# Development of a script for parametrized 3D-meshing around a wind turbine blade using pointwise

**Submitted by**

Karan Jayeshbhai Soni

**Supervising Professors:**

**Primary evaluator:** Prof. Dr. A. P. Schaffarczyk

**External Supervisor and guide**: Brandon Lobo

**Date of submission:** 31.08.2023

## List of Figures

## List of Scripts

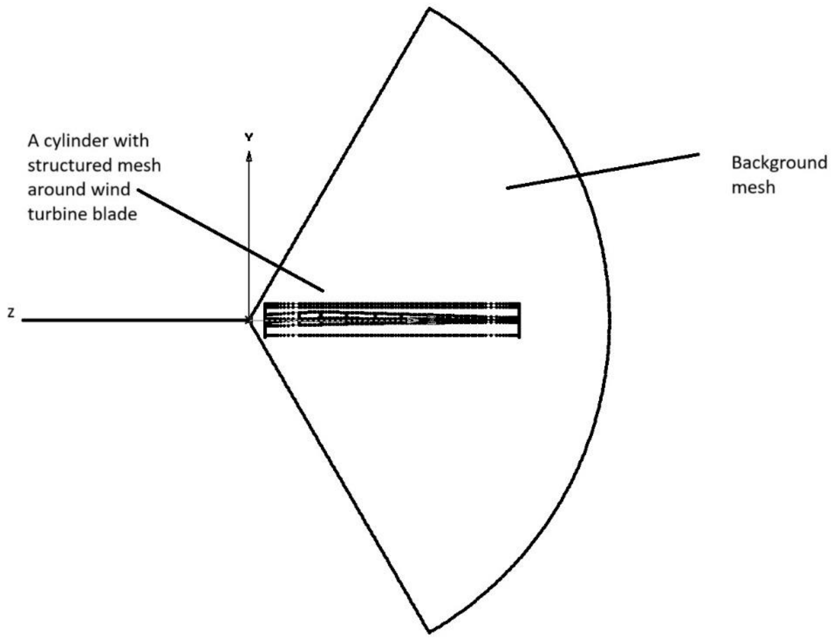
## Table of Contents

## 1. Introduction

The goal of this project is to create a scripting tool for creating a parameterized 3D mesh around wind turbine blades. Using CAD information from the IEA wind dataset is the basis of this project. The geometric design of the wind turbine blade is carefully developed with Pointwise software. Transfinite interpolation (TFI), also known as the Gordon Hall method, is used to create a structured cylinder that encloses the wind turbine blade, which is the main new ideas of the project. The mesh's quality will be determined using the Pointwise software to ensure optimal efficiency.

There are two separate steps to the scripting process. The first step involves the creation of a structured cylinder that includes a fine mesh. The creation of a background mesh made up of 120-degree segment blocks is included in the latter part. This all-inclusive method of mesh creation assures precision and effectiveness in capturing the complex shape of wind turbine blades.

Researchers analyse the complex processes of this scripting-based mesh creation throughout this study, from the collection of CAD data to the use of Pointwise software for quality assessment.



(Figure 1.1 Overall fundamental design idea)

## 2. Coding Languages and Tools

### 2.1 Tool Command Language (TCL):

The open source, interpreted programming language TCL, also known as Tool Command Language, is well recognised for its scripting features. Variables, procedures, and control structures are among the features of this language that users are familiar with from programming. Tcl is very useful for automating processes and communications in a variety of software contexts. It helps users to simplify difficult processes and produce effective, dependable outcomes. Using Tcl to automate the extraction of crucial information from CAD files for wind turbine blade design could serve as an example.

