Table of Contents

[1. Summary 5](#_Toc416107287)

[2. Background and Introduction 6](#_Toc416107288)

[2.1 Modern Turbofan Engines 6](#_Toc416107289)

[2.1.1 Thrust generated in a Turbofan 8](#_Toc416107290)

[2.2 CFD, Optimization and Mesh Morphing 9](#_Toc416107291)

[2.2.1 CFD – Computational Fluid Dynamics 9](#_Toc416107292)

[2.2.2 Optimization using CFD 12](#_Toc416107293)

[2.2.3 Mesh Morphing 15](#_Toc416107294)

[3. Meshing 20](#_Toc416107295)

[3.1 Test Object 20](#_Toc416107296)

[3.2 Types of mesh available 20](#_Toc416107297)

[3.3 Unstructured Meshes 21](#_Toc416107298)

[3.4 Structured Mesh 22](#_Toc416107299)

[3.5 Mappable Volumes 26](#_Toc416107300)

[3.6 Mesh Generation 27](#_Toc416107301)

[3.6.1 Geometry Preparation 27](#_Toc416107302)

[3.6.2 Checking volumes for mappability 30](#_Toc416107303)

[3.6.3 Creating the Mesh 31](#_Toc416107304)

[3.6.4 Boundary Layer Mesh 32](#_Toc416107305)

[3.6.5 Mesh Statistics 34](#_Toc416107306)

[4. CFD Setup and Analysis 35](#_Toc416107307)

[4.1 Baseline configuration analysis 35](#_Toc416107308)

[4.2 Setup and Boundary Conditions Used 35](#_Toc416107309)

[5. Results and Discussion – Baseline Shape 38](#_Toc416107310)

[5.1 Convergence and Residual Ratios 38](#_Toc416107311)

[5.2 Flow parameters 39](#_Toc416107312)

[6. Optimization setup 45](#_Toc416107313)

[7. Results and Discussion – Optimization Study 49](#_Toc416107314)

[7.1 The DOE Case Matrix 49](#_Toc416107315)

[7.2 Flow Parameters 50](#_Toc416107316)

[8. Conclusive Notes and Future Work 52](#_Toc416107317)

[9. References 54](#_Toc416107318)

[10. Appendix 56](#_Toc416107319)

[10.1 Solver input file 56](#_Toc416107320)

List of Figures

[Figure 2‑1: Turbofan engines: high and low bypass flow 6](#_Toc415560607)

[Figure 2‑2: Trend of bypass ratio and specific fuel consumption 7](#_Toc415560608)

[Figure 2‑3: Flowchart of a CFD design process 11](#_Toc415560609)

[Figure 2‑4: Flowchart of a CFD optimization process 13](#_Toc415560610)

[Figure 2‑5: Various optimization approaches and their application to design processes 14](#_Toc415560611)

[Figure 2‑6: Illustration of mesh morphing applied to a grid 17](#_Toc415560612)

[Figure 2‑7: Flowchart of a CFD optimization process with mesh morphing 19](#_Toc415560613)

[Figure 3‑1: The test object: GENX – n series aircraft engine entry fan 20](#_Toc415560614)

[Figure 3‑2: An unstructured tetrahedral mesh illustration 21](#_Toc415560615)

[Figure 3‑3: A structured hexahedral mesh illustration 22](#_Toc415560616)

[Figure 3‑4: Example of geometry decomposition for automated meshing 24](#_Toc415560617)

[Figure 3‑5: Solid regular hexahedral meshing using mesh sweeping 25](#_Toc415560618)

[Figure 3‑6: Map meshing scheme – partial array of hexahedral elements 27](#_Toc415560619)

[Figure 3‑7: Stitching edges 28](#_Toc415560620)

[Figure 3‑8: Modeling of the flow domain 29](#_Toc415560621)

[Figure 3‑9: Creating mappable volumes – the split lines can be seen 30](#_Toc415560622)

[Figure 3‑10: The meshed domain 32](#_Toc415560623)

[Figure 3‑11: Meshing the boundary layer regions 34](#_Toc415560624)

[Figure 4‑1: Boundary conditions and CFD analysis setup 37](#_Toc415560625)

[Figure 5‑1: Residual ratios plot for the baseline shape analysis 38](#_Toc415560626)

[Figure 5‑2: Total pressure variation upstream and downstream of the blade passage 40](#_Toc415560627)

[Figure 5‑3: Total pressure and static pressure contours 40](#_Toc415560628)

[Figure 5‑4: Pressure difference between pressure and suction side of the blade 41](#_Toc415560629)

[Figure 5‑5: Pressure contours on pressure and suction side of the blade 41](#_Toc415560630)

[Figure 5‑6: Streamwise total pressure contour at 50% span 42](#_Toc415560631)

[Figure 5‑7: Streamwise static pressure contour at 50% span 42](#_Toc415560632)

[Figure 5‑8: Streamwise velocity magnitude contour at 50% span 43](#_Toc415560633)

[Figure 5‑9: Streamwise velocity vectors representation at 50% span 43](#_Toc415560634)

[Figure 5‑10: Mass flow through the fan 44](#_Toc415560635)

[Figure 6‑1: Mesh morphing setup and shape variables 46](#_Toc415560636)

[Figure 6‑2: Illustration of morphing applied to the analysis domain 47](#_Toc415560637)

[Figure 7‑1: The shape corresponding to the highest mass flow rate 50](#_Toc415560638)

List of Tables

[Table 6‑1: Optimization experiment design matrix for the current study with two variables 48](#_Toc415560721)

[Table 7‑1: Optimization experiment design matrix populated with the study results 49](#_Toc415560722)

[Table 7‑2: Variation of flow parameters between the baseline case and the selected case 50](#_Toc415560723)

# Summary

The objective of this project work is to obtain the aerodynamically optimized shape of an aircraft engine axial flow fan.

Fans in aircraft engines are used in the modern turbofan type engines. The turbofan engine design employs a fan at the entry of the aircraft which is driven by the engine core. The diameter of the fan is larger than the diameter of the core. Thus, a considerable mass of air entering the engine flows around the core after passing through the fan.

This flow on the outside, also known as the bypass flow, is responsible for a major fraction of the thrust generated by a turbofan. The thrust produced by the engine is directly proportional to the mass flow of the air through the engine (fan + core). Thus, higher thrusts can be obtained from the engine if mass flow through the engine can be maximized.

In the current optimization study mass flow is the parameter selected for optimization. Computational Fluid Dynamics (CFD) analyses are used to generate the results for comparison. Optimization is carried out using a parametric study where the shape of the blade in the given fan model (baseline shape) is modified parametrically. Two parameters are used to define the modified shape of the blade – the rotation and translation of the leading edge of the blade.

For carrying out a CFD study, the geometry must be discretized into a fine grid called the mesh. Mesh morphing technique is used to *morph* (or adapt), the mesh created for the baseline shape to the modified shapes of the blades. Using morphing, an existing mesh can be morphed to fit a modified geometry without having to remesh the new geometry from scratch. This can save a lot of time otherwise spent in remeshing and thus the saved time can be used to test more number of setups for comparison.

# Background and Introduction

## Modern Turbofan Engines

Modern turbofan engines are the most popular engines for commercial civil aircraft propulsion. A turbofan engine is an air breathing jet engine, where a fan driven by a turbine pushes the pressurized air through the rear of the engine thus generating forward thrust which propels the aircraft. Thus a certain percentage of air which enters the fan passes through the engine ‘core’ while the rest of the air bypasses the core. The engine core is the actual power plant of the engine which houses the compressor-turbine arrangement generating the power to drive the engine and the aircraft peripherals. The air passing through the core is compressed in the compressor, mixed with fuel and burnt in the combustor, and this hot mixture is then expanded in the turbine to generate the power. The ratio of the two volumes of air (bypass/core) is called the bypass ratio of the engine. Figure 2‑1 shows schematics of a low-bypass and a high-bypass turbofan.

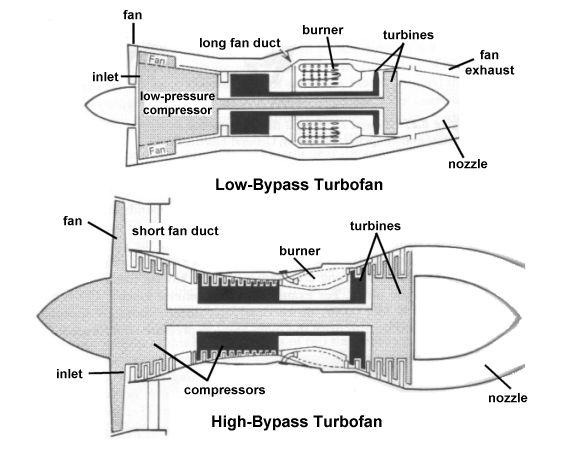


Figure ‑: Turbofan engines: high and low bypass flow

Throughout the development history of the aircraft engines, the trend has been towards development of engines with higher bypass ratio. This can be attributed to the knowledge in propulsion engineering that in propulsion, it is always more efficient to produce a certain amount of propulsive thrust by accelerating a large mass of air by a smaller margin than a small mass of air by a larger margin. Thus when more percentage of air passes through the fan and is accelerated by a relatively smaller value than the small percentage of air which passes through the core, it results in an overall much more efficient engine, thus pushing up the specific fuel consumption. A higher bypass ratio also contributes to lower emissions, and lower noise from the engine. It is thus not uncommon in modern engines that the bypass air passing through the fan can contribute up to 80% of the total thrust of the engine.

In modern aircraft engines, a bypass ratio approaching 10:1 is not uncommon. This has been made possible by the improvements is fan aerodynamics which permit larger size fan blades to be used to accelerate large amounts of air efficiently. Furthermore, improvements in material and manufacturing processes have made possible considerably reduced weight of fan blades, allowing efficient operation of these large size blades. Figure 2‑2 shows how the bypass ratio varies in different types of engines used for propulsion, and also the trend of dropping specific fuel consumption during cruising with higher bypass ratios.

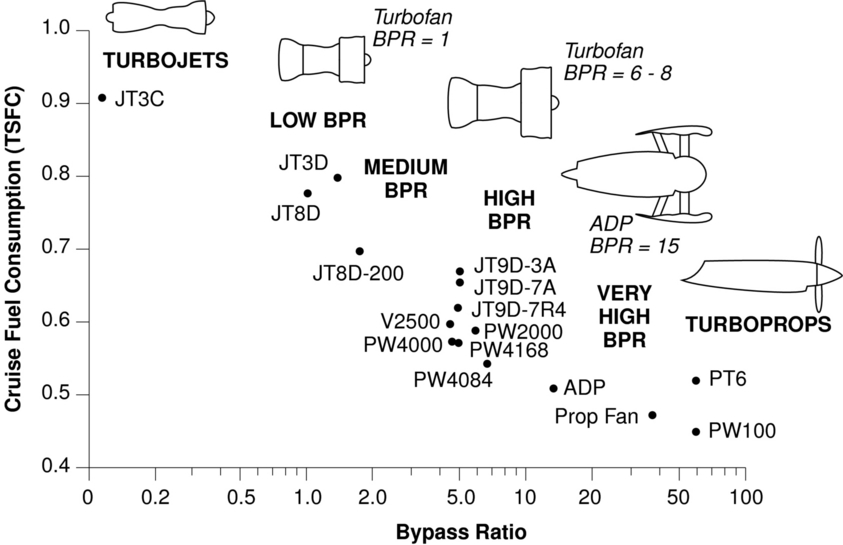


Figure ‑: Trend of bypass ratio and specific fuel consumption for various aircraft engines at cruising

### Thrust generated in a Turbofan

The total thrust generated by a turbofan engine is the sum of two thrust components. One of these components come from the acceleration of the bypass air flow, and the second component is due to the acceleration of the core air flow. Each of these thrust components is the difference in time rate of change of momentum at the inlet and the outlet of each section i.e. the core and the bypass section of the engine.

Total Thrust = Thrust of Fan + Thrust of Core

Mass flow equation,

Bypass ratio equation,

The total thrust equation becomes,

Thus a rise in mass flow rate through the fan, or the increase in bypass ratio increases the total thrust generated by the turbofan.

## CFD, Optimization and Mesh Morphing

### CFD – Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) refers to the branch of fluid dynamics where flow problems are solved numerically, commonly using computers owing to the low cost and high computing power of modern computers. This branch which utilizes numerical techniques to solve for complex real-world flows, for which closed form solutions are often not possible or extremely hard to obtain analytically, has seen tremendous growth in its accuracy and capability for a large number, if not all, of possible types of flows that engineers are often interested to study. Owing to this growth, there has also been a significant increase in the number of commercial CFD software available in the market, which include both general solvers as well as solvers for specialize applications.

A typical CFD design process consists of three stages, namely:

1. Pre-Processing
   1. Geometry modeling and clean-up – This should be done taking into account the mesh topology requirements, as geometry affects the mesh generation process.
   2. Meshing/Grid generation – The geometry is then broken down into a fine grid which forms the basis of numerical calculation which will be dome by the computer in case of CFD.
   3. Specification of Boundary Conditions and Inlet Conditions – Modern CFD techniques allow a wide range of boundary and initial conditions that can be applied to a surface. It Is important to choose the correct boundary and initial conditions to represent the real world situation as closely as possible when solving using CFD.
2. Solver
   1. Problem Specificaion – This can have information like, whether the simulation is a steady state or transient simulation, the convergence value for the residuals of the governing equations, the turbulence models (if present) to be used, etc.
   2. Numerical Simulation – Once the pre-processing and the problem specification is completed, the problem is solved iteratively using in-built numerical procedures (which may differ from solver to solver).
3. Post Processing
   1. The results obtained from the numerical simulation need to be post-processed to infer meaningful information from the result data.
   2. This is done using various contour plots on various surfaces depicting values of parameters of interest, which can be any fluid related properties like pressure, velocity, mach number, density, lift, drag, etc. The property of interest must be chosen according to the application
   3. The obtained values are then compared with values from different sources (for example, experimental results) for deviation.
   4. If the results obtained are satisfactory, the process can be terminated, else it may be repeated with a different geometry which will possibly overcome the shortcomings observed in the staring geometry.

Figure 2‑3 shows a typical process flow of a CFD design process. With recent advances in the robustness of CFD algorithms, and the validation of solutions obtained from CFD with real time experimental solutions, CFD has become the preferred design and analysis tool for a large number of industrial fluid flow related problems. Research is still going on to improve the existing algorithms to further reduce the deviation between the numerical and experimental results, as well as to increase the scope of available models to a more complex range of flows encountered in real world. But even then, the existing models when applied with care and prudence to the correct type of problem being modeled, can yield results which can be said to represent the flow with an accuracy sufficient to validate a design as good or bad based on the results of CFD. Not only that, a CFD analysis result can be used to study the flow characteristics in regions where getting experimental data can be technically or financially prohibitive. Another advantage of being able to perform complex analysis in a virtual setting is the flexibility with which the user can change the operating conditions, the fluid characteristics, or even the whole geometry of the test case altogether. Thus CFD analysis can be used to generate a set of parametric results quickly and without costs of building a large number of prototypes.

