DESIGN CENTRIFUGAL FAN VOLUTE WITH CFD NUMERICAL SIMULATION USING OPENFOAM-MATLAB COUPLING

M.Sc. Gjeta A.[1],
Mechanical Engineering Faculty, Polytechnic University of Tirana, Tirana, Albania [1]
agjeta@fim.edu.al

Abstract: *The main scope of this paper is to fully integrate simulation from open source CFD software with the Matlab App Designer program. The OpenFOAM-Matlab CFD interface allows one to conveniently setup fully turbulent incompressible Reynolds Averaged Navier-Stokes (RANS) CFD case all within an easy to use graphical user interface (GUI). Highlighting built-in CAD tools to create geometry (STL) for automatic mesh generation to OpenFOAM case file as well as a solution with post-processing and visualization with ParaView. Also, the cross-platform OpenFOAM CFD interface for MATLAB should allow the user to rapidly design the spiral casing in an engineering manner as an illustrative platform and coordinator between the inputs chosen by the user and the results which are displayed after calculations. Furthermore, the GUI should be a very flexible and user-friendly platform for spiral casing design according to the efficiency and static pressure recovery coefficient as well.*

Keywords: CFD, MATLAB GUI, OpenFOAM, SPIRAL CASING.

1. Introduction

A Graphical User Interface (GUI) generally presents a graphical display that uses a combination of devices and graphical controls as icons and visual indicators to provide a platform that the user can interact with, imposed on gathering and obtaining information. Designing the visual composition and temporal behavior of a GUI is an important part of software application programming in the area of user-computer interaction. Its goal is to enhance efficiency and presenting simplicity in use. This makes it easier for people with few computer skills to perform with and use this computer software. The components of GUI should be ordered depending on an engineering point of view. While designing the display window the graphical components are represented by the name, the size, the position and function in this window waiting for a user to manipulate control and to respond for each action performed by him. Moreover, all these elements should be adjusted on relevant parts of the GUI as presenting simplicity for the user to handle and easily find them. Properties and their values are set automatically by Matlab, and the user can edit values in-places.

The most important properties of a component are string and tag. A string is the name of the component on the Matlab code referred to as the actions that represent the icon in the interface. It is essential to have names that are identifiable for the component. Moreover it is possible to set the label in a GUI component by its String property. This is a rather procedural approach. It is usually more convenient to use tag in which the action is passed to a function that automatically connects to icon with their string name specified.

After designing the first page of the graphical user interface and estimating the properties for each object component two files are saved automatically. The first as a figure (.fig) and the second as a Matlab file (.m). The code offers empty callbacks functions that are associated with the components. It's necessary to adapt callbacks in order to make the components active. Callbacks are essentially in M-file code as they are executed each time a prescribed action performed near or over an object. So callbacks provide a convenient mean of initially writing and then modifying in M-file code in order to run as user wants to.

2. OpenFOAM structure

OpenFOAM is first and foremost a C++ library, used primarily to create executables, known as applications. The applications fall into two categories: solvers, that are each designed to solve a specific problem in continuum mechanics; and utilities, that are designed to perform tasks that involve data manipulation. New solvers and utilities can be created by its users with some pre-requisite knowledge of the underlying method, physics and programming techniques involved. OpenFOAM is supplied with pre- and post-processing environments. The interface to the pre- and post-processing are themselves OpenFOAM utilities, thereby ensuring consistent data handling across all environments. The overall structure of OpenFOAM is shown in Figure 1 [1].

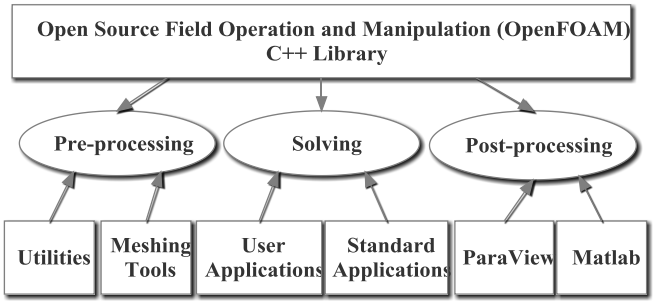

Fig. 1 *Overview of OpenFOAM structure [1]*

3. Volute Shape Design Method

Constant circulation method [2] is a method applied by drawing a spiral case based on the fact that velocity circulation is a constant $rc_u = constant$. In practice, this rule is valid with the restriction that one spiral must be so far displaced from the impeller that deflections conditioned by the consideration of a finite number of blades can be ignored. This rule constitutes the basis for the dimensioning of a volute in the cases where friction has been ignored. The velocity c at an arbitrary place can be calculated from its components c_m and c_u, $rc_u = r_2 c_{u2}$. From the condition that the same volume-flow must flow (the continuity equation) through all the streamline in volute it gives the correlation:

$$Q = 2\pi r_2 b_2 c_{m2} = 2\pi r B c_m \tag{1}$$

From which follows $r_2 b_2 c_{m2} = rBc_m$, by arranging terms in the equation, we obtain the following inclination α of the streamlines:

$$tg(\alpha) = \frac{c_m}{c_u} = \frac{c_{m2}}{c_{u2}}\frac{b_2}{B} \tag{2}$$

Because we obtain the boundary of the volute from the streamline, again it yields, $tg(\alpha) = \frac{dr}{rd\varphi}$;

$$\frac{dr}{r} = d\varphi\ tg(\alpha) = d\varphi\ tg(\alpha_2)\frac{b_2}{B} \tag{3}$$

Finally, the solution states,

$$ln\frac{r}{r_2} = \varphi\ tg(\alpha_2)\frac{b_2}{B} = \varphi\frac{c_{m2}}{c_{u2}}\frac{b_2}{B} \tag{4}$$

Accordingly, the trajectory of fluid particles in the spiral casing is as follows (Carolus 2013) [3],

$$r_{(\varphi)} = r_2 e^{\varphi tg\,(\alpha)} = r_2 e^{\varphi tg\,(\alpha_2)\frac{b_2}{B}} \tag{5}$$

$r_{(\varphi)}$, is the radius of the volute at an angle φ,

r_2, is the outer radius of the impeller that is equal to 150mm in our case

α is the angle that the absolute velocity vector makes with the peripheral direction $tg(\alpha) = c_m/c_u$.