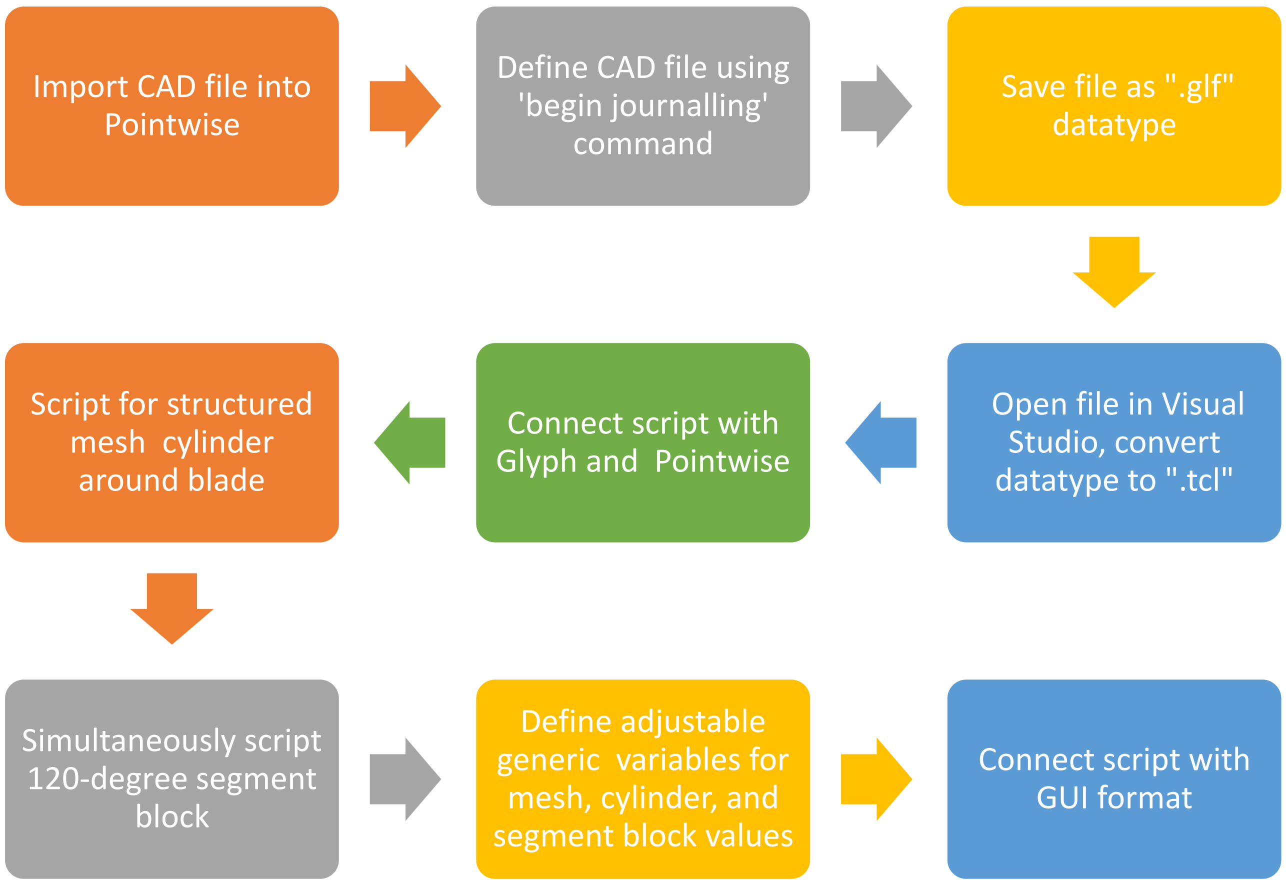
### 2.2 Glyph Scripting Language:

Users can produce meshes automatically and in accordance with their specifications with the help of Glyph, a scripting language that is deeply integrated into the Pointwise meshing software. Glyph gives users immediate access to Pointwise fundamental features and ideas. It was created as an extension of the Tcl programming language. The meshing process can be simplified and customised to the unique needs of wind turbine blade design by using Glyph. A Glyph script, for instance, may automate the creation of structured mesh pieces around complex blade shapes, improving simulation precision.