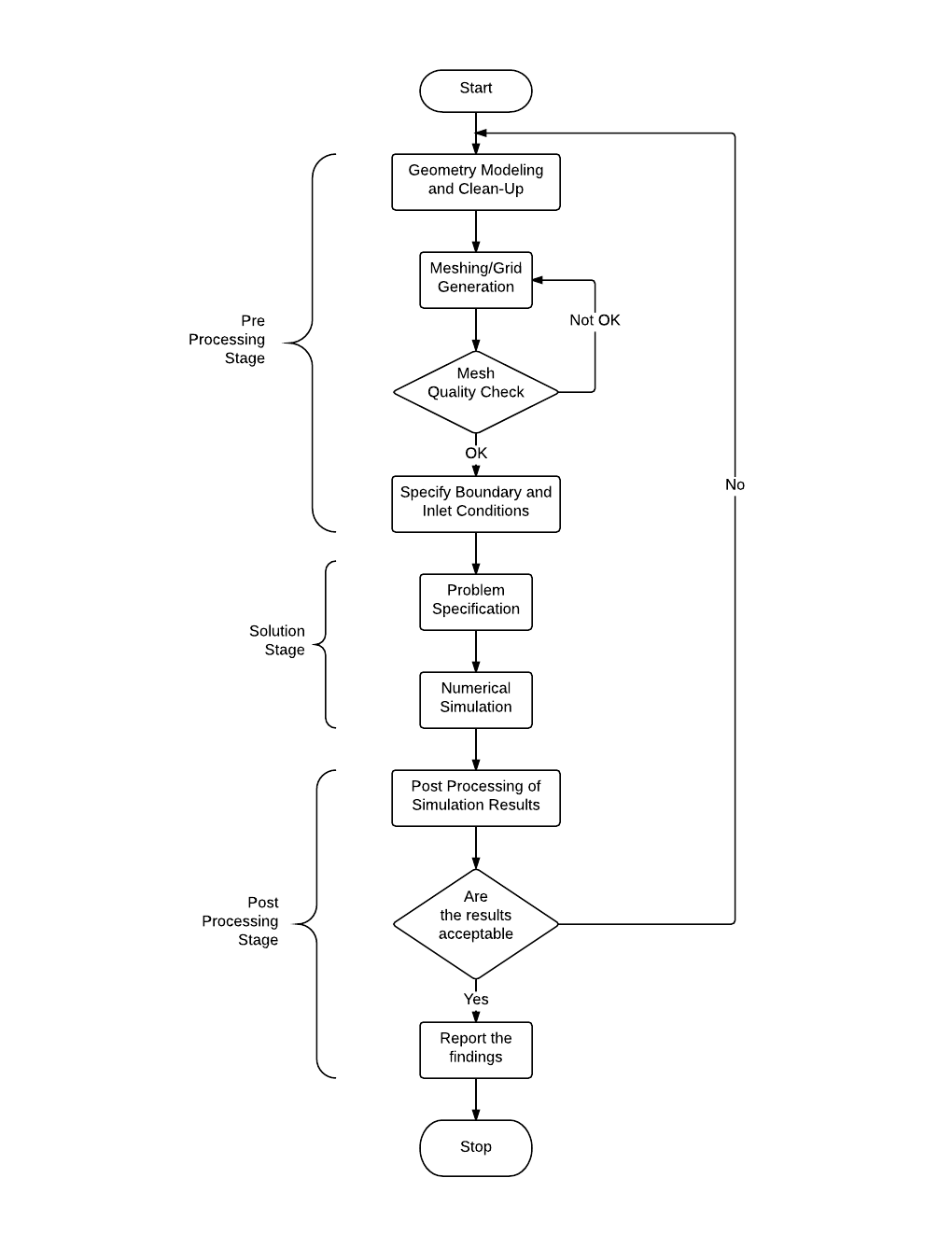


Figure ‑: Flowchart of a CFD design process

### Optimization using CFD

The CFD (and computational analysis methods in general) has reached a stage where it enjoys the confidence of engineers in the results it produces, and also the manner in which it produces these results. With high computing power available cheaply now, and with possibility of further augmenting such with parallelization of processing power across a number of dedicated computing nodes, engineers can quickly solve a large number of complex flow problems in a comparatively very short time as was possible few years ago. And with the constant advancement in both the available algorithms and computing power, the CFD techniques continue to be applied to more and more challenging cases, with complex flows inside complex geometries being one of the drivers of this research.

The ability to be able to solve such complex problems and subsequently carry out a detailed analysis of the results, as was possible never before due to aforementioned practical limitations, has also led to our better understanding of such problems. With better understanding of the problems, engineers continuously strive to improve their designs. But with better understanding of the problems, engineers also need to take into account more variables that may be affecting the solution. With the relationships between these variables often being complex to determine, it often takes a large number of iterative assessments on a particular design to achieve at what can be said as an optimized design.

This is where the power of CFD to generate a large number of parametric results without needing the large number of physical prototypes comes into play. The advantages of CAD and CFD together in terms of amount of time, money and energy utilized to generate an equal number of virtual prototypes and test them is significant. These advantages offered by CFD have caught the attention of designers and researchers alike and there are a number of optimization models available now to the users. Typically these optimization models will work alongside the traditional CFD solver, generating results from various prototypes according to a preset arrangement decided by the user (Figure 2‑4). Thus at the end when the optimization model has finished running, the user will have a set of results to analyze and compare and thus make an informed decision regarding the best design for the application under consideration.

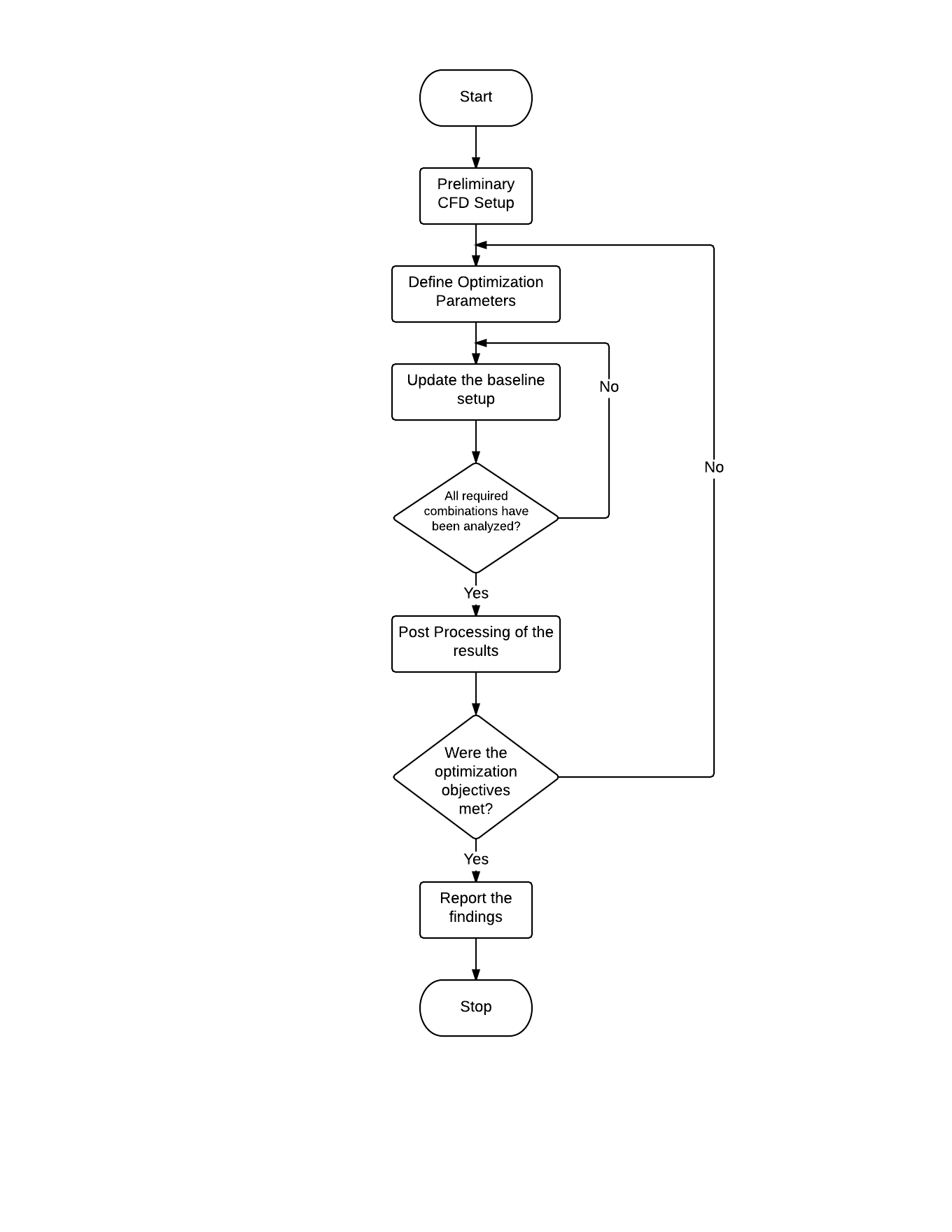


Figure ‑: Flowchart of a CFD optimization process

Optimization approaches can be applied to a new design being developed from scratch, or for optimizing an existing design. In case of a completely new design, all parameters affecting the decision may be open to decision (of course within some boundary constraints). Such optimization is also called *Fully-Parametric Optimization*. Although all parameters in this type of approach are open for optimization as a system, a conceptual design can be helpful to achieve the results quickly. Fully parametric optimization can be a powerful tool in the early development stages of a new design as it brings on the major changes at the early stages itself.

Certain applications may require an existing design created for a similar application to be optimized differently. This is also advantageous to the company because a new design from scratch may not be done. An existing, or baseline design, can be optimized to the output requirements of the new application. In such a case, only a few parameters may be required to be included in the optimization approach. Such an approach is called the *Partially Parametric Optimization*. It is also worthy to note here that a new product design process is often a series of iterative optimization procedures. A partial parametric approach can also be applied to later stages of a new design process for a revised fine-tuning of the original design obtained through a fully parametric optimization. Figure 2‑5 below shows how different parametric optimization approaches can be integrated in the design process.

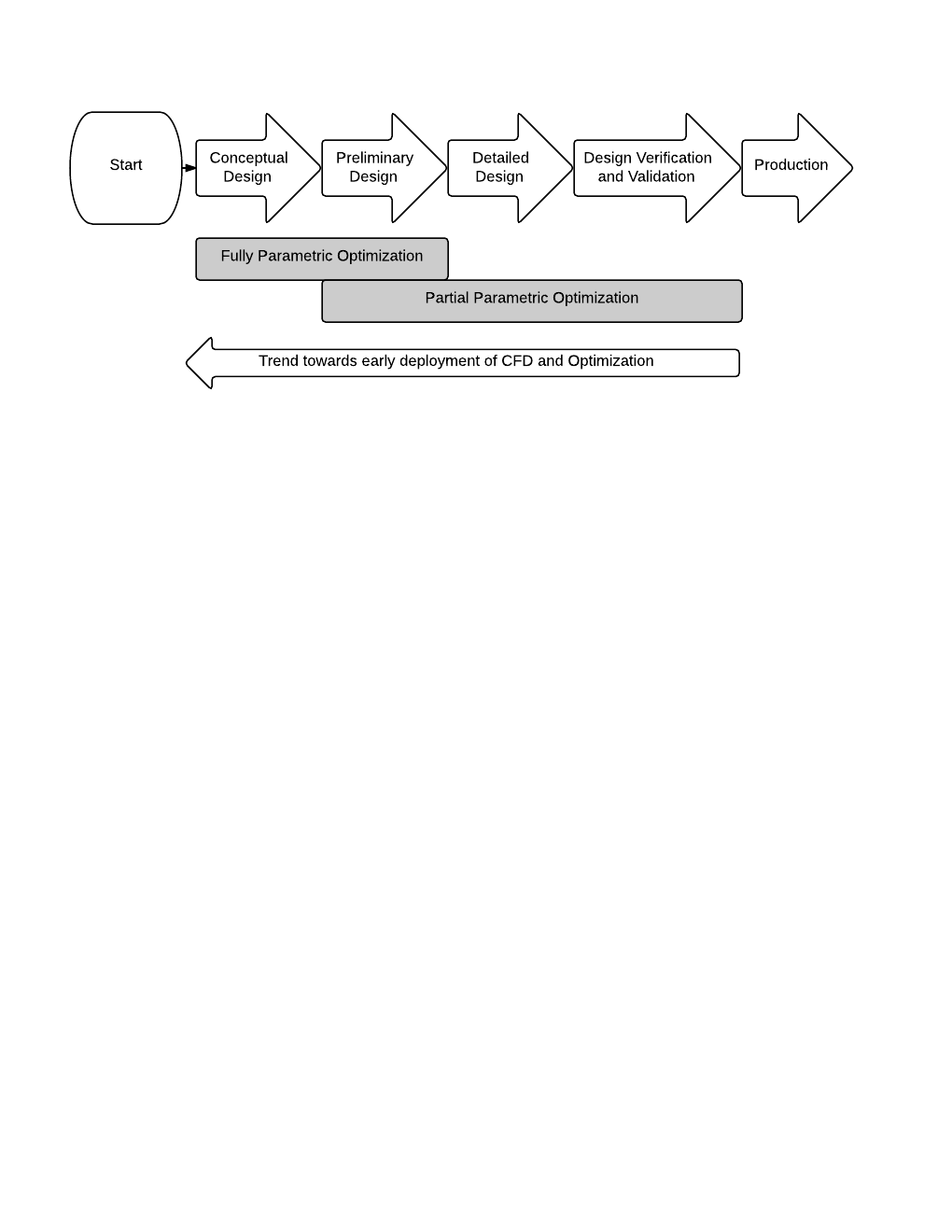


Figure ‑: Various optimization approaches and their application to design processes

Essentially, in both the types of parametric approaches, all the parameters considered important are changed in steps by an optimization algorithm. As many combinations of the different parameters as possible are assessed by running through a CFD solver. If due to resource constraints, all possible combinations are infeasible to test, the design team may select the combinations of permutations which are the most representative of all the combinations. Once all the cases have run through the optimization algorithm, the results can be compared to select the best combination of parameters for the application.

Another set of optimization approaches available to designers is the *parameter free approaches*. One such important approach utilized commonly is known as *Topology Optimization*. In topology optimization, rather than modifying the parameters, the emphasis is on optimizing the shape of the design. The approach is similar to changing the design to reduce stress concentration in certain regions in a structure, or modifying the flow shape in a fluid channel to achieve the best pressure drop etc. In a topology optimization algorithm, the shape of the design changes iteratively during a single run such that at the end of the run an optimized shape is obtained.

### Mesh Morphing

While a big part of the research in CFD has been traditionally focused on development of algorithms, however with the recent advancements which have driven the application of the numerical techniques to more complex geometries, another research area has gained attention of the researchers. In modern CFD analysis process, the bottleneck has now shifted from running the analysis to pre-processing steps. Pre-processing stage in an analysis is the stage where the entire domain of interest to the designer is broken down in a large multi-cell computational grid. The quality of this grid directly affects the quality of the solution the analysis will generate.

This process of breaking down the geometry in a multi-cell grid is referred to as *Meshing*, with each small cell called an element in the mesh. The mesh can be a one-directional mesh with linear elements, a two-dimensional surface mesh with surface elements, or a three-dimensional volume mesh with small volume elements filling the whole volume. The quality of the mesh is defined by the geometric quality of individual elements (parameters like length, skewness, aspect ratio), but also by how the elements are connected to each other in space. Another parameter of interest to the designer is that how closely the mesh conforms to the original geometry being analyzed. While this is not a problem with simpler geometries, it is something that must be taken care of when the geometry being analyzed has lot of curvatures.