b_2, the width of outlet impeller; B, the width of volute

4. Matlab – OpenFOAM implementation scheme

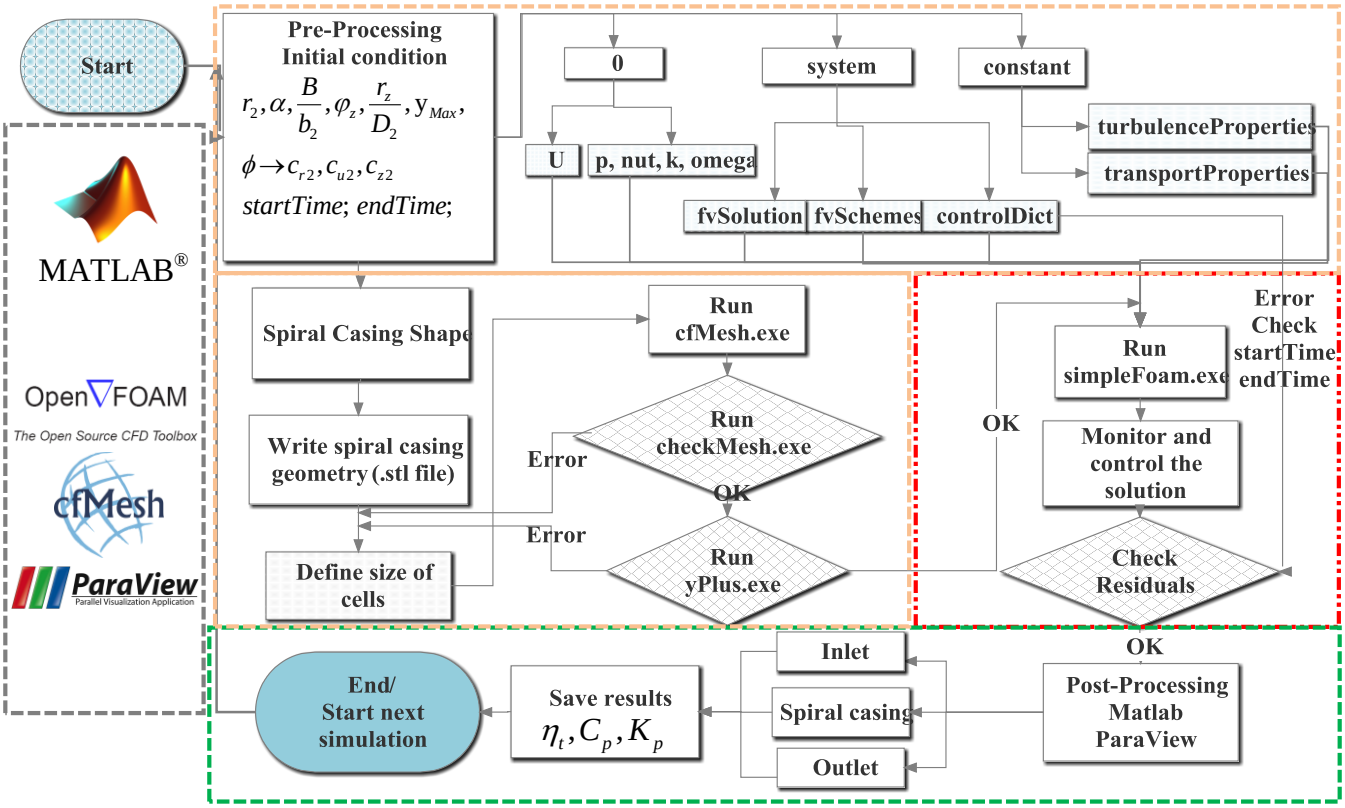

Fig.2 *Matlab – OpenFOAM implementation scheme*

The Matlab-OpenFOAM implementation scheme is built according to the structure of OpenFOAM. In the first stage, preliminary data, which are related to geometry data as well as the necessary numerical simulation data, such as initial and boundary conditions, are defined. The inflow boundary conditions were based on known flow rates and the flow direction. Non-uniform velocity profiles were prescribed at the volute inlet (fan impeller outlet) by implementing radial and tangential velocity components, also axial velocity is included. The front and backside of the impeller as the rotating wall, the other parts wall with no-slip condition and for the outlet ambient pressure is used. Turbulent kinetic energy is $k = 3\ m^2 s^{-2}$, and the specific turbulence dissipation rate is $\omega = 4\ 000 s^{-1}$. Since we have a set of geometric parameters with a wide range of values it is necessary that the process of generating the geometry and then mesh creating to be carried out automatically. The geometry of volutes is generated from MATLAB code as a stereo-lithography (.stl file), than cfMesh v1.1.2 software is used to create mesh. The grid resolution is made according to y^+value $30 < y^+ < 200$.

Table 1: Boundary conditions used in CFD simulations in the OpenFOAM software

Name	Boundary and Initial Condition	
1. Inlet	Constant mass flow rate	*Radial, tangential and axial velocity*
2. Spiral Casing	No slip condition	*fixedValue 0;*
3.FrontAndBackInlet	Rotating wall	*omega;*
4.FrontAndBackVolute	No slip condition	*fixedValue 0;*
5. Outlet	Ambient Pressure	*zeroGradient;*

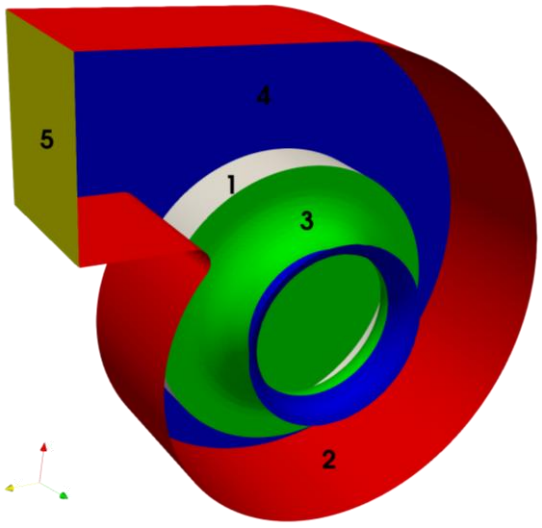

Fig. 3 Schematic view of complete fan geometry

The Matlab code provides file creation in the C++ language format needed to use by the OpenFOAM CFD toolbox. Each file is defined separately in separate folders. This is the advantage of using Matlab, of changing the numerical simulation parameters in a very short time. To simulate a turbulent flow, in the "0" folder, at least the following 5 files are needed: "U" – Velocity file; "p" – Information on pressure value; "*nut*" – kinematic viscosity value; "*k*" – Turbulent kinetic energy and, "omega" – specific turbulent dissipation ratio.