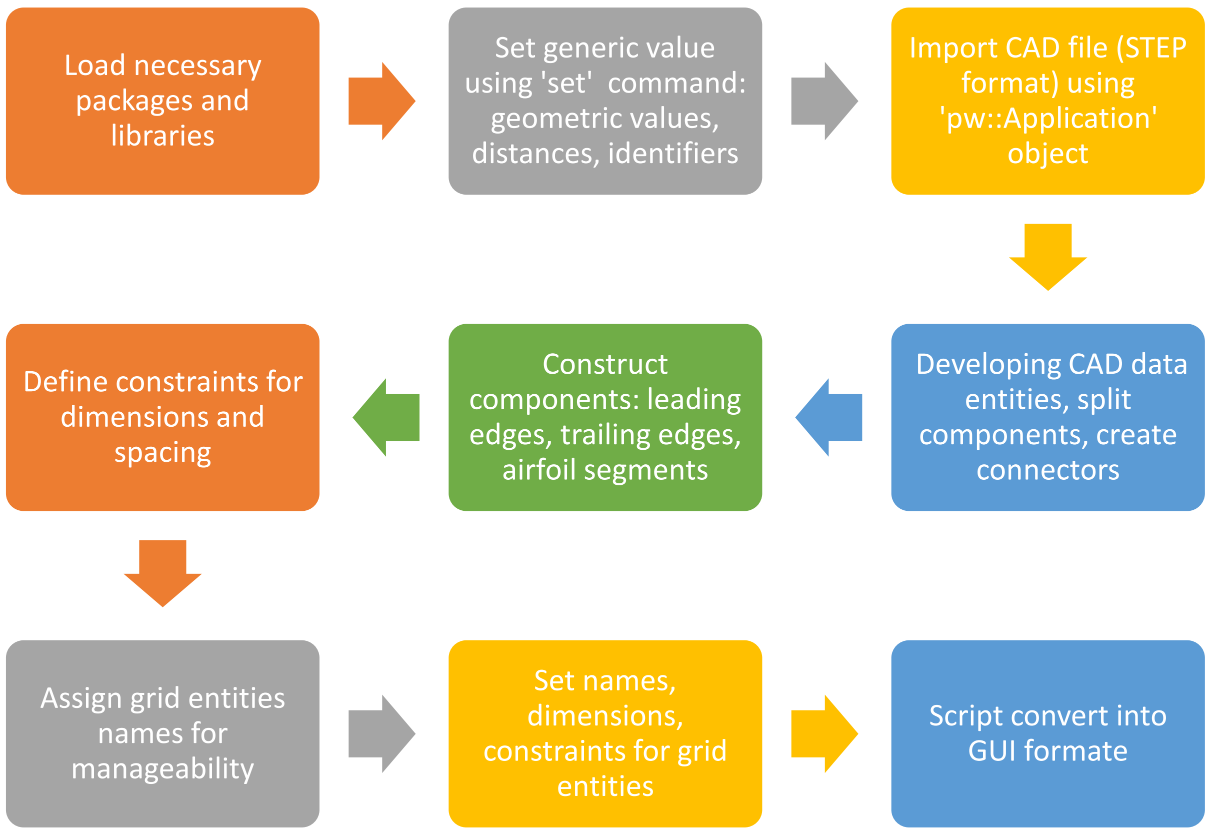
## 3. Project Workflow

The project workflow involves two methodologies: "Overall Project Methodology" and "Script Methodology."

### 3.1 Overall project methodology


Import CAD file into Pointwise
Define CAD file using 'begin journalling' command
Save file as ".glf" datatype
Open file in Visual Studio, convert datatype to ".tcl"
Connect script with Glyph and Pointwise
Script for structured mesh cylinder around blade
Simultaneously script 120-degree segment block
Define adjustable generic variables for mesh, cylinder, and segment block values
Connect script with GUI format

## 3.2 Script Methodology


Load necessary packages and libraries
Set generic value using 'set' command: geometric values, distances, identifiers
Import CAD file (STEP format) using 'pw::Application' object
Developing CAD data entities, split components, create connectors
Construct components: leading edges, trailing edges, airfoil segments
Define constraints for dimensions and spacing
Assign grid entities names for manageability
Set names, dimensions, constraints for grid entities
Script convert into GUI formate

## 4. Parameterization and Geometry Handling

This section explains how the script efficiently parameterizes the 3D meshing procedure and interacts with blade geometry data from sources like CAD files acquired from IEA wind to produce an exact and efficient mesh around the blade.

### 4.1 Parameterization for Adaptability:

For the script to work with different blade geometries and dimensions, it is essential that it be able to parameterize the 3D meshing process. The script allows users to modify important elements of the meshing process by including movable parameters, sometimes known as "generic variables." Geometric values, distances, dimensions, and other important factors are some of these variables. Users can simply change aspects like cylinder dimensions, and 120-degree segment block properties by adjusting these parameters. With the help of this flexible parameterization, the script can produce custom meshes that are suitable for certain wind turbine blade designs, enabling effective simulation and analysis.

### 4.2 Interaction with Blade Geometry Data:

A key component of the script's functioning is how it interacts with blade geometry information. Importing CAD files with the complex airfoil designs of wind turbine blades first happens. The script uses the pw::Application object, which is a crucial component of the pointwise software, to import and work with the CAD data entities. To achieve precision during meshing, this process involves tasks like dividing the airfoil into separate parts and creating connectors.

The foundation for developing numerous components that help in the meshing process is provided by the imported CAD data. The script builds crucial elements including leading edges, trailing edges, and airfoil sections by combining segments, connectors, and splines. To ensure the precision and dependability of the mesh, carefully set parameters on size and spacing are used. The assignment of named entities to grid components allows for effective management and referencing.

### 4.3 Achieving Customization Through Parameterization:

In the customizability of the script, the relationship between parameterization and geometry handling is particularly significant. This flexibility is coupled with the modification of blade geometry data, allowing the script to produce meshes that are specific to a variety of wind turbine blade designs. Users can fine segment block distribution, and cylinder dimensions through parameter-driven modifications, all of which have significant effects on the simulation results and accuracy.

# 5. Mesh Generation Techniques

The Transfinite Interpolation (TFI) technique, often known as the Gordon Hall algorithm, is principally used in the script's mesh generating process. This method is essential for producing structured mesh around wind turbine blade.

## 5.1 Transfinite Interpolation (TFI) or Gordon Hall Algorithm:

The basis of the script's structured meshing strategy is the Transfinite Interpolation (TFI) technique, sometimes referred to as the Gordon Hall algorithm. How to use this method is as follows:

### Step 1: Draw a Circle

Commence by drawing a circle, establishing the initial domain for meshing operations.

### Step 2: Generate the 'O-Mesh' and Split

In the circular domain, create an "O-Mesh" to be used as the starting point for mesh refining.

Set the stage for further operations by strategically splitting the "O-Mesh".

### Step 3: Draw a 120-Degree Segment

Draw a 120-degree section outside the circle's boundaries.

### Step 4: Enclose the Block

Set block boundaries around the 120-degree segment to enclose it. The further development of the Unstructured mesh is built on this stage.

# 6. Script Automation: Changing Dimensions and Values

The range of possibilities of the script allows for the easy changing of dimensions and values, modifying the meshing process to requirements. This section explains how to change important factors, such as the structured cylinder's size and the 120-degree block's orientation. It also looks at the script's graphical user interface (GUI), where these modifications are smoothly incorporated into its operation.