Traditionally the mesh generation process has been largely a manual operation. In earlier stages when CFD was still developing, the geometries analyzed were simpler. Furthermore, unless for research purposes, the meshes were not required to be as robust as they are today. This was because even the solver algorithms available were simpler than they are now. It was only with dedicated research that complexity in models increased allowing engineers and designers to analyze complex geometries. However, as mentioned before, other areas which are part of the process remained mostly untouched from research. Pre-processing and meshing, is one such area.

The traditional method of mesh generation is to first generate the geometry to be meshed in a CAD package. This geometry then needs to be imported in meshing software which can generate and then export the mesh compatible with the solver to be used. In this mesh generator the engineer can create the mesh according to the requirements of the analysis. However, a quality mesh generation is a time consuming process even with computing power available today. This is the case because while the solving algorithms have improved, the meshing algorithms have received researcher’s little attention in comparison.

Modern mesh generators have undoubtedly come a long way from before. It is possible now to generate a fine, highly resolved mesh on a large geometry with a few clicks and minimal human intervention. And moreover, these automated meshes can be trusted to generate fairly accurate, usable results at least for initial conceptual design stages. But when high accuracy of the solution is of utmost importance to the analyst, he or she must step in to monitor the mesh generation process. This addition of human element in mesh generation makes the process subjective, but is essential when the mesh must conform to certain requirements of the analysis, and thus must be monitored carefully.

Now let us look at another aspect of application of CFD. CFD is primarily a design analysis tool. If we look at a typical design process, it starts with a conceptual design which then undergoes a series of analyses and improvements before being deemed fit for use. What this means is that CFD will not be used just once in the design process, but every time a design is changed. In the traditional CFD process this means that for a component with complex geometry, for example a turbomachinery blade, the mesh will have to be a generated again and again every time a change in the geometry is done.

A case in consideration here is when a design is being using one of the optimization algorithms discussed in the last topic. A parametric optimization algorithm might have parameters which define the geometry of the design being tested. As different values of such parameters will be realized during the process, each case might have a difference in geometry thus requiring a mesh to be generated separately for itself. Since in case of such an optimization process, the total number of individual cases may be huge, the need to remesh the geometry every time can be inconvenient and a bottleneck process in the chain.

Recently some researchers have focused their attention on this limitation of the CFD design process and efforts are now being made to streamline the mesh generation process. One of the promising techniques being utilized to this effect is called mesh morphing technique, which as its name implies, is capable of ‘morphing’ an existing mesh from one geometry to another.

Mesh morphing technique is a method of changing the shape, or *morphing* an existing mesh to adapt it to a new geometry. An important characteristic of morphing techniques is that it does not create any new nodes. The existing nodes in a mesh are shifted in space (translated or rotated) as per the requirements of the new shape, preserving the mesh connectivity information. An illustration of how this may be achieved is shown in Figure 2‑6. This has two important implications when morphing is being used as part of an optimization algorithm.

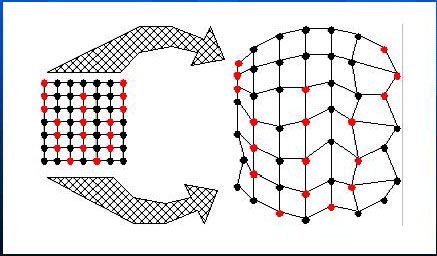


Figure ‑: Illustration of mesh morphing applied to a grid

Since there are no connectivity or topology changes during a morph process, the new shape of the mesh can be defined as a function of independent parameters, usually called the shape parameters. These shape parameters can have different values and can be thus used to create different shapes of the existing mesh just by doing some transformations according to the morphing function with the parameters plugged in. Looking back to the discussion of parametric optimization approach, where one or more parameters may be a shape parameter for the mesh, all that is needed to create the new shape is the original mesh, the value of the shape parameters and the morphing function for each case. Thus instead of having to reload the mesh for each case, only a single mesh with some additional parametric information is needed to run all the possible cases. This is a big save of time, and memory.

The second implication lies in the effect of mesh on the convergence. A remesh of the complete geometry from scratch will most likely have some variation from the original mesh with regards to node connectivity. Thus when the results are analyzed, the variation seen in the results of the mesh of original design and the mesh of modified design can be attributed to two factors – a certain fraction to the change in shape, but also an undesired variation due to the difference in mesh. By preserving the topology information, morphing can minimize the undesired effect of a complete remesh.

Thus, by integrating a mesh morpher in the optimization algorithm, a complete optimization process can be carried out independent of any user intervention, once the parameters have been defined (Figure 2‑7). This makes the process very time effective. Also, the results from such a coupled process can implicitly be assumed to largely reflect only the changes in the design parameters, and not any remeshing or other similar effects.

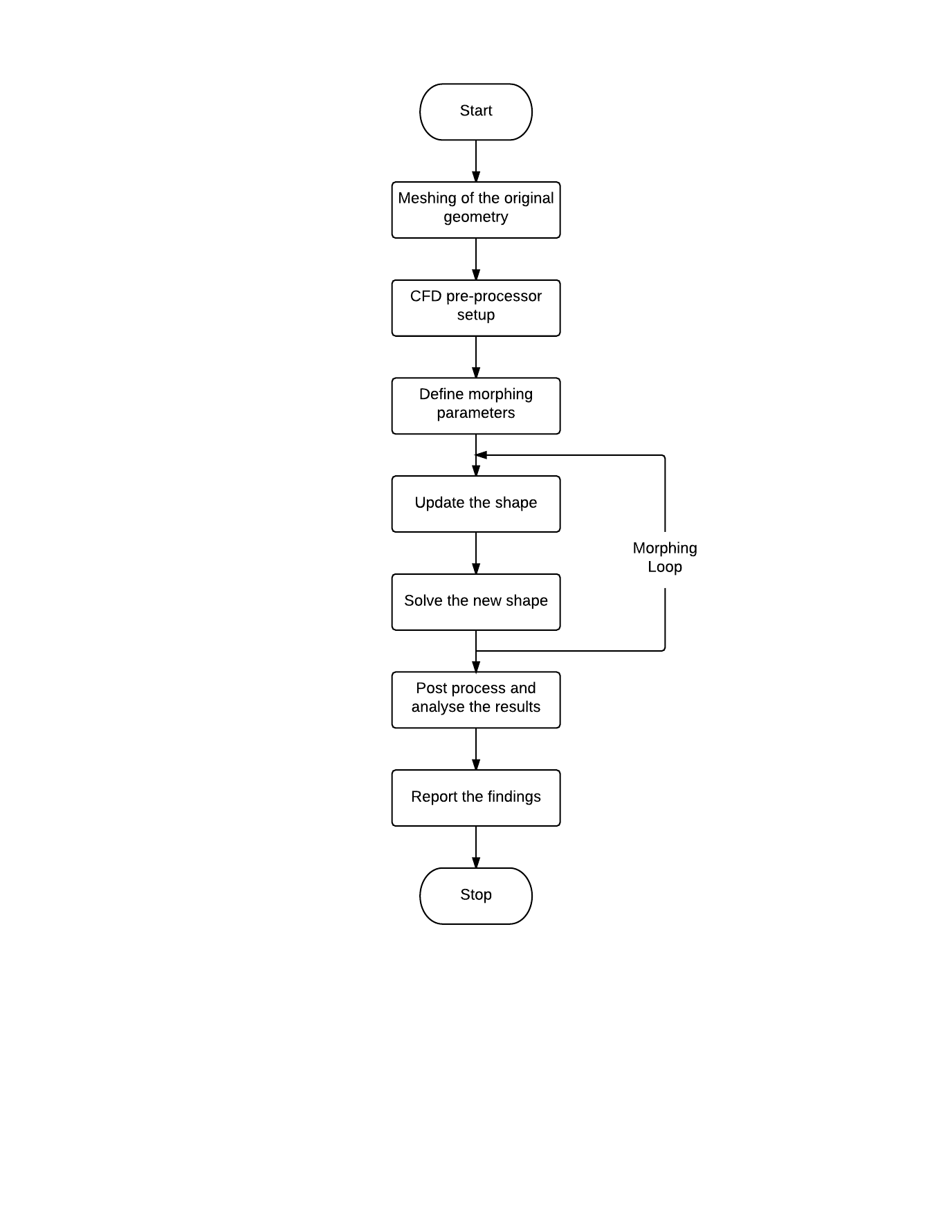


Figure ‑: Flowchart of a CFD optimization process with mesh morphing

# Meshing

## Test Object

The test object for this thesis project is an axial entry aircraft engine fan, being developed by General Electric Aviation. The fan is part of the GENX – n series engines. However, the exact make and model of the engine were not disclosed due to confidential reasons. The fan model was provided as a Solidworks CAD model file. The meshing software used is capable of importing surface and solid data from this particular CAD format. Figure 3‑1 shows how the test object looks like when imported in the meshing program. The following data was extracted from the provided model.

* Fan diameter – 1 m
* Fan Hub Diameter – 0.29 m
* Fan RPM – 1080 min-1 clockwise when looked from front

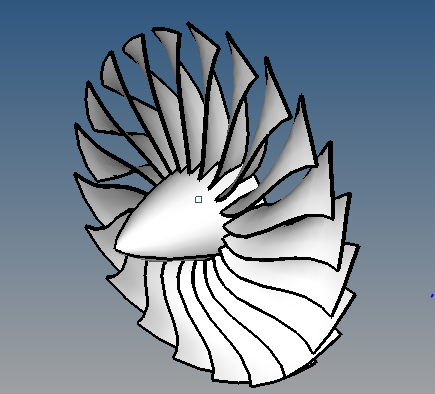


Figure ‑: The test object: GENX – n series aircraft engine entry fan

## Types of mesh available: Structured and Unstructured Mesh

CFD meshes used in the analysis are predominantly of two types:

1. Structured Meshes (mostly formed of quadrilateral elements for 2D and hexahedral elements for 3D)
2. Unstructured Meshes (mostly formed using triangular elements for 2D and tetrahedral elements for 3D)

Below the two types of meshes are discussed briefly.

## Unstructured Meshes

Unstructured or hybrid mesh, as the name implies, is an unstructured arrangement of elements in the meshing domain. Nodes are placed arbitrarily in the region to be meshed and connected via straight lines. However while connecting nodes it is ensured that no nodes lie inside an element volume. Figure 3‑2 shows how an unstructured solid mesh would look like.

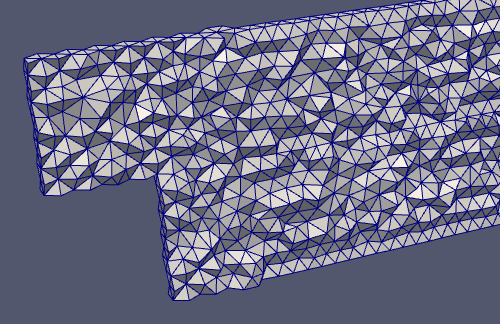


Figure ‑: An unstructured tetrahedral mesh illustration

The big advantage of using unstructured meshes is the level of automation available to the meshing engineer. Since the node arrangement is arbitrary in such a mesh, complex geometries can often be discretized with only a few parameters required to be entered manually. However, this automation process also means that the level of control available to the engineer is also limited. There are some important considerations for certain types of analyses for which the control offered by a structured mesh is very useful to get a good mesh quality and hence, a good result. We will discuss these in the following section under structured meshes.

Other disadvantages of using unstructured meshes are related to the grid size and computational time. For a 2D triangular element mesh, twice the numbers of elements are required than for a quad element mesh. For a 3D tetrahedral element mesh, six times the numbers of elements are required than for a hexahedral element mesh. Thus it can be seen how large an unstructured mesh can be compared to a structured mesh. Moreover, since the arrangement of nodes is arbitrary, information (such as coordinates) about each node has to be stored separately along with a connectivity table. This leads to a tremendous increase in computational time as would be required for a similar structured mesh.

## Structured Mesh

A structured mesh, as opposed to an unstructured mesh, is a structured arrangement of elements where the connectivity between the nodes and elements is implicitly defined. The elements in a structured mesh (elements for 2D and hexahedral elements for 3D) are arranged in a I x J x K array (K=1 for 2D) such that if (0, 0, 0) are the coordinates of the first node in the array, the coordinates (i, j, k) of any node can be determined using the information that the (i, j, k) has neighbors at (i+1, j, k), (i-1, j, k), (i, j+1, k), (i, j-1, k), (i, j, k+1) and (i, j, k-1). This simple structural arrangement has a powerful effect on the performance of this type of mesh, since the neighboring node information is easier to locate, making the process much faster and also one requiring less storage space. Figure 3‑3 below shows a volume meshed with a structured hexahedral mesh arrangement.

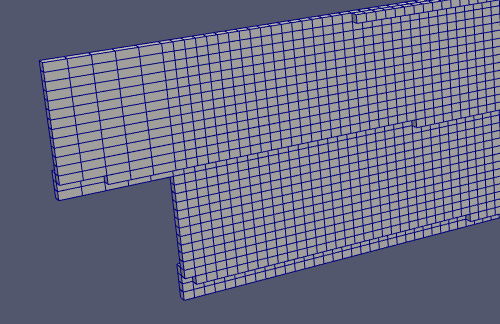


Figure ‑: A structured hexahedral mesh illustration

There are other advantages of using a structural mesh too:

* Less number of elements for a given number of nodes
  + A hexahedral mesh with a certain number of nodes has considerably less number of elements than a tetrahedral mesh with the same number of nodes. This is due to the more regular arrangement of elements in a hex mesh. The asymptotic estimates for number of elements for a mesh of *N* nodes, is *5N* for a tetrahedral mesh, and *N* for a hexahedral mesh. Since in a Finite Volume CFD method, number of degrees of freedoms is equal to the number of elements, this corresponds to a much more efficient solution.
* A hexahedral mesh arrangement can be used to optimize mesh density by reducing number of elements in streamwise direction.
  + This is especially useful in the regions near inlet and outlet, where gradients in properties are low in the streamwise direction (along the flow) and thus the elements can be skewed longitudinally without affecting the results considerably. Such an arrangement is difficulty to achieve with a tetrahedral mesh without giving up the element quality, thus leading to either an inefficient mesh or inefficient mesh distribution.
* A better boundary layer resolution can be obtained with a hexahedral mesh.
  + Since hexahedral mesh allows us to control the element density in our direction of choice, such a mesh can be resolved to have very fine element density in the direction perpendicular to the wall to capture the wall effects without compromising the total mesh size. Even the meshes with tetrahedral elements in the bulk domain are often coupled with a hexahedral mesh near the walls.
* Information flow between nodes is ‘smoother’ because the arrangement is regular.
  + This is a clear advantage when the gradients in the flow are concentrated along the edges of the hex elements. However this advantage becomes more and more subtle as geometric complexity increases, as then the gradients are not always aligned to the edges. However, this is almost never possible in a tetrahedral mesh.

However structured meshes do not come without problems. Creating a structured mesh requires much more human effort and time. Even with a simple geometry, the process is not really straightforward, and can involve considerably increasing user interaction as the geometries become more complex. A structured mesh arrangement has an equal number of nodes and elements on the opposite faces of the geometry being meshed. This is not always possible in the geometry being meshed and thus it mostly falls upon the user to decompose the global domain into smaller geometries the sum of which makes up the complete geometry, and each of these individual volumes must be ready to be meshed to produce a structured grid. Such a volume is commonly referred to as a mappable volume. Research is ongoing in this area to automate the geometry decomposition process by helping the meshing software identify features and corners before beginning the actual meshing process. While the available methods have shown promise, they are still far from being able to apply independently on complex real geometries. Figure 3‑4 shows an example of how the decomposition process might work.

