/3rd octave frequency bands, from 10 [Hz] to 20000 [Hz]. The noise sound pressure level values is presented for each frequency band, for the five noise sources in 5 columns, and a final column for the total sound pressure levels produced, corresponding to the sum of all the sources. An example of an output file nafnoise.out is included in the appendix of this thesis.

4.2 Experimental & Modeled data Analysis

Since the experimental data contained the sound pressure levels for each 1/3rd octave frequency band, for the four velocities of 34 m/s, 44 m/s, 54 m/s and 64 m/s, and for each one, four angles of attack of 4, 8, 12, 16 degrees, the modeled data had to be presented in the same manner for comparison purposes. To do this, a MATLAB code was developed so that by using NAFNoise software, the modeled and experimental data is compared and analyzed. For each case, the experimental and modeled data are presented in a single chart.

First, for an angle of attack of 4 degrees, for the each of the four velocities, the data is presented in Figures 34 a, b, c and d.

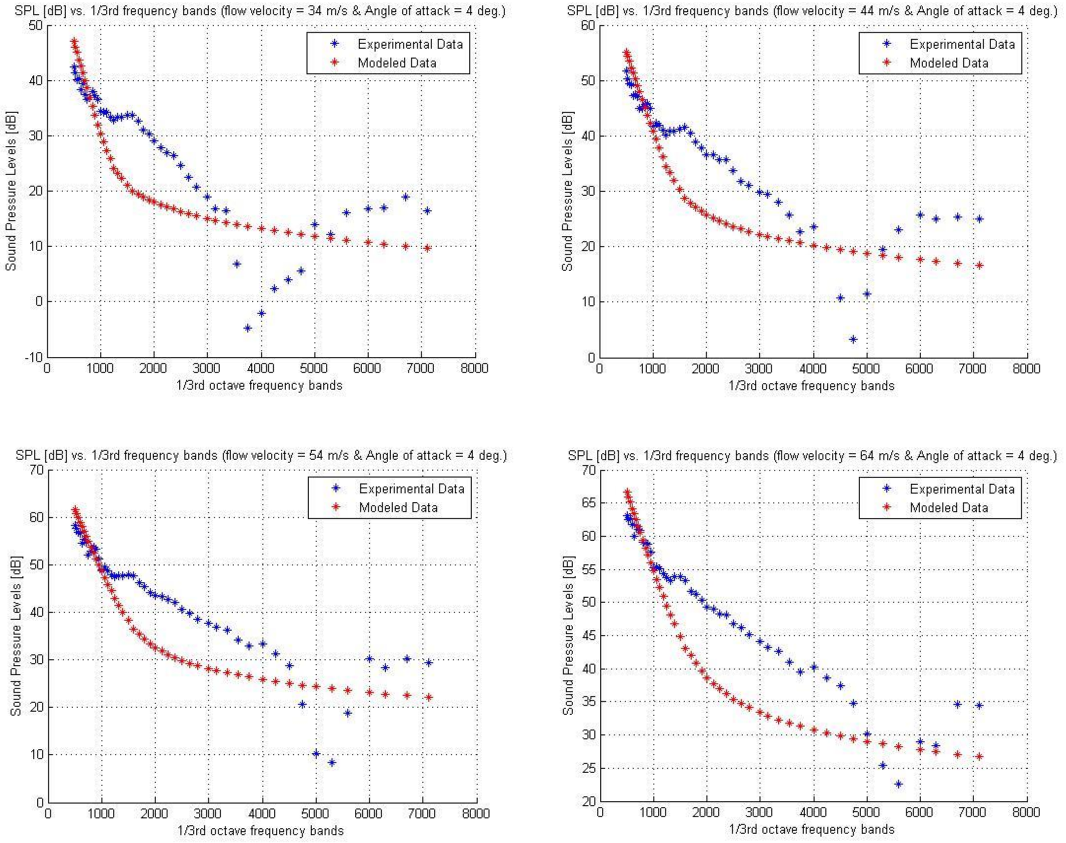


Figure 34. Experimental and Modeled Data for a) flow velocity = 34 m/s & AOA=4 deg, b) flow velocity = 44 m/s & AOA=4 deg, c) flow velocity = 54 m/s & AOA=4 deg, d) flow velocity = 64 m/s & AOA=4 deg.

It is observed that for the four cases, the experimental data follows a similar trend line as the modeled data. For the flow velocities of 34 m/s and 44 m/s, the experimental data has a large deviation from the modeled one, between 4000[Hz] and 5000[Hz], and for 54 m/s and 64 m/s, between 5000[Hz] and 6000[Hz]. The large sound pressure level drop for these frequencies indicates that the experimental measurements for three or four bands was probably inaccurate. This could have happened as a result of faulty measuring equipment or a mistake done while performing frequency analysis.

Apart from this, the program has under-predicted the sound pressure levels by a value of approximately 5 [dB] for all frequencies staring at a 1000 [Hz]. The under-prediction could be caused by the fact that the *NAFNoise.exe* uses an empirical modeled boundary layer thickness computation method, based on a NACA 0012 airfoil which has a different geometry from a DU96 airfoil. Moreover, for this case the Turbulent Boundary Layer-Trailing Edge Noise would be dominant, and the Separated Flow Noise grows as the angle of attack increases. This means that there is a clear dependence of the sound pressure levels on the boundary layer thickness, as observed in equation 4.1. Additionally, it should be taken into account that the boundary layer thickness is dependent on the Reynolds number and it determines how turbulent the flow is. Therefore, it models the frequencies at which the flow separates and vortices are created, that ultimately determine the noise produced at any given frequency, as shown in equation 4.1 where empirical functions depending on the Strouhal number are introduced.

An explanation on how to compute the overall sound pressure levels (OSPL) is contained in the theoretical framework of this thesis. On Table 6, it is shown that the resulting difference between the experimental and modeled overall sound pressure levels, for all the cases are between 1 [dB] and 2 [dB]. Moreover, the modeled overall sound pressure levels deviate by very low percentages from the experimental ones, specifically between 1.36% and 3.92%. This means that both the experimental and modeled overall sound pressure level data fit relatively well. It can also be concluded that for an angle of attack of 4 degrees, as the flow velocity increases, the modeled and experimental data have a closer fit.

CASE (flow velocity, angle of attack)	Experimental OSPL [dB]	Modeled OSPL [dB]	%
34 m/s ,4 deg.	51	53	3.92
44 m/s ,4 deg.	60	62	3.33
54 m/s ,4 deg.	67	69	2.98
64 m/s ,4 deg.	73	74	1.36

Table 6. Experimental and Modeled Overall Sound Pressure levels for a) flow velocity = 34 m/s & AOA=4 deg, b) flow velocity = 44 m/s & AOA=4 deg, c) flow velocity = 54 m/s & AOA=4 deg, d) flow velocity = 64 m/s & AOA=4 deg.

For an angle of attack of 8 degrees, for each of the four flow velocities, the data is presented in Figures 35 a, b, c and d.

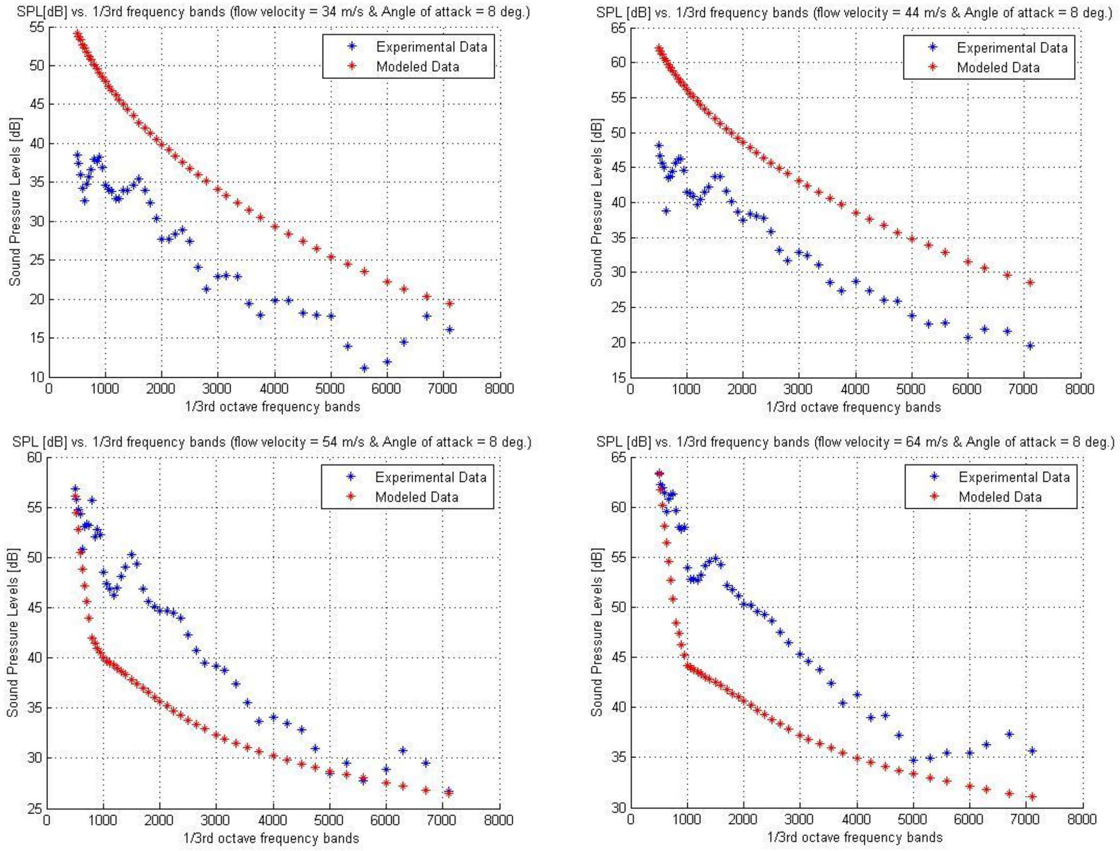


Figure 35. Experimental and Modeled Data for a) flow velocity = 34 m/s & AOA=8 deg, b) flow velocity = 44 m/s & AOA=8 deg, c) flow velocity = 54 m/s & AOA=8 deg, d) flow velocity = 64 m/s & AOA=8 deg.

For these cases, where the angle of attack is 8 degrees, it is observed that for the velocities of 34 m/s and 44 m/s, both the experimental and the modeled data follow the same trend line. In these cases there is an approximate over-prediction of 10 [dB] to 15[dB] for all frequencies, between the experimental and modeled data. For the cases where the flow velocities are 54 m/s and 64 m/s, the modeled data is under-predicted by values of 5 [dB] to 15 [dB] from 500 [Hz] and 4000 [Hz]. The over-prediction and under-prediction could be caused by the boundary layer thickness computation and empirical functions that depend on the Strouhal number, as explained previously for the cases shown in Figure 34.

CASE (flow velocity, angle of attack)	**Experimental OSPL [dB]**	**Modeled OSPL [dB]**	**%**
34 m/s ,8 deg.	50	64	28.00
44 m/s ,8 deg.	58	72	24.14
54 m/s ,8 deg.	66	61	8.19
64 m/s ,8 deg.	73	68	6.85

Table 7. Experimental and Modeled Overall Sound Pressure levels for a) flow velocity = 34 m/s & AOA=8 deg, b) flow velocity = 44 m/s & AOA=8 deg, c) flow velocity = 54 m/s & AOA=8 deg, d) flow velocity = 64 m/s & AOA=8 deg.

The overall sound pressure levels obtained are shown in Table 7. It is observed that for the under-predicted cases, the modeled OSPL values are deviated from the experimental ones by 6.85% and 8.19 %, and for the over-predicted cases the percentages are higher and have a value of 24.14% and 28%. This shows that for an angle of attack of 8 degrees, the modeled data did not fit very well. Nevertheless, for both the under-predicted cases and over-predicted cases, it is evidenced the percentages become lower as the flow velocity increases.

For an angle of attack of 12 degrees, for each of the four flow velocities, the data is presented in Figures 36 a, b, c and d.

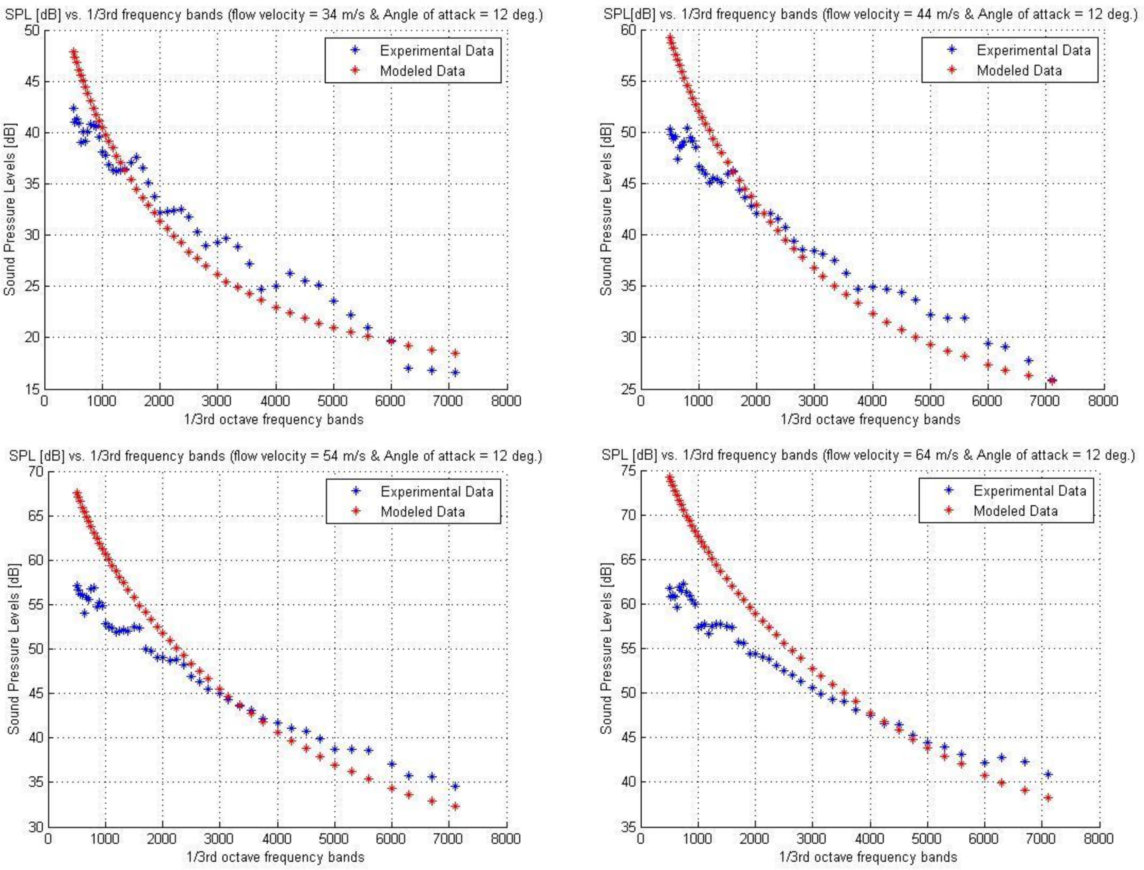


Figure 36. Experimental and Modeled Data for a) flow velocity = 34 m/s & AOA=12 deg, b) flow velocity = 44 m/s & AOA=12 deg, c) flow velocity = 54 m/s & AOA=12 deg, d) flow velocity = 64 m/s & AOA=12 deg.

For a 12 degree angle of attack, for all flow velocities, the experimental and modeled data fits very well, except for frequencies lower to a 1000[Hz]. The values are over-predicted for these frequencies and have a maximum of 10 [dB]. This could be because at these frequencies the modeled noise is different from the experimental one due to innacurate boundary layer computations and Strouhal number dependent empirical functions (as explained for Figure 34).

CASE (flow velocity, angle of attack)	**Experimental OSPL [dB]**	**Modeled OSPL [dB]**	**%**
34 m/s ,12 deg.	53	57	7.55
44 m/s ,12 deg.	62	68	9.68
54 m/s ,12 deg.	69	77	11.59
64 m/s ,12 deg.	74	83	12.16

Table 8. Experimental and Modeled Overall Sound Pressure levels for a) flow velocity = 34 m/s & AOA=12 deg, b) flow velocity = 44 m/s & AOA=12 deg, c) flow velocity = 54 m/s & AOA=12 deg, d) flow velocity = 64 m/s & AOA=12 deg.

The OSPL for the modeled and experimental data differ from 4[dB] to 9 [dB]. This is shown in Table 8, where there are also the percentages showing the modeled OSPL deviation from the experimental ones. Unlike the two previous cases for an angle of attack of 4 degrees and 8 degrees, in this case, as the flow velocity increases the percentage grows from 7.55 % to 12.16%.

Finally, for an angle of attack of 16 degrees, for each of the four flow velocities, the data is presented in Figures 37 a, b, c and d. It can be observed that for all these cases, for all frequencies the modeled data is under-predicted with values varying from 1 to 15 [dB]. As explained before this could be because of the boundary layer computation inaccuracy and the use of empirical Strouhal number dependent empirical functions.

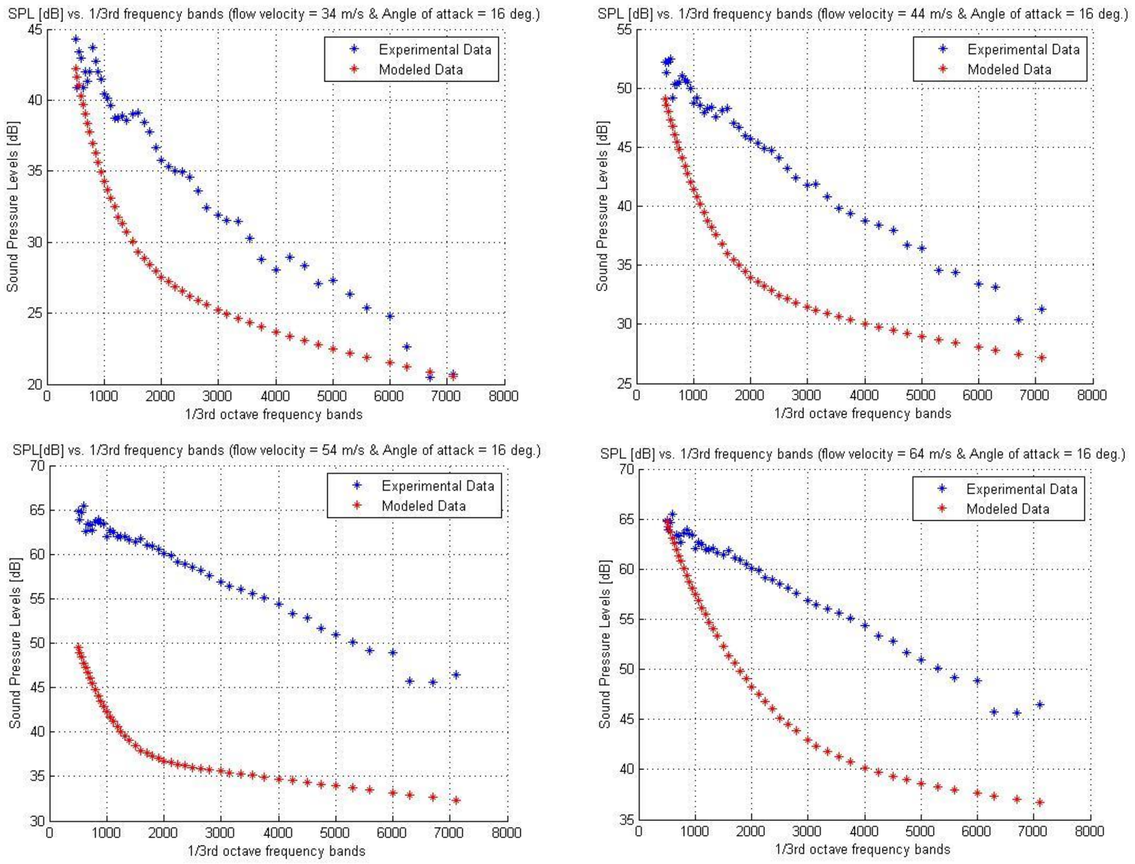


Figure 37. Experimental and Modeled Data for a) flow velocity = 34 m/s & AOA=16 deg, b) flow velocity = 44 m/s & AOA=16 deg, c) flow velocity = 54 m/s & AOA=16 deg, d) flow velocity = 64 m/s & AOA=16 deg.

As observed in Table 9, the values for the OSPL for these two velocities the modeled overall sound pressure levels differ from the experimental by 4 to 11 [dB]. The percentages that determine the modeled OSPL deviation from the experimental values vary from 5.13% to 18.06%. In this case, these percentages are random to any of the four velocities used.

CASE (flow velocity, angle of attack)	Experimental OSPL [dB]	Modeled OSPL [dB]	%
34 m/s ,16 deg.	55	51	7.27
44 m/s ,16 deg.	64	58	9.38
54 m/s ,16 deg.	72	59	18.06
64 m/s ,16 deg.	78	74	5.13

Table 9. Experimental and Modeled Overall Sound Pressure levels for a) flow velocity = 34 m/s & AOA=16 deg, b) flow velocity = 44 m/s & AOA=16 deg, c) flow velocity = 54 m/s & AOA=16 deg, d) flow velocity = 64 m/s & AOA=16 deg.

After comparing the modeled and experimental data, it can be concluded that the sound pressure levels determined by the NAFNoise.exe program are relatively inaccurate when compared to the experimental wind tunnel test data, from Chapter 3. This may be because of the turbulent boundary layer computation method used, as well as the empirical functions that depend on the Strouhal number that the program uses. Because a NACA 0012 was used for these calculations, the geometric differences with a DU-96 airfoil could introduce errors. This is reflected on the fact that for certain angles of attack, the modeled data is more closely predicted than for others. Nevertheless, most of the predicted curves follow the same trend line as the experimental data, and from all the cases a maximum of a 28% of OSPL deviation was obtained, but a minimum of 1.36% also reflects very accurate predictions for some cases.

4.3 NAFNOISE_main.m MATLAB Code Description

The routine *NAFNOISE_main.m* is contained in the appendix of this thesis. This code consists of 5 sub-routines that perform different tasks, as observed in the flow chart in Figure 38. The first sub-routine is *NAFNOISE_RUN.m*, and the objective of it is to run *NAFNoise.exe* for all the four angles of attack of 4, 8, 12 and 16 degrees, for each of the four flow velocities of 34 m/s, 44 m/s, 54 m/s and 64 m/s. To do this, the lines corresponding to the angle of attack and flow velocity on the input file *nafnoise.ipt*, must be changed. Within a loop for each flow velocity and angle of attack, the code opens the input file, saves each line of the file in a cell, closes the file, changes the required lines of the file, writes a new input file, runs NAFNoise.exe, and finally retrieves the data from the output file *nafnoise.out.*

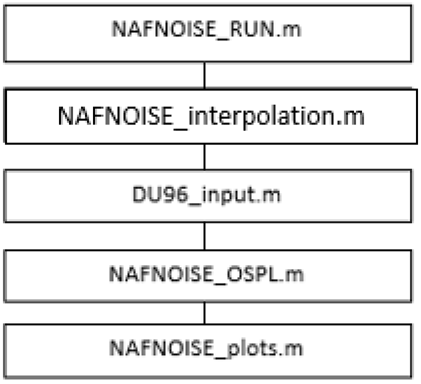


Figure 38. NAFNOISE_main.m Flow chart.

The sub-routine *NAFNOISE_interpolation.m* is required because the output file nafnoise.out, from the NAFNoise.exe software, does not contain the sound pressure levels generated by the airfoil, for all the 1/3rd octave frequency bands used on the experimental data. This means that some modeled sound pressure level values must be interpolated, so that the modeled and experimental data are both as a function of the same frequency bands.

The sub-routine *DU96_input.m*, extracts the experimental sound pressure levels for each flow velocity and angle of attack, from an excel file *DU96.xlsx* (this file is contained in the appendix of this thesis). The sub-routine *NAFNOISE_OSPL.m* computes the overall sound pressure levels for each flow velocity and angle of attack, of both the modeled and experimental data. The values computed in this part of the code, for each case are presented in Tables 6, 7, 8 & 9.

Finally, the sub-routine *NAFNOISE_plots.m,* plots the experimental and modeled sond pressure levels, as a function of 1/3rd octave frequency bands. This is, for every case of flow velocity and angle of attack combination. In this way the experimental and modeled results can be compared and analyzed. The plots generated by this sub-routine are presented in Figures 34, 35, 36 & 37.

CHAPTER V

WIND TURBINE NOISE PREDICTION

The main purpose of this chapter is to show how a wind turbine noise prediction tool has been developed using MATLAB. In this case, the prediction was performed on a Nordtank NTK 500/41 wind turbine, because all its geometric properties were able to be obtained from the book *Aerodynamics of Wind Turbines*, by Martin O. L. Hansen. All the implemented computations, used methods and results, are thoroughly explained, and a description of the programmed MATLAB code is presented.