## 6.1 Modifying Cylinder and Block Dimensions:

When the script executes, a table with 10 generic values is displayed. Critical features of the meshing process, such as the number of connectors, spacing constraints, cylinder dimensions, and more, are affected by these variables taken together. The GUI format of the script is represented by this table, which enables users to make modifications by altering these values directly.

6.1.1 Number of Connectors: Find the correct value in the GUI table and change it to your specifications to change the number of connectors.

6.1.2 Spacing Constraint: Similarly, modify the spacing constraint (first cell height) as necessary, ensuring its alignment with the simulation objectives.

## 6.2 Adapting Cylinder Characteristics:

The script offers precise control over the structure of the cylinder while simplifying the changing of its size.

6.2.1 Cylinder Length (z1 and z2): In the GUI table, look for the values corresponding to z1 and z2 to change the cylinder's length. These numbers represent the cylinder's vertical axis length. To get the desired cylinder length, modify them.

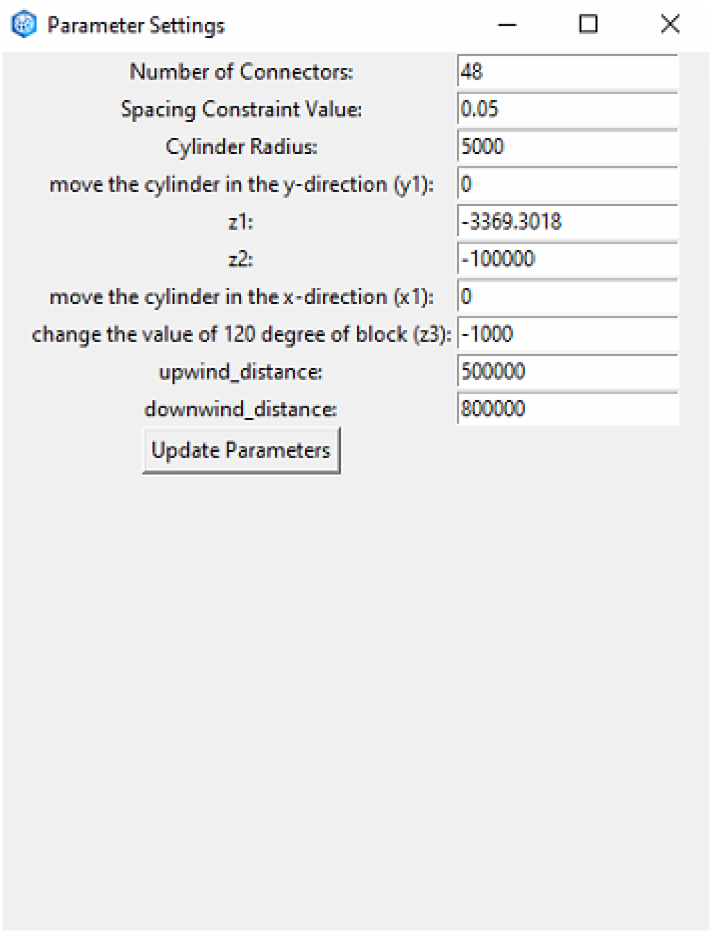
6.2.2 Cylinder Radius: In the GUI table, look for the value corresponding to Cylinder Radius to make modifications to the cylinder's radius. To successfully change the cylinder's horizontal measurements, change this value.

## 6.3 Positional Adjustments:

The script makes it easier to move the cylinder's position dynamically.

6.3.1 Shifting in the x-Direction: To move the cylinder in the x-direction, change the value of x1 in the GUI table.

6.3.2 Moving along the y-Direction: In a similar manner, change the value of y1 to move the cylinder in the y-direction.



| | |
|---|---|
| Number of Connectors: | 48 |
| Spacing Constraint Value: | 0.05 |
| Cylinder Radius: | 5000 |
| move the cylinder in the y-direction (y1): | 0 |
| z1: | -3369.3018 |
| z2: | -100000 |
| move the cylinder in the x-direction (x1): | 0 |
| change the value of 120 degree of block (z3): | -1000 |
| upwind_distance: | 500000 |
| downwind_distance: | 800000 |

(Figure 6.1 GUI format of script)

## 6.4 Wind Turbine Blade Replacement:

There are a few simple processes involved in changing the wind turbine blades:

I. To change the script, open it and go to the "IMPORT FILE" section on script (refer script 6.1). Change this to point to the new wind turbine blade you want to mesh by updating the file location.

II. Set the Wind Turbine Blade Geometry in Pointwise: Set the wind turbine blade geometry in the Pointwise programme. The four main components that must be specified are the root side airfoil, tip side airfoil, leading edge, and trailing edge.