A "*system*" directory for setting parameters associated with the solution procedure itself. It contains at least the following 3 files:"*controlDict*", where run control parameters are set including start/end time, time step and parameters for data output; "*fvSchemes*", where discretization schemes used in the solution may be selected at run-time; and, "*fvSolution*", where the equation solvers, tolerances and other algorithm controls are set for the run.

A "*constant"* directory that contains a full description of the case mesh in a subdirectory "*polyMesh*" and files specifying physical properties for the application concerned, e.g. "*turbulenceProperties*" – includes turbulence modeling: e.g. komegaSST, "*transportProperties*" – includes kinematic viscosity value of air.

The second stage is mesh generation, which should be checked for any errors, and if there is any problem with the mesh or the values of y+ than changing the size of cells. Only after these two conditions have been met, we can proceed with solving. Steady-state solver for incompressible flows with the turbulence model is "*simpleFoam*", which is using the SIMPLE algorithm [7], [8]. While the solving process continues it is possible monitoring residual values. If the number of step iterations is not necessary to reach the residual value of 10^{-5} then changing the "*startTime"* and "*endTime"* values. Replace "*startTime*" with the "*endTime*" value, and double the new value of "*endTime*". This change should be reflected in "controlDict" file. Only after the residuals value has been reached then the solving process is automatically stopped.

The next step is post-processing which is implemented through Matlab. OpenFOAM provides a set of sampling function objects to sample field data, either through a 1D line for plotting on graphs or a 2D plane and 3D surfaces for displaying as images. We have used a 3D sampling function for "*Inlet*", "*Spiral Casing*" and for the "*Outlet*". The sampling parameter for the "*Inlet*" is the static pressure since the velocity is known and for the "*Outlet*", is the velocity since we have the pressure data.

The outlet dynamic pressure is determined by the square of each velocity component in each outlet cells. Since we define these parameters it is easy to determine the performance of the spiral casing. The overall performance of the volute can be analyzed by using:

Total Efficiency of volute:

$$\eta_T = \frac{p_{t3}}{p_{t2}} \tag{6}$$

Static pressure recovery coefficient of volute:

$$C_p = \frac{p_3 - p_2}{p_{t2} - p_2} = \frac{p_3 - p_2}{\frac{\rho}{2} c_2^2} \tag{7}$$

C_p, is defined as the ratio between the static pressure recovered in the volute to the dynamic pressure at the impeller exit.

Total pressure loss coefficient of volute:

$$K_p = \frac{p_{t2} - p_{t3}}{p_{t2} - p_2} = \frac{p_{t2} - p_{t3}}{\frac{\rho}{2} c_2^2} \tag{8}$$

K_p, is defined as the ratio between the total pressure losses in the volute to the dynamic pressure at the impeller exit.

After defining the 3 parameters for evaluating the performance of the spiral casing, all results are stored in the form of a Matlab file (.m), with the date and time of the numeric solution and almost any parameter needed. The file with results is easily readable by Matlab code and then processed graphically, saving it to Matlab figure form (.fig). Refer to figures 4-9 for more details.