5.1 Prediction Model Description

The process used to obtain wind turbine noise predictions will be described, by dividing it into six sections. Each section concatenates with the next one, until an explanation on how to determine the final results is explained in the last one. The first one describes how the turbine's incoming wind velocity was modeled, the second one corresponds to the geometric modelling of each of the wind turbine's blades, the third one explains how each blade rotate around the turbine hub, the fourth one explains the grid in which the predicted noise was calculated, the fifth one corresponds to the noise source modeling, and the last one explain how the predicted overall A-weighted sound pressure levels were computed on the spherical shell grid.

The modeled incoming wind velocity determined in section 5.1.1 is necessary in order to obtain geometric variables from each of the blades, which is the task in section 5.1.2. On the other hand, section 5.1.3 explains how the positions of the blades are determined, as they rotate. Section 5.1.4 explains how the positions for all the points on a spherical shell grid were obtained, so the predicted noise is computed at these points. Only when all the data from the previous four sections has been determined, the noise sources modeling can be completed, as explained in section 5.1.5. Finally, the predicted noise is obtained by determining the contribution from all noise sources, at each point on the grid.

5.1.1 Incoming Wind Velocity Modeling.

The motion of air on the earth's surface is dependent of both surface friction and the pressure and temperature gradients determined by solar heating. These factors determine the formation of a boundary layer or wind velocity profile. Usually, it extends for up to one kilometer above the surface, and it is influenced by the time of the day. An example of wind vleocity profiles for different types of terrain are shown in Figure 39. Here it can be observed that in an urban area, wind profiles are higher than for suburb and level country areas.

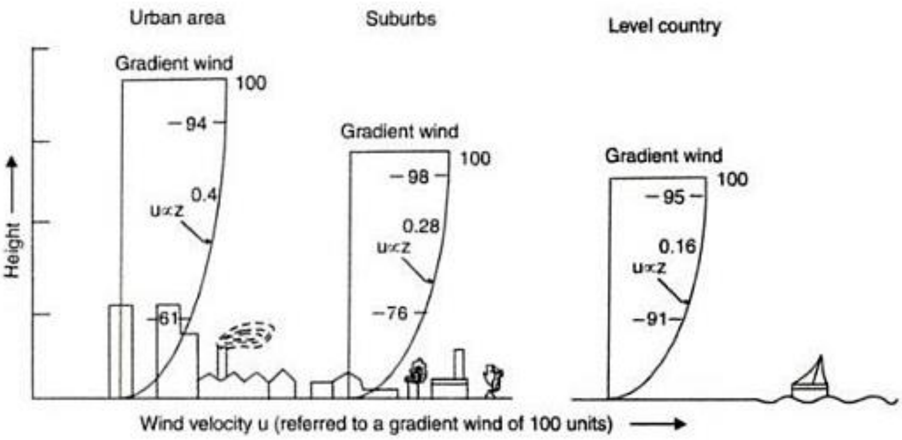
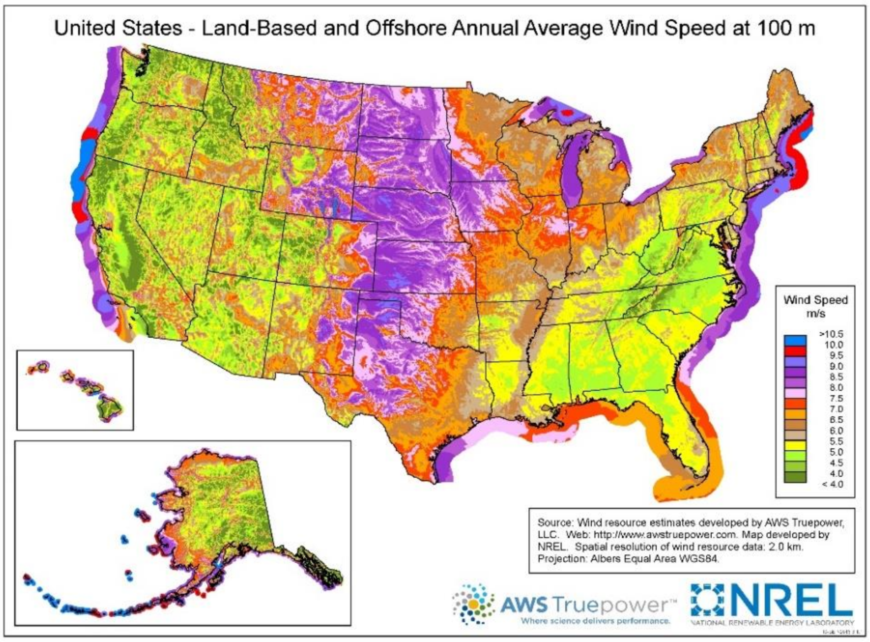


Figure 39. Effect of terrain roughness on the wind velocity profile. Retrieved from: Rao,C.S. (2006). Environmental Pollution Control Engineering. New Age International Ltd., New Delhi.

A wind velocity profile is determined by equation 5.1, where u [m/s] is the wind velocity at height z [m], and $u1$ [m/s] is the wind velocity at height $z1$ [m], α is a unit-less exponent that varies depending on the surface's roughness (Rao, 2006).

$$\frac{u}{u1} = \left(\frac{z}{z1}\right)^{\alpha} \quad (5.1)$$

This equation was used to determine the incoming wind flow velocity experienced by the wind turbine, at different heights. The value of α was determined to be 0.16, which is the value for flat open country, as it was established by Rao (2006). The values of $z1$ and $u1$ were obtained from Figure 40, where the wind velocities at a 100[m] height is shown for different parts of the United States. A value of a 100[m] was chosen for $z1$, and a value of 10.5 [m/s] for $u1$. Figure 40 was retrieved from the National Renewable Energy Laboratory's, Dynamic Maps, GIS Data, & Analysis Tools web page.


United States - Land-Based and Offshore Annual Average Wind Speed at 100 m

Source: Wind resource estimates developed by AWS Truepower, LLC. Web: http://www.awstruepower.com. Map developed by NREL. Spatial resolution of wind resource data: 2.0 km. Projection: Albers Equal Area WGS84.

Figure 40. United States-Land Based and Offshore Annual Average Wind Speed at 100 m. Retrieved from: http://www.nrel.gov/gis/wind.html

5.1.2 Wind Turbine Blade Geometry Modeling.

In order to determine the noise produced by each one of the wind turbine blades, as they rotate, they were modeled as a rigid body and broken down into equal sections, as shown in Figure 41. In the case of the Nordtank NTK 500/41 the blade has a length of 20.5 [m], and the twist and chord distributions along the blade's span is shown in Table 10. Because the number of sections into which the blade was divided for computational purposes is different from those shown in table 10, all the values of the twist were interpolated accordingly to the chosen value of blade sections. In this case, the blades were divided into

20 sections, as shown in table 11, where the interpolated radius and twist appear. It should be noted that the chord was not interpolated, because this variable is not necessary on noise computations.

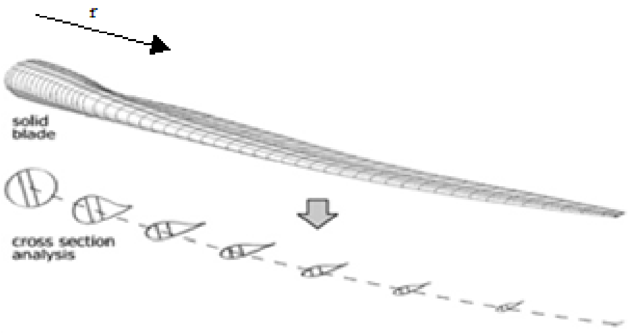
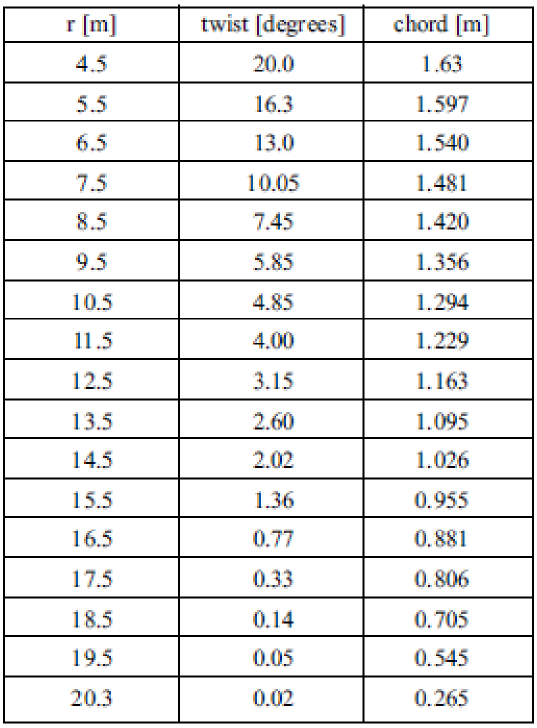


Figure 41. Wind Turbine Blade divided into sections. Retrieved From: http://www.becas.dtu.dk/About



r [m]	twist [degrees]	chord [m]
4.5	20.0	1.63
5.5	16.3	1.597
6.5	13.0	1.540
7.5	10.05	1.481
8.5	7.45	1.420
9.5	5.85	1.356
10.5	4.85	1.294
11.5	4.00	1.229
12.5	3.15	1.163
13.5	2.60	1.095
14.5	2.02	1.026
15.5	1.36	0.955
16.5	0.77	0.881
17.5	0.33	0.806
18.5	0.14	0.705
19.5	0.05	0.545
20.3	0.02	0.265

Table 10. Twist and Chord distributions for a Nordtank NTK 500/41 blade. Retrieved From: Hansen, M. (2008). Aerodynamics of Wind Turbines. (pp.58) Earthscan, Sterling, VA-USA.

r [m]	twist [degrees]
4.9	18.5200
5.7	15.6400
6.5	13.0000
7.3	10.6400
8.1	8.4900
8.9	6.8100
9.7	5.6500
10.5	4.8500
11.3	4.1700
12.1	3.4900
12.9	2.9300
13.7	2.4840
14.5	2.0200
15.3	1.4920
16.1	1.0060
16.9	0.5940
17.7	0.2920
18.5	0.1400
19.3	0.0680
20.1	0.0275

Table 11. Interpolated Twist Distribution for a Nordtank NTK 500/41 blade

Each section of the blade was considered to be an airfoil profile, and three angles were computed for each one, as shown in Figure 42. These three angles were computed because they determine the noise radiation directivity produced by a noise source located at each blade section, as it will be explained in section 5.1.5. The first one corresponds to the sum of the segment's pitch and local twist and was called β. The second one was determined to be the angle between the rotational flow velocity and the relative wind velocity, and it was called γ. In order to compute γ, equation 5.2 was used. The rotational flow velocity magnitude $|\bar{W}|$ was computed for each segment by using the distance from the turbine's hub to the segment position, and the fact that this turbine is designed to rotate at 27.1 [rpm]. This is shown in equation 5.3. On the other hand, the wind velocity $|\bar{V}|$ computation was obtained, as explained in section 5.1.1.

$$\gamma = tan^{-1} \frac{|\bar{V}|}{|\bar{W}|}$$

(5.2)

$$|\bar{W}|[{}^{m}/_{s}] = 27.1\ [rpm] \times \frac{2\pi}{60} \times r[m]$$

(5.3)

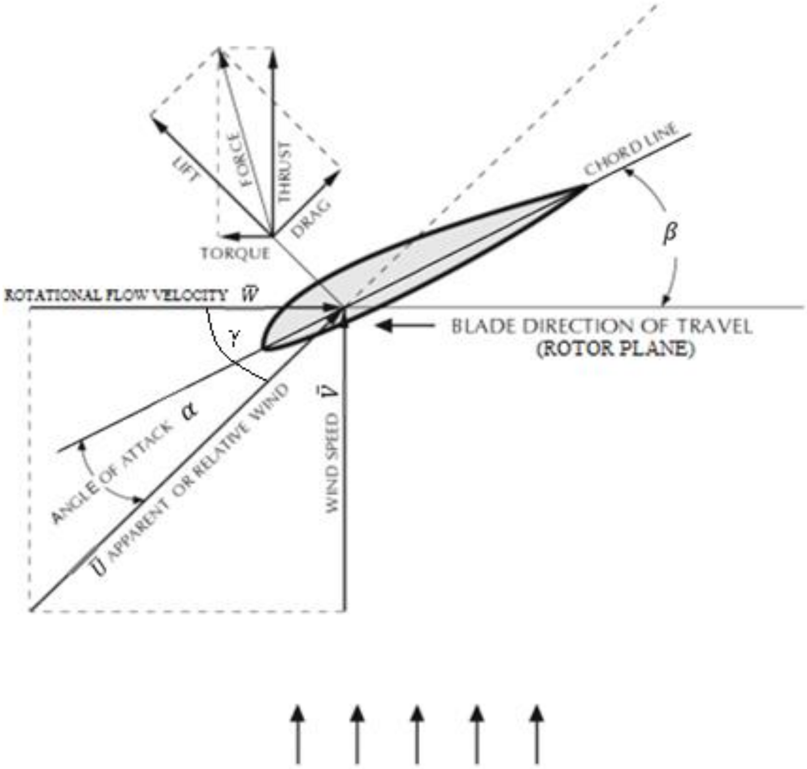


Figure 42. Airfoil with Definition of Flow Angles and Forces. Retrieved from: Gipe, P. (2004), Renewable Power: Renewable Energy for Home, Farm, and Business, Chelsea Green Publishing Company, USA.

The third angle computed for each segment of the blade was the angle of attack, which was computed as the subtraction of the angle γ minus β. Additionally, with the angle γ and the rotational flow velocity, the apparent or relative wind was also computed as shown in equation 5.4.

$$|\bar{U}| = |\bar{W}| \times \cos\gamma \qquad (5.4)$$

5.1.3 Rotational Positions Modeling.

The blade positions, determined by their rotation, were obtained by using a rectangular coordinate system located at the hub of the turbine, as shown in Figure 43. The direction parallel to the turbine's tower was determined to be the z –axis, the x-axis direction was perpendicular to the rotation of the blades, and finally the y-axis direction was determined to be parallel to the rotation of the blades. This is shown in Figure 43.

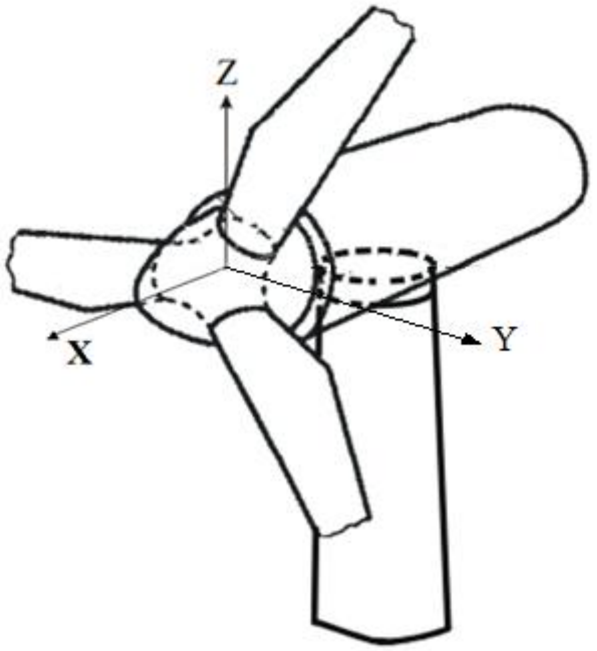


Figure 43. Wind Turbine Coordinates. Retrieved From: van Rooij, R. (2001). Terminology, Reference Systems and Conventions. Duwind 2001.004

The first step in order to obtain the rotational positions was to determine a vector $\bar{d}$ from the center of the hub towards all the sections contained in one blade. This, when one of the blades was on a position parallel to the turbine's tower, as shown on Figure 44. After, the vectors corresponding to each blade section were rotated around the x –axis with the rotation matrix shown in equation 5.5a, for various equal azimuth angles $\Delta\theta$, until 360 degrees were reached. In this way, all the positions of the blade's sections, for all azimuth angles, were determined.

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Delta\theta) & \sin(\Delta\theta) \\ 0 & -\sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \tag{5.5a}$$

For example, for the first blade section, the vector $\bar{d} = [0\ 0\ 4.9]$, and the rotation positions were computed as shown in equation 5.5b. This is also shown in Figure 44.

$$\bar{d}\,rotated = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Delta\theta) & \sin(\Delta\theta) \\ 0 & -\sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 4.9 \end{bmatrix} = \begin{bmatrix} 0 \\ 4.9\sin(\Delta\theta) \\ 4.9\cos(\Delta\theta) \end{bmatrix} \tag{5.5b}$$

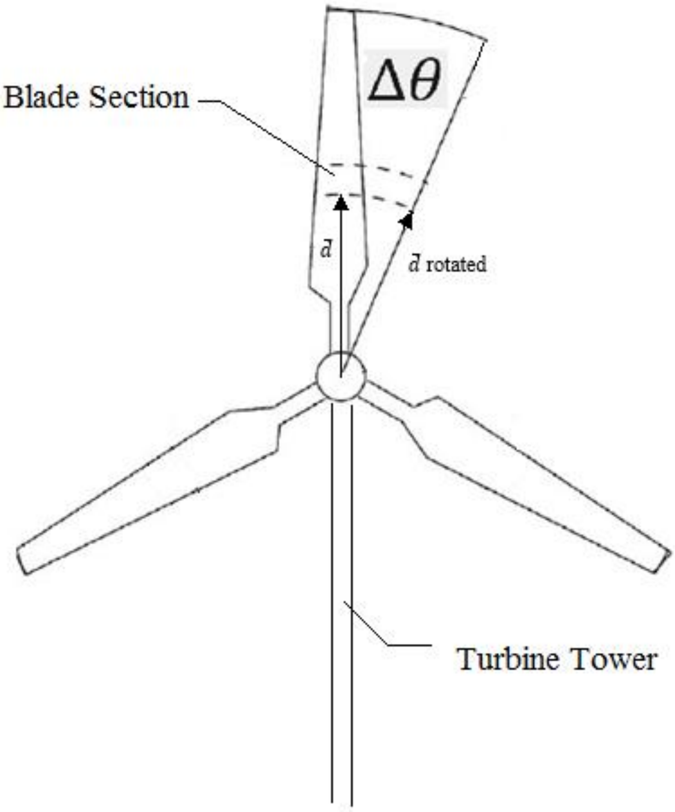


Figure 44. Blades Rotation Given by Azimuth Angle. Retrieved from: Hansen, M. (2008). Aerodynamics of Wind Turbines. (pp.58) Earthscan, Sterling, VA-USA.

5.1.4 Spherical Grid Modeling.

A spherical shell grid in which the predicted sound pressure levels were calculated, was determined. It was constructed, as shown in Figure 45. To do so, the location for each point on the grid was determined by using vectors with a spherical coordinate system centered at the turbine's hub. It was defined by a radius p, an angle θ and an angle $\emptyset$, as shown in Figure 46. Additionally, the spherical coordinates were transformed to rectangular coordinates, for computational purposes, as explained in section 5.1.5. The sphere grid was located at the center of the hub, with a radius of 35 [m]. It should be taken into account that the turbine's hub height is also 35[m], so part of the predicted sound pressure levels would be calculated at the ground or near it.

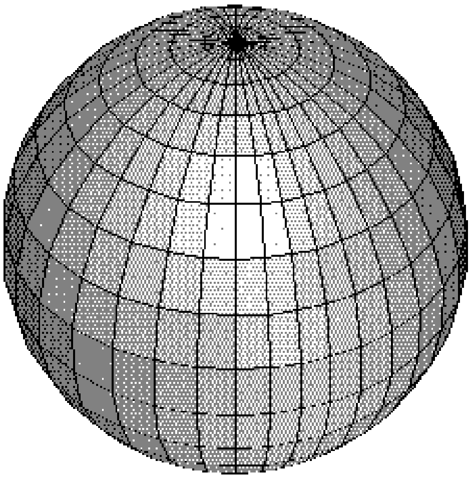


Figure 45. Spherical Grid. Retrieved From: http://www-personal.umich.edu/~cjablono/dycore_test_suite.html.

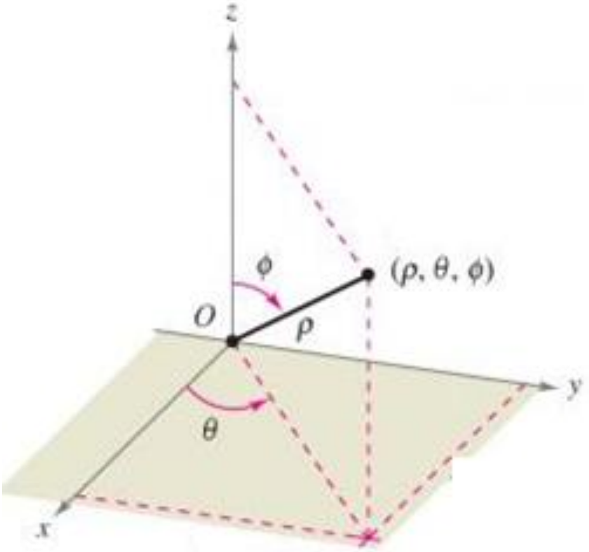


Figure 46. Spherical Coordinates Definition. Retrieved From: Larson, R. &Edwards, B. (2010). Multivariable Calculus. Cengage Learning, Inc. Belmont, CA, USA.

5.1.5 Noise Source Modeling.

As explained in chapter four, all the noise sources for an airfoil are determined by fluctuating instabilities on the flow. The noise produced by these sources can be modeled as a compact dipole at the trailing edge of the airfoil, as shown in Figure 47. This means that for all the positions computed in section 5.1.3, a compact dipole was modeled to be at the trailing edge of all the blades' sections airfoil profiles.

A dipole is determined by a directivity pattern that has maximum radiation in the direction perpendicular to the mean flow (Bowdler & Leventhall, 2011). For a dipole, the root mean square sound pressure value at an angle θ from a line perpendicular to the chord line and a distance r [m], is given by equation 5.6. Here, $W_D [W]$ is the sound power of the dipole, $\rho \left[{Kg}/{m^3}\right]$ is the air density and $c\ [{m}/{s}]$ is the speed of sound (Bies & Hansen, 2009).

$$P_{rms}^2 = 3\, W_D\, \frac{\rho\, c}{4\, \pi\, r^2}\ cos^2\, \theta \quad [\text{Pa}] \tag{5.6}$$

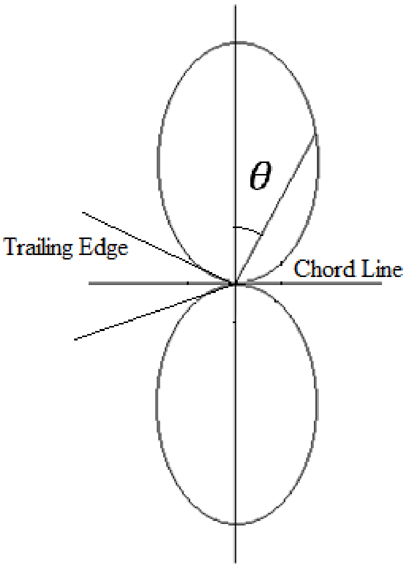


Figure 47. Radiation Pattern of a Dipole at the Trailing Edge of an Airfoil

Using equation 5.6 to obtain the sound pressure levels produced by a dipole, the expression on equation 5.7a is resultant, which is simplified into equation 5.7b. It is observed that the value of $10 * log10\left(\frac{3\,W_D\,\rho\,c}{4\,\pi P_{ref}^2}\right)$ is equivalent to the sound pressure levels produced by a dipole, when $\cos\theta = 0$ and r=*1*, as shown on equation 5.7c.

$$LP_{dipole} = 10 * log10\left(\frac{3\,W_D\,\rho\,c}{4\,\pi r^2 P_{ref}^2}\,cos^2\,\theta\right) \quad [\text{dB}] \qquad (5.7a)$$

$$LP_{dipole} = 10 * log10\left(\frac{3\,W_D\,\rho\,c}{4\,\pi P_{ref}^2}\right) + 10 * log10\left(\frac{cos^2\,\theta}{r^2}\right) \quad [\text{dB}] \qquad (5.7b)$$

$$LP_{dipole} = LP_{(cos\theta=0,r=1)} + 10 * log10\left(\frac{cos^2\,\theta}{r^2}\right) \quad [\text{dB}] \qquad (5.7c)$$

In Chapter 3, equation 3.21 was established by using the experimental data from a wind tunnel airfoil test. It determined the sound pressure levels produced by an airfoil as a function of Strouhal number and the angle of attack. These values were normalized to a 1[m] distance from the airfoil to the used microphone array (distance source-observer, perpendicular to the airfoil's chord line). This means that the sound pressure levels from equation 3.21 are equivalent to the value of $LP_{(cos\theta=0,r=1)}$ on equation 5.7c, for any given angle of attack and Strouhal number (The Strouhal number introduces sound pressure level values for all frequency bands).