III. Execute the script after modifying it to point to the new file location and defining the new blade geometry. Automation of the script will make it easier to create a structured mesh adapted to the recently introduced wind turbine blade.

```
# ------------------------------------------------------------
# IMPORT FILE
# -------------------------------------------------------------

set IMPORT_FILE [pw::Application begin DatabaseImport]
$IMPORT_FILE initialize -strict -type Automatic
{C:/Users/soni/Desktop/Airfoil Data/CIG10MW-blade.STEP}
$IMPORT_FILE setAttribute FileUnits Millimeters
$IMPORT_FILE read
$IMPORT_FILE convert
$IMPORT_FILE end
```

(Script 6.1 Script import CAD file)

# 7. Integration with Pointwise: Script and Pointwise interface for meshing tasks

The foundation of the meshing process is the combination of the script and Pointwise software. Through this interface, the script may interact with and command the Pointwise the programme, making it easier to do challenging meshing jobs. Here is an explanation of how this interface works:

## 7.1 Interface with Script-Glyph:

Through the usage of Glyph, a scripting language built into the Pointwise software, the integration is performed. Glyph serves as a pathway by which external scripts, like the one you've created, can speak to and give instructions to the Pointwise application. The script essentially communicates via Glyph, which then transforms the commands into actions inside of Pointwise.

```
# Load Pointwise Glyph package and Tk
package require Tk
package require PWI_Glyph 6.22.1
```

(Script 7.1 How to import Glyph to connect Pointwise with Tk/Tcl programming)

## 7.2 Meshing Procedures Execution:

When the script is run, Pointwise receives commands and instructions in Glyph. The methods included in these instructions are necessary to build the structured mesh that goes around the wind turbine blade. The script gives Pointwise instructions to perform complex meshing activities such as building the structured cylinder, producing the 120-degree segment block, providing dimensions, defining connectors, and other related tasks.

## 7.3 Data Exchange and Manipulation

The script communicates with the entities and data structures of Pointwise in a way that is flexible. The geometry and mesh elements in Pointwise's database are modified. For instance, the script might direct Pointwise to adjust lines, connectors, and segments such that the meshing entities appropriately reflect the geometry of the blade.

## 7.4 Guided Meshing Process:

The script essentially guides Pointwise through the mesh generation process. It provides a series of step-by-step instructions that enable Pointwise to generate, refine, and structure the mesh elements as required.

## 7.5 Automaton and effectiveness:

The meshing procedure is significantly more automated and efficient as a result of the script's interaction with Pointwise. The script now manages tasks that would ordinarily require manual execution, saving designers of time-consuming tasks. By reducing the chance of errors and speeding up the meshing workflow, this automation increases productivity and frees engineers to concentrate on more in-depth studies.

# 8. Summary of Project Achievements (Results)

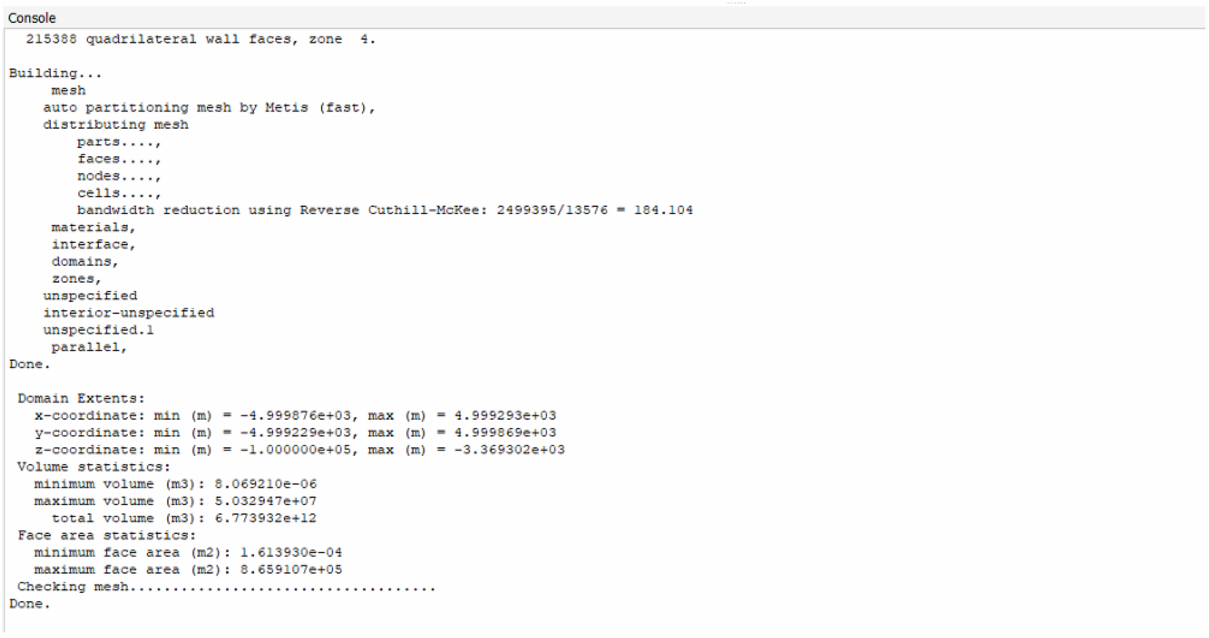
In the subject of wind turbine blade meshing, the project's efforts resulted in the achievement of key objectives. The project's main successes can be summarised in the following list:

## 8.1 Utilization of Original CAD File:

The project begins by following the instructions in the original CAD file. This initial phase makes sure that the complicated geometry of the wind turbine blades becomes the basis for further meshing processes.