5. Matlab GUI for spiral casing design

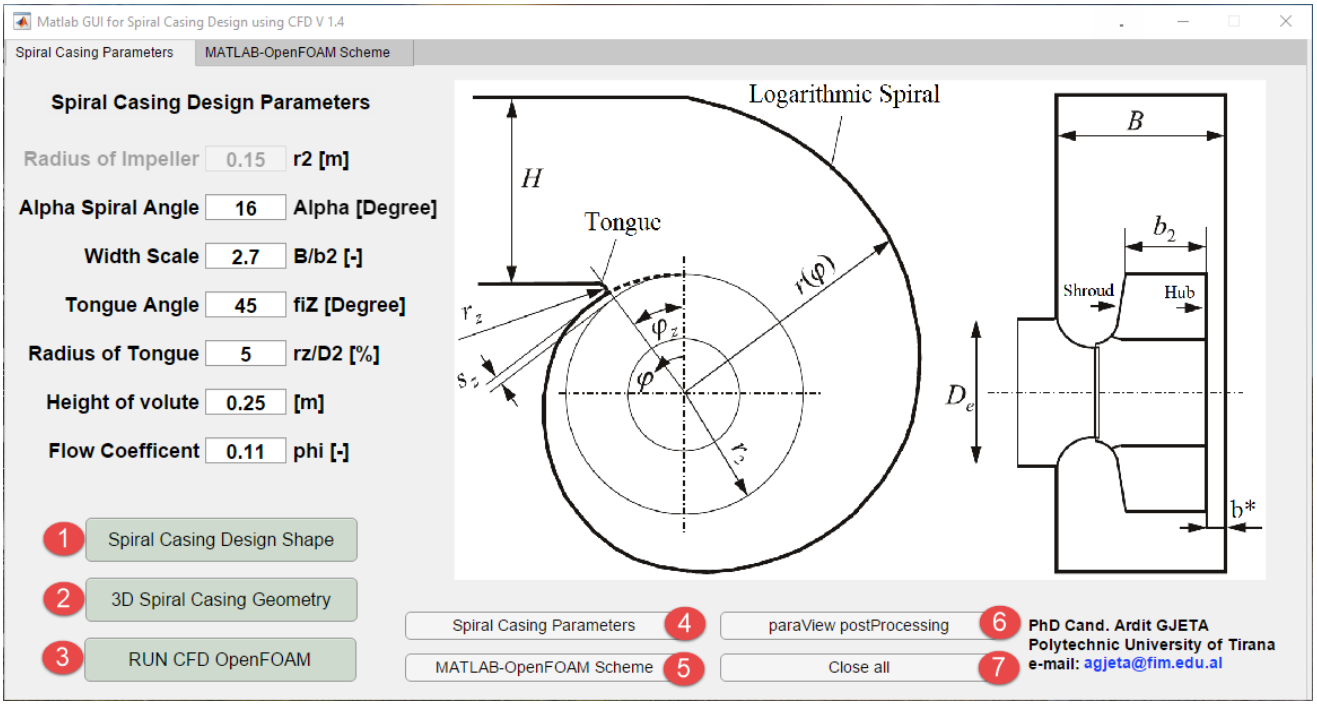

Fig. 4 *Matlab graphical user interface for spiral casing design*

Each of the spiral casing design parameters can be changed according to the desired value imposed by the user. Except for the rotor radius, which for this set of simulations, is assumed unchanged.

1. "*Spiral Casing Design Shape*" button gives the shape of the spiral casing for specific values of parameters, see fig. 5. In addition to the parameters that the user set at the beginning from GUI, in the Matlab results file can be found more information, e.g. on the clearance gap space, which in this particular case is s_z/D_2=5.5%, as well as the maximum aperture of spiral casing H=0.177m.

2. "*3D Spiral Casing Geometry*" button gives us an overview of 3D complete fan geometry.

Through this feature, the user can view the position of the rotor and the spiral casing, as well as check for any possible geometry errors before performing the geometry meshing. This can be done directly using the Matlab interface. To check in more detail, Matlab code gives us the ability to create triangulate geometry in a separate file (.stl), and can then be viewed through paraView software. ParaView software must be pre-installed, and can be opened directly from Matlab GUI while clicking the button "paraView postProcessing".

Additional information can be obtained about the outlet surface area of the spiral casing as well as the relative position of the rotor (impeller) within the spiral casing, which we have defined as the asymmetry parameter.