Each blade section position computed in section 5.1.3 has a specific airfoil span, and is exposed to different relative incoming flow velocities. Therefore, before replacing equation 3.21 into $LP_{(cos\theta=0,r=1)}$ of equation 5.7c, a rescaling procedure is required for all blade section positions, according to their local span and relative flow velocity. This is because all the values from equation 3.21 were normalized to an incoming relative flow velocity of 1[m/s] and an airfoil span of 1[m]. The rescaling procedure is a normalization to new values of incoming relative flow velocities, and span, and it is done in the same way as it was explained in section 3.3.2

It has to be taken into account that the second expression, $10 * log10\left(\frac{cos^2\theta}{r^2}\right)$ on equation 5.7c, permits the computation of noise at all the grid locations from section 5.1.4. Therefore, for each of the blade section positions from section 5.1.3, the noise on the grid was computed by using equation 5.7c. For example, at a specific blade section position, in order to obtain the values of $\cos\theta$ towards each of the points on the gird, a dot product between two unit vectors has to be determined. The first one is the unit vector $\bar{s}$ corresponding to the direction perpendicular to the blade's section chord at the trailing edge of its airfoil profile, and parallel to the maximum radiation from the dipole, as shown in Figure 48. The second one is a unit vector $\bar{g}$ that is directed from the trailing edge of the blade section towards each point on the spherical shell grid.

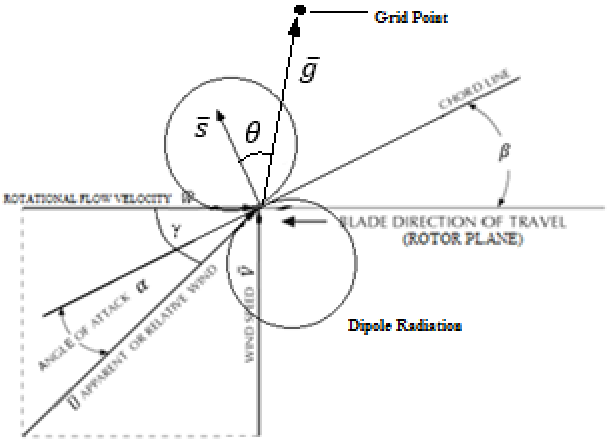


Figure 48. Dipole Unit Vector for each blade section for all azimuth rotation blade positions.

Finally, the value of r^2 from the second expression on equation 5.7c, is defined as the square of the distance between the blade section positions from section 5.1.3, and the spherical grid points from section 5.1.4.

5.1.6 Overall A-Weighted Sound Pressure Levels Computation.

The sound pressure levels on the grid, predicted by applying equation 5.7c for all the blade section rotational positions from section 5.1.3, have to be A-weighted. The reason for this, is that by adding the frequency dependent values from equation 2.14b, the resulting sound pressure levels are adapted to the loudness perceived by the human ear.

On the other hand, since equation 5.7c determines the sound pressure levels on the spherical grid, originated at a dipole located at a specific blade section position, for various Strouhal numbers, then the resultant sound pressure levels from all frequencies have to be summed. To do so, an overall A-weighted sound pressure level corresponding to a specific noise source ($OASPL_{source}$), has to be obtained by applying equation 5.9. In this case, n is the number of frequency bands (or Strouhal numbers), and $Lp_{source-st(i)}$ are the A-weighted sound pressure levels corresponding to each frequency band (or Strouhal number).

$$OASPL_{source} = \sum_{i=1}^{n} 10^{Lp_{source-st(i)}/10} \quad \text{[dBa]} \tag{5.9}$$

Afterwards, all the $OASPL_{source}$ contained in one blade are summed into overall A-weighted sound pressure levels on the spherical grid called $OASPL_{Blade}$. This is performed for all the azimuth rotational positions from section 5.1.3, and equation 5.10 is used, where m is the number of sources contained in one blade, which is the same number of sections contained in one blade.

$$OASPL_{Blade} = \sum_{i=1}^{m} 10^{OASPL_{source(i)}/10} \quad \text{[dBa]} \tag{5.10}$$

Since the sound pressure levels on the grid are obtained from the contribution of three different blades at different positions in time, the overall A-weighted sound pressure levels from three blades $OASPL_{turbine}$, are obtained using equation 5.11. This is done for every possible azimuth rotational position of the three blades (the blades are carefully positioned at 120 degrees from each other as observed in Figure 44), therefore three $OASPL_{bade}$ values have to be carefully chosen. In this way, the predicted values of the OASPL produced by the turbine on the grid are obtained, at every instant of rotation of the three blades.

$$OASPL_{turbine} = \sum_{i=1}^{3} 10^{OASPL_{bade(i)}/10} \quad \text{[dBa]} \tag{5.11}$$

5.2 Noise Prediction Results

Given the importance of the angles of attack on the noise computations and aerodynamic performance of the blades, they were plotted as shown in Figure 49, where they are observed for all rotational positions, from an upstream view that is centered at the hub of the wind turbine. The number of segments into which the blade was divided was 20 and the number of azimuth rotational positions was 45. It is observed that the maximum angle of attack is close to 14 degrees, and the minimum of 8 degrees. The reason why the highest angles of attack are distributed right below the turbine’s hub, and the lowest are located at the tip on the top of the wind turbine’s hub, is because the angle of attack is ultimately dependent of the incoming wind velocity, as shown in equation 5.12. At the top of the turbine's hub the incoming wind has higher values than at the bottom of it because of the modeled wind velocity profile.

$$\propto = \gamma - \beta = \left(tan^{-1} \frac{|\bar{V}|}{|\bar{W}|} \right) - \beta \qquad (5.12)$$

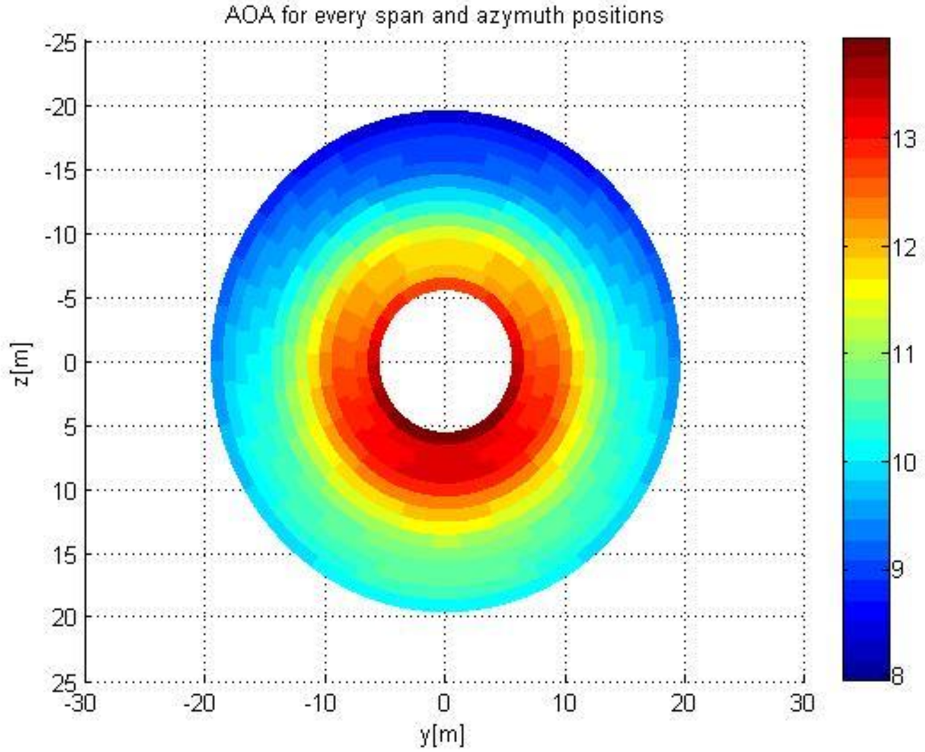


Figure 49. Angles of Attack for all Rotational Positions of the Blades.

The apparent flow velocity experienced by all the blade's sections at all positions is also of interest, because its values determine the incoming flow rescaling of the experimental sound pressure levels. The relative flow velocities are shown in Figure 50, where they are observed for all rotational positions, from an upstream view that is centered at the hub of the wind turbine. It is observed that the maximum relative flow velocity is close to 55 [m/s] at the tip of the blades, and the minimum close to 20 [m/s] at the hub.

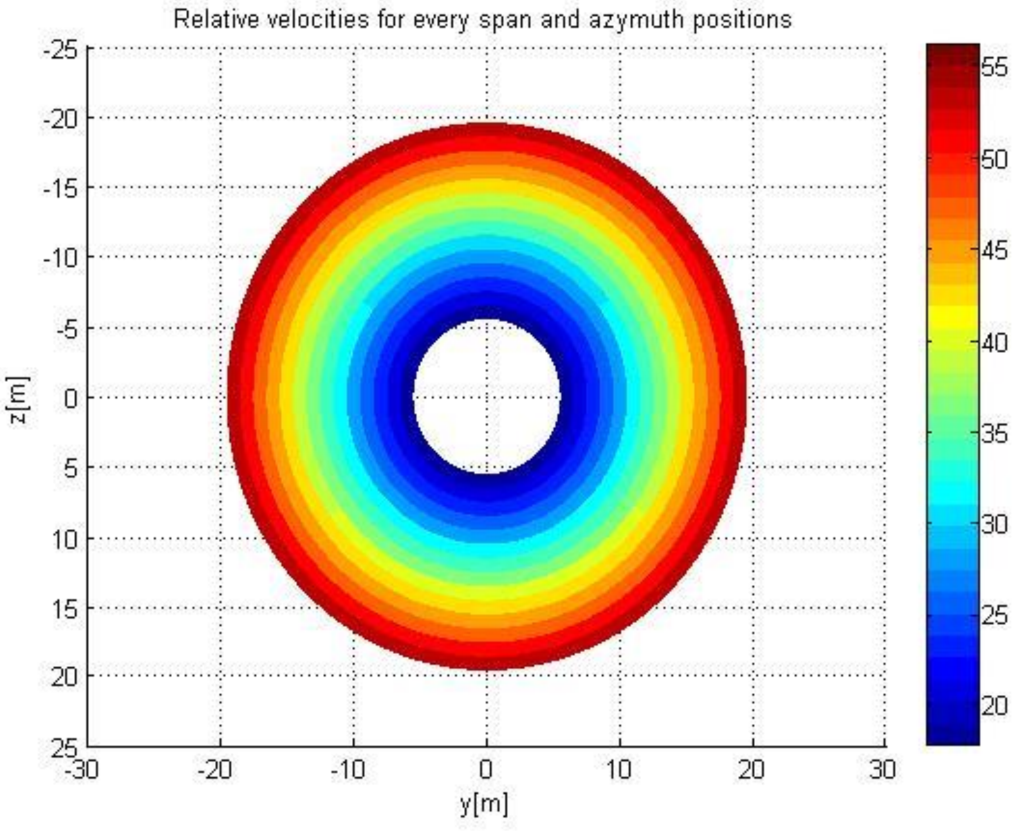


Figure 50. Apparent or Relative Flow Velocities for all Rotational Positions of the Blades.

The predicted OASPL on the sphere grid were obtained for all the sources at each blade section positions and the azimuth rotational positions of the three blades of the wind turbine. In this way, the fluctuation of the OASPL on the grid could be observed, as the blades rotate. In Figure 51 the OASPL for the first position of the three blades is observed, for a spherical grid containing 2501 points. The first position is determined as the one where one of the blades is at the top of the hub, parallel to the wind turbine tower, as seen in Figure 44. On Figure 51, it is observed that the OASPL have a maximum value of 35 [dBa] on the sphere's poles, perpendicular to the turbine's rotation plane. At the center of the sphere, close to the x-values of zero, the OASPL values have a minimum of 0 [dBa]. For all the rotational positions of the blades, the OASPL grid distribution fluctuates between these two values.

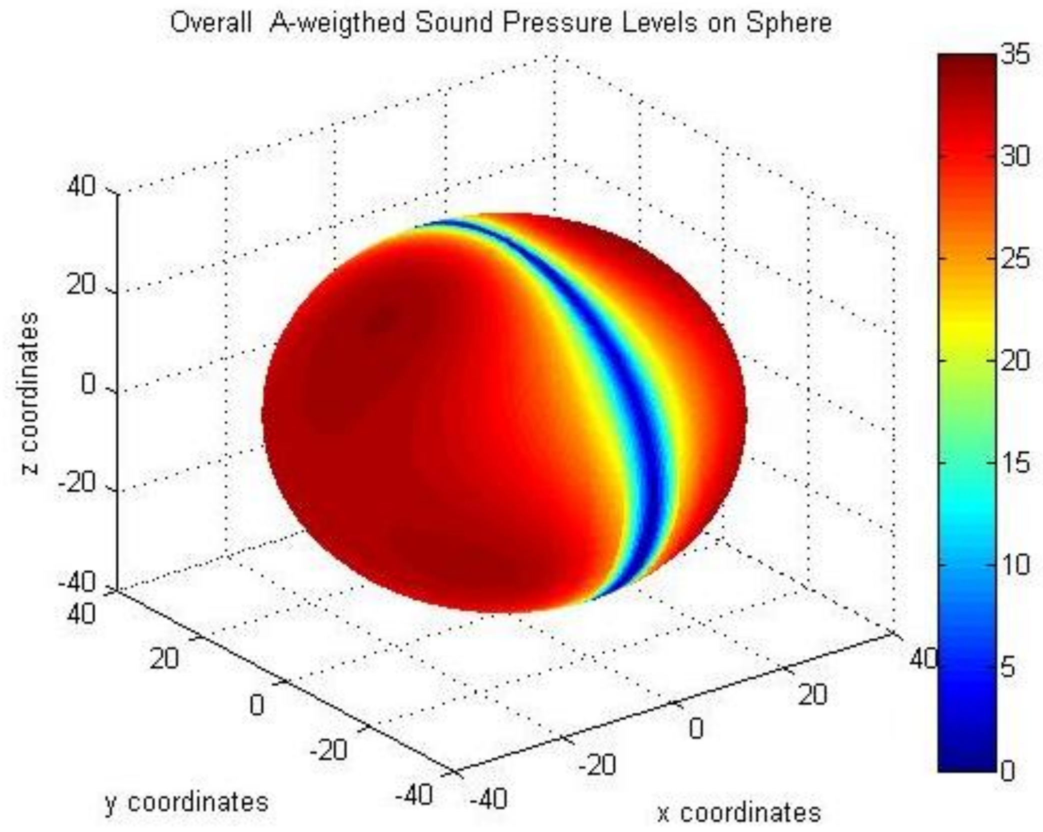


Figure 51. Predicted OASPL Over Spherical Grid around Turbine (First Position of Blades), with 2501 Grid Points.

The reason for the phenomena where there are various points on the gird with very low sound pressure level values, is determined by the directivity from all the sources towards these grid points. In these cases the unit vector $\bar{s}$ from the sources (shown in Figure 48), and the unit vector $\bar{g}$, form angles that are close to 90 degrees. This means that the value of $cos^2\theta$ from equation 5.7c is close to zero, and therefore the values of $10 * log10\left(\frac{cos^2\theta}{r^2}\right)$ are negative. This results in OASPL values that are close to zero when the summations on section 5.1.6 are performed.

The number of grid points is very important so more accurate results are obtained. For example, in Figure 52, the OASPL grid had 121 points and it was for the first position of the blades. It is observed that the OASPL values that are very low, are only positioned at the sphere upper and lower poles. This is because a spherical grid has more points on these poles, as observed in Figure 45, and on other parts of the sphere, the interpolated values are not as accurate. Therefore, a larger number of grid points of 2501 was used to obtain more accurate results, as shown in Figure 51.

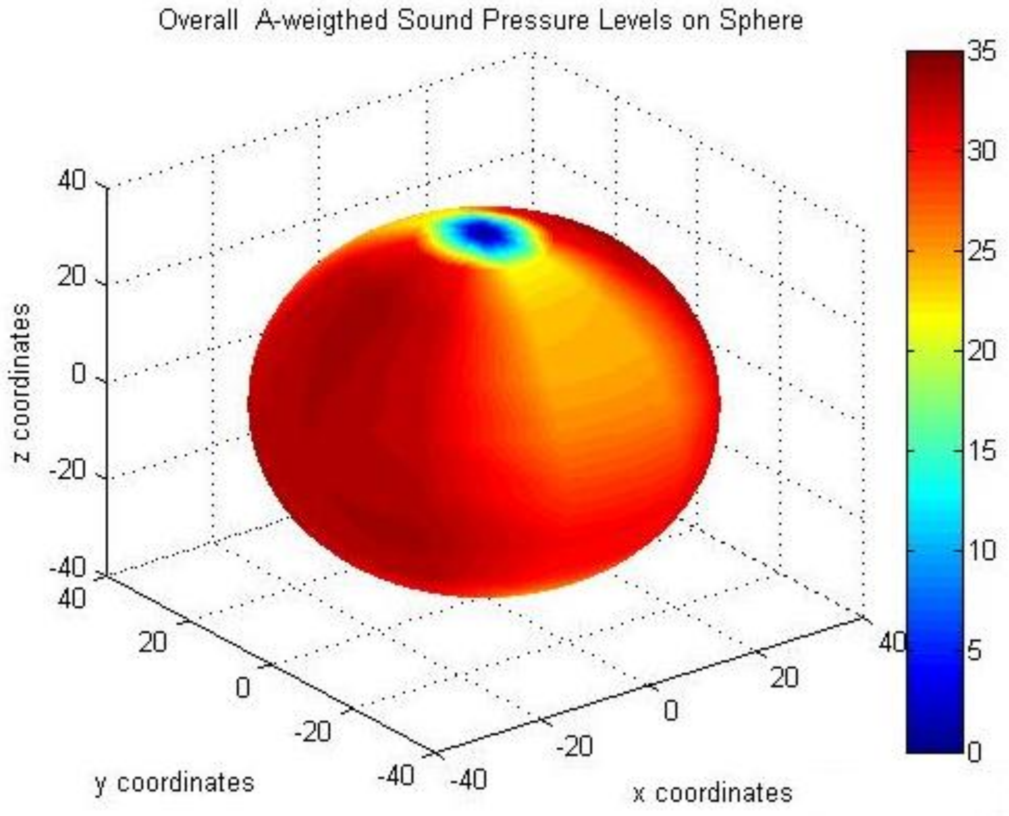


Figure 52. Predicted OASPL Over Spherical Grid around Turbine (First Position of Blades), with 121 Grid Points.

Additionally, a video was constructed with the OASPL distribution over the sphere grid, as the turbine's blades rotate.

5.3 WTNOISE_main.m MATLAB Code Description

The routine *WTNOISE_main.m* is contained in the appendix of this thesis. This code consists of 18 sub-routines, as shown in the flowchart on Figure 52. The first sub-routine is *WTNOISE_input.m*, and the objective of it is to read and save the input variables from three *.txt* input files. The first file is called *weather_parameter.txt*, and it there is contained wind speed, temperature, pressure, density and relative humidity, as a function of the height above the ground. From this file, only the wind speed was used. The second file is called *general_parameters.txt*, and in it there are general variables from the wind turbine such as hub height, pitch of the blades, number of blades, rotational blade speed, cut in wind speed, and cut out wind speed. The third file is called *blade_parameters.txt*, and it contains the length of the blades, the chord distribution, and the twist distribution along the blade. The blade parameters and general parameters from the Nordtank NTK 500/41 wind turbine were obtained from the book *Aerodynamics of Wind Turbines*, by Martin O. L. Hansen.

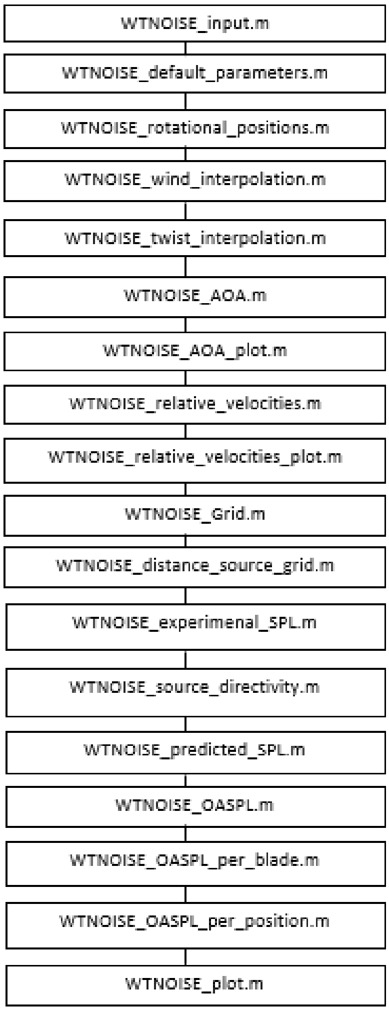

WTNOISE_input.m
WTNOISE_default_parameters.m
WTNOISE_rotational_positions.m
WTNOISE_wind_interpolation.m
WTNOISE_twist_interpolation.m
WTNOISE_AOA.m
WTNOISE_AOA_plot.m
WTNOISE_relative_velocities.m
WTNOISE_relative_velocities_plot.m
WTNOISE_Grid.m
WTNOISE_distance_source_grid.m
WTNOISE_experimenal_SPL.m
WTNOISE_source_directivity.m
WTNOISE_predicted_SPL.m
WTNOISE_OASPL.m
WTNOISE_OASPL_per_blade.m
WTNOISE_OASPL_per_position.m
WTNOISE_plot.m

Figure 53. WTNOISE_main.m flowchart.

The second sub-routine is called *WTNOISE_default_parameters.m,* and it defines default variables that are needed for the noise computations. These are, the air density, speed of sound on air, reference pressure, and all 1/3rd octave frequency bands. The third sub-routine is called *WTNOISE_rotational_positions.m,* and what it does is the computation of the locations of each blade sections, for all azimuth rotational positions, as well as their corresponding height and rotation velocity.

The fourth sub-routine is called *WTNOISE_wind_interpolation.m*, and what it does is to interpolate the wind velocity values for all the heights of the sources corresponding to each blade section rotational position, by using the input wind profile data. The subsequent sub-routine *WTNOISE_twist_interpolation.m*, interpolates the twist for the blade sections, by using the input twist distribution for the Nordtank NTK 500/41 wind turbine.

The sixth sub-routine is called *WTNOISE_AOA.m* and it calculates the angles β, γ and angle of attack for all the blade section rotational positions. Accordingly, the sub-routine *WTNOISE_AOA_plot.m* plots the angles of attack, as observed in Figure 49. The sub-routine *WTNOISE_relative_velocities.m* computes the relative velocities, and *WTNOISE_relative_velocities_plot.m,* plots the results, as observed in Figure 50.

The sub-routine *WTNOISE_Grid.m* computes the location of all the points on the spherical grid in which the OASPL are obtained. *WTNOISE_distance_source_grid.m* determines the distance from each source to the points on the grid. This sub-routine also calculates the unit vector corresponding to the direction of the source-grid distances.

WTNOISE_experimental_SPL.m computes the sound pressure levels using equation 3.21, for all the blade segment positions for all azimuth rotational positions, and Strouhal numbers. Additionally, all the rescaling procedures described in section 5.1.4 are performed. In order to apply equation 5.7c, the directivity factor was computed for all blade segment positions for all azimuth rotational positions. This was done in the sub-routine *WTNOISE_source_directivity.m*. Finally, equation 5.7c is applied in the sub-routine *WTNOISE_predicted_SPL.m*.

WTNOISE_OASPL.m computes the overall A-weighted sound pressure levels for all the points of the grid, using equation 5.9. The sub-routine *WTNOISE_OASPL_per_blade.m* applies equation 5.10, and *WTNOISE_OASPL_per_positon.m* applies equation 5.11.

The final sub-routine is called *WTNOISE_plot.m*, and generates all the plots corresponding to the predicted OASPL produced by the three blades, as they rotate. Additionally, it creates a video that records in time all the OASPL, depending on the position of the three blades.

CHAPTER VI

CONCLUSIONS & RECOMMENDATIONS

This thesis determined the propagation of noise produced by wind turbines. To do so, the predicted overall A-weighted sound pressure levels on a grid around a Nordtank NTK 500/41 wind turbine, were computed. In order to achieve this goal, the usage of experimental data from airfoil noise tests on a wind tunnel, provided the necessary inputs for noise sources modeling. Moreover, the experimental data was compared with semi-empirical models developed by other researchers, so a better understanding of the aero-acoustic behavior of an airfoil is achieved.

The aero-acoustic noise prediction of the propagation of noise produced by a wind turbine was successfully obtained. Even though many other factors have to be added to the model, so more realistic results are obtained, the empirical airfoil sound pressure level data along with theoretical procedures, have determined very accurate results.