## 8.2 Structured Cylinder Design and Meshing:

The project's successful design of a structural cylinder surrounds the wind turbine blade is a key result. A special script is used to carry out this process, producing a carefully defined mesh consequently. This structural mesh has been carefully adjusted to fit the complex shape of the blade. The resulting mesh has been designed to be integrated with the ANSYS Fluent software, allowing for thorough simulation and analysis.



Console

```
  215388 quadrilateral wall faces, zone  4.

Building...
     mesh
    auto partitioning mesh by Metis (fast),
    distributing mesh
        parts....,
        faces....,
        nodes....,
        cells....,
        bandwidth reduction using Reverse Cuthill-McKee: 2499395/13576 = 184.104
     materials,
     interface,
     domains,
     zones,
    unspecified
    interior-unspecified
    unspecified.1
     parallel,
Done.

 Domain Extents:
   x-coordinate: min (m) = -4.999876e+03, max (m) = 4.999293e+03
   y-coordinate: min (m) = -4.999229e+03, max (m) = 4.999869e+03
   z-coordinate: min (m) = -1.000000e+05, max (m) = -3.369302e+03
 Volume statistics:
   minimum volume (m3): 8.069210e-06
   maximum volume (m3): 5.032947e+07
     total volume (m3): 6.773932e+12
 Face area statistics:
   minimum face area (m2): 1.613930e-04
   maximum face area (m2): 8.659107e+05
 Checking mesh...................................
Done.
```

(Figure 8.1 show resulting mesh in ANSYS Fluent Software)

## 8.3 Creation of 120-Degree Segment Block:

Using the functions of the script, the project also creates a 120-degree segment block.

## 8.4 Enhanced User Interface (GUI) Format:

The project's framework has a graphical user interface (GUI) that is user-friendly. The GUI format improves the script's functionality by providing users with a table of dimensions and parameters. Users have the freedom to change variable values directly within this interface, which is significant.

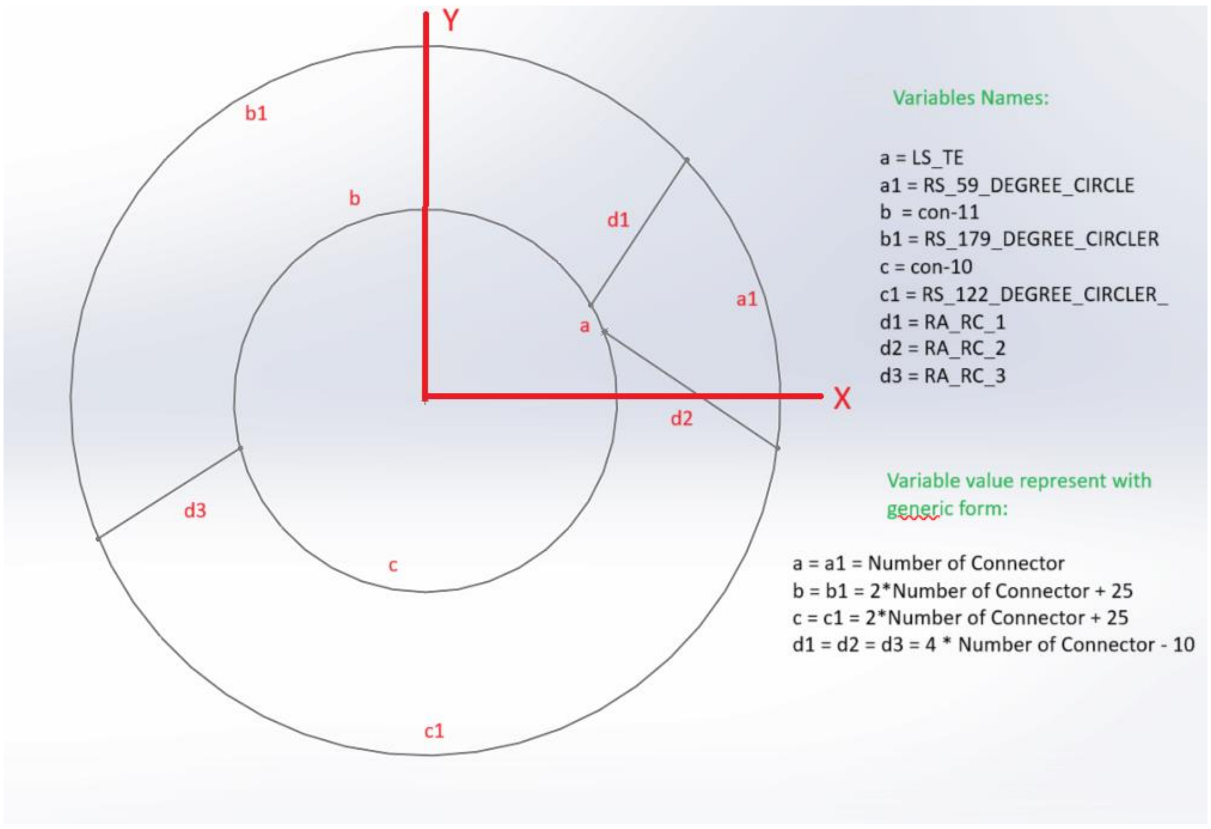
## 9. Pending Work

1. The Unstructured background volume mesh remains to be designed.

## 10. Appendices

### 10.1 Defined variables names used in script

Variables (See Figure no.10.1, 10.2 and 10.3) are executed in scripts with the 'set' command, which receive as a return argument.