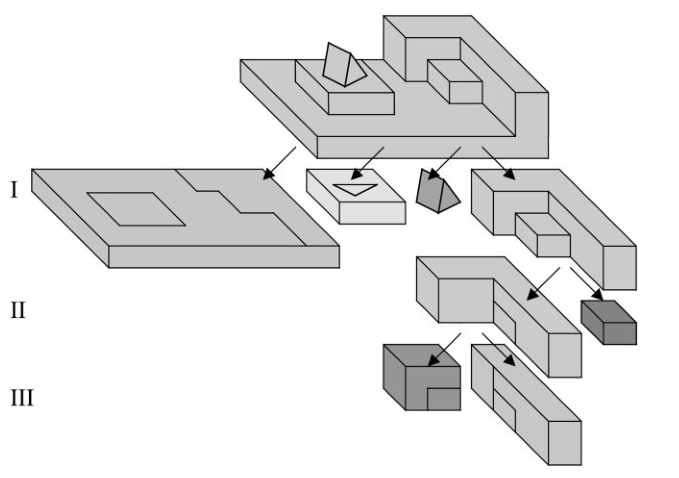
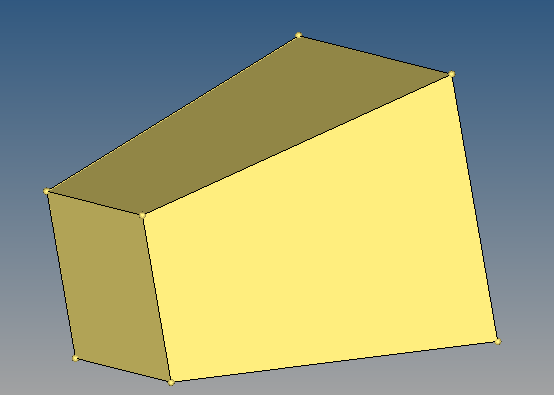
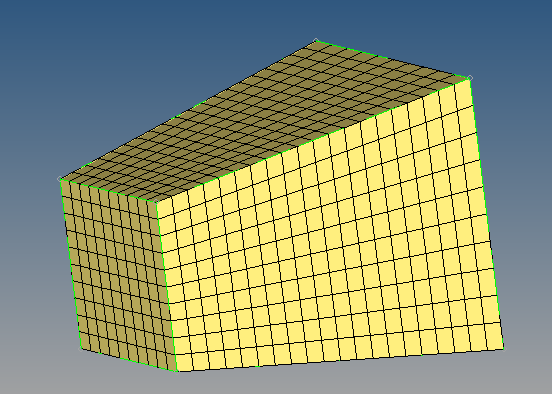
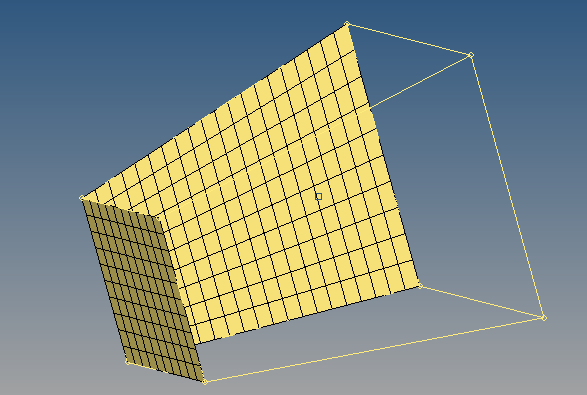


Figure ‑: Example of geometry decomposition into smaller, logical volumes for automated meshing

One of the most common methods to generate a structured mesh for complex geometries is using a sweeping algorithm. As with any structured mesh algorithm, the primary requirement here is for the volume of interest to be a mappable volume. If the volume as a whole is not mappable, it must be logically decomposed into smaller blocks each of which must be mappable. Once the decomposition is complete, each smaller volume must be meshed using the following steps. First, a quadrilateral structured 2D mesh must be generated on a carefully selected face, and then this surface mesh can be swept in a logical direction such as to obtain layers of hexahedral elements with the same topology as the swept quadrilateral surface mesh. The distribution of these hexahedral elements can be controlled by having the target as well as some of the across faces meshed with desired mesh topology. The process is illustrated in Figure 3‑5 with enumerated steps.

**1**

**3**

**2**

Figure ‑: Solid regular hexahedral meshing using mesh sweeping – 1) Create a mappable volume, 2) Mesh a source surface and (optionally) an across surface, 3) Create solid mesh using sweeping

A quick review of the sweeping algorithm makes it clear how a combination of manual and automated tasks can be used in a controlled manner to get a high quality mesh as output of the process. In the above example, the manual steps were the decomposition of the big domain into smaller mappable domains, and also the mesh generation on the source and target surfaces. As long as these tasks are done in a satisfactory manner, the algorithm can be trusted to provide the designer with a high quality robust mesh for further analysis. However, geometry decomposition is a difficult task, and it becomes more and more tedious as the complexity of the geometry increases. And also it may lead to a situation where the logical decomposition of the geometry leaves the designer with little control over the topology of the mesh. These challenges must be met before a fully automatic and reliable mesh generation algorithm can be realized.

## Mappable Volumes

As discussed in the section above, a hexahedral structured mesh can only be generated if the volume being meshed is a mappable volume. In a mappable volume, a surface mesh existing on one face of the volume can be swept, or extruded, to the opposite face, along a logical direction, to obtain a regular solid map mesh.

The map volume meshing scheme can only be applied to volumes that can be meshed such that the mesh represents a logical cube. To represent a logical cube, a volume mesh must satisfy the following general requirements:

* There must exist exactly eight mesh nodes that are attached to only three mesh element faces. (These eight mesh nodes comprise the corners of the logical cube.)
* Each of the eight corner mesh nodes must be connected to three other corner mesh nodes by means of a straight chain of mesh edges-that is, a chain of mesh edges all of which belong to a single logical row of mesh nodes.

According to the criteria described above, the most basic form of a mappable volume is a rectangular brick (Figure 3‑6). For such a volume, the mesh nodes located at the corner vertices of the brick constitute the corners of the mesh cube.

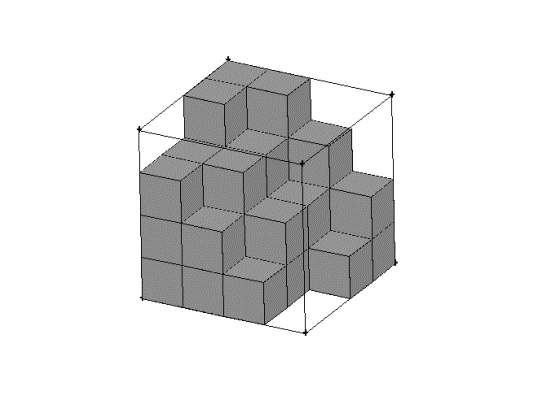


Figure ‑: Map meshing scheme – partial array of hexahedral elements

## Mesh Generation for the provided test object

A fully structured hexahedral mesh was generated for the analysis. The mesh mapping method used to create solid elements by sweeping 2D surface meshes has been explained briefly above. The same method was applied for creating the mesh for this project.

### Geometry Preparation

The model of the test object was provided as a Solidworks CAD solid model file. The model could be imported in the meshing software where pre-processing on the model before meshing could be done to prepare the geometry for grid generation.

The CAD model provided was a solid model of the fan. However, the region of interest for this study is the flow passages between the fan blades i.e. where the fluid flows through the fan. The flow passages are concentric and are assumed to be similar throughout the circumference of the fan. Thus, instead of modeling the flow through the whole fan, only one passage section of the fan is considered for analysis, and the results are assumed to be similar through the rest of the passages.

The solid component is deleted before proceeding, however preserving the surface information. The surfaces of the blades and the hub (or axis) surface in the chosen passage also become the bounding surfaces of the fluid passing through the passage. The passage is closed with another bounding surface on the top side of the passage (near the tip of the blade). Appropriate boundary conditions will be applied to these surfaces and these will be discussed later.

It is a good practice in CFD analyses to keep the inlet and outlet of a flow domain far away from the actual region of interest. This helps mitigate the influence of the applied inlet and outlet boundary conditions on the result of the analysis, unless the objective of the study is to study the effect of changing inlet or outlet conditions. By taking away the perturbations caused by the boundaries further from the region of interest, this helps ensure with maximum accuracy that the results obtained in this region are representative of the actual conditions.

A similar practice is observed while modeling the domain for this analysis, and the inlet and outlet surfaces of the passage are extended in respective directions to obtain the actual inlet and outlet surfaces for the flow domain which will be used in the analysis. For this model, keeping in mind the steep curvature of the fan blades at the leading and trailing edges, the inlet and outlet are extended by a length equal to ten times and twenty times respectively of the axial chord length of the blade.

Once all the required surfaces have been created, it is necessary to stitch the overlapping edges wherever the surfaces join with each other (Figure 3‑7). This is important because the flow domain must be a closed volume within which the flow equations will be solved with respect to the applied boundary conditions. Also, some edges might need to be suppressed. Suppressing an edge treats the edge as if it is non-existent.

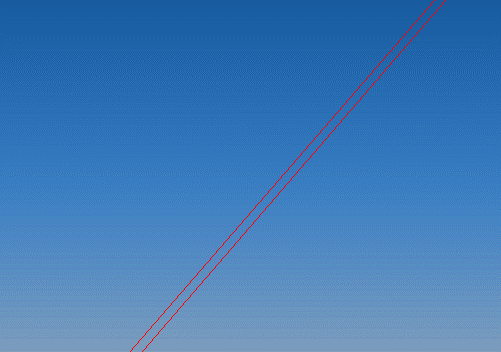
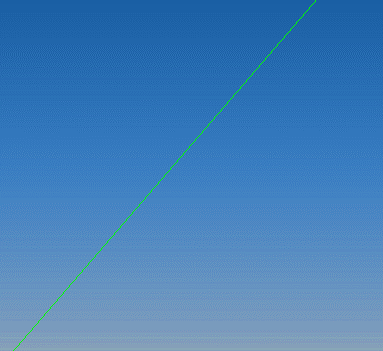
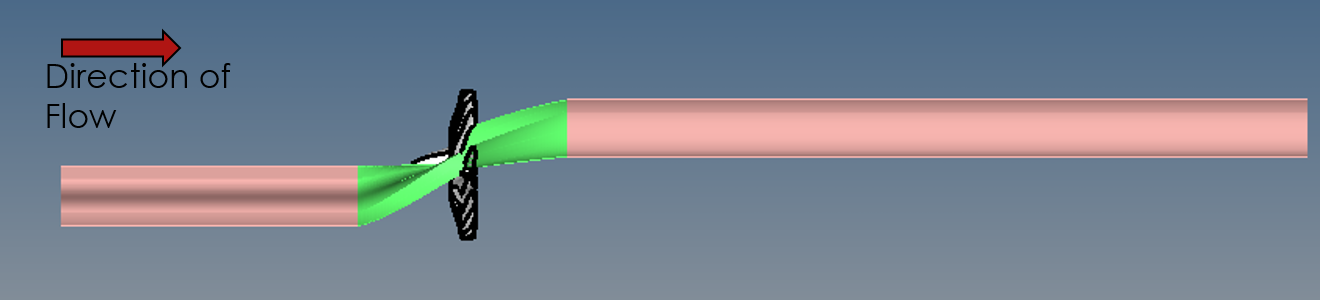
 

Figure ‑: Stitching edges

Another step in geometry preparation is separation of the rotating volume from the non-rotating volume. When a fan is placed in an air flow, in a large room for example, only the volume of air which is passing through the fan, or in close vicinity of the fan can be assumed to have some swirl due to the movement of the fan blades. The rest of the air in the room is still, or unaffected. Of course, the draft created by the rotation of the fan affects the movement of flow in the room, but only the air in the vicinity of the fan has the swirl generated by the fan blades. Analogizing the same conditions for this analysis, the volume of the air which is flowing through the passage and in vicinity of the passage is to be separated as a different volume collector which, for the purposes of analysis will be assigned a different reference frame than the rest of the volume. This reference frame will be a rotating frame with the rotational velocity equal to the angular velocity of the fan. Figure 3‑8 shows the modeled flow domain for one blade passage of the fan.



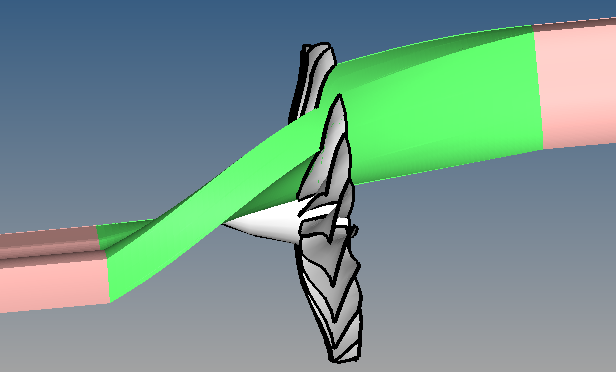


Figure ‑: Modeling of the flow domain from the given test object – the volume labeled green is the rotating volume, the upstream and downstream volumes are stationary

### Checking volumes for mappability

Since it is being aimed to generate a regular hexahedral volume mesh for this analysis, all the individual volumes to be meshed should be mappable volumes. The definition of mappability of a volume, and what makes a volume mappable has been explained previously. However, as also mentioned before, when the geometries become complex, it is at times difficult for the meshing software to be able to gauge a given volume as mappable, even if the volume looks as if it is satisfying the required criteria.

In such a case, the user must intervene to logically split the volume into bits of smaller volumes, such that the sum of these individual bits makes up the whole volume. This process must be repeated until each small volume is mappable individually. For this model, this splitting was done along the span of the blade.

Different sections of the volume were created by splitting it longitudinally (along the flow passage) at different spans of the blade. The first attempt was made with the split in the center of the blade span, resulting in two half sections of the volume. However, this still resulted in a few non-mappable volumes. Again, it was attempted to create three sections of volumes, by splitting at one-third of the blade height each from the tip and base of the blade. While this resulted in all resulting volumes as mappable, some volumes were mappable in all three directions, some were however mappable only in one direction. At this step, it was again ensured that all the overlapping edges have been stitched together. Figure 3‑9 shows the domain as split in the smaller mappable volumes.

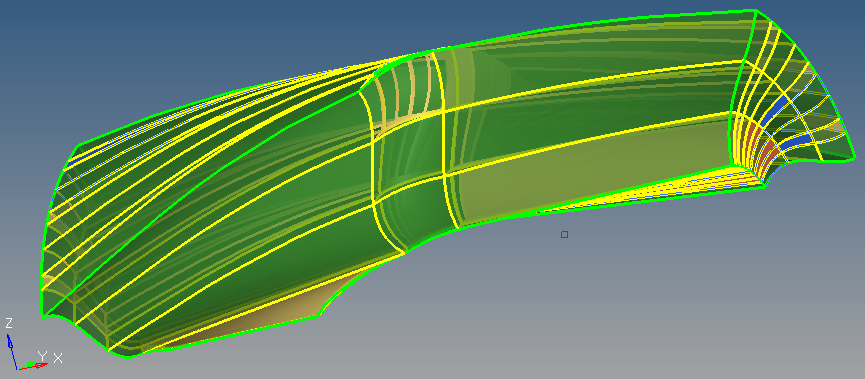


Figure ‑: Creating mappable volumes – the split lines can be seen

Direction of mappability is the direction in which the sweeping, or the extrusion process can be carried out within a volume so as to obtain a solid element mesh. A volume which is mappable in all three directions allows to use any two opposite faces as the source and target faces and any connecting edge as the across edge. A volume which is not mappable in all three directions can still be meshed with a regular solid element arrangement, however the mapping can only occur between a certain pair of opposite faces and not any pair of opposite faces in the volume.