The data obtained from airfoil wind tunnel experimentation was able to be processed, and a useful sound pressure level equation, only dependent on the angle of attack and Strouhal number, was obtained. This straightforward equation is crucial because it effectively demonstrated how it can be applied to a modeled dipole noise source, and simplify unknown terms such as the source's noise power.

On the other hand, a comparison of modeled data by other researchers and the experimental data, permitted the development of certain conclusions. It was resolved that all the noise sources explained by Moriarty and Migliore were thoroughly determined, and close to real physical phenomena. Nevertheless, the expressions used to obtain the sound pressure levels of the noise caused by the sources, relies on semi-empirical methods. This causes inaccuracies at certain frequencies, for some the angles of attack and relative inflow velocities.

The characteristic trailing edge noise source used for a wind turbine airfoil, was a dipole. It allowed the prediction of overall A-weighted sound pressure levels on a spherical grid. The maximum sound pressure levels obtained were parallel to the plane of rotation of the turbine, and that the minimum perpendicular to it. This behavior pattern was determined to be characteristic of a wind turbine. Furthermore, as the blades rotate, an oscillating effect was observed, proving the existence of the beating character of the sound that causes high annoyance levels for human beings. The frequency of the beating character was determined to be dependent of the rotation velocity of the blades.

Even though the maximum overall sound pressure levels were of approximately 35 [dBa], which is not a considerable amount of noise, the overall sound pressure levels produced by a wind power generation farm would be higher. Additionally, the wind turbine used for the model prediction was relatively not as large as new modern wind turbines, which would also produce higher sound pressure levels.

From the obtained results, it could also be concluded that if a small number of points are used in the grid, the prediction results will not be accurate. Therefore, a relatively large number of points on the grid should be used to guarantee a more accurate model.

Finally, even though the model developed is fairly accurate, other variables that are present in reality should be taken into account for future research. These include, noise ground reflections, atmospheric attenuation on the sound pressure levels, deflection of the turbine’s blades (by not assuming the blades are rigid bodies), vibration analysis on the blades ,and determining how the Doppler effect (apparent change of frequency of a wave for an observer moving relative to the source) affects the sound pressure levels prediction.

REFERENCES

Technial University of Denmark. (2015, 04 19). *BECAS*. Retrieved from http://www.becas.dtu.dk/About

Amiet, R. (1975). Acoustic radiation from an airfoil in a turbulent stream. *Journal of Sound and Vibration, 41*(4), 407-420.

Bader, R. (2014). Microphone Array. In T. Rossing, *Springer Handbook of Acoustics* (pp. 1179-1204). Stanford, California: Springer-Verlag Berlin Heidelberg.

Barron, M. (2003). *Industrial Noise Control and Acoustics.* New York, NY: Marcel Dekker, INC.

Bell, L., & Bell, D. (1994). *Industrial Noise Control: Fundamentals and Applications.* New York, New York: Marcel Dekker, INC.

Bies, D., & Hansen, C. (2009). *Engineering Noise Control: Theory and Practice* (4th ed.). New York, NY, USA: Taylor and Francis.

Bowdler, D., & Leventhall, G. (2011). *Wind Turbine Noise.* Essex, UK: Multi-Science Publishing Co. Ltd.

Brooks, T., Pope, D., & Marcolini, M. (1989). *Airfoil Self-Noise Prediction.* NASA Reference Publication 1218.

Davenport, W., Burdisso, R., Camargo, H., Crede, E., Remillieux, M., Rasnick, M., & Van Seeters, P. (2010). Aeroacoustic Testing of Wind Turbine Airfoils. *Subcontract Report NREL/SR-500-43471*.

Gipe, P. (2004). *Wind Power: Renewable Energy for Home, Farm, and Buisness.* White River Junction, VT: Chelsea Green Pub. Co.

Global Wind Energy Council. (2015). *Global Wind Statistics 2014.* Burssels, Belgium: Global Wind Energy Council.

Granger, R. (1995). *Fluid Mechanics.* Mineola, NY: Courier Corporation.

Hansen, M. (2008). *Aerodynamics of Wind Turbines* (2nd edition ed.). Sterling, VA, USA: EARTHSACAN.

Hau, E. (2013). *Wind Turbines: Fundamentals, Technologies, Application, Economics.* New York, NY: Springer.

Jablonowski, C. (2015, 04 19). *Idealized Dynamical Core Test Cases for Weather and Climate Models*. Retrieved from Test of the Dynamical Core of General Circulation

Models: Short deterministic and long climate simulations: http://www-personal.umich.edu/~cjablono/dycore_test_suite.html

Kleiner, M., & Tichy, J. (2014). *Acoustics of Small Rooms.* Boca Raton, Florida: CRC Press.

Kumar, A., Dwivedi, A., Paliwal, V., & Patil, P. (2014). Free Vibration Analysis of AL 2024 Wind Turbine Blade Designed for Uttarakhand Region Based on FEA. *Procedia Technology*, 14 (2014), 336-347.

Larson, R., & Edwards, B. (2010). *Calculus Multivariable.* Cengage Learning Inc. : Belmont, CA. USA.

Lowson, M. (1993). *Assesment and Prediction of Wind Turbine Noise.* Bristol, England: Department of Trade and Industry W/13/00284/REP.

Makarewicz, R. (2013). The influence of refractipon on turbine noise. *International journal of aeroacoustics*, 12 (4), 349-362.

Makarewicz, R., & Golebiewski, R. (2014). The partially ensonified zone of wind turbine noise. *Journal of Wind Engineering and Industrial Aerodynamics*, 132 (2014), 49-53.

Mollasalehi, E., Qiao, S., & Wood, D. (2013). Contribution of Small Wind Turbine Structural Vibration to Noise Emission. *Energies (19961073)*, 6(8), 3669-3691. doi: 10.3390/en6083669.

Moriarty, P. (2005). *NAFNoise User's Guide.* Golden, Colorado: National Wind Technology Center, National Renewable Energy Laboratory.

Moriarty, P., & Migliore, P. (2003). *Semi-Empirical Aeroacoustic Noise Prediction Code for Wind Turbines.* Golden, Colorado: National Renewable Energy Laboratiry, NREL/TP-500-34478.

Moriarty, P., Guidati, G., & Migliore, P. (2005). Prediction of Turbulent Inflow and Trailing Edge Noise for Wind Turbines. *11 th AIAA/CEAS Aeroacoustics Conference (26th AIAA Aeroacoustics Conference)* (pp. 2005-2881). Monterey, California: AIAA.

Munson, B., Okiishi, T. H., & Rothmayer, A. (2013). *Fundamentals of Fluid Mechanics.* Hoboken, NJ: John Wiley & Sons.

National Renewabl Energy Laboratory . (2015, 04 19). *Dynamic Maps, GIS Data, & Analysis Tools*. Retrieved from http://www.nrel.gov/gis/wind.html

Norton, M., & Karczub, D. (2003). *Fundamentals of Noise and Vibration Analysis for Engineers.* Cambridge, UK: Cambridge University Press.

Oerlemans, S., & Schepers, J. (2009). Prediction of wind turbine noise and validation against experiment. *International Journal of Aeroacoustics*, 8(6), 555-584.

Pierce, A. (2014). Basic Linear Acoustics. In T. Rossing, *Springer Handbook of Acoustics* (pp. 29-112). Stanford, California: Springer-Verlag Berlin Heidelberg.

Randall, R. (2005). *An Introduction to Acoustics.* Mineola, New York: Dover Publications INC.

Rao, C. (2006). *Enviromental Pollution Control Engineering.* New Delhi, India: New Age International Publishers.

Ryi, J., Choi, J., Lee, S., & Lee, S. (2014). A full-scale prediction method for wind turbine rotor noise by using wind tunnel test data. *Renewable Energy*, 65 (2014), 257-264.

Timmer, W., & vanRoiij, R. (2001). Some aspects of high angle-of-attac flow on airfoils for wind turbine application. *DUWIND, the Delft University Wind Energy Reseacrh Institute*.

Tonin, R. (2012). SOURCES OF WIND TURBINE NOISE AND SOUND PROPAGATION. *Acoustics Australia*, 40(1), 20-27.

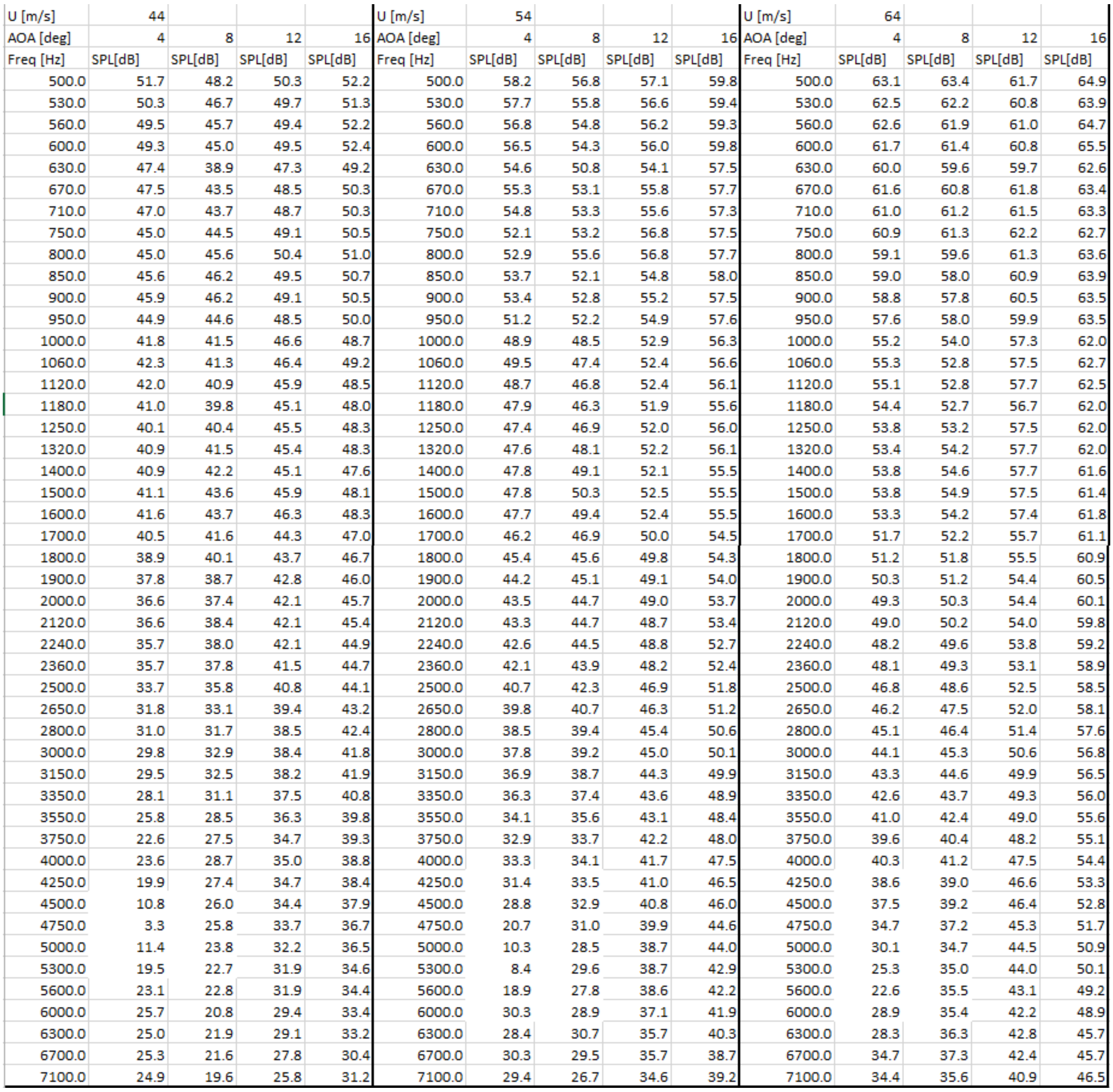
U.S. Energy Information Administration. (2014). *Levelized Cost and Levelized Avoided Cost of New Generation Resources in the Annual Energy Outlook 2014.* U.S. Energy Information Administration.

van Rooij, R. (2001, October). Terminology, Refernce Systems and Conventions. *Duwind 2001.004.*

Virginia Polytechnic Institute and State University. (2015, February 11). *Virginia Tech Stability Wind Tunnel*. Retrieved from http://www.aoe.vt.edu/research/facilities/stabilitytunnel/index.html

Wagner, S., Bareib, R., & Guidati, G. (1996). *Wind Turbine Noise.* Stuttgart, Germany: Springer-Verlag Berlin Heidelberg.

APPENDIX A: DU96.xlsx File



U [m/s]	44				U [m/s]	54				U [m/s]	64			
AOA [deg]	4	8	12	16	AOA [deg]	4	8	12	16	AOA [deg]	4	8	12	16
Freq [Hz]	SPL[dB]	SPL[dB]	SPL[dB]	SPL[dB]	Freq [Hz]	SPL[dB]	SPL[dB]	SPL[dB]	SPL[dB]	Freq [Hz]	SPL[dB]	SPL[dB]	SPL[dB]	SPL[dB]
500.0	51.7	48.2	50.3	52.2	500.0	58.2	56.8	57.1	59.8	500.0	63.1	63.4	61.7	64.9
530.0	50.3	46.7	49.7	51.3	530.0	57.7	55.8	56.6	59.4	530.0	62.5	62.2	60.8	63.9
560.0	49.5	45.7	49.4	52.2	560.0	56.8	54.8	56.2	59.3	560.0	62.6	61.9	61.0	64.7
600.0	49.3	45.0	49.5	52.4	600.0	56.5	54.3	56.0	59.8	600.0	61.7	61.4	60.8	65.5
630.0	47.4	38.9	47.3	49.2	630.0	54.6	50.8	54.1	57.5	630.0	60.0	59.6	59.7	62.6
670.0	47.5	43.5	48.5	50.3	670.0	55.3	53.1	55.8	57.7	670.0	61.6	60.8	61.8	63.4
710.0	47.0	43.7	48.7	50.3	710.0	54.8	53.3	55.6	57.3	710.0	61.0	61.2	61.5	63.3
750.0	45.0	44.5	49.1	50.5	750.0	52.1	53.2	56.8	57.5	750.0	60.9	61.3	62.2	62.7
800.0	45.0	45.6	50.4	51.0	800.0	52.9	55.6	56.8	57.7	800.0	59.1	59.6	61.3	63.6
850.0	45.6	46.2	49.5	50.7	850.0	53.7	52.1	54.8	58.0	850.0	59.0	58.0	60.9	63.9
900.0	45.9	46.2	49.1	50.5	900.0	53.4	52.8	55.2	57.5	900.0	58.8	57.8	60.5	63.5
950.0	44.9	44.6	48.5	50.0	950.0	51.2	52.2	54.9	57.6	950.0	57.6	58.0	59.9	63.5
1000.0	41.8	41.5	46.6	48.7	1000.0	48.9	48.5	52.9	56.3	1000.0	55.2	54.0	57.3	62.0
1060.0	42.3	41.3	46.4	49.2	1060.0	49.5	47.4	52.4	56.6	1060.0	55.3	52.8	57.5	62.7
1120.0	42.0	40.9	45.9	48.5	1120.0	48.7	46.8	52.4	56.1	1120.0	55.1	52.8	57.7	62.5
1180.0	41.0	39.8	45.1	48.0	1180.0	47.9	46.3	51.9	55.6	1180.0	54.4	52.7	56.7	62.0
1250.0	40.1	40.4	45.5	48.3	1250.0	47.4	46.9	52.0	56.0	1250.0	53.8	53.2	57.5	62.0
1320.0	40.9	41.5	45.4	48.3	1320.0	47.6	48.1	52.2	56.1	1320.0	53.4	54.2	57.7	62.0
1400.0	40.9	42.2	45.1	47.6	1400.0	47.8	49.1	52.1	55.5	1400.0	53.8	54.6	57.7	61.6
1500.0	41.1	43.6	45.9	48.1	1500.0	47.8	50.3	52.5	55.5	1500.0	53.8	54.9	57.5	61.4
1600.0	41.6	43.7	46.3	48.3	1600.0	47.7	49.4	52.4	55.5	1600.0	53.3	54.2	57.4	61.8
1700.0	40.5	41.6	44.3	47.0	1700.0	46.2	46.9	50.0	54.5	1700.0	51.7	52.2	55.7	61.1
1800.0	38.9	40.1	43.7	46.7	1800.0	45.4	45.6	49.8	54.3	1800.0	51.2	51.8	55.5	60.9
1900.0	37.8	38.7	42.8	46.0	1900.0	44.2	45.1	49.1	54.0	1900.0	50.3	51.2	54.4	60.5
2000.0	36.6	37.4	42.1	45.7	2000.0	43.5	44.7	49.0	53.7	2000.0	49.3	50.3	54.4	60.1
2120.0	36.6	38.4	42.1	45.4	2120.0	43.3	44.7	48.7	53.4	2120.0	49.0	50.2	54.0	59.8
2240.0	35.7	38.0	42.1	44.9	2240.0	42.6	44.5	48.8	52.7	2240.0	48.2	49.6	53.8	59.2
2360.0	35.7	37.8	41.5	44.7	2360.0	42.1	43.9	48.2	52.4	2360.0	48.1	49.3	53.1	58.9
2500.0	33.7	35.8	40.8	44.1	2500.0	40.7	42.3	46.9	51.8	2500.0	46.8	48.6	52.5	58.5
2650.0	31.8	33.1	39.4	43.2	2650.0	39.8	40.7	46.3	51.2	2650.0	46.2	47.5	52.0	58.1
2800.0	31.0	31.7	38.5	42.4	2800.0	38.5	39.4	45.4	50.6	2800.0	45.1	46.4	51.4	57.6
3000.0	29.8	32.9	38.4	41.8	3000.0	37.8	39.2	45.0	50.1	3000.0	44.1	45.3	50.6	56.8
3150.0	29.5	32.5	38.2	41.9	3150.0	36.9	38.7	44.3	49.9	3150.0	43.3	44.6	49.9	56.5
3350.0	28.1	31.1	37.5	40.8	3350.0	36.3	37.4	43.6	48.9	3350.0	42.6	43.7	49.3	56.0
3550.0	25.8	28.5	36.3	39.8	3550.0	34.1	35.6	43.1	48.4	3550.0	41.0	42.4	49.0	55.6
3750.0	22.6	27.5	34.7	39.3	3750.0	32.9	33.7	42.2	48.0	3750.0	39.6	40.4	48.2	55.1
4000.0	23.6	28.7	35.0	38.8	4000.0	33.3	34.1	41.7	47.5	4000.0	40.3	41.2	47.5	54.4
4250.0	19.9	27.4	34.7	38.4	4250.0	31.4	33.5	41.0	46.5	4250.0	38.6	39.0	46.6	53.3
4500.0	10.8	26.0	34.4	37.9	4500.0	28.8	32.9	40.8	46.0	4500.0	37.5	39.2	46.4	52.8
4750.0	3.3	25.8	33.7	36.7	4750.0	20.7	31.0	39.9	44.6	4750.0	34.7	37.2	45.3	51.7
5000.0	11.4	23.8	32.2	36.5	5000.0	10.3	28.5	38.7	44.0	5000.0	30.1	34.7	44.5	50.9
5300.0	19.5	22.7	31.9	34.6	5300.0	8.4	29.6	38.7	42.9	5300.0	25.3	35.0	44.0	50.1
5600.0	23.1	22.8	31.9	34.4	5600.0	18.9	27.8	38.6	42.2	5600.0	22.6	35.5	43.1	49.2
6000.0	25.7	20.8	29.4	33.4	6000.0	30.3	28.9	37.1	41.9	6000.0	28.9	35.4	42.2	48.9
6300.0	25.0	21.9	29.1	33.2	6300.0	28.4	30.7	35.7	40.3	6300.0	28.3	36.3	42.8	45.7
6700.0	25.3	21.6	27.8	30.4	6700.0	30.3	29.5	35.7	38.7	6700.0	34.7	37.3	42.4	45.7
7100.0	24.9	19.6	25.8	31.2	7100.0	29.4	26.7	34.6	39.2	7100.0	34.4	35.6	40.9	46.5