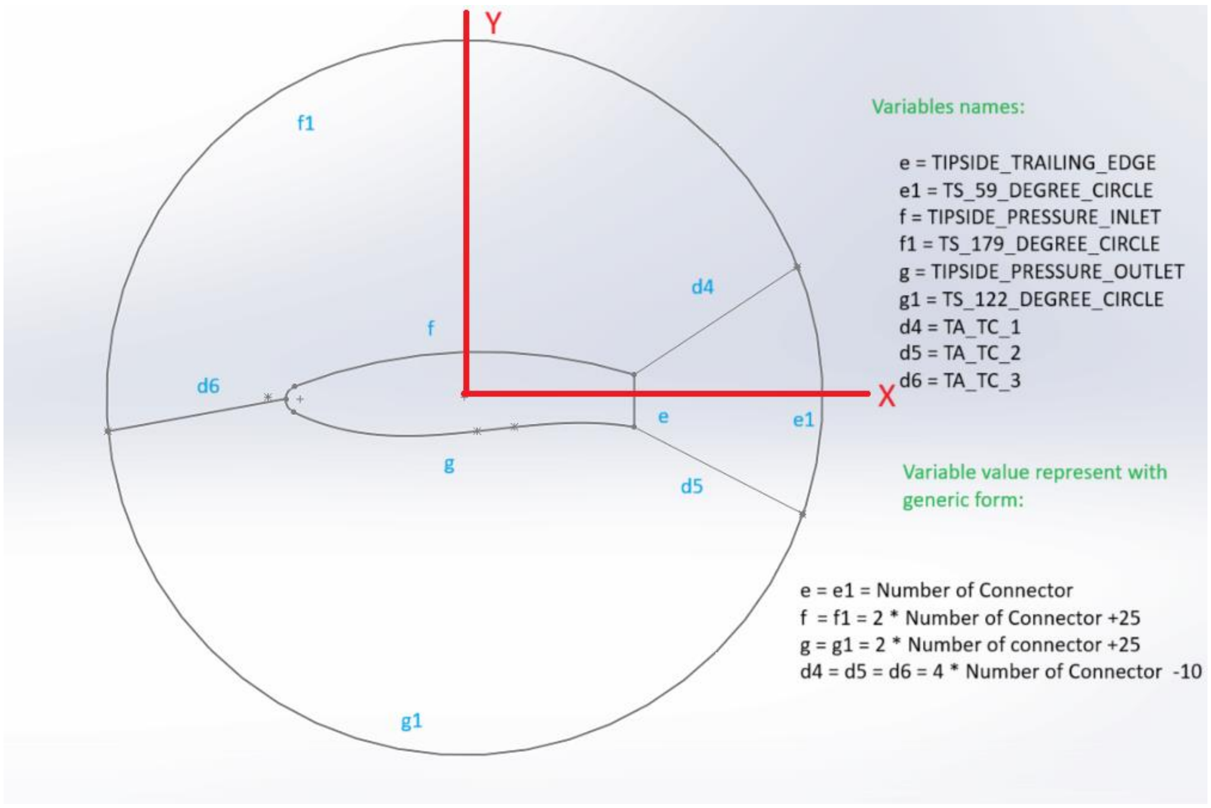
Variables Names:

a = LS_TE
a1 = RS_59_DEGREE_CIRCLE
b  = con-11
b1 = RS_179_DEGREE_CIRCLER
c = con-10
c1 = RS_122_DEGREE_CIRCLER_
d1 = RA_RC_1
d2 = RA_RC_2
d3 = RA_RC_3

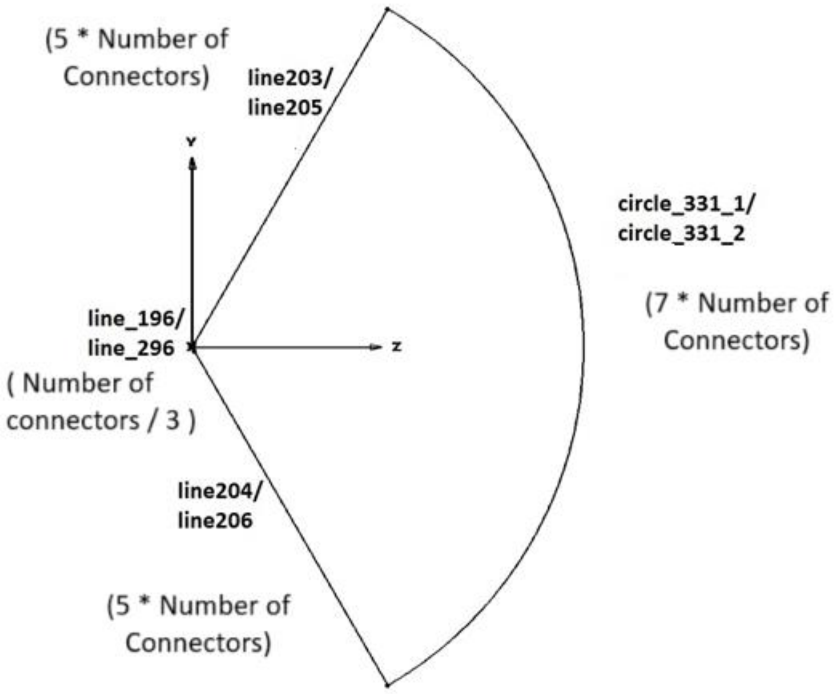
Variable value represent with generic form:

a = a1 = Number of Connector
b = b1 = 2*Number of Connector + 25
c = c1 = 2*Number of Connector + 25
d1 = d2 = d3 = 4 * Number of Connector - 10

(Figure 10.1 Show the defined variable name and number of connectors in script)


Y
X
f1
f
d4
d6
e
e1
g
d5
g1
Variables names:
e = TIPSIDE_TRAILING_EDGE
e1 = TS_59_DEGREE_CIRCLE
f = TIPSIDE_PRESSURE_INLET
f1 = TS_179_DEGREE_CIRCLE
g = TIPSIDE_PRESSURE_OUTLET
g1 = TS_122_DEGREE_CIRCLE
d4 = TA_TC_1
d5 = TA_TC_2
d6 = TA_TC_3
Variable value represent with generic form:
e = e1 = Number of Connector
f = f1 = 2 * Number of Connector +25
g = g1 = 2 * Number of connector +25
d4 = d5 = d6 = 4 * Number of Connector -10

*(Figure 10.2* Show the defined variable name and number of connectors in script)


(5 * Number of Connectors)
line203/
line205
Y
circle_331_1/
circle_331_2
(7 * Number of Connectors)
line_196/
line_296
z
( Number of connectors / 3 )
line204/
line206
(5 * Number of Connectors)

*(Figure 10.3* Show the defined variable name and number of connectors in script)

## 11. References

1. A.P.Schaffarczyk. (2020). Introduction to Wind Turbine Aerodynamics. 124-126.
2. Brilliant Worldwide, Inc. (2021). *Geometric Progressions.* Retrieved from https://brilliant.org/
3. Cadence. (2023). *Compute Grid Spacing for a Given Y+*. (Cadence Design Systems, Inc)
4. Cadence Design Systems, Inc. (2021). *Glyph Version 7.22.2*. (Cadence Design Systems, Inc.) Retrieved from https://www.pointwise.com/doc/glyph2/files/Glyph/cxx/GgGlyph-cxx.html
5. CASIO COMPUTER CO., LTD. (2023). *Geometric progression Calculator*. (CASIO COMPUTER CO., LTD.)
6. Chawner, J. (2021). The Handiest CFD App – the Y+ Calculator.
7. Eivind Fonn, A. R. (2015). Spline Based Mesh Generator for High Fidelity Simulation of Flow around Turbine Blades. *Energy Procedia*.
8. Flanagan, D. (2012). Generic Types. *O.REILY*.
9. Fonn, E. (2015). Spline Based Mesh Generator for High FIelity SImulation of Flow around Turbine Blades. 80.
10. George Duval. (2021, July 13). What are Wind Turbine Blades Made of? *The blades of a wind turbine explained*. Retrieved from https://www.semprius.com/what-are-wind-turbine-blades-made-of
11. IEA Wind. (2020). *Definition of the IEA Wind 15-Megawatt Offshore Reference Wind Turbine.*
12. Khurma, M. (2013). *Master Math*. (CUEMATH) Retrieved from https://www.cuemath.com/
13. LearnCAx. (n.d.). Basics of Y Plus, Boundary Layer and Wall Function in Turbulent Flows.
14. mathsisfun. (2022). *Introduction to Trigonometry*. Retrieved from https://www.mathsisfun.com
15. Microsoft. (n.d.). Visual Studio code.
16. NPTEL. (n.d.). Renewable Energy Engineering: Solar, Wind and Biomass Energy Systems, IIT Guwahati. Retrieved from https://nptel.ac.in
17. Pointwise, Inc. (1996). *Pointwise*. Retrieved from Pointwise: http://pointwise.com/
18. RTSlabs. (2019). The 5 Stages of Software Development (2019 Update).

19. Schaffarczyk, A. (2020). *Introduction to Wind Turbine Aerodynamics.* Springer.

20. SIMSCALE. (2022, September 12). Mesh Quality.

21. Stephens, D. (2019). Scripting Pointwise Meshing.

22. Tutorials Point. (2006). Tutorials Point. *LEARN TCL/TK*.

23. University of Manchester. (n.d.). All there is to know about different mesh types in CFD! Retrieved from https://www.manchestercfd.co.uk/post/all-there-is-to-know-about-different-mesh-types-in-cfd

24. Varotsis, A. B. (2021). Meshing in FEA, CFD & Manufacturing.

25. Vector Renewable. (2023). Do you know what the 'wake effect' is in a wind farm?