### Creating the Mesh

Once all the volumes have been verified to be mappable, all the preparations for the actual meshing have been done and the meshing process can be started.

Since satisfying the conditions of mappability has resulted in number of smaller, different volumes which sum up as the total flow passage volume modeled as domain of interest, each of these small volumes must be meshed independently using the sweeping process to generate the volume mesh. It is also important to ensure the continuous mesh connectivity between all these individual meshes. Mesh connectivity is important to ensure that the information is passed from each node to all the connected nodes properly during solving the governing equations over the grid.

As discussed before, the mesh generation starts from creating a surface mesh on one or more faces of the volume. The mesh generation process was started from the passage volumes since it is the region of interest, and hence it is desirable to have most control over the mesh in this region. Due to steep curvatures of the geometry, meshing only one face was not enough to obtain a satisfactory volume mesh. Thus, two opposite faces and two across faces were meshed with a surface mesh before proceeding with the mapping operation. Care was taken to have higher element densities in critical regions, for instance the blade surface, and the leading and trailing edges. While a boundary layer was created along the direction perpendicular to the wall surface, element sizes were gradually changed along the chord of the blade as well, with elements being closer near the edges of the blades. A similar bias was introduced in the element density along the length of the blade, with a progressively smaller element size and thus higher element density near the hub and the tip section of the blade, beginning from the center of the blade span (Figure 3‑10).

All the passage sections were meshed keeping in mind the guidelines mentioned above. Once the critical passage volumes were meshed, the remaining volumes i.e. the volumes in front and rear of the blade were meshed similarly. While meshing these volumes, there were no boundary layers to be taken care of but element sizes were progressively kept smaller in the streamwise direction as the flow approached the passage (Figure 3‑10).

As mentioned before, mesh connectivity in such a meshing process must be ensured. When creating volume mesh with a set of source surface elements, mesh connectivity can be ensured by using the faces of the neighboring meshed volumes, wherever exist, as the source elements. However, sometimes when mappability is not available such that an existing mesh may be used, the elements should later be connected by equivalencing the nodes. Equivalencing the nodes is similar to the stitching of edges during geometry preparation stage, where coincident entities are merged together as one.

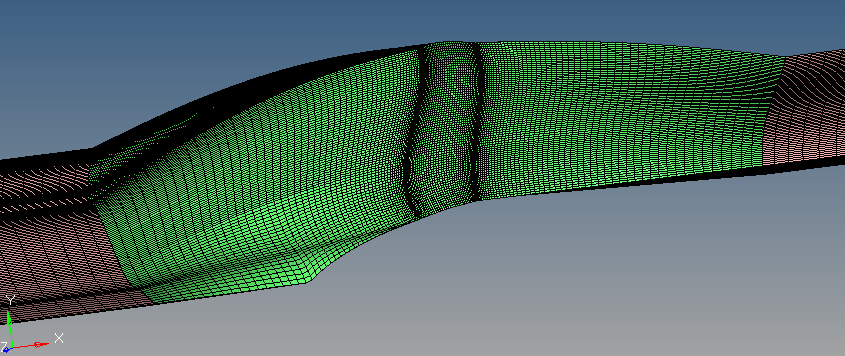


Figure ‑: The meshed domain – the higher density of elements near the blade edges, and also the varying element density in the stream-wise direction, can be seen.

### Boundary Layer Mesh

Boundary layer meshes are important to capture the effects of boundary layer when a flow is occurring close to a wall surface. These effect occur due to the viscous nature of the fluid. When a fluid passes over a no-slip wall, velocity of the local fluid layer attached to the wall is the lower than the free stream velocity. The region where this transition happens, from the free stream velocity to the boundary layer velocity, is called the boundary layer region. Boundary layer has a thickness which is dependent on the Reynolds number of the flow.

To efficiently capture the gradients in a flow using CFD, the resolution of the mesh should be higher than the resolution of the gradients desired to be measured. Boundary layers fall in the category of flows which exhibit steep gradients, hence it is important that a significantly finer mesh be used for modeling the boundary layer flow i.e the flow close that will occur close to a wall.

A common method used by the CFD community to determine a suitable element size for the first element in the boundary layer is the *y+ (y-plus)* factor. This factor takes into account the Reynolds number of the flow, and the shear stress factor due to the wall friction to estimate what will be a good value of the first element size in a boundary layer i.e. the element adjacent to the wall.

Different guidelines are available for what is a good target value of y+ for a suitably resolved mesh. Usually, lower value of y+ means a smaller element size, and thus a better boundary layer mesh. Since this study is an optimization study and the wall bounded effects are of secondary importance, a very low value of y+ may not be essential for a reasonable result. Using an extremely small y+ value will also lead to very small elements in the boundary layer and when applied morphing algorithm, these elements are likely to get distorted which is also not desirable. Hence it was decided to use a neither very high, nor very low y+ value of ~20 to estimate the boundary layer size.

With this choice of the y+, the first element height was estimated to be 0.5 mm and this is the height used in this mesh. The growth rate for the boundary layer was kept at 1.2 which is well within the guidelines of CFD best practices. A total of 8 individual boundary layers were used to define the boundary region. In Figure 3‑11, the red trapezoidal region is the region where the boundary layer propagates from the blade wall (in pink), with progressively finer element sizes closer to the wall.

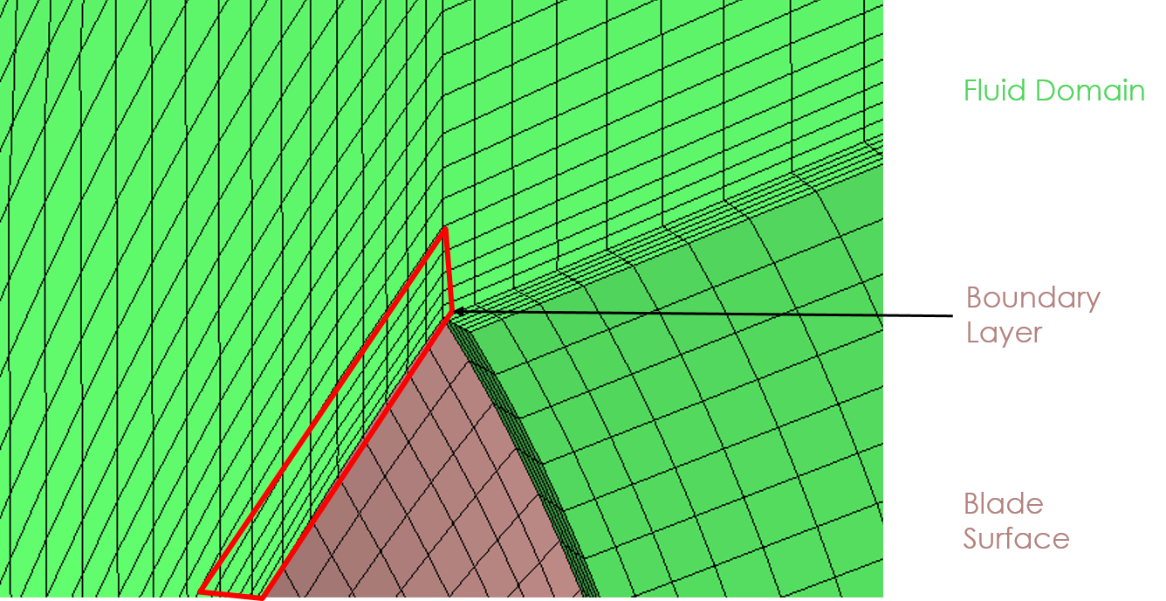


Figure ‑: Meshing the boundary layer regions

### Mesh Statistics

The following are the details of the mesh count and quality statistics.

* Total element count – 838200
* Total node count – 807576
* Total surface elements – 61950
* Total solid elements – 776250
* Minimum Jacobian – 0.44
* Maximum skew – 78 degrees
* Maximum interior angle of quad faces – 168 degrees
* Minimum interior angle of quad faces – 12 degrees
* Maximum aspect ratio – 2755
* Maximum aspect ratio in the passage elements – 38
* Minimum Jacobian in 2D elements – 0.46
* Maximum skew – 78 degrees

# CFD Setup and Analysis

## Baseline configuration analysis

The baseline configuration in this study is the blade shape of the provided model of the fan. In this section the setup used for CFD analysis applied to the baseline configuration has been specified. A number of different trials were done with varying setup parameters before finalizing this setup to be used for further analyses. In the subsequent analyses, the CFD setup used remains the same. Only the shape of the model is changed for each step of optimization, so that the changes can be attributed only to the change in shape, and not any change in setup. The setup is also illustrated in the Figure 4‑1.

## Setup and Boundary Conditions Used

* Type of analysis – Steady State
* Convergence level for residuals – 1e-4
* Maximum iterations – 300
* Under relaxation factor – 0.5
* Temperature equation – Off
* Turbulence model – One equation Spalart-Allmaras model
* Material model
  + Fluid
  + Constant density air
  + Standard atmospheric air density – 1.225 kg/m3
* Reference frame
  + Rotational reference frame.
  + Rotation speed – 1080 min-1
* Upstream and downstream volumes
  + Stationary volumes.
* Passage volume
  + Rotating volume with the reference frame applied.
* Inlet
  + Inlet is specified as a stagnation pressure inflow.
  + Inlet pressure – standard atmospheric pressure.
  + No swirl at inlet.
* Outlet
  + Outlet is specified as a static pressure outflow.
  + Outlet pressure – standard atmospheric pressure.
* Blade surfaces
  + Blade surfaces are no-slip walls.
  + Reference frame applied on the included elements, since the blades are rotating components.
* Hub
  + Hub surface is also set as a no-slip wall.
  + Reference frame is applied to the hub faces too, since hub rotates with the blades.
* Periodic surfaces
  + The surfaces on either side of the rotational direction of the flow passage are periodic surfaces (Figure 4‑1).
  + These surfaces are coincident with the other similar surfaces of the remaining flow passages, had they been modeled for analysis as well.
  + Since for simplification purposes only one passage has been modeled for analysis, thus assuming all the passages are equivalent, it can be said that the flow conditions within the individual passages will be repeating over the whole circumference, and hence are periodic over these surfaces.
* Interior/Interface surfaces
  + Interface surfaces are the surfaces between two adjacent volumes (Figure 4‑1).
  + In this case, interfaces are present between the upstream and passage volume, and the passage and downstream volume.
  + Interface surfaces are only interior surfaces and have no boundary conditions, and are only there to facilitate a closed volume
* Remaining surfaces
  + The remaining enclosing surfaces around the flow volume are modeled as slip surfaces.
  + Slip surface condition does not allow the flow to leak out from the volume, but does not affect the flow otherwise.

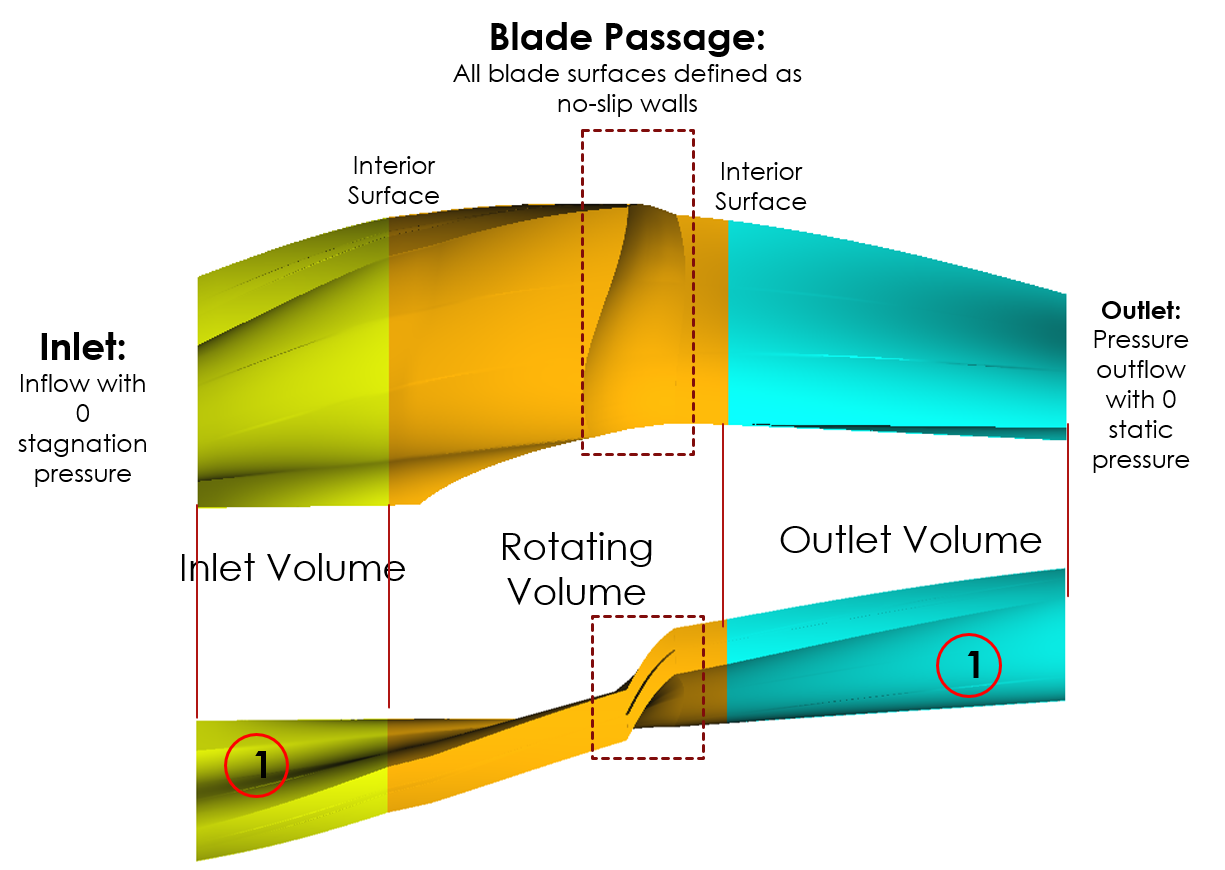


Figure ‑: Boundary conditions and CFD analysis setup – the surfaces marked **1** are the periodic surfaces

# Results and Discussion – Baseline Shape

In this section results obtained from the analysis of the baseline shape are presented and discussed briefly. These results will later on be compared with the results of the analysis of numerous shapes that will be generated as part of the optimization stage of the study.

## Convergence and Residual Ratios

With the setup defined in the previous section, the baseline model of the fan solved with residual ratios reaching convergence level in 148 iterations. The solution was completed in a little under eight hours while running on a high performance node with twenty processing cores and 64 GB RAM. The run time for a model is important to be optimized because a large number of runs will be performed on a range of different shapes for selecting the optimized shape among them. As can be seen in Figure 5‑1, which is the plot of the residual ratios with respect to the iteration steps, the convergence is smooth with residual ratios falling smoothly until below the desired convergence value.