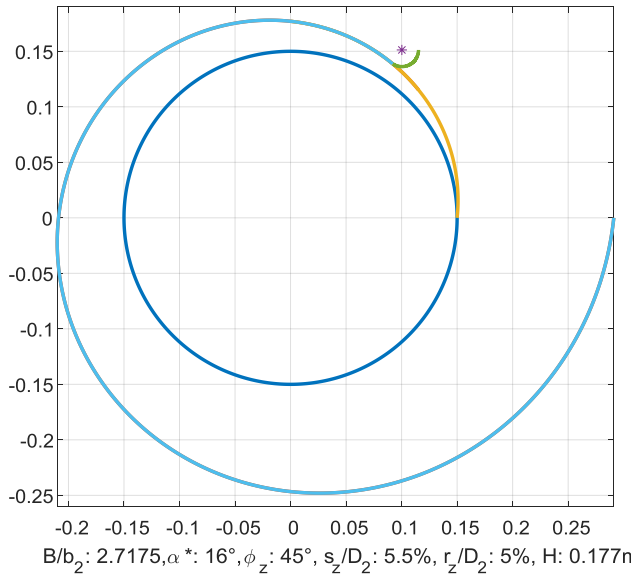

Fig. 5 *Spiral casing shape design parameters*

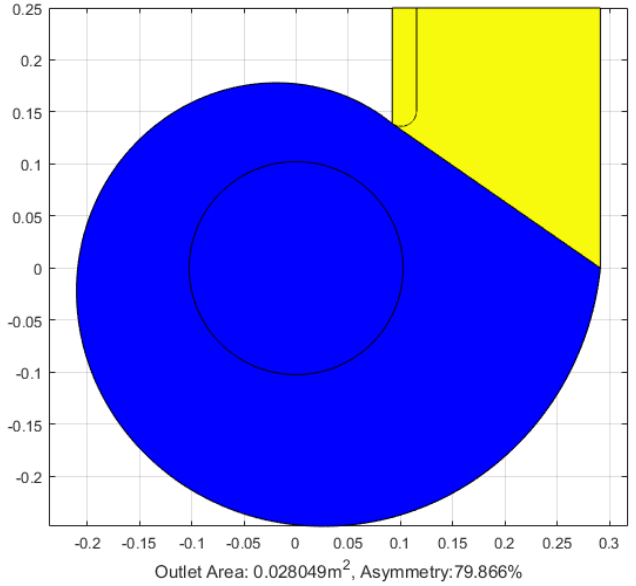

Fig. 6 *Complete fan geometry*

3. "*RUN CFD OpenFOAM*" button will start CFD simulation with the parameters that the user has defined.

Detailed information according to the performance of the spiral casing, boundary and initial conditions you can refer to the papers [4-6].

4. "*Spiral Casing Parameters*" button opening a picture with information on the parameters of the spiral casing and the recommended range values [9-10].

5. "*Matlab-OpenFOAM Scheme*" button opens the figure no.1, which is provided if the user wants to know the implementation method of CFD simulation. The same information can be viewed also in the second tab of the Matlab GUI.

6. "*paraView Post-Processing*" button open paraView software, is an open-source, multi-platform data analysis and visualization application.

7. "*Close all*" button closes the windows with the results as well as creating the conditions for the next simulation.

After the numerical simulation is completed, the first parameter to be checked is the residual control value. Since the numerical simulation is finished, the residual value is equal to 10^{-5}. In this case, it can be seen which of the parameters converges slowly, e.g. from fig.6 can be observed that the last parameter that converges is axial velocity "*Uz*" and the pressure. Additional information can be obtained on the number of iteration as well as the PC computational time required for the numerical solution. It is observed that the solution of numerical simulations for this case is achieved for 604 iterations, and the computational time of the solution is about 42 minutes.

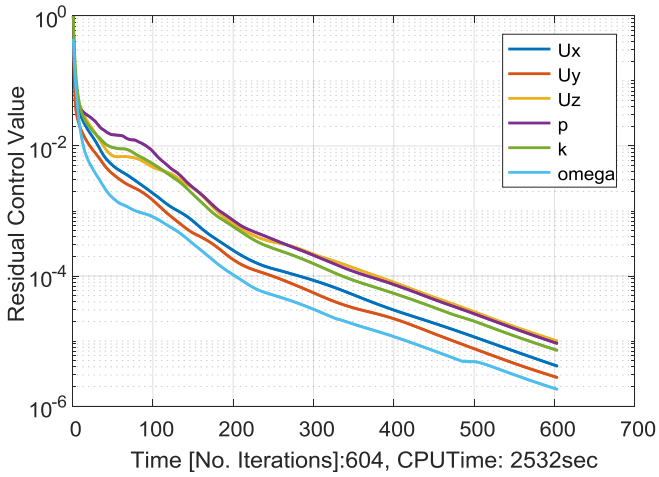

Fig. 6 *Residual control value*

6. CFD simulation results

One of the ways to determine the appropriate using of the spiral casing is the efficiency value. Normally in most cases, the highest value is required, but the performance of spiral casing depends on the operating conditions of the centrifugal fan. In certain working conditions, the spiral casing is required to provide a high increase in static pressure recovery, as a result, the main focus is not only the efficiency value.