APPENDIX B: DU96_main.m

```
%% DU96_input.m
% This routine reads the input to the code from an excel file
%
% Authors: Sterling McBride
%          Ricardo Burdisso
%
% DATE:June 2014
%
% INPUTS:
%
%   DU96.xslx
%
% OUTPUTS:
%
%   U_1_data = Frequency and SPL for each AOA for a flow velocity of 34 m/s
%   U_2_data = Frequency and SPL for each AOA for a flow velocity of 44 m/s
%   U_3_data = Frequency and SPL for each AOA for a flow velocity of 54 m/s
%   U_4_data = Frequency and SPL for each AOA for a flow velocity of 64 m/s
%   U_1 = First velocity of 34 m/s
%   U_2 = First velocity of 44 m/s
%   U_3 = First velocity of 54 m/s
%   U_4 = First velocity of 64 m/s
%   AOA_1_vec = Vector containing an angle of attack of 4 degrees
%   AOA_2_vec = Vector containing an angle of attack of 8 degrees
%   AOA_3_vec = Vector containing an angle of attack of 12 degrees
%   AOA_4_vec = Vector containing an angle of attack of 16 degrees
%   U_ref = reference flow velocity;
%   chord = chord for the airfoil;
%   Span = Span used;
%----------------------------------------------------------------------
% Reference and geometric values

U_ref = 1;
chord = 0.9;
Span = 1.8288;  %6 feet
Sm = 1;
distance = 1.6;
ref_distance = 1;

%-------------------------------------------------------------------------
% Data extraction from the excel file
% SPL and Frequency
filename = 'DU96.xlsx';
sheet = 1;


xlRange_1 = 'A4:E50';
U_1_data = xlsread(filename, sheet, xlRange_1);
xlRange_2 = 'F4:J50';
U_2_data = xlsread(filename, sheet, xlRange_2);
xlRange_3 = 'K4:O50';
U_3_data = xlsread(filename, sheet, xlRange_3);
xlRange_4 = 'P4:T50';
U_4_data = xlsread(filename, sheet, xlRange_4);
```

```
%---------------------------------------------------------
% Data extraction from the excel file
% Flow velocities

xlRange_5 = 'B1';
U_1 = xlsread(filename, sheet, xlRange_5);
xlRange_6 = 'G1';
U_2 = xlsread(filename, sheet, xlRange_6);
xlRange_7 = 'L1';
U_3 = xlsread(filename, sheet, xlRange_7);
xlRange_8 = 'Q1';
U_4 = xlsread(filename, sheet, xlRange_8);


%-----------------------------------------------------------
% Data extraction from the excel file
% Angles of attack

xlRange_9 = 'B2';
AOA_1 = xlsread(filename, sheet, xlRange_9);
xlRange_10 = 'C2';
AOA_2 = xlsread(filename, sheet, xlRange_10);
xlRange_11 = 'D2';
AOA_3 = xlsread(filename, sheet, xlRange_11);
xlRange_12 = 'E2';
AOA_4 = xlsread(filename, sheet, xlRange_12);

AOA_1_vec = ones(length(U_1_data),1)*AOA_1;
AOA_2_vec = ones(length(U_2_data),1)*AOA_2;
AOA_3_vec = ones(length(U_3_data),1)*AOA_3;
AOA_4_vec = ones(length(U_4_data),1)*AOA_4;
%----------------------------------------------------------------------
% End of DU96_input.m
%----------------------------------------------------------------------
```

```
%% DU96_STROUHAL.m
% This routine calculates the Strouhal number for every
%
% Authors: Sterling McBride
%          Ricardo Burdisso
%
% DATE:June 2014
%
% INPUTS:
%
%   chord = airfoil chord
%   U_1_data = Frequencies used
%   U_1 = flow velocities used
%
% OUTPUTS:
%   st_1 = Strouhal numbers when flow velocity is 34 m/s
%   st_2 = Strouhal numbers when flow velocity is 44 m/s
%   st_3 = Strouhal numbers when flow velocity is 54 m/s
%   st_4 = Strouhal numbers when flow velocity is 64 m/s
%----------------------------------------------------------------------
% Strouhal number calculations

for n=1:length(U_1_data)
st_1(n)=chord*U_1_data(n,1)/U_1;
end

for n=1:length(U_2_data)
st_2(n)=chord*U_2_data(n,1)/U_2;
end

for n=1:length(U_3_data)
st_3(n)=chord*U_3_data(n,1)/U_3;
end

for n=1:length(U_4_data)
st_4(n)=chord*U_4_data(n,1)/U_4;
end

%----------------------------------------------------------------------
% End of DU96_STROUHAL.m
%----------------------------------------------------------------------
```

```
%% DU96_Uref_Correction.m
% This routine does the SPL correction using the a refrence flow velocity
%
% Authors: Sterling McBride
%          Ricardo Burdisso
%
% DATE:June 2014
%
% INPUTS:
%
%   U_ref = reference velocity
%   U_1_data = Frequencies used
%   U_1 = flow velocities used
%
% OUTPUTS:
%   SPL_1_1 = SPL normalized using U_ref when the flow velocity is 34 m/s
%   SPL_1_2 = SPL normalized using U_ref when the flow velocity is 44 m/s
%   SPL_1_3 = SPL normalized using U_ref when the flow velocity is 54 m/s
%   SPL_1_4 = SPL normalized using U_ref when the flow velocity is 64 m/s
%----------------------------------------------------------------
%   Normalization procedure
%
%   Loop over every SPL for a flow velocity of 34 m/s
power_law = 5;
power_law = power_law*10;
for m = 2:size(U_1_data,2)
  for l = 1: size(U_1_data,1)
      SPL_1_1(l,m-1) = U_1_data(l,m)-power_law*log10(U_1/U_ref);
  end
end

%   Loop over every SPL for a flow velocity of 44 m/s
for m = 2:size(U_2_data,2)
  for l = 1: size(U_2_data,1)
      SPL_1_2(l,m-1) = U_2_data(l,m)-power_law*log10(U_2/U_ref);
  end
end

%   Loop over every SPL for a flow velocity of 54 m/s
for m = 2:size(U_3_data,2)
  for l = 1: size(U_3_data,1)
      SPL_1_3(l,m-1) = U_3_data(l,m)-power_law*log10(U_3/U_ref)
  end
end

%   Loop over every SPL for a flow velocity of 64 m/s
for m = 2:size(U_4_data,2)
  for l = 1: size(U_4_data,1)
      SPL_1_4(l,m-1) = U_4_data(l,m)-power_law*log10(U_4/U_ref);
  end
end
%----------------------------------------------------------------
% End of DU96_Uref_Correction.m
%----------------------------------------------------------------
```

```
%% DU96_Span_Correction.m
% This routine does the SPL correction using the a refrence flow velocity
%
% Authors: Sterling McBride
%          Ricardo Burdisso
%
% DATE:June 2014
%
% INPUTS:
%
%   SPL_1_1 = SPL normalized using U_ref when the flow velocity is 34 m/s
%   SPL_1_2 = SPL normalized using U_ref when the flow velocity is 44 m/s
%   SPL_1_3 = SPL normalized using U_ref when the flow velocity is 54 m/s
%   SPL_1_4 = SPL normalized using U_ref when the flow velocity is 64 m/s
%
% OUTPUTS:
%   SPL_4deg = SPL corrected using the Span when the flow velocity is 34
m/s
%   SPL_8deg = SPL corrected using the Span when the flow velocity is 44
m/s
%   SPL_12deg = SPL corrected using the Span when the flow velocity is 54
m/s
%   SPL_16deg = SPL corrected using the Span when the flow velocity is 64
m/s
%
%---------------------------------------------------------------------------
%   correction procedure
%   Loop over every SPL for a flow velocity of 34 m/s
for m = 2:size(U_1_data,2)
  for l = 1: size(U_1_data,1)
      SPL_4deg(l,m-1) = SPL_1_1(l,m-1)-10*log10(Span/Sm);
  end
end

%   Loop over every SPL for a flow velocity of 44 m/s
for m = 2:size(U_1_data,2)
  for l = 1: size(U_1_data,1)
      SPL_8deg(l,m-1) = SPL_1_2(l,m-1)-10*log10(Span/Sm);
  end
end
%   Loop over every SPL for a flow velocity of 54 m/s
for m = 2:size(U_1_data,2)
  for l = 1: size(U_1_data,1)

      SPL_12deg(l,m-1) = SPL_1_3(l,m-1)-10*log10(Span/Sm);
  end
end

%   Loop over every SPL for a flow velocity of 64 m/s
for m = 2:size(U_1_data,2)
  for l = 1: size(U_1_data,1)

      SPL_16deg(l,m-1) = SPL_1_4(l,m-1)-10*log10(Span/Sm);
  end
end
%---------------------------------------------------------------------------
% End of DU96_Span_Correction.m
%-------------------------------------------------------------------------
```

```
%% DU96_distance_Correction.m
% This routine does the SPL correction using the a refrence flow velocity
%
% Authors: Sterling McBride
%          Ricardo Burdisso
%
% DATE:June 2014
%
% INPUTS:
%
%   SPL_4deg = SPL corrected using the Span when the flow velocity is 34
m/s
%   SPL_8deg = SPL corrected using the Span when the flow velocity is 44
m/s
%   SPL_12deg = SPL corrected using the Span when the flow velocity is 54
m/s
%   SPL_16deg = SPL corrected using the Span when the flow velocity is 64
m/s
%
% OUTPUTS:
%
%   SPL_1 = SPL corrected using the Span when the flow velocity is 34 m/s
%   SPL_2 = SPL corrected using the Span when the flow velocity is 44 m/s
%   SPL_3 = SPL corrected using the Span when the flow velocity is 54 m/s
%   SPL_4 = SPL corrected using the Span when the flow velocity is 64 m/s
%-------------------------------------------------------------------------
%   correction procedure
%   Loop over every SPL for a flow velocity of 34 m/s
for m = 2:size(U_1_data,2)
  for l = 1: size(U_1_data,1)
      SPL_1(l,m-1) = SPL_4deg(l,m-1)-20*log10(distance/ref_distance);
  end
end

%   Loop over every SPL for a flow velocity of 44 m/s
for m = 2:size(U_1_data,2)
  for l = 1: size(U_1_data,1)

      SPL_2(l,m-1) = SPL_8deg(l,m-1)-20*log10(distance/ref_distance);
  end
end

%   Loop over every SPL for a flow velocity of 54 m/s
for m = 2:size(U_1_data,2)
  for l = 1: size(U_1_data,1)
      SPL_3(l,m-1) = SPL_12deg(l,m-1)-20*log10(distance/ref_distance)
  end
end
%   Loop over every SPL for a flow velocity of 64 m/s
for m = 2:size(U_1_data,2)
  for l = 1: size(U_1_data,1)

     SPL_4(l,m-1) = SPL_16deg(l,m-1)-20*log10(distance/ref_distance);
  end
end
%-------------------------------------------------------------------------
% End of DU96_distance_correction.m
%-------------------------------------------------------------------------
```

```
%% DU96_Plots.m
% This routine plots the SPL as a function of the Strouhal number
%
% Authors: Sterling McBride
%          Ricardo Burdisso
%
% DATE:June 2014
%
% INPUTS:
%
%   SPL_1 = Corrected SPL when flow speed is 34 m/s
%   SPL_2 = Corrected SPL when flow speed is 44 m/s
%   SPL_3 = Corrected SPL when flow speed is 54 m/s
%   SPL_4 = Corrected SPL when flow speed is 64 m/s
%
% OUTPUT:
%
%  plots of the SPL as a function of the Strouhal number for each AOA and
%  flow velocity
%
% ROUTINES USED: none
%
%--------------------------------------------------------------------------
% Create a folder in the directory
mkdir('SPL(corrected) vs Strouhal')
%--------------------------------------------------------------------------
%Plot the SPL for AOA of 4 degrees as a function of the stouhal number for
all wind speeds
figure(1);
st_total_1 = [st_1 st_2 st_3 st_4];
SPL_inv_1 = [SPL_1(:,1); SPL_2(:,1); SPL_3(:,1); SPL_4(:,1)];
SPL_total_1 = SPL_inv_1';
scatter(st_1,SPL_1(:,1),'r','*')
hold on
scatter(st_2,SPL_2(:,1),'b','*')
hold on
scatter(st_3,SPL_3(:,1),'g','*')
hold on
scatter(st_4,SPL_4(:,1),'k','*')
title('SPL corrected vs. Strouhal for AOA of 4 degrees');
xlabel('Strouhal number');
ylabel('SPL[dB]');
grid on
hleg1 = legend('34 m/s','44 m/s','54 m/s','64 m/s');
saveas(figure(1),[pwd '/SPL(corrected) vs Strouhal/AOA 4.fig']);

%Plot the SPL for AOA of 8 degrees as a function of the stouhal number for
all wind speeds
figure(2);
st_total_2 = [st_1 st_2 st_3 st_4];
SPL_inv_2 = [SPL_1(:,2); SPL_2(:,2); SPL_3(:,2); SPL_4(:,2)];
SPL_total_2 = SPL_inv_2';
scatter(st_1,SPL_1(:,2),'r','*')
hold on
scatter(st_2,SPL_2(:,2),'b','*')
hold on
scatter(st_3,SPL_3(:,2),'g','*')
hold on
```

```
scatter(st_4,SPL_4(:,2),'k','*')
title('SPL corrected vs. Strouhal for AOA of 8 degrees');
xlabel('Strouhal number');
ylabel('SPL[dB]');
grid on
hleg2 = legend('34 m/s','44 m/s','54 m/s','64 m/s');
saveas(figure(2),[pwd '/SPL(corrected) vs Strouhal/AOA 8.fig']);

%Plot the SPL for AOA of 12 degrees as a function of the stouhal number for
all wind speeds
figure(3);
st_total_3 = [st_1 st_2 st_3 st_4];
SPL_inv_3 = [SPL_1(:,3); SPL_2(:,3); SPL_3(:,3); SPL_4(:,3)];
SPL_total_3 = SPL_inv_3';
scatter(st_1,SPL_1(:,3),'r','*')
hold on
scatter(st_2,SPL_2(:,3),'b','*')
hold on
scatter(st_3,SPL_3(:,3),'g','*')
hold on
scatter(st_4,SPL_4(:,3),'k','*')
title('SPL corrected vs. Strouhal for AOA of 12 degrees');
xlabel('Strouhal number');
ylabel('SPL[dB]');
grid on
hleg3 = legend('34 m/s','44 m/s','54 m/s','64 m/s');
saveas(figure(3),[pwd '/SPL(corrected) vs Strouhal/AOA 12.fig']);

%Plot the SPL for AOA of 16 degrees as a function of the stouhal number for
all wind speeds
figure(4);
st_total_4 = [st_1 st_2 st_3 st_4];
SPL_inv_4 = [SPL_1(:,4); SPL_2(:,4); SPL_3(:,4); SPL_4(:,4)];
SPL_total_4 = SPL_inv_4';
scatter(st_1,SPL_1(:,4),'r','*')
hold on
scatter(st_2,SPL_2(:,4),'b','*')
hold on
scatter(st_3,SPL_3(:,4),'g','*')
hold on
scatter(st_4,SPL_4(:,4),'k','*')
grid on
title('SPL corrected vs. Strouhal for AOA of 16 degrees');
xlabel('Strouhal number');
ylabel('SPL[dB]');
hleg4 = legend('34 m/s','44 m/s','54 m/s','64 m/s');
saveas(figure(4),[pwd '/SPL(corrected) vs Strouhal/AOA 16.fig']);
close all
%-----------------------------------------------------------------------------
% End of routine DU96_Plots.m
%-----------------------------------------------------------------------------
```

```
%% DU96_3DPlot.m
% This routine plots the SPL as a function of the of the AOA and Strouhal
number
%
% Authors: Sterling McBride
%          Ricardo Burdisso
%
% DATE:June 2014
%
% ROUTINES USED: none
%
%-------------------------------------------------------------------------
% Create a folder in the directory
mkdir('SPL(corrected) vs Strouhal vs AOA')
%--------------------------------------------------------------------------
% each curve is plotted, for AOA of 4 degrees
figure(5);
plot3(st_1,AOA_1_vec,SPL_1(:,1),'r')
hold on
plot3(st_2,AOA_1_vec,SPL_2(:,1),'b')
hold on
plot3(st_3,AOA_1_vec,SPL_3(:,1),'g')
hold on
plot3(st_4,AOA_1_vec,SPL_4(:,1),'y')


% each curve is plotted, for AOA of 8 degrees
plot3(st_1,AOA_2_vec,SPL_1(:,2),'r')
hold on
plot3(st_2,AOA_2_vec,SPL_2(:,2),'b')
hold on
plot3(st_3,AOA_2_vec,SPL_3(:,2),'g')
hold on
plot3(st_4,AOA_2_vec,SPL_4(:,2),'y')

% each curve is plotted, for AOA of 12 degrees
plot3(st_1,AOA_3_vec,SPL_1(:,3),'r')
hold on
plot3(st_2,AOA_3_vec,SPL_2(:,3),'b')
hold on
plot3(st_3,AOA_3_vec,SPL_3(:,3),'g')
hold on
plot3(st_4,AOA_3_vec,SPL_4(:,3),'y')


% each curve is plotted, for AOA of 16 degrees
plot3(st_1,AOA_4_vec,SPL_1(:,4),'r')
hold on
plot3(st_2,AOA_4_vec,SPL_2(:,4),'b')
hold on
plot3(st_3,AOA_4_vec,SPL_3(:,4),'g')
hold on
plot3(st_4,AOA_4_vec,SPL_4(:,4),'y')

grid on
title('SPL(Str,AOA)')
xlabel('Strouhal number')
```

```
ylabel('Angle of Attack')
zlabel('SPL [dB]')
view([135 45])
hleg5 = legend('34 m/s','44 m/s','54 m/s','64 m/s');
saveas(figure(5),[pwd '/SPL(corrected) vs Strouhal vs AOA/AOA 16.fig']);
close all
%--------------------------------------------------------------------------
% End of routine DU96_3DPlot.m
%--------------------------------------------------------------------------
```

```
%%   DU96_Power_fit.m
% The curve fitted is of the form  f(x) = a*x^b + c
%
%  INPUTS:
%
%  Data for 'untitled fit 1' fit:
%      X Input : st_total_1
%      Y Output: SPL_total_1
%  Data for 'untitled fit 2' fit:
%      X Input : st_total_2
%      Y Output: SPL_total_2
%  Data for 'untitled fit 3' fit:
%      X Input : st_total_3
%      Y Output: SPL_total_3
%  Data for 'untitled fit 4' fit:
%      X Input : st_total_4
%      Y Output: SPL_total_4
%  Output:
%      fitresult : a cell-array of fit objects representing the fits.
%      gof : structure array with goodness-of fit info.
%
%  Auto-generated by MATLAB on 10-Jun-2014 13:18:04, edited by Sterling
%  McBride

%% Initialization.
% Initialize arrays to store fits and goodness-of-fit.
fitresult = cell( 4, 1 );
gof = struct( 'sse', cell( 4, 1 ), ...
    'rsquare', [], 'dfe', [], 'adjrsquare', [], 'rmse', [] );
%---------------------------------------------------------------------------
% Create a folder in the directory
mkdir('AOA curve fits')
%% Fit: 'Fit for AOA of 4 degrees'.
[xData, yData] = prepareCurveData( st_total_1, SPL_total_1 );

% Set up fittype and options.
ft = fittype( 'power2' );
opts = fitoptions( ft );
opts.Display = 'Off';
opts.Lower = [-Inf -Inf -Inf];
opts.StartPoint = [0.913375856139019 0.63235924622541 0.0975404049994095];
opts.Upper = [Inf Inf Inf];

% Fit model to data.
[fitresult{1}, gof(1)] = fit( xData, yData, ft, opts );
coeffs_1 = coeffvalues(fitresult{1});

% Create a figure for the plots.
figure(6);

% Plot fit with data.
subplot( 2, 1, 1 );
h = plot( fitresult{1}, xData, yData, 'predobs' );
legend( h, 'SPL [dB] vs. Strouhal Number', 'Fit for AOA of 4 degrees',
'Lower bounds (Fit for AOA of 4 degrees)', 'Upper bounds (Fit for AOA of 4
degrees)', 'Location', 'NorthEast' );
% Label axes
xlabel( 'Strouhal Number' );
```

```
ylabel( 'SPL [dB]' );
title('Fit for AOA of 4 degrees');
grid on

% Plot residuals.
subplot( 2, 1, 2 );
h = plot( fitresult{1}, xData, yData, 'residuals' );
legend( h, 'untitled fit 1 - residuals', 'Zero Line', 'Location',
'NorthEast' );
% Label axes
xlabel( 'Strouhal Number' );
ylabel( 'SPL [dB]' );
title('Residuals');
grid on
saveas(figure(6),[pwd '/AOA curve fits/AOA 4 fit.fig']);
%% Fit: 'Fit for AOA of 8 degrees'.
[xData, yData] = prepareCurveData( st_total_2, SPL_total_2 );

% Set up fittype and options.
ft = fittype( 'power2' );
opts = fitoptions( ft );
opts.Display = 'Off';
opts.Lower = [-Inf -Inf -Inf];
opts.StartPoint = [0.964888535199277 0.157613081677548 0.970592781760616];
opts.Upper = [Inf Inf Inf];

% Fit model to data.
[fitresult{2}, gof(2)] = fit( xData, yData, ft, opts );
coeffs_2 = coeffvalues(fitresult{2});

% Create a figure for the plots.
figure(7);

% Plot fit with data.
subplot( 2, 1, 1 );
h = plot( fitresult{2}, xData, yData, 'predobs' );
legend( h, 'SPL [dB] vs. Strouhal Number', 'Fit for AOA of 8 degrees',
'Lower bounds (Fit for AOA of 8 degrees)', 'Upper bounds (Fit for AOA of 8
degrees)', 'Location', 'NorthEast' );
% Label axes
xlabel( 'Strouhal Number' );
ylabel( 'SPL [dB]' );
title('Fit for AOA of 8 degrees');
grid on

% Plot residuals.
subplot( 2, 1, 2 );
h = plot( fitresult{2}, xData, yData, 'residuals' );
legend( h, 'untitled fit 2 - residuals', 'Zero Line', 'Location',
'NorthEast' );
% Label axes
xlabel( 'Strouhal Number' );
ylabel( 'SPL [dB]' );
title('Residuals');
grid on
saveas(figure(7),[pwd '/AOA curve fits/AOA 8 fit.fig']);
%% Fit: 'Fit for AOA of 12 degrees'.
[xData, yData] = prepareCurveData( st_total_3, SPL_total_3 );
```

```
% Set up fittype and options.
ft = fittype( 'power2' );
opts = fitoptions( ft );
opts.Display = 'Off';
opts.Lower = [-Inf -Inf -Inf];
opts.StartPoint = [0.915735525189067 0.792207329559554 0.959492426392903];
opts.Upper = [Inf Inf Inf];

% Fit model to data.
[fitresult{3}, gof(3)] = fit( xData, yData, ft, opts );
coeffs_3 = coeffvalues(fitresult{3});

% Create a figure for the plots.
figure(8);

% Plot fit with data.
subplot( 2, 1, 1 );
h = plot( fitresult{3}, xData, yData, 'predobs' );
legend( h, 'SPL [dB] vs. Strouhal Number', 'Fit for AOA of 12 degrees',
'Lower bounds (Fit for AOA of 12 degrees)', 'Upper bounds (Fit for AOA of
12 degrees)', 'Location', 'NorthEast' );
% Label axes
xlabel( 'Strouhal Number' );
ylabel( 'SPL [dB]' );
title('Fit for AOA of 12 degrees');
grid on

% Plot residuals.
subplot( 2, 1, 2 );
h = plot( fitresult{3}, xData, yData, 'residuals' );
legend( h, 'untitled fit 3 - residuals', 'Zero Line', 'Location',
'NorthEast' );
% Label axes
xlabel( 'Strouhal Number' );
ylabel( 'SPL [dB]' );
title('Residuals');
grid on
saveas(figure(8),[pwd '/AOA curve fits/AOA 12 fit.fig']);
%% Fit: 'Fit for AOA of 16 degrees'.
[xData, yData] = prepareCurveData( st_total_4, SPL_total_4 );

% Set up fittype and options.
ft = fittype( 'power2' );
opts = fitoptions( ft );
opts.Display = 'Off';
opts.Lower = [-Inf -Inf -Inf];
opts.StartPoint = [0.757740130578333 0.743132468124916 0.392227019534168];
opts.Upper = [Inf Inf Inf];

% Fit model to data.
[fitresult{4}, gof(4)] = fit( xData, yData, ft, opts );
coeffs_4 = coeffvalues(fitresult{4});

% Create a figure for the plots.
figure(9);
```

```
% Plot fit with data.
subplot( 2, 1, 1 );
h = plot( fitresult{4}, xData, yData, 'predobs' );
legend( h, 'SPL [dB] vs. Strouhal Number', 'Fit for AOA of 16 degrees',
'Lower bounds (Fit for AOA of 16 degrees)', 'Upper bounds (Fit for AOA of
16 degrees)', 'Location', 'NorthEast' );
% Label axes
xlabel( 'Strouhal Number' );
ylabel( 'SPL [dB]' );
title('Fit for AOA of 16 degrees');
grid on

% Plot residuals.
subplot( 2, 1, 2 );
h = plot( fitresult{4}, xData, yData, 'residuals' );
legend( h, 'untitled fit 4 - residuals', 'Zero Line', 'Location',
'NorthEast' );
% Label axes
xlabel( 'Strouhal Number' );
ylabel( 'SPL [dB]' );
title('Residuals');
grid on
saveas(figure(9),[pwd '/AOA curve fits/AOA 16 fit.fig']);
%%
%-------------------------------------------------------------------------
% End of DU96_Power_fit.m
%-------------------------------------------------------------------------
```

```
%% DU96_coefficients.m
% This routine plots the coefficients from the polynomial fit of the
% corrected SPL vs. AOA
%
% Authors: Sterling McBride
%          Ricardo Burdisso
%
% DATE:June 2014
%
% INPUTS:
%
%  alpha = angles of attack used
%
%  coeff_one = coefficient number one of the polynomial
%  coeff_two = coefficient number one of the polynomial
%  coeff_three = coefficient number one of the polynomial
%  coeff_four = coefficient number one of the polynomial
%
% OUTPUTS:
%
%  Plots of the coefficients from the polynomial fit of the
%  corrected SPL vs. AOA
%
%  Routines used:none
%--------------------------------------------------------------------------
% Create a folder in the directory
mkdir('Fitted coefficients vs. AOA')
%--------------------------------------------------------------------------
% First coefficient plot
figure(10);
alpha = [4,8,12,16];
coeff_one = [coeffs_1(1),coeffs_2(1),coeffs_3(1),coeffs_4(1)];
plot(alpha,coeff_one);

title('First Coefficient vs. Strouhal');
xlabel('Strouhal number');
ylabel('Fit coefficients');
grid on
saveas(figure(10),[pwd '/Fitted coefficients vs. AOA/First
coefficient.fig']);

% Second coefficient plot
figure(11);
alpha = [4,8,12,16];
coeff_two = [coeffs_1(2),coeffs_2(2),coeffs_3(2),coeffs_4(2)];
plot(alpha,coeff_two);

title('Second coefficients vs. Strouhal');
xlabel('Strouhal number');
ylabel('Fit coefficients');
grid on
saveas(figure(11),[pwd '/Fitted coefficients vs. AOA/Second
coefficient.fig']);

% Third coefficient plot
figure(12);
alpha = [4,8,12,16];
coeff_three = [coeffs_1(3),coeffs_2(3),coeffs_3(3),coeffs_4(3)];
```

```
plot(alpha,coeff_three);

title('Third coefficients vs. Strouhal')
ylabel('Fit coefficients');
grid on
saveas(figure(12),[pwd '/Fitted coefficients vs. AOA/Third
coefficient.fig']);
close all
%--------------------------------------------------------------------------
% End of DU96_coefficients.m
%--------------------------------------------------------------------------
```

```
%% DU96_coefficients_power_fit.m
% The curve fitted is of the form  f(x) = a*x^b + c
%
%  Data for 'untitled fit 1' fit:
%      X Input : alpha
%      Y Output: coeff_one
%  Data for 'untitled fit 2' fit:
%      X Input : alpha
%      Y Output: coeff_two
%  Data for 'untitled fit 3' fit:
%      X Input : alpha
%      Y Output: coeff_three
%  Output:
%      fitresult : a cell-array of fit objects representing the fits.
%      gof : structure array with goodness-of fit info.
%
%  See also FIT, CFIT, SFIT.

%  Auto-generated by MATLAB on 10-Jun-2014 15:14:18 edited by Sterling
McBride

%% Initialization.

% Initialize arrays to store fits and goodness-of-fit.
fitresult = cell( 3, 1 );
gof = struct( 'sse', cell( 3, 1 ), ...
    'rsquare', [], 'dfe', [], 'adjrsquare', [], 'rmse', [] );
%--------------------------------------------------------------------------
% Create a folder in the directory
mkdir('Coefficients Curve Fits')
%% Fit: 'Coefficient one fit'.
[xData, yData] = prepareCurveData( alpha, coeff_one );

% Set up fittype and options.
ft = fittype( 'power2' );
opts = fitoptions( ft );
opts.Display = 'Off';
opts.Lower = [-Inf -Inf -Inf];
opts.StartPoint = [0.317099480060861 0.950222048838355 0.0344460805029088];
opts.Upper = [Inf Inf Inf];

% Fit model to data.
[fitresult{1}, gof(1)] = fit( xData, yData, ft, opts );
coefficient1 = coeffvalues(fitresult{1});
save('coefficient1');

% Create a figure for the plots.
figure(13);

% Plot fit with data.
subplot( 2, 1, 1 );
h = plot( fitresult{1}, xData, yData, 'predobs' );
legend( h, 'First coefficient vs. AOA', 'Coefficient one fit','Lower bounds
(Coefficient one fit)', 'Upper bounds (Coefficient one fit)', 'Location',
'NorthEast' );
% Label axes
xlabel( 'alpha [deg]' );
```