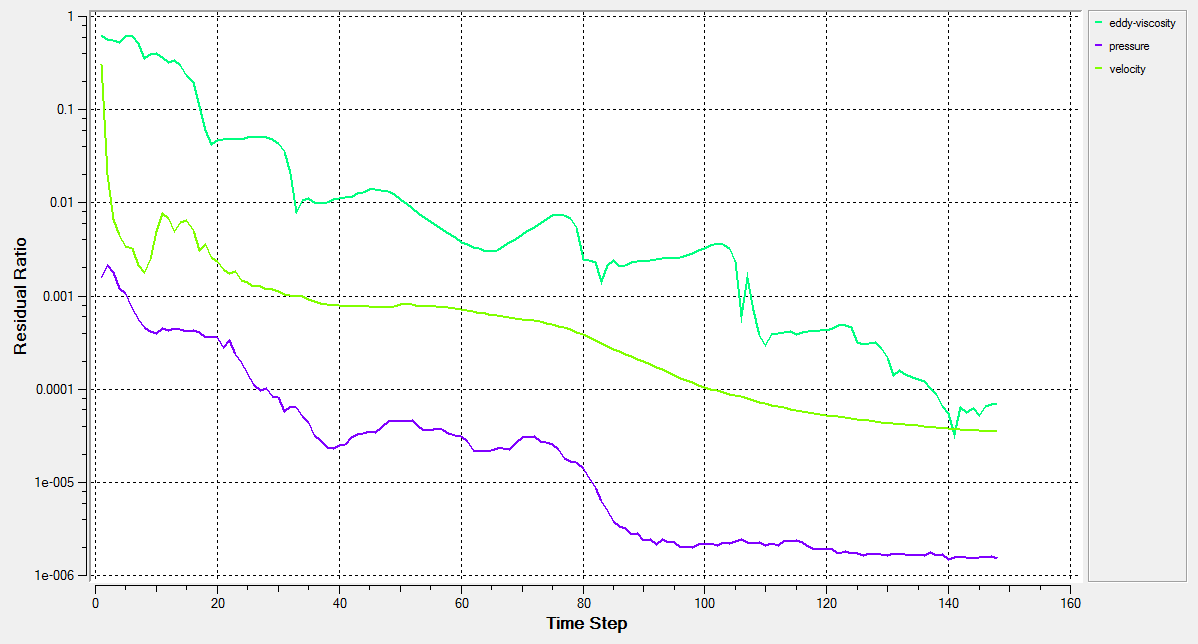


Figure ‑: Residual ratios plot for the baseline shape analysis

## Flow parameters

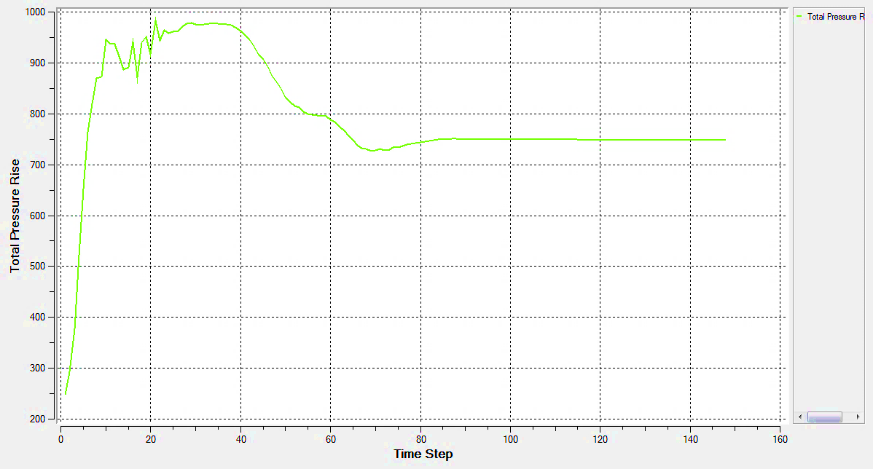
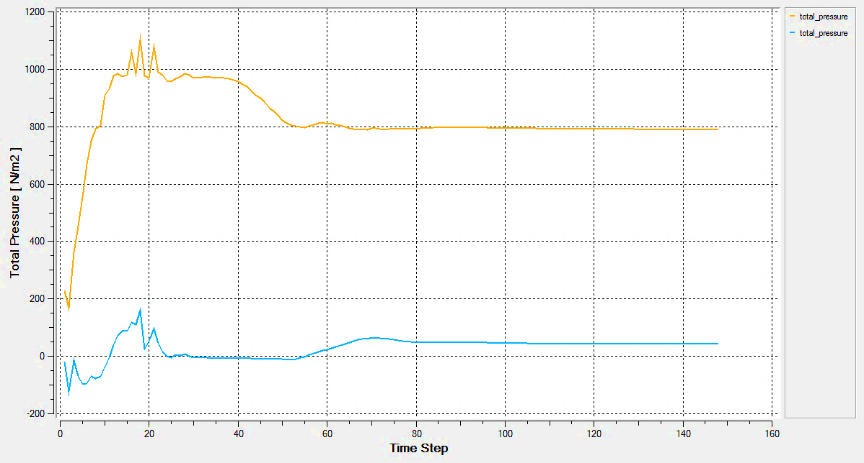
The pressure and velocity values were evaluated at different sections in the flow region. Some of the contours obtained from post-processing the results are shown below.

The mean total pressure rise in the air as the flow passes through the fan is observed to be **747.5 pascal**. This total pressure rise in the flow is later expanded in the exhaust nozzle to atmospheric pressure. The expansion also causes acceleration in air velocity as it exits the engine hence generating the forward thrust as described in the Section 1. This was evaluated by differencing the mean values of total pressure at the sections just upstream and downstream of the blade passage (Figure 5‑2). The total pressure rise can be attributed to the partial rise in static pressure and the dynamic pressure. The static pressure rise through the passage was observed to be 250 pascal, and the remaining 497.5 pascal is due to the dynamic pressure rise.

The pressure difference between the two sides of the blade, namely the pressure side and the suction side was found to be **454.4 pascal** (Figure 5‑4). This is an important parameter as it is directly related to the loading on the blade (Figure 5‑5). A higher pressure difference between the two sides translates to a higher loading on the blade which is undesirable in a turbomachine, especially with fan blades which are usually slender and of a high aspect ratio.

The mass flow through the fan was observed to be **0.8837** **kg/s** (Figure 5‑10). Mass flow is the parameter of interest in this study as the fan blade is being optimized for maximizing the mass flow. A higher mass flow as explained earlier leads to a higher total thrust generation in the engine. The mass flow value in this baseline model will be compared with the mass flow values in the other shapes to select the shape which has the maximum mass flow among the tested options.

A look at the pressure and total pressure contours also shows how the pressure varies as it passes through the blade row (Figure 5‑3, Figure 5‑6, and Figure 5‑7). Velocity vectors at different sections also show the rise in velocity magnitude as the flow passes through the blade row (Figure 5‑8, Figure 5‑9). The differential rise in velocity through the blade span is due to the large variation in blade velocity from hub to base. As blade velocity at a section is directly proportional to the radius of the section from the rotational axis, blade velocity sharply increases from hub to the tip. Thus the flow passing closer to the tip section also sees more augmentation in its velocity as it passes through the blade row.



Total Pressure Rise

Total Pressure Downstream

Total Pressure Upstream

Figure ‑: Total pressure variation upstream and downstream of the blade passage

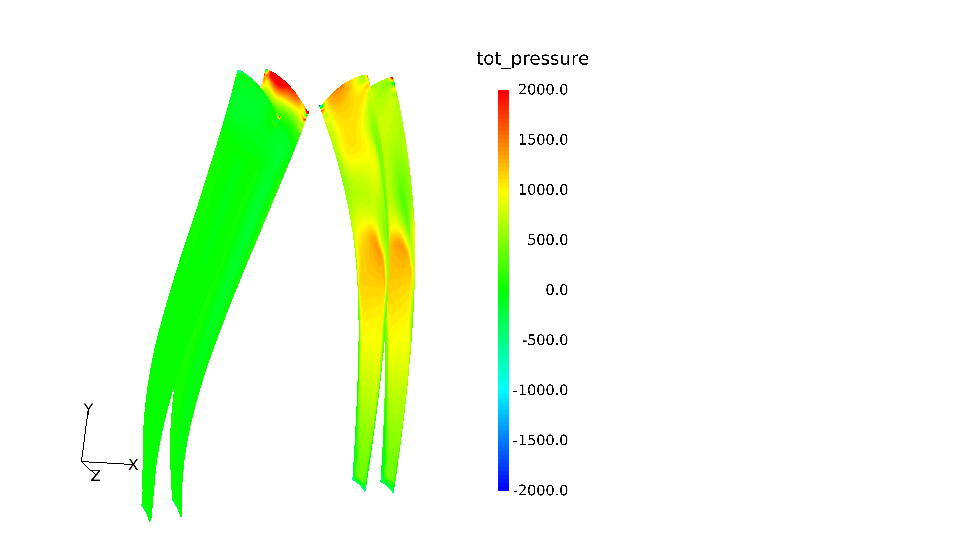
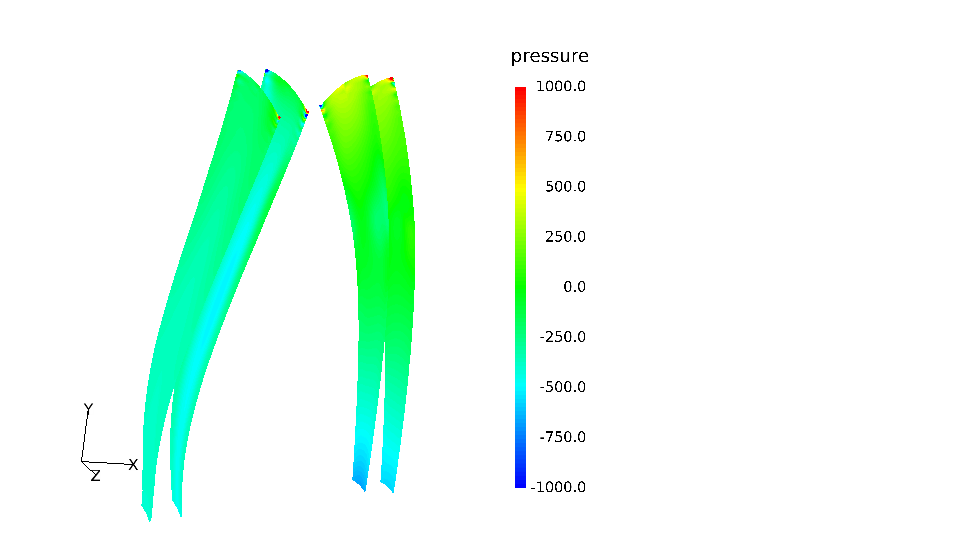
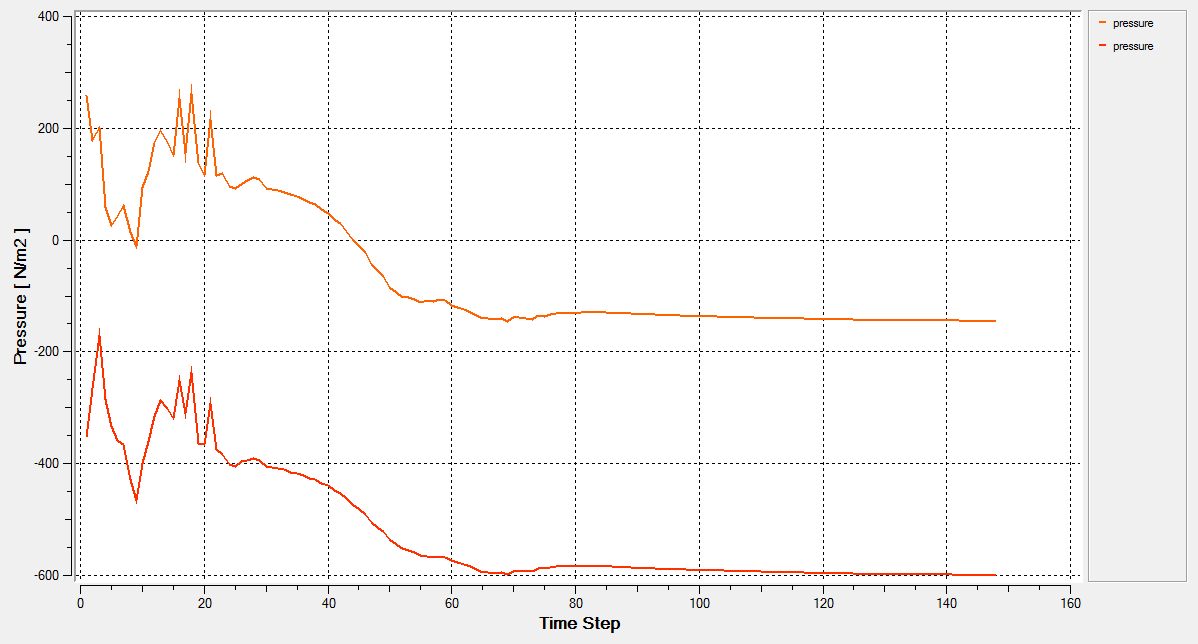
 

Figure ‑: Total pressure (left) and static pressure (right) contours upstream and downstream of the blade passage (flow direction left to right)



Suction Side

Pressure Side

Figure ‑: Pressure difference between the pressure and suction side of the blade

Suction Side

Pressure Side

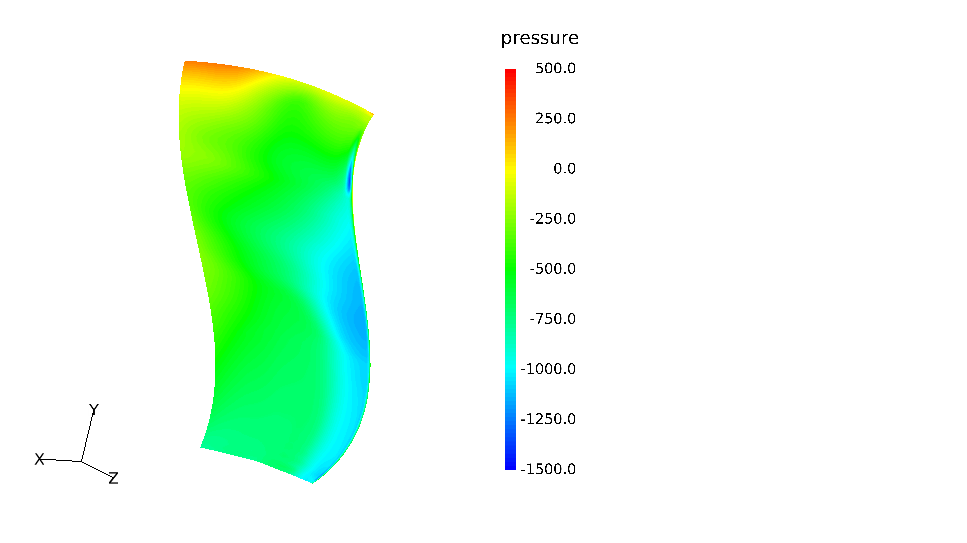
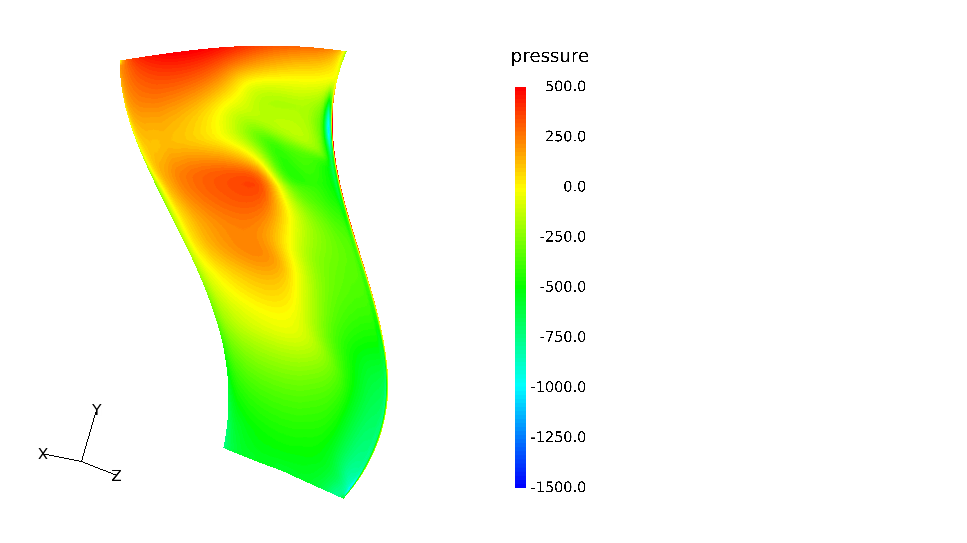