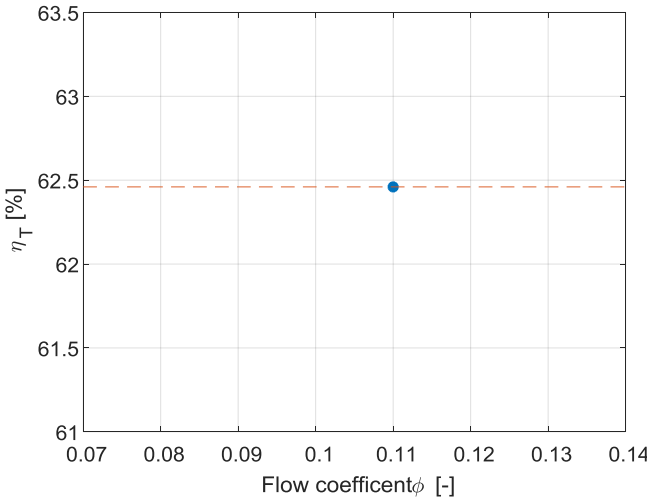

Fig. 7 *The total efficiency of the volute function of flow coefficient*

In this specific case study, the overall efficiency of the spiral casing is $\eta_T = 62.46\%$, corresponding to a flow coefficient of $\phi = 0.11$, which correspond flow rare, $Q = 0.36641\ m^3/s$ since the diameter of the impeller is $D = 0.3m$ and the rotational speed is $n = 3000\ rpm$.

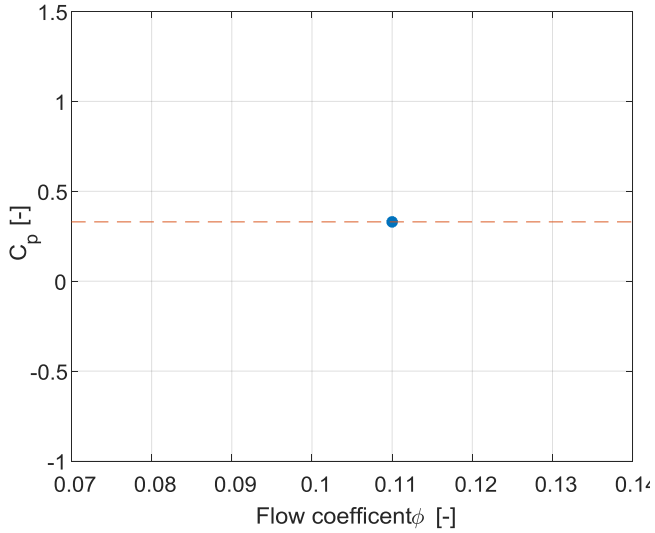

Fig. 8 *Static pressure recovery coefficient of the volute function of flow coefficient*

From all we have explained above, from the graph in fig.8, it obtains information on the value of static pressure recovery coefficient. The exact value of Cp=0.33. We must pay attention to the negative values of this coefficient, as the use of spiral casing in those cases is completely unnecessary, due to the wrong spiral casing design.