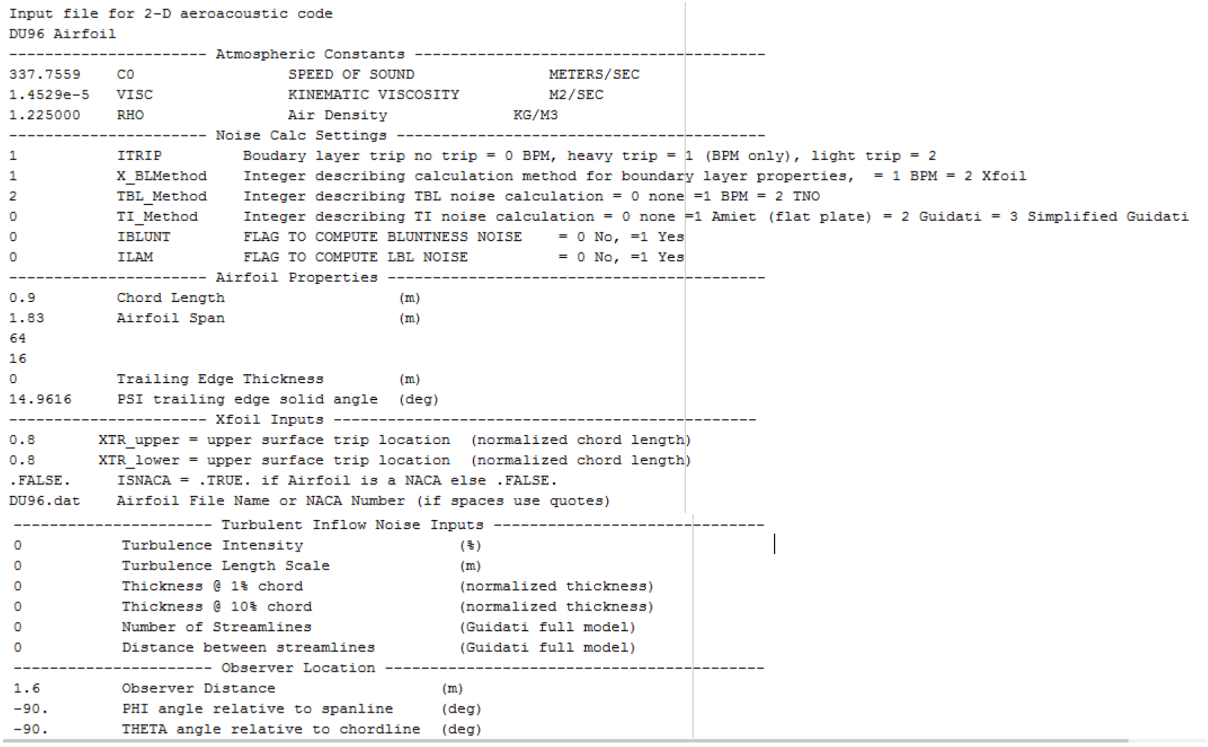
```
ylabel( 'coefficient one' );
title('First Coefficient vs. AOA [deg]');
grid on

% Plot residuals.
subplot( 2, 1, 2 );
h = plot( fitresult{1}, xData, yData, 'residuals' );
legend( h, 'Coefficient one fit - residuals', 'Zero Line', 'Location',
'NorthEast' );
% Label axes
xlabel( 'alpha [deg]' );
ylabel( 'coefficient one' );
grid on
saveas(figure(13),[pwd '/Coefficients Curve Fits/First coefficient.fig']);

%% Fit: 'Coefficient two fit'.
[xData, yData] = prepareCurveData( alpha, coeff_two );

% Set up fittype and options.
ft = fittype( 'power2' );
opts = fitoptions( ft );
opts.Display = 'Off';
opts.Lower = [-Inf -Inf -Inf];
opts.StartPoint = [0.031498938662723 1.07375914827211 0.0032142660933091];
opts.Upper = [Inf Inf Inf];

% Fit model to data.
[fitresult{2}, gof(2)] = fit( xData, yData, ft, opts );
coefficient2 = coeffvalues(fitresult{2});
save('coefficient2');

% Create a figure for the plots.
figure(14);

% Plot fit with data.
subplot( 2, 1, 1 );
h = plot( fitresult{2}, xData, yData, 'predobs' );
legend( h, 'Second coefficient vs. AOA', 'Coefficient two fit','Lower
bounds (Coefficient two fit)', 'Upper bounds (Coefficient two fit)',
'Location', 'NorthEast' );
% Label axes
xlabel( 'alpha [deg]' );
ylabel( 'coefficient two' );
title('Second Coefficient vs. AOA [deg]');
grid on

% Plot residuals.
subplot( 2, 1, 2 );
h = plot( fitresult{2}, xData, yData, 'residuals' );
legend( h, 'Coefficient two fit - residuals', 'Zero Line', 'Location',
'NorthEast' );
% Label axes
xlabel( 'alpha [deg]' );
ylabel( 'coefficient two' );
grid on
saveas(figure(14),[pwd '/Coefficients Curve Fits/Second coefficient.fig']);

%% Fit: 'Coefficient three fit'.
```

```
[xData, yData] = prepareCurveData( alpha, coeff_three );

% Set up fittype and options.
ft = fittype( 'power2' );
opts = fitoptions( ft );
opts.Display = 'Off';
opts.Lower = [-Inf -Inf -Inf];
opts.StartPoint = [0.0971317812358475 0.823457828327293 0.694828622975817];
opts.Upper = [Inf Inf Inf];

% Fit model to data.
[fitresult{3}, gof(3)] = fit( xData, yData, ft, opts );
coefficient3 = coeffvalues(fitresult{3});
save('coefficient3');

% Create a figure for the plots.
figure(15);

% Plot fit with data.
subplot( 2, 1, 1 );
h = plot( fitresult{3}, xData, yData, 'predobs' );
legend( h, 'Third coefficient vs. AOA', 'Coefficient three fit','Lower
bounds (Coefficient three fit)', 'Upper bounds (Coefficient three fit)',
'Location', 'NorthEast' );
% Label axes
xlabel( 'alpha [deg]' );
ylabel( 'coefficient three' );
title('Third Coefficient vs. AOA [deg]');
grid on

% Plot residuals.
subplot( 2, 1, 2 );
h = plot( fitresult{3}, xData, yData, 'residuals' );
legend( h, 'Coefficient three fit - residuals', 'Zero Line', 'Location',
'NorthEast' );
% Label axes
xlabel( 'alpha [deg]' );
ylabel( 'coefficient three' );
grid on
saveas(figure(15),[pwd '/Coefficients Curve Fits/Third coefficient.fig']);
close all
%%
%------------------------------------------------------------------------
% End of DU96_coefficients_power_fit.m
%------------------------------------------------------------------------
```

```
%% DU96_SPLplot.m
% This routine plots the SPL as a function of the of the AOA and Strouhal
number
%
% Authors: Sterling McBride
%          Ricardo Burdisso
%
% DATE:June 2014
%
% ROUTINES USED: none
%
%--------------------------------------------------------------------------
% The directory is created inb the same folder
mkdir('SPL corrected final curve')
%--------------------------------------------------------------------------
% Variables are defined
wind_velocity = 10;
chord=0.9;
AOA = [4,8,12,16];
P_ref = 20e-6;

f_bands = [40,42.5,45,47.5,50,53,56,60,63,67,71,75,80, ...
           85,90,95,100,106,112,118,125,132,140,150,160,170,180, ...
           190,200,212,224,236,250,265,280,300,315,335,355,375,400,425, ...
           450,475,500,530,560,600,630,670,710,750,800,850,900,950,1000,
...
           1060,1120,1180,1250,1320,1400,1500,1600,1700,1800,1900,2000, ...
           2120,2240,2360,2500,2650,2800,3000,3150,3350,3550,3750,4000, ...
           4250,4500,4750,5000];

Nb_3bands = length(f_bands);
AOA_count = 0;

% The final SPL equation is  applied

 for l = 1: length(AOA)
      AOA_count = AOA_count+1;
  for n = 1:Nb_3bands
 st(n) = f_bands(n)*chord/wind_velocity;

 SPL_eq(n,AOA_count) =
((coefficient1(1)*(AOA(l))^coefficient1(2))+(coefficient1(3)))...

*st(n)^((coefficient2(1)*(AOA(l))^coefficient2(2))+(coefficient2(3)))...
             +
((coefficient3(1)*(AOA(l))^coefficient3(2))+(coefficient3(3))) ;

 Prms_square_source(n) = (P_ref^2)*(10^(SPL_eq(n,AOA_count)/10));
 end
 end

% The figures are plot

figure(16)
plot(st,SPL_eq(:,1));
title('SPL [dB] for AOA of 4 Degrees vs. Strouhal Number')
xlabel('Strouhal Number');
ylabel('SPL [dB]');
```

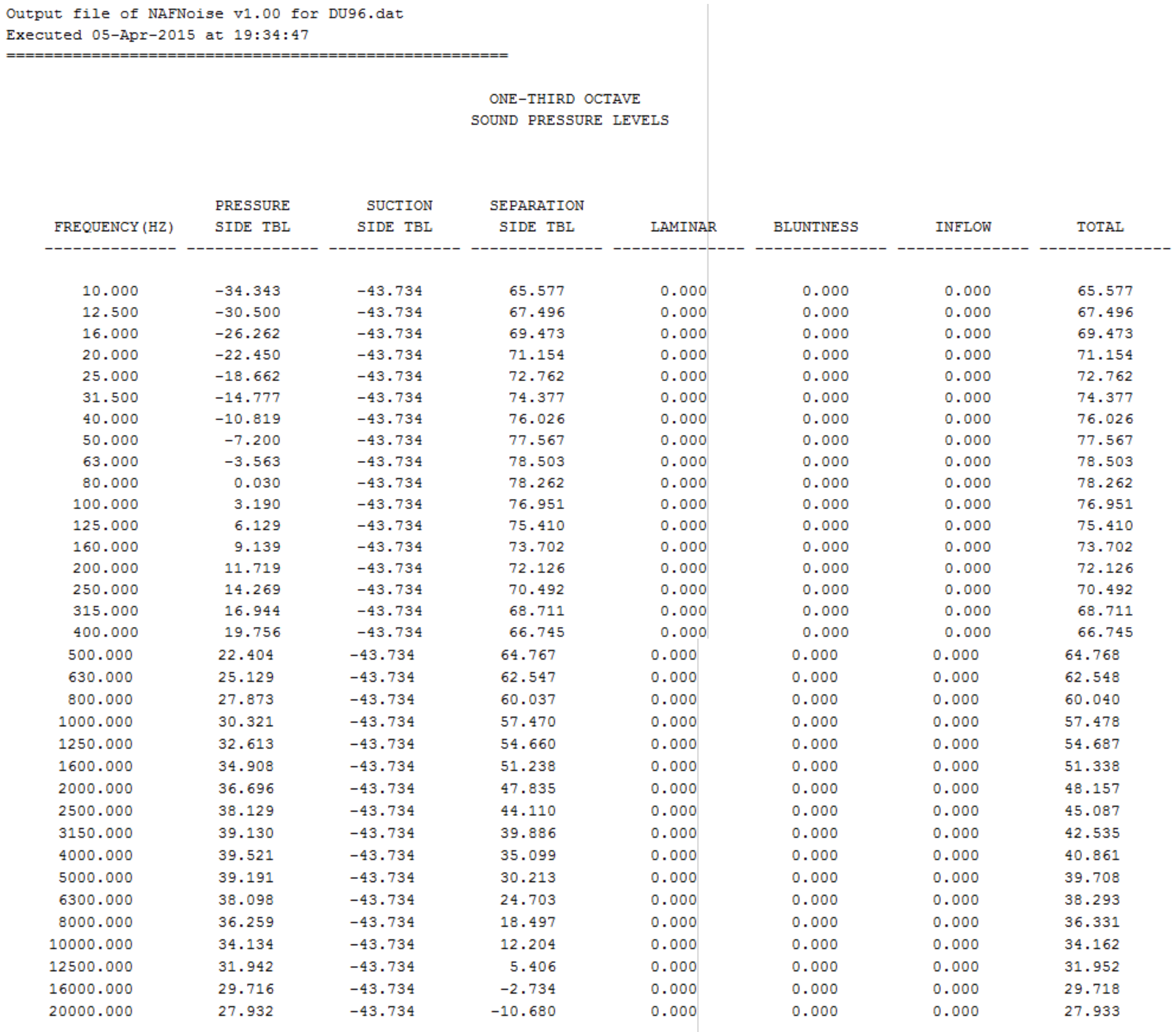
```
grid on
saveas(figure(16),[pwd '/SPL corrected final curve/SPL vs st1.fig']);
close all


figure(17)
plot(st,SPL_eq(:,2));
title('SPL [dB] for AOA of 8 Degrees vs. Strouhal Number')
xlabel('Strouhal Number');
ylabel('SPL [dB]');
grid on
saveas(figure(17),[pwd '/SPL corrected final curve/SPL vs st2.fig']);
close all

figure(18)
plot(st,SPL_eq(:,3));
title('SPL [dB] for AOA of 12 Degrees vs. Strouhal Number')
xlabel('Strouhal Number');
ylabel('SPL [dB]');
grid on
saveas(figure(18),[pwd '/SPL corrected final curve/SPL vs st3.fig']);
close all

figure(19)
plot(st,SPL_eq(:,4));
title('SPL [dB] for AOA of 16 Degrees vs. Strouhal Number')
xlabel('Strouhal Number');
ylabel('SPL [dB]');
grid on
saveas(figure(19),[pwd '/SPL corrected final curve/SPL vs st4.fig']);
close all
%%
%-------------------------------------------------------------------------
% End of DU96_SPLplot.m
%-------------------------------------------------------------------------
```

APPENDIX C: nafnoise.ipt file



```
Input file for 2-D aeroacoustic code
DU96 Airfoil
---------------------- Atmospheric Constants ---------------------------------------
337.7559    C0                 SPEED OF SOUND               METERS/SEC
1.4529e-5   VISC               KINEMATIC VISCOSITY          M2/SEC
1.225000    RHO                Air Density              KG/M3
---------------------- Noise Calc Settings -----------------------------------------
1           ITRIP         Boudary layer trip no trip = 0 BPM, heavy trip = 1 (BPM only), light trip = 2
1           X_BLMethod    Integer describing calculation method for boundary layer properties,  = 1 BPM = 2 Xfoil
2           TBL_Method    Integer describing TBL noise calculation = 0 none =1 BPM = 2 TNO
0           TI_Method     Integer describing TI noise calculation = 0 none =1 Amiet (flat plate) = 2 Guidati = 3 Simplified Guidati
0           IBLUNT        FLAG TO COMPUTE BLUNTNESS NOISE    = 0 No, =1 Yes
0           ILAM          FLAG TO COMPUTE LBL NOISE          = 0 No, =1 Yes
---------------------- Airfoil Properties ------------------------------------------
0.9         Chord Length                   (m)
1.83        Airfoil Span                   (m)
64
16
0           Trailing Edge Thickness        (m)
14.9616     PSI trailing edge solid angle  (deg)
---------------------- Xfoil Inputs -----------------------------------------------
0.8       XTR_upper = upper surface trip location  (normalized chord length)
0.8       XTR_lower = upper surface trip location  (normalized chord length)
.FALSE.     ISNACA = .TRUE. if Airfoil is a NACA else .FALSE.
DU96.dat    Airfoil File Name or NACA Number (if spaces use quotes)
---------------------- Turbulent Inflow Noise Inputs -------------------------------
0           Turbulence Intensity                 (%)
0           Turbulence Length Scale              (m)
0           Thickness @ 1% chord                 (normalized thickness)
0           Thickness @ 10% chord                (normalized thickness)
0           Number of Streamlines                (Guidati full model)
0           Distance between streamlines         (Guidati full model)
---------------------- Observer Location -------------------------------------------
1.6         Observer Distance                  (m)
-90.        PHI angle relative to spanline     (deg)
-90.        THETA angle relative to chordline  (deg)
```

APPENDIX D: nafnoise.out File



```
Output file of NAFNoise v1.00 for DU96.dat
Executed 05-Apr-2015 at 19:34:47
=====================================================

                                                   ONE-THIRD OCTAVE
                                                 SOUND PRESSURE LEVELS


                      PRESSURE       SUCTION      SEPARATION
     FREQUENCY(HZ)    SIDE TBL       SIDE TBL       SIDE TBL       LAMINAR      BLUNTNESS        INFLOW         TOTAL
    -------------- -------------- -------------- -------------- -------------- -------------- -------------- --------------

        10.000        -34.343        -43.734         65.577          0.000          0.000          0.000         65.577
        12.500        -30.500        -43.734         67.496          0.000          0.000          0.000         67.496
        16.000        -26.262        -43.734         69.473          0.000          0.000          0.000         69.473
        20.000        -22.450        -43.734         71.154          0.000          0.000          0.000         71.154
        25.000        -18.662        -43.734         72.762          0.000          0.000          0.000         72.762
        31.500        -14.777        -43.734         74.377          0.000          0.000          0.000         74.377
        40.000        -10.819        -43.734         76.026          0.000          0.000          0.000         76.026
        50.000         -7.200        -43.734         77.567          0.000          0.000          0.000         77.567
        63.000         -3.563        -43.734         78.503          0.000          0.000          0.000         78.503
        80.000          0.030        -43.734         78.262          0.000          0.000          0.000         78.262
       100.000          3.190        -43.734         76.951          0.000          0.000          0.000         76.951
       125.000          6.129        -43.734         75.410          0.000          0.000          0.000         75.410
       160.000          9.139        -43.734         73.702          0.000          0.000          0.000         73.702
       200.000         11.719        -43.734         72.126          0.000          0.000          0.000         72.126
       250.000         14.269        -43.734         70.492          0.000          0.000          0.000         70.492
       315.000         16.944        -43.734         68.711          0.000          0.000          0.000         68.711
       400.000         19.756        -43.734         66.745          0.000          0.000          0.000         66.745
       500.000         22.404        -43.734         64.767          0.000          0.000          0.000         64.768
       630.000         25.129        -43.734         62.547          0.000          0.000          0.000         62.548
       800.000         27.873        -43.734         60.037          0.000          0.000          0.000         60.040
      1000.000         30.321        -43.734         57.470          0.000          0.000          0.000         57.478
      1250.000         32.613        -43.734         54.660          0.000          0.000          0.000         54.687
      1600.000         34.908        -43.734         51.238          0.000          0.000          0.000         51.338
      2000.000         36.696        -43.734         47.835          0.000          0.000          0.000         48.157
      2500.000         38.129        -43.734         44.110          0.000          0.000          0.000         45.087
      3150.000         39.130        -43.734         39.886          0.000          0.000          0.000         42.535
      4000.000         39.521        -43.734         35.099          0.000          0.000          0.000         40.861
      5000.000         39.191        -43.734         30.213          0.000          0.000          0.000         39.708
      6300.000         38.098        -43.734         24.703          0.000          0.000          0.000         38.293
      8000.000         36.259        -43.734         18.497          0.000          0.000          0.000         36.331
     10000.000         34.134        -43.734         12.204          0.000          0.000          0.000         34.162
     12500.000         31.942        -43.734          5.406          0.000          0.000          0.000         31.952
     16000.000         29.716        -43.734         -2.734          0.000          0.000          0.000         29.718
     20000.000         27.932        -43.734        -10.680          0.000          0.000          0.000         27.933
```