Figure ‑: Pressure contours on pressure and suction side of the blade

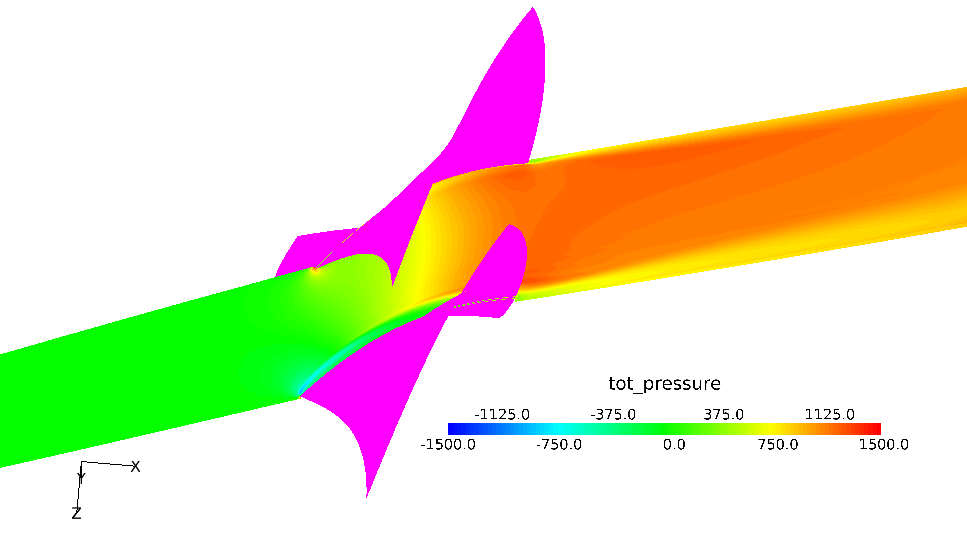


Figure ‑: Streamwise total pressure contour at 50% span

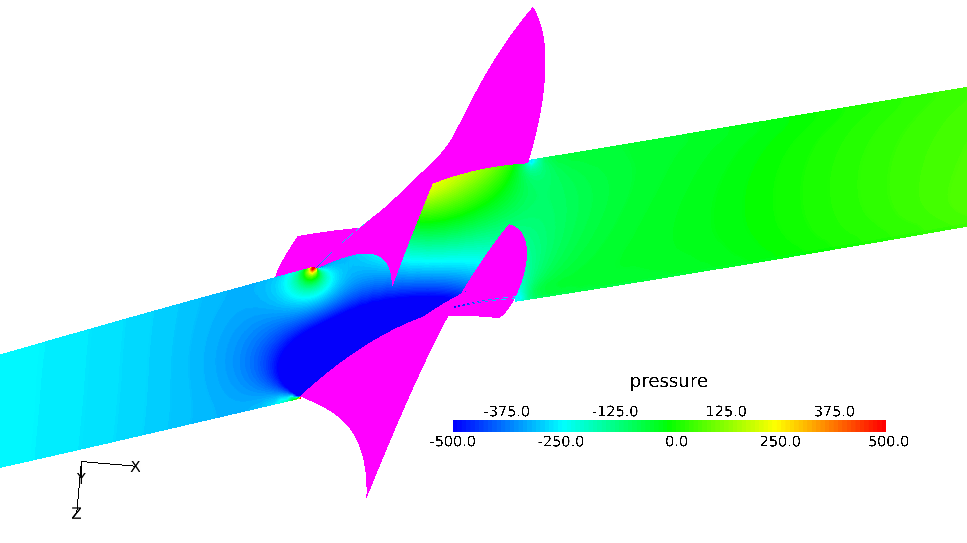


Figure ‑: Streamwise static pressure contour at 50% span

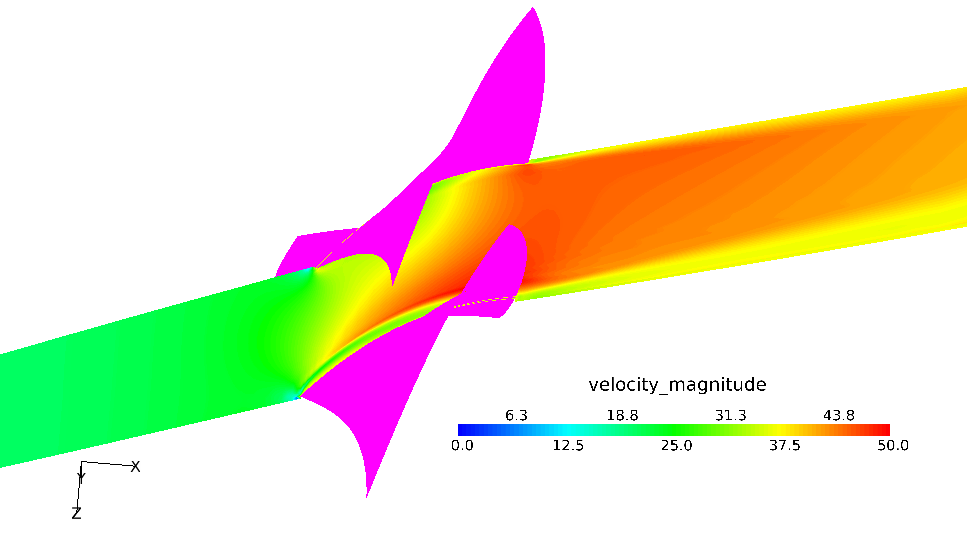


Figure ‑: Streamwise velocity magnitude contour at 50% span

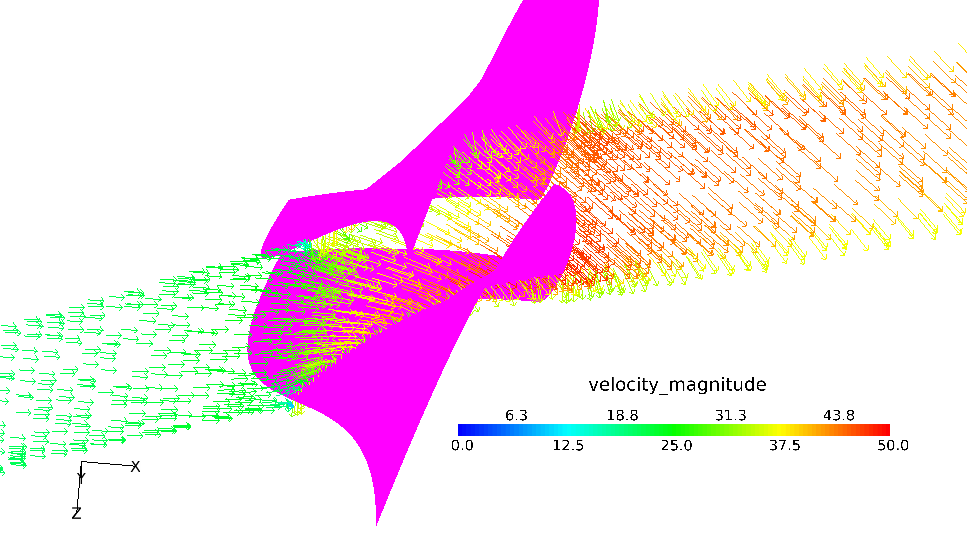


Figure ‑: Streamwise velocity vectors representation at 50% span

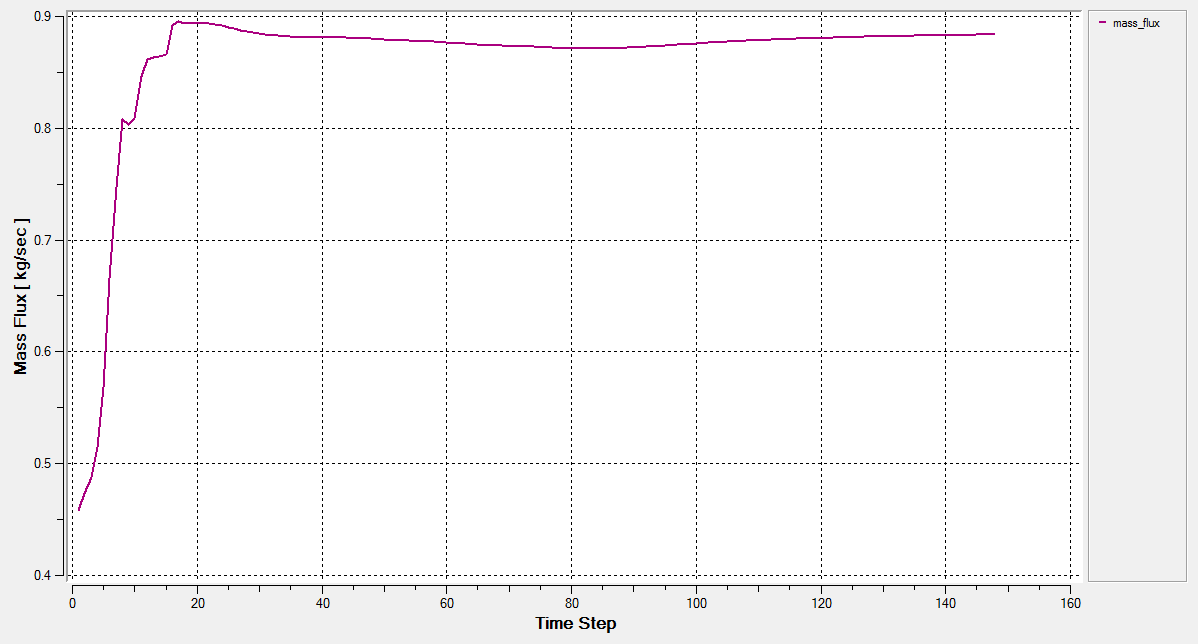


Figure ‑: Mass flow through the fan

# Optimization setup

Once the CFD setup has been finalized and the satisfactory solution for the baseline fan shape model have been obtained, the next step is to setup the optimization study. The different optimization approaches which are used in a CFD optimization program have been discussed earlier. This study uses one of these approaches, namely the *partial parametric* optimization approach.

However, all the parameters being used in this study relate to the change in only the shape of the blade. As mentioned previously, the setup remains identical for all the cases that will be run for optimization. However, each case will have a slightly different shape and thus when comparing the results, the difference in the results between the different cases can be attributed only to the change in shape.

Also, as mentioned in a previous section, the mesh for the different shapes will not be generated by remeshing, but by morphing the mesh created for the baseline, original shape. This will again, by keeping the mesh topology unchanged, restrict the variation in results between different cases only to the change in shape of the geometry and not any variations due to the change in the mesh.

Before a morphing algorithm is applied to a mesh, the elements are segregated in what may be called as morphing domains. Each of these domains can be morphed independently of the others, but keeping the connectivity of the nodes intact. Weights and biases can also be specified if needed, to control the deformation of elements in the neighboring domains due to morphing in a particular domain.

In this model, the domains for morphing are defined such that only the leading edge of the blade is modified. Since the outlet angles of the fan blade row are constrained by the downstream components in the engine, such as the compressor rows, the trailing edge of the blade is set to be unaffected by the morphing. The elements between the trailing edge and the leading edge of the blade are defined as one domain, the elements in front of the blade row form the upstream domain, and the elements behind the blade row form the downstream domain. Figure 6‑1 illustrates how the mesh morphing has been defined for this study.

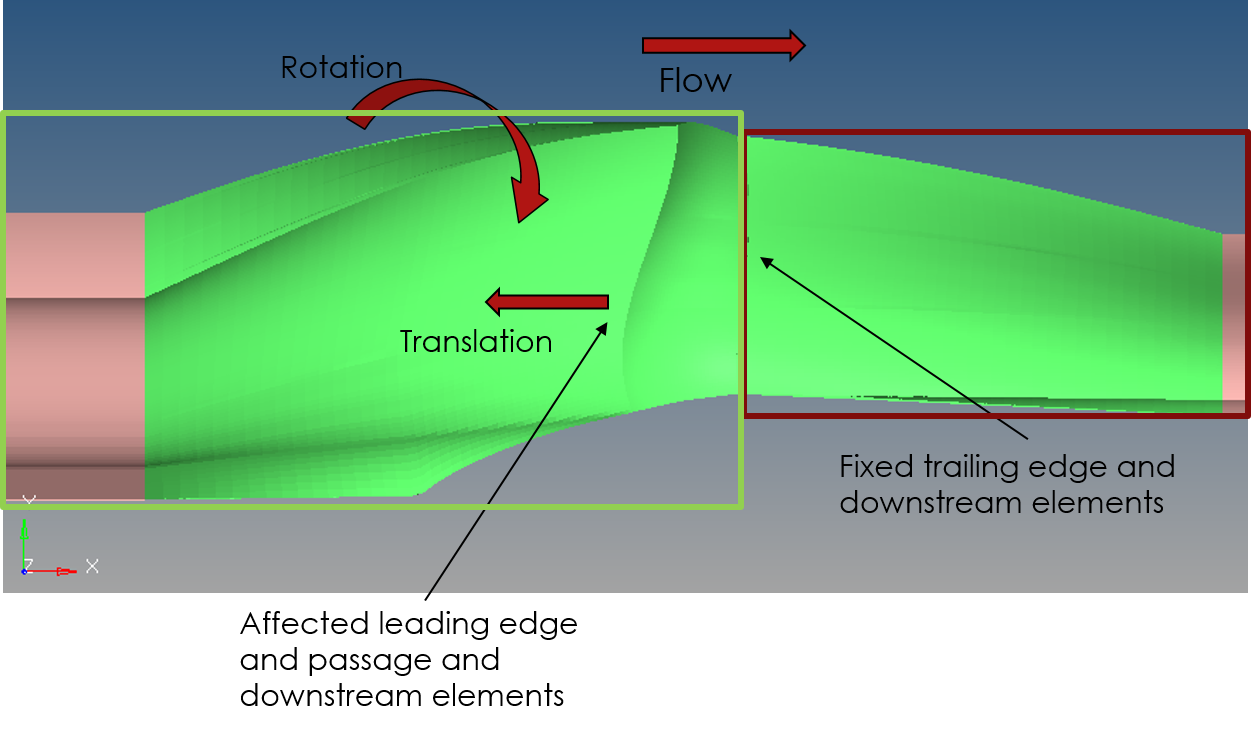


Figure ‑: Mesh morphing setup and shape variables

Once the morphing domains have been decided, the actual morphing process can be proceeded with. To define the new shapes for this study, the upstream elements will be shifted in space using two shape variables – rotation and translation of the nodes. The downstream elements are kept fixed because of the constraints explained above. The in-between passage elements will however be affected due to the shifting of the elements in the upstream domain, thus yielding a new shape for the fan blade. The translation of nodes is along the direction of the blade chord i.e. x direction. The rotation of the nodes is along the direction of rotation of the fan i.e. about the x axis. Figure 6‑2 illustrates the morphed shape of the domain when a certain rotation and translation is applied to the original shape.