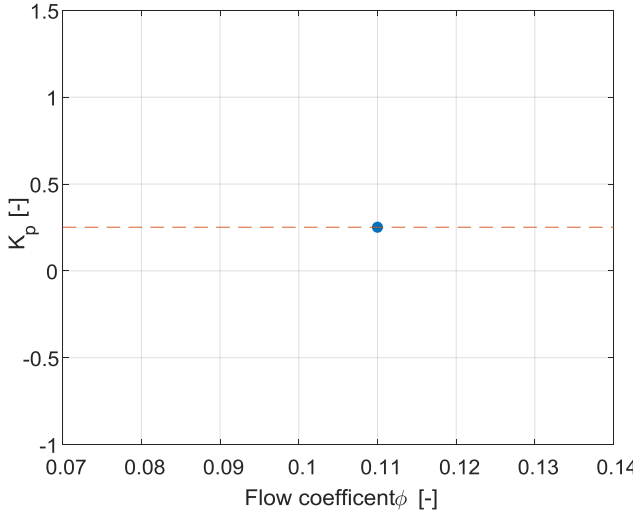

Fig. 9 *Total pressure loss coefficient of the volute function of flow coefficient*

Another parameter that can be obtained is the total pressure loss coefficient. Unlike the static pressure recovery coefficient, we must make sure that this value is as low as possible. In this particular case the value of the total pressure loss coefficient is 0.25.

7. Conclusion

A Graphical User Interface has been created to facilitate the design of the spiral casing using CFD. This program functionality now allows the user to quickly design the spiral casing geometries while providing convenience. The convenience of the GUI is that the user can set any possible parameters of the spiral casing. The program can be modified and adapted to any type of rotor size as well as to any operating point.

Many studies concerning centrifugal fans have investigated the impeller but only to a smaller extends to the spiral casing (volute). The volute may take up a substantial part of the fan's hydraulic loss. Currently, minimization of energy loss is dependent on the characteristics of the spiral casing. Hence, the appropriate design of the fan volute has significant meaning to centrifugal fan performance. For that, this study of volutes is carried out by leading to advanced best practice recommendations for the volute design shape parameter. A qualitative understanding of the effects of parameters will enable the performance of a real product to be improved. Evaluation is carried out by analyzing the performance of the volute, based on the efficiency, static pressure recovery coefficient as well as the total pressure loss coefficient through the use of OpenFOAM-Matlab coupling. Finally, the geometric results performed by the CFD simulation, are a key element to build CAD design format.

Nomenclature

Indices

2	impeller outlet (volute inlet)
3	volute outlet

Greek Symbols

ϕ	flow coefficient
α	alpha spiral angle
η	efficiency
ρ	air density

Abbreviations

CFD	Computational Fluid Dynamics
RANS	Reynolds Averaged Navier-Stokes
SST	Shear stress transport
FOAM	Field Operation And Manipulation
SIMPLE	Semi-Implicit Method for Pressure Linked Equations
GUI	Graphical User Interface
CAD	Computer-Aided Design

References

[1] OpenFOAM, "The Open Source CFD Toolbox, User Guide" Version 3.0.1, 2015.

[2] Eck, B.: "Fans, Design, and operation of centrifugal, axial-flow and cross-flow fans". *Pergamon Press*, 1973, chap. 11, pp. 189-192.

[3] Carolus, Th.: "Ventilatoren", *Springer*, 2013, chap. 2, pp. 40-44.

[4] E. Ayder R. Van den Braembussche: "Experimental and Theoretical Analysis of the Flow in a Centrifugal Compressor Volute", *Journal of Turbomachinery*, 115(3): 582-589 1993.

[5] Gjeta A., Bamberger K., Carolus Th, Londo A.: " Parametric Study of Volutes for Optimal Centrifugal Fan Impellers", *FAN 2018 - International Conference on Fan Noise, Aerodynamics, Applications & Systems, Darmstadt, Germany*, April 18-20, 2018.

[6] Gjeta, A. 2019. "Effect of Clearance Gap in Spiral Casing Design of a Centrifugal Fan with Optimized Impellers". *European Journal of Engineering Research and Science*. 4, 9 (Sep. 2019), 181-185. DOI: https://doi.org/10.24018/ejers.2019.4.9.1533.

[7] Ferziger, Joel H.; Perić, M.: "Computational methods for fluid dynamics". 3rd, rev. ed. Berlin, London: Springer, 2002

[8] Patankar, S. V.: "Numerical Heat Transfer and Fluid Flow", Hemisphere Publishing Corporation, 1980

[9] Bommes, L., Fricke, J., Grundmann, R.: " Ventilatoren ". Vulkan-Verlag, Essen, 2003.

[10] R.K. Turton: "Principles of Turbomachinery", 1995, ch. 6, pp. 121-13.