APPENDIX E: NAFNoise_main.m

```
%% NAFNOISE_RUN.m
% This is the main code the runs NAFNoise.exe, and extracts the total SPL
% for 4 different angles of attack and 4 different flow velocities.
%
% Author: Sterling McBride
%
%
% DATE:February 2015
%
%% ------------------------------------------------------------------------
clear
clc
tic
%% ------------------------------------------------------------------------
% The angles of attack and flow velocity are defined
AOA = [4,8,12,16];
Vel = [34,44,54,64];
ncount_1 = 0;

for n = 1:length(AOA)
    for m = 1:length(Vel)

    ncount_1 = ncount_1 + 1;
%% ------------------------------------------------------------------------
% Read nafnoise.inp file and save into cell A
fid = fopen('nafnoise.ipt','r'); %% open file for read

if fid == -1 % True: the file does not exist
    error(['ERROR: The file ','nafnoise.ipt',' could not be opened.'])
end

l = 1;
tline = fgetl(fid);          % Returns the next line of the specified file,
removing the newline characters.
                             % fid is an integer file identifier obtained
from fopen. tline is a text string
                             % unless the line contains only the end-of-
file marker. In this case, tline is the numeric value -1.
A{l} = tline;                % Define cell A.
while ischar(tline)          % ischar(A) returns logical 1 (true) if A{1}
is a character array and logical 0 (false) otherwise.
    l = l+1;                 % Save each line contained in the file in cell
A.
    tline = fgetl(fid);
    A{l} = tline;
end
fclose(fid);                 % Close the open file.

% Change data in cell A
A{17} = sprintf('%d', Vel(m));     % Format the required data into a string
(Free Stream Velocity).
A{18} = sprintf('%d', AOA(n));          % Format the required data into a
string (Angle of Attack).
% Write the changed Cell A, into the new nafnoise.ipt file
```

```
fid = fopen('nafnoise.ipt', 'w');  % Create new file for writing.
for l = 1:numel(A)                 % Loop for the number of array elements
within cell A.
    if A{l+1} == -1                % If the array element contains numeric
value -1, it means end-of-line-file marker has been reached.
        fprintf(fid,'%s', A{l});   % Write the end-of-line-file marker in
the new nafnoise.ipt.
        break                      % Terminate Execution of for loop
    else
        fprintf(fid,'%s\n', A{l}); % In other cases write data into file,
from cell A.
    end
end
fclose(fid);                        % Close the open file.

%% -----------------------------------------------------------------------
% Run NAFNoise.exe
system('Directory of NAFNOISE.exe');

%% -----------------------------------------------------------------------
% Retreive and save data from nafnoise.out file
fid = fopen('nafnoise.out','r'); %% open file for read

if fid == -1 % True: the file does not exist
    error(['ERROR: The file ','nafnoise.out',' could not be opened.'])
end
% Read file into cell C line by line
C = textscan(fid,'%s','delimiter', '\n');
% Copy C cell into a string array
Datastring = char(C{1});
% Close file
fclose(fid);
%% -----------------------------------------------------------------------
% This section converts and saves data to proper format
ncount = 13;
% The sound pressure level data and the used frequencies are saved
for nl = 1:34
ncount = ncount + 1;

NAFnoise_SPL(nl,:) = textscan(Datastring(ncount,:), '%f %f %f %f %f %f %f
%f');
NAFNoise_TotalSPL(nl,ncount_1) = NAFnoise_SPL{nl,8};
Frequencies(nl) = NAFnoise_SPL{nl,1};
end

    end
end
toc
%% -----------------------------------------------------------------------
% End of NAFNOISE_RUN.m
%-------------------------------------------------------------------------
```

```
%% NAFNOISE_interpolation.m
% This code interpolates the nafnoise data for the frequency bands used on
DU-96 experimental data.
%
% Author: Sterling McBride
%
%
% DATE:February 2015
%% -----------------------------------------------------------------------
% The interpolation procedure
% The frequency bands used on the DU-96 experimental data are defined

Du96_bands =
[500,530,560,600,630,670,710,750,800,850,900,950,1000,1060,1120,1180,1250,1
320,1400,1500,1600,1700,1800,1900,2000,2120,2240,2360,2500,...

2650,2800,3000,3150,3350,3550,3750,4000,4250,4500,4750,5000,5300,5600,6000,
6300,6700,7100];

% Frequency bands used by nafnoise thaat are within 500 Hz and 7100 Hz are
defined in a vector

nafnoise_bands = Frequencies (17:30);

% The nafnoise data is interpolated for all the 1/3rd octave frequency
% bands used on the experimental DU96 data.

    for m = 1:16


        nafnoise_SPL_interpolated (:,m)= interp1(nafnoise_bands,
NAFNoise_TotalSPL(17:30,m),Du96_bands);

    end
%% -----------------------------------------------------------------------
% End of NAFNOISE_interpolation.m
%-------------------------------------------------------------------------
```

```
%% NAFNOISE_OSPL.m
% This routine computes the OASPL for all flow velocities for eacha ngle of
attack
% for experimental and modeled data
%
% DATE:March 2015
% INPUTS:
%
%
% OUTPUT:
%
%
% ROUTINES USED: none
%
%------------------------------------------------------------------------
% First, all the power coefficientes from the formula are computed
f_count = 0;

for n=1:length (U_1_data(:,2))

f_count = f_count+1;

% 34 m/s, 4 degrees
power(f_count) = 10^((U_1_data(f_count,2))/10);
power1(f_count) = 10^((nafnoise_SPL_interpolated(f_count,1))/10);

% 34 m/s, 8 degrees
power2(f_count) = 10^((U_1_data(f_count,3))/10);
power3(f_count) = 10^((nafnoise_SPL_interpolated(f_count,5))/10);

% 34 m/s, 12 degrees
power4(f_count) = 10^((U_1_data(f_count,4))/10);
power5(f_count) = 10^((nafnoise_SPL_interpolated(f_count,9))/10);

% 34 m/s, 16 degrees
power6(f_count) = 10^((U_1_data(f_count,5))/10);
power7(f_count) = 10^((nafnoise_SPL_interpolated(f_count,13))/10);

% 44 m/s, 4 degrees
power8(f_count) = 10^((U_2_data(f_count,2))/10);
power9(f_count) = 10^((nafnoise_SPL_interpolated(f_count,2))/10);

% 44 m/s, 8 degrees
power10(f_count) = 10^((U_2_data(f_count,3))/10);
power11(f_count) = 10^((nafnoise_SPL_interpolated(f_count,6))/10);

% 44 m/s, 12 degrees
power12(f_count) = 10^((U_2_data(f_count,4))/10);
power13(f_count) = 10^((nafnoise_SPL_interpolated(f_count,10))/10);

% 44 m/s, 16 degrees
power14(f_count) = 10^((U_2_data(f_count,5))/10);
power15(f_count) = 10^((nafnoise_SPL_interpolated(f_count,14))/10);

% 54 m/s, 4 degrees
power16(f_count) = 10^((U_3_data(f_count,2))/10);
power17(f_count) = 10^((nafnoise_SPL_interpolated(f_count,3))/10);
```

```
% 54 m/s, 8 degrees
power18(f_count) = 10^((U_3_data(f_count,3))/10);
power19(f_count) = 10^((nafnoise_SPL_interpolated(f_count,7))/10);

% 54 m/s, 12 degrees
power20(f_count) = 10^((U_3_data(f_count,4))/10);
power21(f_count) = 10^((nafnoise_SPL_interpolated(f_count,11))/10);

% 54 m/s, 16 degrees
power22(f_count) = 10^((U_3_data(f_count,5))/10);
power23(f_count) = 10^((nafnoise_SPL_interpolated(f_count,15))/10);

% 64 m/s, 4 degrees
power24(f_count) = 10^((U_4_data(f_count,2))/10);
power25(f_count) = 10^((nafnoise_SPL_interpolated(f_count,4))/10);

% 64 m/s, 8 degrees
power26(f_count) = 10^((U_4_data(f_count,3))/10);
power27(f_count) = 10^((nafnoise_SPL_interpolated(f_count,8))/10);

% 64 m/s, 12 degrees
power28(f_count) = 10^((U_4_data(f_count,4))/10);
power29(f_count) = 10^((nafnoise_SPL_interpolated(f_count,12))/10);

% 64 m/s, 16 degrees
power30(f_count) = 10^((U_4_data(f_count,5))/10);
power31(f_count) = 10^((nafnoise_SPL_interpolated(f_count,16))/10);

end
%--------------------------------------------------------------------------
% The total SPL for each case areobtained
% 34 m/s, 4 degrees
Total_spl_v34a4_experiment = 10*log10(sum(power));
Total_spl_v34a4_modeled = 10*log10(sum(power1));

% 34 m/s, 8 degrees
Total_spl_v34a8_experiment = 10*log10(sum(power2));
Total_spl_v34a8_modeled = 10*log10(sum(power3));

% 34 m/s, 12 degrees
Total_spl_v34a12_experiment = 10*log10(sum(power4));
Total_spl_v34a12_modeled = 10*log10(sum(power5));

% 34 m/s, 16 degrees
Total_spl_v34a16_experiment = 10*log10(sum(power6));
Total_spl_v34a16_modeled = 10*log10(sum(power7));

% 44 m/s, 4 degrees
Total_spl_v44a4_experiment = 10*log10(sum(power8));
Total_spl_v44a4_modeled = 10*log10(sum(power9));

% 44 m/s, 8 degrees
Total_spl_v44a8_experiment = 10*log10(sum(power10));
Total_spl_v44a8_modeled = 10*log10(sum(power11));
```

```
% 44 m/s, 12 degrees
Total_spl_v44a12_experiment = 10*log10(sum(power12));
Total_spl_v44a12_modeled = 10*log10(sum(power13));

% 44 m/s, 16 degrees
Total_spl_v44a16_experiment = 10*log10(sum(power14));
Total_spl_v44a16_modeled = 10*log10(sum(power15));

% 54 m/s, 4 degrees
Total_spl_v54a4_experiment = 10*log10(sum(power16));
Total_spl_v54a4_modeled = 10*log10(sum(power17));

% 54 m/s, 8 degrees
Total_spl_v54a8_experiment = 10*log10(sum(power18));
Total_spl_v54a8_modeled = 10*log10(sum(power19));

% 54 m/s, 12 degrees
Total_spl_v54a12_experiment = 10*log10(sum(power20));
Total_spl_v54a12_modeled = 10*log10(sum(power21));

% 54 m/s, 16 degrees
Total_spl_v54a16_experiment = 10*log10(sum(power22));
Total_spl_v54a16_modeled = 10*log10(sum(power23));

% 64 m/s, 4 degrees
Total_spl_v64a4_experiment = 10*log10(sum(power24));
Total_spl_v64a4_modeled = 10*log10(sum(power25));

% 64 m/s, 8 degrees
Total_spl_v64a8_experiment = 10*log10(sum(power26));
Total_spl_v64a8_modeled = 10*log10(sum(power27));

% 64 m/s, 12 degrees
Total_spl_v64a12_experiment = 10*log10(sum(power28));
Total_spl_v64a12_modeled = 10*log10(sum(power29));

% 64 m/s, 16 degrees
Total_spl_v64a16_experiment = 10*log10(sum(power30));
Total_spl_v64a16_modeled = 10*log10(sum(power31));

%-------------------------------------------------------------------------
% End of routine NAFNOISE_OSPL.m
%-------------------------------------------------------------------------
```

```
%% NAFNOISE_plots.m
% This routine plots the SPL as a function of 1/3rd frequency bands, for
% experimental and modeled data
%
% DATE:March 2015
% INPUTS:
%
%
% OUTPUT:
%
%
% ROUTINES USED: none
%
%------------------------------------------------------------------------------
% Create a folder in the directory
mkdir('SPL vs band')
%------------------------------------------------------------------------------
% 34 m/s, 4 degrees
figure(1)
scatter(U_1_data(:,1),U_1_data(:,2),'b','*');
hold on
scatter(U_1_data(:,1),nafnoise_SPL_interpolated(:,1),'r','*');
grid on
title('SPL [dB] vs. 1/3rd frequency bands (flow velocity = 34 m/s & Angle
of attack = 4 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg1 = legend('Experimental Data','Modeled Data');
saveas(figure(1),[pwd '/SPL vs band/AOA434ms.fig']);


% 34 m/s, 8 degrees
figure (2)
scatter(U_1_data(:,1),U_1_data(:,3),'*');
hold on
scatter(U_1_data(:,1),nafnoise_SPL_interpolated(:,5),'r','*');
grid on
title('SPL[dB] vs. 1/3rd frequency bands (flow velocity = 34 m/s & Angle of
attack = 8 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg2 = legend('Experimental Data','Modeled Data');
saveas(figure(2),[pwd '/SPL vs band/AOA834ms.fig']);


% 34 m/s, 12 degrees
figure(3)
scatter(U_1_data(:,1),U_1_data(:,4),'*');
hold on
scatter(U_1_data(:,1),nafnoise_SPL_interpolated(:,9),'r','*');
grid on
title('SPL [dB] vs. 1/3rd frequency bands (flow velocity = 34 m/s & Angle
of attack = 12 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg3 = legend('Experimental Data','Modeled Data');
saveas(figure(3),[pwd '/SPL vs band/AOA1234ms.fig']);
```

```
% 34 m/s, 16 degrees
figure(4)
scatter(U_1_data(:,1),U_1_data(:,5),'*');
hold on
scatter(U_1_data(:,1),nafnoise_SPL_interpolated(:,13),'r','*');
grid on
title('SPL [dB] vs. 1/3rd frequency bands (flow velocity = 34 m/s & Angle
of attack = 16 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg4 = legend('Experimental Data','Modeled Data');
saveas(figure(4),[pwd '/SPL vs band/AOA1634ms.fig']);

% 44 m/s, 4 degrees
figure(5)
scatter(U_2_data(:,1),U_2_data(:,2),'*');
hold on
scatter(U_2_data(:,1),nafnoise_SPL_interpolated(:,2),'r','*');
grid on
title('SPL [dB] vs. 1/3rd frequency bands (flow velocity = 44 m/s & Angle
of attack = 4 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg5 = legend('Experimental Data','Modeled Data');
saveas(figure(5),[pwd '/SPL vs band/AOA444ms.fig']);

% 44 m/s, 8 degrees
figure(6)
scatter(U_2_data(:,1),U_2_data(:,3),'*');
hold on
scatter(U_2_data(:,1),nafnoise_SPL_interpolated(:,6),'r','*');
grid on
title('SPL [dB] vs. 1/3rd frequency bands (flow velocity = 44 m/s & Angle
of attack = 8 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg6 = legend('Experimental Data','Modeled Data');
saveas(figure(6),[pwd '/SPL vs band/AOA844ms.fig']);

% 44 m/s, 12 degrees
figure(7)
scatter(U_2_data(:,1),U_2_data(:,4),'*');
hold on
scatter(U_2_data(:,1),nafnoise_SPL_interpolated(:,10),'r','*');
grid on
title('SPL[dB] vs. 1/3rd frequency bands (flow velocity = 44 m/s & Angle of
attack = 12 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg7 = legend('Experimental Data','Modeled Data');
saveas(figure(7),[pwd '/SPL vs band/AOA1244ms.fig']);

% 44 m/s, 16 degrees
figure(8)
scatter(U_2_data(:,1),U_2_data(:,5),'*');
hold on
scatter(U_2_data(:,1),nafnoise_SPL_interpolated(:,14),'r','*');
```

```
grid on
title('SPL [dB] vs. 1/3rd frequency bands (flow velocity = 44 m/s & Angle
of attack = 16 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg8 = legend('Experimental Data','Modeled Data');
saveas(figure(8),[pwd '/SPL vs band/AOA1644ms.fig']);

% 54 m/s, 4 degrees
figure(9)
scatter(U_3_data(:,1),U_3_data(:,2),'*');
hold on
scatter(U_3_data(:,1),nafnoise_SPL_interpolated(:,3),'r','*');
grid on
title('SPL [dB] vs. 1/3rd frequency bands (flow velocity = 54 m/s & Angle
of attack = 4 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg9 = legend('Experimental Data','Modeled Data');
saveas(figure(9),[pwd '/SPL vs band/AOA454ms.fig']);

% 54 m/s, 8 degrees
figure(10)
scatter(U_3_data(:,1),U_3_data(:,3),'*');
hold on
scatter(U_3_data(:,1),nafnoise_SPL_interpolated(:,7),'r','*');
grid on
title('SPL [dB] vs. 1/3rd frequency bands (flow velocity = 54 m/s & Angle
of attack = 8 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg10 = legend('Experimental Data','Modeled Data');
saveas(figure(10),[pwd '/SPL vs band/AOA854ms.fig']);

% 54 m/s, 12 degrees
figure(11)
scatter(U_3_data(:,1),U_3_data(:,4),'*');
hold on
scatter(U_3_data(:,1),nafnoise_SPL_interpolated(:,11),'r','*');
grid on
title('SPL [dB] vs. 1/3rd frequency bands (flow velocity = 54 m/s & Angle
of attack = 12 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg11 = legend('Experimental Data','Modeled Data');
saveas(figure(11),[pwd '/SPL vs band/AOA1254ms.fig']);

% 54 m/s, 16 degrees
figure(12)
scatter(U_4_data(:,1),U_4_data(:,5),'*');
hold on
scatter(U_4_data(:,1),nafnoise_SPL_interpolated(:,15),'r','*');
grid on
title('SPL[dB] vs. 1/3rd frequency bands (flow velocity = 54 m/s & Angle of
attack = 16 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg12 = legend('Experimental Data','Modeled Data');
```

```
saveas(figure(12),[pwd '/SPL vs band/AOA1654ms.fig']);


% 64 m/s, 4 degrees
figure(13)
scatter(U_4_data(:,1),U_4_data(:,2),'*');
hold on
scatter(U_4_data(:,1),nafnoise_SPL_interpolated(:,4),'r','*');
grid on
title('SPL [dB] vs. 1/3rd frequency bands (flow velocity = 64 m/s & Angle
of attack = 4 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg13 = legend('Experimental Data','Modeled Data');
saveas(figure(13),[pwd '/SPL vs band/AOA464ms.fig']);

% 64 m/s, 8 degrees
figure(14)
scatter(U_4_data(:,1),U_4_data(:,3),'*');
hold on
scatter(U_4_data(:,1),nafnoise_SPL_interpolated(:,8),'r','*');
grid on
title('SPL [dB] vs. 1/3rd frequency bands (flow velocity = 64 m/s & Angle
of attack = 8 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg14 = legend('Experimental Data','Modeled Data');
saveas(figure(14),[pwd '/SPL vs band/AOA864ms.fig']);

% 64 m/s, 12 degrees
figure(15)
scatter(U_4_data(:,1),U_4_data(:,4),'*');
hold on
scatter(U_4_data(:,1),nafnoise_SPL_interpolated(:,12),'r','*');
grid on
title('SPL [dB] vs. 1/3rd frequency bands (flow velocity = 64 m/s & Angle
of attack = 12 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg15 = legend('Experimental Data','Modeled Data');
saveas(figure(15),[pwd '/SPL vs band/AOA1264ms.fig']);

% 64 m/s, 16 degrees
figure(16)
scatter(U_4_data(:,1),U_4_data(:,5),'*');
hold on
scatter(U_4_data(:,1),nafnoise_SPL_interpolated(:,16),'r','*');
grid on
title('SPL [dB] vs. 1/3rd frequency bands (flow velocity = 64 m/s & Angle
of attack = 16 deg.)');
xlabel('1/3rd octave frequency bands');
ylabel('Sound Pressure Levels [dB]');
hleg16 = legend('Experimental Data','Modeled Data');
saveas(figure(16),[pwd '/SPL vs band/AOA1664ms.fig']);
%--------------------------------------------------------------------------
% End of routine NAFNOISE_plots.m
%--------------------------------------------------------------------------
```

APPENDIX F: WTNOISE_main.m

```
%% WTNOISE_input.m
% This routine reads the input to the code
%
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
%
% INPUTS:
%
% OUTPUT:
%
% ROUTINES USED: none
%
%---------------------------------------------------------------------------
disp('    ==> WTNOISE_input.m')
%%--------------------------------------------------------------------------
% ===> Read weather parameters input file
% Open and check that 'weather_parameters.txt' exists and it can be openned
fid = fopen(strcat('Directory if file','weather_parameters.txt'),'r');
if fid == -1 % True: the file does not exist.
    error(['ERROR: The file ','weather_parameters.txt',' could not be
opened.'])
end
% Read file into cell C line by line (comment lines indicated by "/") are
ignored
C = textscan(fid,'%s','delimiter', '\n','commentStyle', '/');
% Copy C cell into a string array
Weather_datastring = char(C{1});
% Close file
fclose(fid);
%% -------------------------------------------------------------------------
% ===> Read general parameters input file
% Open and check that 'general_parameters.txt' exists and it can be openned
fid = fopen(strcat('Directory of File','general_parameters.txt'),'r');
if fid == -1 % True: the file does not exist
    error(['ERROR: The file ','general_parameters.txt',' could not be
opened.'])
end
% Read file into cell C line by line (comment lines indicated by "/") are
ignored
C = textscan(fid,'%s','delimiter', '\n','commentStyle', '/');
% Copy C cell into a string array
General_datastring = char(C{1});
% Close file
fclose(fid);
%%--------------------------------------------------------------------------
% ===> Read blade parameters input file
% Open and check that 'blade_parameters.txt' exists and it can be openned
fid = fopen(strcat('Directory of File','blade_parameters.txt'),'r');
if fid == -1 % True: the file does not exist
    error(['ERROR: The file ','blade_parameters.txt',' could not be
opened.'])
end
```

```
% Read file into cell C line by line (comment lines indicated by "/") are
ignored
C = textscan(fid,'%s','delimiter', '\n','commentStyle', '/');
% Copy C cell into a string array
Blade_datastring = char(C{1});
% Close file
fclose(fid);

%% ------------------------------------------------------------------------
% This section converts and saves the weather data to proper format
weather_ncount = 0;
%
% Weather variables:
% height  x-wind   y-wind    Temp         Press       Densisty         RH
fraction
% m         m/s      m/s     Kelvin       kPa         kg/m^3
RH%/100
for nl = 1:31
weather_ncount = weather_ncount + 1;
weather_variables(nl,:) = textscan(Weather_datastring(weather_ncount,:),'%f
%f %f %f %f %f %f %f');
end
%  velocity_height = height at which weather data is reported [m]
velocity_height = [weather_variables{:,1}]';
%  xwind_velocity = velocities on x direction reported at different heights
[m/s]
xwind_velocity = [weather_variables{:,2}]';
%  ywind_velocity = velocities on x direction reported at different heights
[m/s]
ywind_velocity = [weather_variables{:,3}]';
%  temperature
temperature_velocity = [weather_variables{:,4}]';
%  Air Density
air_density = [weather_variables{:,6}]';
%% ------------------------------------------------------------------------
% This section converts and saves the wind turbine general data to proper
format
general_ncount = 0;
%
%  hub_height = height of wind turbine tower [m]
general_ncount = general_ncount+1;
C = textscan(General_datastring(general_ncount,:),'%f');
hub_height = C{1,1};
%  Pitch = pitch of the blades [rad]
general_ncount = general_ncount+1;
C = textscan(General_datastring(general_ncount,:),'%f');
Pitch = C{1,1};
%  Nb_blades = number of blades
general_ncount = general_ncount+1;
C = textscan(General_datastring(general_ncount,:),'%f');
Nb_blades = C{1,1};
%  rotational_speed = rotational speed of the blades [rpm]
general_ncount = general_ncount+1;
C = textscan(General_datastring(general_ncount,:),'%f');
rotational_speed = C{1,1};
%  cutin_wind_speed = the cut in wind speed of the turbine [m/s]
general_ncount = general_ncount+1;
C = textscan(General_datastring(general_ncount,:),'%f');
cutin_wind_speed = C{1,1};
```

```
%  cutout_wind_speed = the cut in wind speed of the turbine [m/s]
general_ncount = general_ncount+1;
C = textscan(General_datastring(general_ncount,:),'%f');
cutout_wind_speed = C{1,1};
%% ------------------------------------------------------------------------
% This section converts and saves the wind turbine general data to proper
format
blade_ncount = 1;
%
%  Blade_length = length of the blade [m]
blade_ncount = blade_ncount+1;
C = textscan(Blade_datastring(blade_ncount,:),'%f');
blade_length = C{1,1};
%   distribution_span = the axial span distances for the chord and twist
distributions [m]
blade_ncount = blade_ncount+1;
C = textscan(Blade_datastring(blade_ncount,:), '%f %f %f %f %f %f %f %f %f
%f %f %f %f %f %f %f %f');
distribution_span = [C{1,:}];
%   chord_distribution = chord distribution [m]
blade_ncount = blade_ncount+1;
C = textscan(Blade_datastring(blade_ncount,:), '%f %f %f %f %f %f %f %f %f
%f %f %f %f %f %f %f %f');
chord_distribution = [C{1,:}];
%   twist_distribution = pitch of the blades [deg]
blade_ncount = blade_ncount+1;
C = textscan(Blade_datastring(blade_ncount,:), '%f %f %f %f %f %f %f %f %f
%f %f %f %f %f %f %f %f');
twist_distribution = [C{1,:}];
%  Nb_span_locations = number of axial span locations
blade_ncount = blade_ncount+1;
C = textscan(Blade_datastring(blade_ncount,:),'%f');
Nb_span_locations = C{1,1};
%-------------------------------------------------------------------------
% End of WTNOISE_input.m
%-------------------------------------------------------------------------
```

```
%%  WTNOISE_default_parameters.m
%  This script file defines default parameters
%
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
%
%  INPUTS:
%
%  OUTPUTS:
%   CAIR        = speed of sound
%   DENSITYAIR  = air density
%   pres_ref    = reference pressure in air
%   W_ref       = reference power in air
%
%   Nb_3rd_octave_band      = number of 1/3rd octave bands (30);
%   Oct_Freq_Band_3rd       = center frequencies of the
%                            1/3rd octave bands 12.5Hz through 10kHz
%   Oct_Band_Limits_3rd(Nb_3rd_octave_band,2) =
%                    lower and upper frequency limits of octave bands
%
%  ROUTINE USED:
%
%---------------------------------------------------------------------------
tstart = tic; % start clock to measure execution time
disp('    ==> WTNOISE_default_parameters.m')
%---------------------------------------------------------------------------
% Speed of sound [m/s] and air density [kg/m^3] are obtained as the
avergage of those given for each height
CAIR   =
20.04*((sum(temperature_velocity)/(length(temperature_velocity))))^0.5;   %
Speed of sound in [m/s]
DENSITYAIR = (sum(air_density)/(length(air_density)))  ;    % Air density in
[kg/m^3]
%--------------------------------------------------------------------------
% Reference pressure  p_ref = 20x10^-6 Pa
%           power     W_ref = 10^-12 watts
%
pres_ref  = 0.000020; % Pa
W_ref = 10^-12;       % watts
%---------------------------------------------------------------------------
% The 1/3rd octave frequency band center
%
Oct_Freq_Band_3rd  = [
10.0  ...
                        12.5   16.0   20.0   25.0   31.5   40.0   50.0
63.0   80.0   100.0  ...
                       125.0  160.0  200.0  250.0  315.0  400.0  500.0
630.0  800.0  1000.0  ...
                      1250.0 1600.0 2000.0 2500.0 3150.0 4000.0 5000.0
6300.0 8000.0 10000.0 12500.0, 16000.0, 20000.0];

Nb_3rd_octave_band = length(Oct_Freq_Band_3rd);
% 1/3rd oct. band number
Oct_Band_num_3rd = round(10*log10(Oct_Freq_Band_3rd));
%%------------------------------------------------------------------------
% End of WTNOISE_default_parameters.m
%%-------------------------------------------------------------------------
```

```
%% WTNOISE_rotational_positions.m
% This routine computes all the possible span and azymuth angular positions
% of the blade.
%
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
%
% INPUTS:
%
%
% OUTPUTS:
%
%
%
% ROUTINES USED: none
%--------------------------------------------------------------------------
disp('    ==> WTNOISE_rotational_positions.m')
%--------------------------------------------------------------------------
%   The length of each segment is calculated, for a defined value of ten
span
%   segments
%
Nb_span_segments = 20;
segment_length = (blade_length-distribution_span(1))/Nb_span_segments;
%
%--------------------------------------------------------------------------
%   The number of azymuth angular positions is defined
%
Nb_azymuth_angles = 45; % This number must be a multiple of 3
%--------------------------------------------------------------------------
%   Loop over the number of span segments and azymuth angles
rotation_count =0;
for n = 1:Nb_azymuth_angles+1

    for k = 1:Nb_span_segments


    rotation_count = rotation_count +1;

%   The axial span positions and azymuth angles are computed.

    blade_azymuth_angle(rotation_count) = (2*pi/Nb_azymuth_angles)*(n-1);
    span_axial_position(rotation_count) = ((2*k)-1)*(segment_length/2)+
distribution_span(1);

%   The rotational velocity for each span and azymuth angle is computed.

    rotational_velocity(rotation_count) =
(rotational_speed*2*pi/60)*span_axial_position(rotation_count); %[rad/s]

%   The initial segment location for each span axial position is
determined. It
%   always for a position parallel to the wind turbine tower (x,y,z).

    initial_segment_location = [0  0  span_axial_position(rotation_count)];
    %   The roatation matrix that determines the new location of the span
    %   axial position when rotated an azymuth angle
    Blade_rot_mat = zeros(3,3);
    Blade_rot_mat(1,1) = 1;
    Blade_rot_mat(1,2) = 0;
    Blade_rot_mat(1,3) = 0;
    Blade_rot_mat(2,1) = 0;
    Blade_rot_mat(2,2) = cos(blade_azymuth_angle(rotation_count));
    Blade_rot_mat(2,3) = sin(blade_azymuth_angle(rotation_count));
    Blade_rot_mat(3,1) = 0;
    Blade_rot_mat(3,2) = -sin(blade_azymuth_angle(rotation_count));
    Blade_rot_mat(3,3) = cos(blade_azymuth_angle(rotation_count));

    %  Source location in terms of a coordinate system at the center of the
    %  hub
    local_position(rotation_count,:) = (Blade_rot_mat *
initial_segment_location');

    source(k).azymuth(n).local_position = local_position(rotation_count,:);
    source(k).azymuth(n).span_location  =
span_axial_position(rotation_count);
    source(k).azymuth(n).azymuth_angle  =
blade_azymuth_angle(rotation_count);
    source(k).azymuth(n).height =
local_position(rotation_count,3)+hub_height;
    source(k).azymuth(n).rotation_velocity =
rotational_velocity(rotation_count);

    end
end

% The span and azymuth locations for each position are saved in an array.

rotation_positions = [reshape(blade_azymuth_angle,[],1)
reshape(span_axial_position,[],1)];

% The heights of each span and azymuth locations are saved in an array.

rotation_heights = reshape(local_position(:,3),[],1)+hub_height;

% The rotation velocities of each span and azymuth location are saved in an
array.

rotation_velocities = reshape(rotational_velocity,[],1);

%-----------------------------------------------------------------------
% End of routine WTNOISE_rotational_positions.m
%-----------------------------------------------------------------------
```

```
%% WTNOISE_wind_interpolation.m
% This routine computes values of the wind velocity according to the
heights given by the blade's
% rotation are interpolated with the wind input data.
%
%
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
%
% INPUTS:
%
% OUTPUT:
%
%
% ROUTINES USED: none
%--------------------------------------------------------------------------
disp('    ==> WTNOISE_wind_interpolation.m')
%--------------------------------------------------------------------------
% The values of the wind velocity according to the heights given by the
blade's
% rotation are interpolated with the wind input data.

wind_interp_velocities =
interp1(velocity_height,xwind_velocity,rotation_heights);

for n = 1:Nb_azymuth_angles+1

    for k = 1:Nb_span_segments

    source(k).azymuth(n).wind_interp_velocities  =
interp1(velocity_height,xwind_velocity,source(k).azymuth(n).height);

    end
 end

%--------------------------------------------------------------------------
% End of routine WTNOISE_wind_interpolation.m
%--------------------------------------------------------------------------
```

```
%% WTNOISE_twist_interpolation.m
% This routine computes values of the twist distribution according to the
% span locations, interpolated with the twist distribution input data.
%
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
%
% INPUTS:
%
% OUTPUT:
%
%
% ROUTINES USED: none
%--------------------------------------------------------------------------
disp('    ==> WTNOISE_twist_interpolation.m')
%--------------------------------------------------------------------------
% The values of the twist distribution according to the
% span locations, interpolated with the twist distribution input data.
twist_interpolated_values =
interp1(distribution_span,twist_distribution,span_axial_position);

for n = 1:Nb_azymuth_angles+1

    for k = 1:Nb_span_segments

        source(k).azymuth(n).twist_interpolated_values =
interp1(distribution_span,twist_distribution,span_axial_position(k)) ;

    end
end
%-------------------------------------------------------------------------
% End of routine WTNOISE_twist_interpolation.m
%-------------------------------------------------------------------------
```

```
%% WTNOISE_AOA.m
% This routine computed the angle of attack for every  azymuth angular and
% span position of the blade
%
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
% INPUTS:
%
% OUTPUT:
%
%
% ROUTINES USED: none
%-------------------------------------------------------------------------
disp('    ==> WTNOISE_AOA.m')
%-------------------------------------------------------------------------
% For each span position, the sum of the pitch and the twist is obtained.
% All angles for these computations are in radians.

Beta = (twist_interpolated_values*pi/180)+ Pitch;

for n = 1:Nb_azymuth_angles+1

    for k = 1:Nb_span_segments

source(k).azymuth(n).Beta =
(source(k).azymuth(n).twist_interpolated_values*pi/180)+ Pitch;

    end
end

%The calculation of the angle of attack.
%FOR PLOT
for m=1:length(wind_interp_velocities)
Gamma(m)  = atan(wind_interp_velocities(m)/rotation_velocities(m));
AOA (m)= (Gamma(m)-Beta(m));
end

% FOR OTHER CALCULATIONS
for n = 1:Nb_azymuth_angles+1
    for k = 1:Nb_span_segments

source(k).azymuth(n).Gamma =
atan(source(k).azymuth(n).wind_interp_velocities/source(k).azymuth(n).rotat
ion_velocity);
source(k).azymuth(n).AOA = source(k).azymuth(n).Gamma-
source(k).azymuth(n).Beta;
    end
end

%-------------------------------------------------------------------------
% End  of WTNOISE_AOA.m
%-------------------------------------------------------------------------
```

```
%% WTNSRC_AOA_plot.m
% This routine plots the angle of attack for every span and azymuth angular
% position
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
%
% INPUTS:
%
%
% OUTPUT:
%
%
% ROUTINES USED: none
%--------------------------------------------------------------------------
disp('    ==> WTNOISE_AOA_plot.m')
%--------------------------------------------------------------------------
mkdir('AERO Graphics')
% The circular grid is created
[angle_wind, radius_wind] =
meshgrid(0:360,distribution_span(1):blade_length);
% The AOA data is interpolated along a surface at the query points
% specified by the variables: angle and radial positions along the turbine
% rotation.
C_wind =
griddata(rotation_positions(:,1)*180/pi,rotation_positions(:,2),AOA*180/pi,
angle_wind,radius_wind);
% The querry points are changed from a polar coordinates to cartesian
coordinates system
[x_wind, y_wind] = pol2cart(angle_wind*pi/180,radius_wind);
% The surface is plotted
p_wind = surf(x_wind,y_wind,C_wind);
view([90,90])
set(p_wind, 'edgecolor','none')
colormap(jet(32))
colorbar;
title('AOA for every span and azymuth positions');
xlabel('z[m]');
ylabel('y[m]');
saveas(gcf,'AERO Graphics/AOA.fig');
close all
%--------------------------------------------------------------------------
% End of WTNOISE_AOA_plot.m
%--------------------------------------------------------------------------
```

```
%% WTNOISE_relative_velocities.m
% This routine computed the relative wind velocities for each azymuth
angular and span position of the blade.
%
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
% INPUTS:
%
%
% OUTPUT:
%
%
% ROUTINES USED: none
%--------------------------------------------------------------------------
disp('    ==> WTNOISE_relative_velocities.m')
%--------------------------------------------------------------------------

%The calculation of the wind relative velocities.
% FOR PLOT
for m=1:length(wind_interp_velocities)

relative_velocity(m) =
(((wind_interp_velocities(m))^2+(rotation_velocities(m))^2)^0.5);

end

% FOR COMPUTATIONS
for n = 1:Nb_azymuth_angles+1

    for k = 1:Nb_span_segments

      source(k).azymuth(n).relative_velocity =
((((source(k).azymuth(n).wind_interp_velocities)^2)+((source(k).azymuth(n).
rotation_velocity)^2))^0.5);


    end
end

%--------------------------------------------------------------------------
% End  of WTNOISE_relative_velocities.m
%--------------------------------------------------------------------------
```

```
%% WTNOISE_relative_velocities_plot.m
% This routineplots the relative velocities for every span and azymuth
angular
% position
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
%
% INPUTS:
%
%
% OUTPUT:
%
%
% ROUTINES USED: none
%-----------------------------------------------------------------------------
disp('    ==> WTNOISE_relative_velocities_plot.m')
%----------------------------------------------------------------------------
% The circular grid is created
[angle_wind, radius_wind] =
meshgrid(0:360,distribution_span(1):blade_length);
% The relaative velocities data is interpolated along a surface at the
query points
% specified by the variables: angle and radial positions along the turbine
% rotation.
C_relative_vel =
griddata(rotation_positions(:,1)*180/pi,rotation_positions(:,2),relative_ve
locity,angle_wind,radius_wind);
% The querry points are changed from a polar coordinates to cartesian
coordinates system
[x_wind, y_wind] = pol2cart(angle_wind*pi/180,radius_wind);
% The surface is plotted
p_relative_vel = surf(x_wind,y_wind,C_relative_vel);
view([90,90])
set(p_relative_vel, 'edgecolor','none')
colormap(jet(32))
colorbar;
title('Relative velocities for every span and azymuth positions');
xlabel('z[m]');
ylabel('y[m]');
saveas(gcf,'AERO Graphics/Relative_velocity.fig');
close all
%-----------------------------------------------------------------------------
% End of WTNOISE_relative_velocities_plot.m
%-----------------------------------------------------------------------------
```

```
%% WTNOISE_distance_source_grid.m
% This routine computed:
% a) the distance between source and grid
% b) the unit vector from source to each grid point
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
%
% INPUTS:
%   source_location = coordinates of the sources
%
%   Nb_grid_points    = number of grid points
%
% OUTPUT:
%   distance_source_observer(Nb_grid_points,1) = distance from source to
%                                                grid point
%   unit_vec_src_grid(Nb_grid_points,3)        = unit vector from source to
%                                              grid point
%
% ROUTINES USED: none
%
%---------------------------------------------------------------------------
disp('    ==> WTNOISE_distance_source_grid.m')
%---------------------------------------------------------------------------
% Compute 1) distance from the dipole to each point of the grid
%         2) unit vector from source-grid
%
for m = 1:Nb_azymuth_angles+1
   for k = 1:Nb_span_segments
    grid_count = 0;

    for n = 1:Nb_sphere_points
     grid_count = grid_count+1;
    % for the nth grid, compute distance source-grid
    distance_source_observer(grid_count) = ( (sphere_coordinates(n,1) -
source(k).azymuth(m).local_position(1))^2 + ...
                                       (sphere_coordinates(n,2) -
source(k).azymuth(m).local_position(2))^2 + ...
                                       (sphere_coordinates(n,3) -
source(k).azymuth(m).local_position(3))^2 )^0.5;
    % for the nth grid, compute unit vector from source-grid
    unit_vector_grid (grid_count,:) = (sphere_coordinates(n,:) -
source(k).azymuth(m).local_position) / ...
                                   distance_source_observer(grid_count);

    end
    source(k).azymuth(m).distance_source_observer =
distance_source_observer;
    source(k).azymuth(m).unit_vector_grid  = unit_vector_grid;
   end
end

%---------------------------------------------------------------------------
% End of routine WTNOISE_distance_source_grid.m
%---------------------------------------------------------------------------
```

```
%% WTNOISE_experimental_SPL.m
% This routine compute:
% a) The source strength as the spatial average sound pressure
%    at a distance of 1 m, using experimental wind tunnel test data.
%
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
%
% INPUTS:
%
%
%
% OUTPUT:
%
%
% ROUTINES USED: none
%
%---------------------------------------------------------------------------
disp('    ==> WTNOISE_experimental_SPL.m')
%---------------------------------------------------------------------------
% Compute the source strength as the spatial average sound pressure
% at a distance of 1 m
exp_chord = 0.9; % the airfoil chord from the wind tunnel experiment is
defined

% The coefficients required for the experimental expression obtained with
% wind tunnel data, are loaded
load('coefficient1.mat')
load('coefficient2.mat')
load('coefficient3.mat')

p_counter = 0;

for m = 1:Nb_azymuth_angles+1

    for k = 1:Nb_span_segments

       p_counter = p_counter + 1;

        for n = 1:Nb_3rd_octave_band



 % For each band & local position, the strouhal number is obtained

 st(p_counter,n) =
(Oct_Freq_Band_3rd(n)*exp_chord)/(source(k).azymuth(m).relative_velocity);


 % The sound pressure levels are obtained, using the experimental
 % expression
 SPL(p_counter,n) =
(((coefficient1(1)*(source(k).azymuth(m).AOA*180/pi)^coefficient1(2))+(coef
ficient1(3)))...

*st(p_counter,n)^((coefficient2(1)*(source(k).azymuth(m).AOA*180/pi)^coeffi
cient2(2))+(coefficient2(3))))...
             +
((coefficient3(1)*(source(k).azymuth(m).AOA*180/pi)^coefficient3(2))+(coeff
icient3(3))) ;

% The data is re-scaled for the used turbine airfoil span & relative
% velocity

 SPL_velocity_corrected(p_counter,n) =
SPL(p_counter,n)+50*log10(source(k).azymuth(m).relative_velocity/1);

 SPL_span_corrected(p_counter,n) =
SPL_velocity_corrected(p_counter,n)+10*log10(segment_length/1);



        end
    source(k).azymuth(m).st = st(p_counter,:);
    source(k).azymuth(m).SPL_span_corrected =
SPL_span_corrected(p_counter,:);
end
end

% The data is re-scaled according to the distance between the source and
% the observer. This means that for each local position (source position),
% there are distances towards each point on the sphere grid (observer
position)
for m = 1:Nb_azymuth_angles+1

    for k = 1:Nb_span_segments


grid_count1 = 0;
 for n = 1:Nb_sphere_points
grid_count1 = grid_count1 + 1;



 SPL_distance_corrected(grid_count1,:) =
source(k).azymuth(m).SPL_span_corrected + 20*log10(1/1);

 end
 source(k).azymuth(m).SPL_distance_corrected = SPL_distance_corrected;

    end

end
%-------------------------------------------------------------------------
% End of routine WTNOISE_experimental_SPL.m
%-------------------------------------------------------------------------
```

```
%% WTNOISE_source_directivity.m
% This routine computes the directivity of the source for each point on the
% grid, for each of the sources.
%
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
%
% INPUTS:
%
%
% OUTPUT:
%
%
% ROUTINES USED: none
%--------------------------------------------------------------------------
% Compute the directivity factor for each source
%--------------------------------------------------------------------------
disp('    ==> WTNOISE_source_directivity.m')
%--------------------------------------------------------------------------
% the unit vector corresponding to each source at each position
unit_vec_src = [1 0 0]; % a unit vector on the x direction is determined

% the unit vector corresponding to each vector is rotated accordingly with
% each blade segment rotation
for n = 1:Nb_azymuth_angles+1

    for k = 1:Nb_span_segments

        source_rot_angle = (pi)-source(k).azymuth(n).Beta;
        source(k).azymuth(n).source_rot_angle = source_rot_angle;

        % Rotation matrix
        Rot_mat = zeros(3,3);
        Rot_mat(1,1) = cos(-source_rot_angle);
        Rot_mat(1,2) = sin(-source_rot_angle);
        Rot_mat(1,3) = 0;
        Rot_mat(2,1) = -sin(-source_rot_angle);
        Rot_mat(2,2) = cos(-source_rot_angle);
        Rot_mat(2,3) = 0;
        Rot_mat(3,1) = 0;
        Rot_mat(3,2) = 0;
        Rot_mat(3,3) = 1;

        unit_vec_src_rot1 = Rot_mat*unit_vec_src';
        %unit_vec_src_rot_array1 =
repmat(unit_vec_src_rot1',Nb_sphere_points,1);
        source(k).azymuth(n).unit_vec_src_rot1 = unit_vec_src_rot1;
        end
end

unit_vec_rot_count=0;
for n = 1:Nb_azymuth_angles+1
```

```
    for k = 1:Nb_span_segments

        unit_vec_rot_count = unit_vec_rot_count + 1;

        blade_azymuth_angle(unit_vec_rot_count) =
(2*pi/Nb_azymuth_angles)*(n-1);

        Src_rot_mat = zeros(3,3);
        Src_rot_mat(1,1) = 1;
        Src_rot_mat(1,2) = 0;
        Src_rot_mat(1,3) = 0;
        Src_rot_mat(2,1) = 0;
        Src_rot_mat(2,2) = cos(blade_azymuth_angle(unit_vec_rot_count));
        Src_rot_mat(2,3) = sin(blade_azymuth_angle(unit_vec_rot_count));
        Src_rot_mat(3,1) = 0;
        Src_rot_mat(3,2) = -sin(blade_azymuth_angle(unit_vec_rot_count));
        Src_rot_mat(3,3) = cos(blade_azymuth_angle(unit_vec_rot_count));


        source(k).azymuth(n).unit_vec_src_rot2 =
Src_rot_mat*source(k).azymuth(n).unit_vec_src_rot1;
        unit_vec_src_rot_array2 =
repmat(source(k).azymuth(n).unit_vec_src_rot2',Nb_sphere_points,1);
        source(k).azymuth(n).unit_vec_src_rot = unit_vec_src_rot_array2;
    end
end
% The directivity is obtained as the square of the dot product between the
% unit vector of each source and the unit vector corresponding to the
% direction from each source towards each grid point.
for n = 1:Nb_azymuth_angles+1

    for k = 1:Nb_span_segments

        d_count = 0;

        for m = 1:Nb_sphere_points
        d_count = d_count + 1;

        Directivity_factor(d_count) =
((dot(source(k).azymuth(n).unit_vec_src_rot(m,:),source(k).azymuth(n).unit_
vector_grid(m,:))) ^2);
        Directivity_angle(d_count) =
acos(dot(source(k).azymuth(n).unit_vec_src_rot(m,:),source(k).azymuth(n).un
it_vector_grid(m,:)))*180/pi;

        end
        source(k).azymuth(n).Directivity_factor = Directivity_factor;
        source(k).azymuth(n).Directivity_angle= Directivity_angle;
    end
end

%--------------------------------------------------------------------------
% End of routine WTNOISE_source_directivity.m
%--------------------------------------------------------------------------
```

```
%% WTNOISE_predicted_SPL.m
% This routine computes the predicted sound pressure levels on all the
% grid points, from each source in the blade and all 1/3rd frequency bands.
%
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
%
% INPUTS:
%
% OUTPUT:
%
% ROUTINES USED: none
%-------------------------------------------------------------------------
% Compute the predicted sound pressure levels
%-------------------------------------------------------------------------
disp('    ==> WTNOISE_predicted_SPL.m')
%-------------------------------------------------------------------------


for m = 1:Nb_azymuth_angles+1

    for k = 1:Nb_span_segments

    spl_count = 0;

    for n = 1:Nb_sphere_points
    spl_count = spl_count+1;

    SPL_predicted(spl_count,:)=
source(k).azymuth(m).SPL_distance_corrected(n,:)+...

(10*log10((source(k).azymuth(m).Directivity_factor(n))/((source(k).azymuth(
m).distance_source_observer(n))^2)));

    end
    source(k).azymuth(m).SPL_predicted = SPL_predicted;

    end
end
%-------------------------------------------------------------------------
% End of routine WTNOISE_predicted_SPL.m
%-------------------------------------------------------------------------
```

```
%% WTNOISE_OASPL.m
% This routine computes the overall A-weighted sound pressure levels from
% each source, for all the grid points.
%
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
%
% INPUTS:
%
%
% OUTPUT:
%
%
% ROUTINES USED: none
%---------------------------------------------------------------------------
disp('    ==> WTNOISE_OASPL.m')
%---------------------------------------------------------------------------
% The A-weighted correction is determined for each frequency band
for n = 1:Nb_3rd_octave_band

    ra(n)  = ( 12200^2*Oct_Freq_Band_3rd(n)^4)  /
...
             ( (Oct_Freq_Band_3rd(n)^2 + 20.6^2 ) *
...
             ( (Oct_Freq_Band_3rd(n)^2 + 107.7^2) *
(Oct_Freq_Band_3rd(n)^2+737.9^2) )^0.5 *  ...
               (Oct_Freq_Band_3rd(n)^2 + 12200^2)
);
    dba(n) = 2 + 20*log10(ra(n));

end


% The A-weighted correction is applied to all the sound pressure levels
% (all sources, grid points & frequency bands)

for m = 1:Nb_azymuth_angles+1

    for k = 1:Nb_span_segments

        spl1_count = 0;

        for n = 1:Nb_3rd_octave_band
        spl1_count = spl1_count + 1;

            SPL_A_weighted(:,spl1_count) =
source(k).azymuth(m).SPL_predicted(:,n)+dba(n);

        end

        source(k).azymuth(m).SPL_A_weighted = SPL_A_weighted;
    end
end


% The value of 10^(Lp/10), for all the sound pressure levels is obtained.

for m = 1:Nb_azymuth_angles+1

    for k = 1:Nb_span_segments

        for n = 1:Nb_3rd_octave_band
            for l = 1:Nb_sphere_points

        power(l,n) = 10.^(source(k).azymuth(m).SPL_A_weighted(l,n)/10);

            end
        end
        source(k).azymuth(m).power = power;
    end
end


% The OASPL is obtained for al soruces & all grid points.
for m = 1:Nb_azymuth_angles+1

    for k = 1:Nb_span_segments

        for n = 1:Nb_sphere_points


        OASPL(n) = 10*log10(sum(source(k).azymuth(m).power(n,:)));


        end
            source(k).azymuth(m).OASPL = OASPL;
        end

    end

%-------------------------------------------------------------------------
% End  of WTNOISE_OASPL.m
%-------------------------------------------------------------------------
```

```
%% WTNOISE_OASPL_per_blade.m
% This routine computes the overall A-weighted sound pressure levels from
% each blade position, for all the grid points.
%
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
%
% INPUTS:
%
%
% OUTPUT:
%
%
% ROUTINES USED: none
%---------------------------------------------------------------------------
disp('    ==> WTNOISE_OASPL_per_blade.m')
%---------------------------------------------------------------------------
% The value of 10^(Lp/10), for all the sound pressure levels is obtained.

for m = 1:Nb_azymuth_angles+1

    for k = 1:Nb_span_segments


        source(k).azymuth(m).OASPL_power =
10.^(source(k).azymuth(m).OASPL/10);


    end
end

% The OASPL for each blade position is obtained
 for m = 1:Nb_azymuth_angles+1

     blade_addition = zeros(1,Nb_sphere_points);

    for k = 1:Nb_span_segments

        blade_addition = source(k).azymuth(m).OASPL_power + blade_addition;

    end

    OASPL_blade(m).blade_position = 10*log10(blade_addition);

 end

%---------------------------------------------------------------------------
% End  of WTNOISE_OASPL_per_blade.m
%---------------------------------------------------------------------------
```

```
%% WTNOISE_OASPL_per_position.m
% This routine computes the overall A-weighted sound pressure levels from
% each position of the three blades of the turbine, for all grid points.
%
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
%
% INPUTS:
%
%
% OUTPUT:
%
%
% ROUTINES USED: none
%--------------------------------------------------------------------------
disp('    ==> WTNOISE_OASPL_per_position.m')
%--------------------------------------------------------------------------

% The sound pressure levels from each blade, for all grid points, are
% determined by the value of 10^(Lp/10).

for m = 1:Nb_azymuth_angles+1

power_blade_OASPL = 10.^(OASPL_blade(m).blade_position/10);
OASPL_blade(m).power_blade_OASPL = power_blade_OASPL;

end

% The number of positions of the three blades for 1/3rd of a revolution is
% determined, as well as the corresponding predicted OASPL for each
% position of the 3 blades.

turbine_positions = Nb_azymuth_angles/3;

for m = 1:turbine_positions

    total_position_power = OASPL_blade(m).power_blade_OASPL +
OASPL_blade(m+turbine_positions).power_blade_OASPL+...
        OASPL_blade(m+(2*turbine_positions)).power_blade_OASPL;

    OASPL_turbine(m).OASPL_position = 10*log10(total_position_power);
end

%--------------------------------------------------------------------------
% End  of WTNOISE_OASPL_per_position.m
%--------------------------------------------------------------------------
```

```
%% WTNOISE_plot.m
% This routine plots and saves the OASPL for every position of the blades
%
% AUTHORS: Sterling McBride Granda
%          Ricardo A. Burdisso, Professor
%
% DATE:NOVEMBER 2014
%
%
% ROUTINES USED: none
%
%--------------------------------------------------------------------------
--------------
% Create a new folder on the directory
mkdir('OASPL Graphics')
% Loop over each position of the blades
for l = 1:turbine_positions

 %Define the color distribution
 color = OASPL_turbine(l).OASPL_position;

 % develop a uniform grid
 [Az, El] = meshgrid(0:360,-90:90);
 % interpolate nonuniformly spaced points
 C =
griddata(sphere_angles(:,1)*(180/pi),sphere_angles(:,2)*(180/pi),color,Az,E
l);
 % convert to cartesian coordinates
 [x, y, z] = sph2cart(Az*pi/180,El*pi/180,r);
 % plot
 h = surf(x,y,z,C);
 %axis equal off vis3d
 lighting phong
 %camlight('right')
 set(h, 'edgecolor','none')
 colorbar;
 caxis([0 35]);
 xlabel('x coordinates (Rotation Plane)')
 ylabel('y coordinates')
 zlabel('z coordinates')
 title('Overall  A-weigthed Sound Pressure Levels on Sphere')
 F(l) = getframe(gcf);
 saveas(h,sprintf('OASPL Graphics/FIG%d.fig',l))
 close all
end
% There are any number of turbine positions per 1/3rd of revolution,
therefore the fps
% is obtained, for a real OSPL-time video.
frames_per_second = 3*rotational_speed*turbine_positions/60;

movie2avi(F, 'OASPL.avi','fps',frames_per_second,'compression','None');
%--------------------------------------------------------------------------
% End of WTNOISE_plot.m
%--------------------------------------------------------------------------
```