In the optimizer algorithm being used for this study, the external bounds for the parameters can be defined in the optimization setup. Then as the parameter information is loaded in the optimizer, the optimization algorithm traverses between the external bounds specified, stopping at certain intervals in between according to the number of levels that can be specified by the user. For example, if the normalized external bounds for the shape variable are -1 and +1, and the number of levels specified is 3, the intervals which will be used by the optimization algorithm for the variables are (-1, 0, +1).

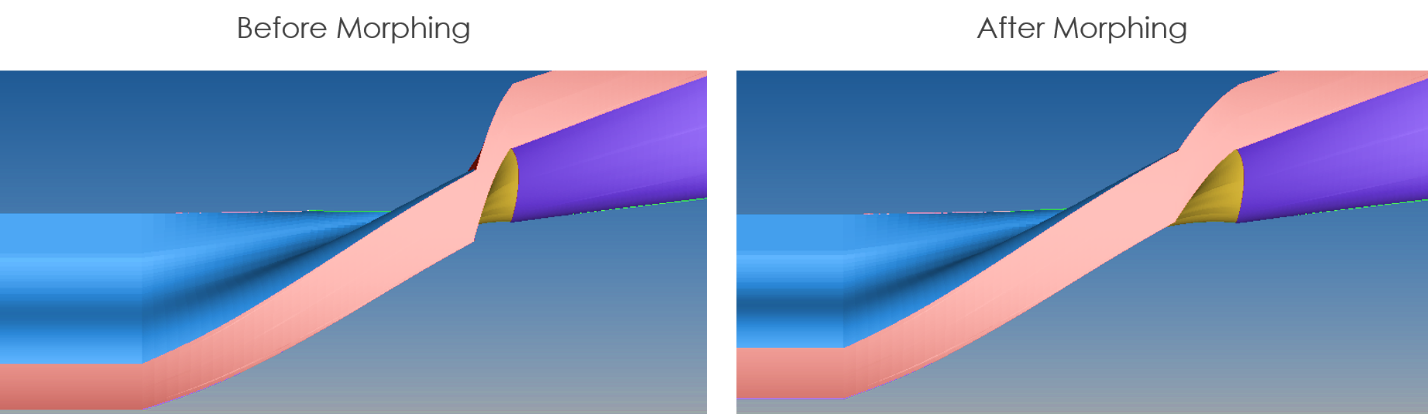


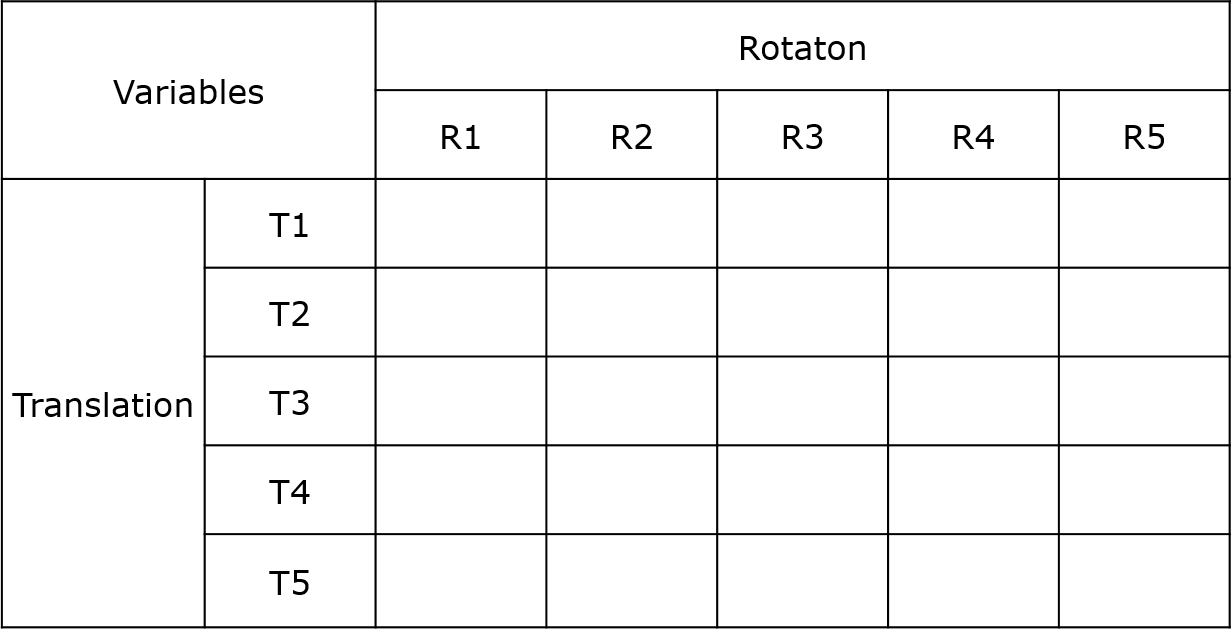
Figure ‑: Illustration of morphing applied to the analysis domain

In this study, the shape variables are the optimization parameters, and the external bounds for these are defined in the morphing tool. These are, as mentioned earlier, the rotation and translation of the nodes in the upstream morphing domain. When one or more variables are present, as in this case, each variable traverses through the specified levels, and all the possible combinations of the different levels of two variables make up the complete repertoire of cases that will be analyzed. The experiment design is similar to the full factorial design, enabling the user to study the effect of all possible input parameters on the response parameter, and also the interaction between the parameters. For example, if R is the response parameter and X and Y are input parameters, variation can be studied in terms of how R varies when X and Y are varied simultaneously.

The optimization, or input parameters X and Y in this study are the translation and rotation of the upstream nodes. The number of levels for both these variables are specified as 5. Full factorial design will be used for the *design of experiment* (DOE), thus yielding a total of 25 cases that can be evaluated according to the design variable matrix, as shown in Table 6‑1.

The levels specified in the matrix are normalized bounds for the variables. The actual bounds specified in the morphing tool while creating the shapes are 50 mm for the translation, and 10 degrees for the rotation of the moved nodes. Thus a level of -1 for translation refers to a translation of nodes by -50 mm, or 50 mm in negative x direction. Similarly a level of -1 for rotation refers to the rotation of nodes by -10 degrees about x axis.

Table ‑: Optimization experiment design matrix for the current study with two variables



Once the design variables are specified and the experiment design with the number of levels is input to the optimization algorithm, the algorithm works automatically by reading the original baseline mesh, applying the morphing parameters to generate the new shape for the case being evaluated and sending the modified shape mesh along with the setup to the CFD solver. The optimizer software interacts with the morphing tool and the CFD solver via scripts. As soon as the CFD solver completes running a case the optimizer repeats the process for the next case in the queue until the experiment has finished. The results for all the cases are saved, and the desired data can be obtained from the post processing of these results, just like the baseline case.

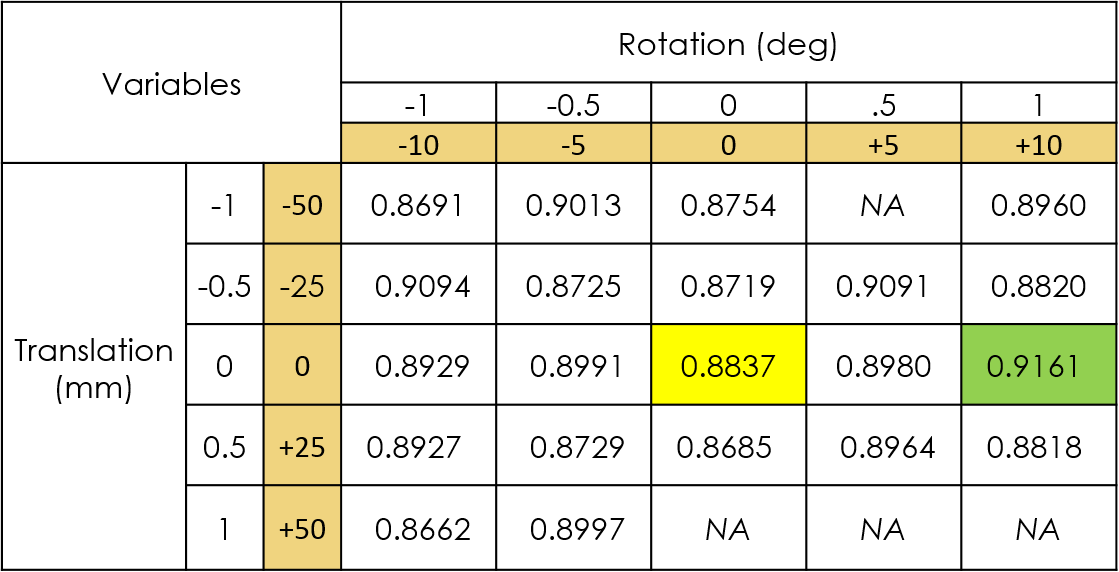
# Results and Discussion – Optimization Study

In this section we will discuss the results obtained from the cases which were run as part of the optimization study, along with the comparison of these results with the results of the baseline shape.

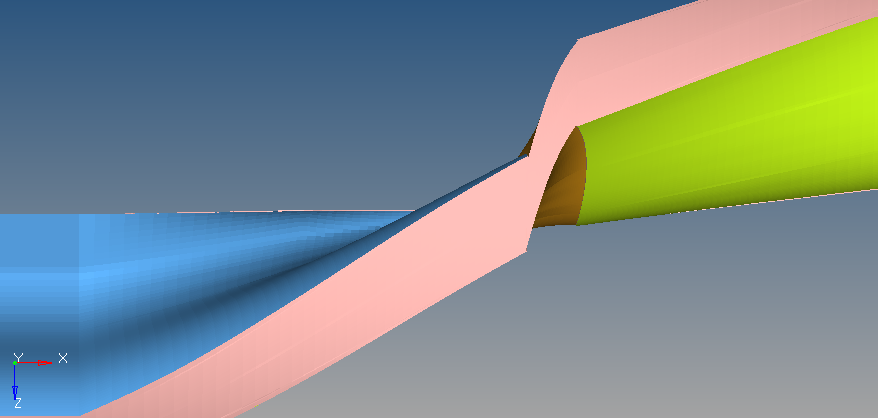
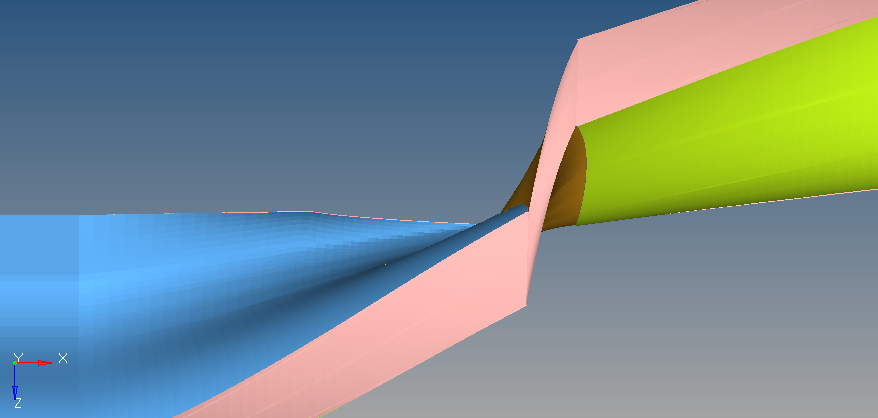
## The DOE Case Matrix

The DOE matrix which was presented in the last section can now be filled with results from the cases run for optimization. Since the response parameter for this study is mass flow, the Table 7‑1 shows the experiment matrix as discussed in the previous section, with the mass flow values for all the different cases for which the results were obtained. There were a few cases for which the results could not be obtained because the solution failed due to problems with the morphed mesh. Those cases have been marked with *NA* in the matrix.

Table ‑: Optimization experiment design matrix populated with the study results



From the above calculated mass flow values for different runs, it is seen that the maximum mass flow is observed in the case with the normalized variable levels as (0, +1) for translation and rotation of the upstream nodes respectively. As actual values of the variables, this translates to no translation of the nodes but their rotation by +10 degrees about x axis. The corresponding shape of the blade row with this shape applied is shown in Figure 7‑1.

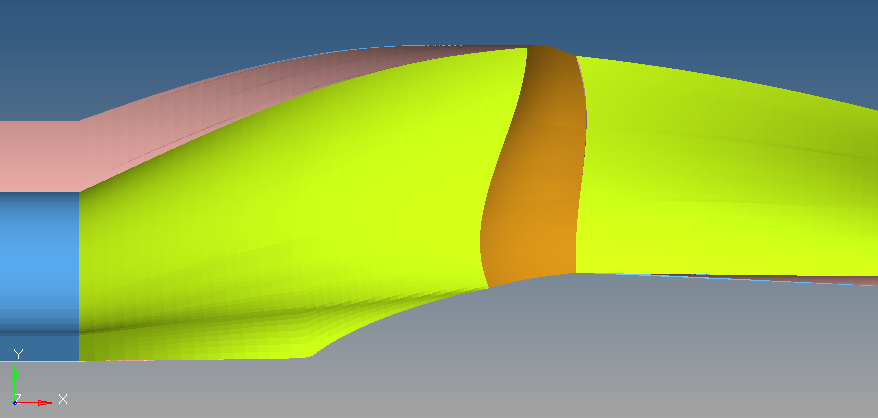
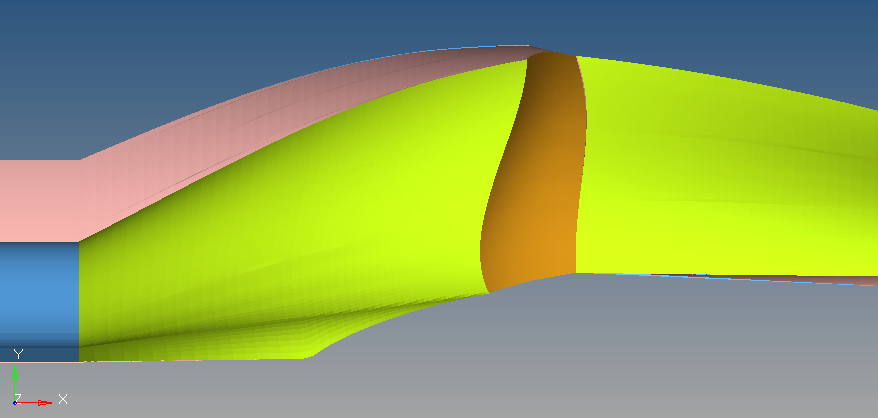
 

Figure ‑: The shape corresponding to the highest mass flow rate through the fan

## Flow Parameters

Table 7‑2 shows the variation between the important flow parameters between the baseline case discussed earlier and the selected case from the optimization runs.

Table ‑: Variation of flow parameters between the baseline case and the selected case

|  |  |  |
| --- | --- | --- |
| **Flow Parameter** | **Optimized Case** | **Baseline Case** |
| Total Pressure Rise through the fan | 731.5 Pa | 747.5 Pa |
| Differential pressure between blade surfaces | 453.0 Pa | 454.4 Pa |
| Mass Flux | 0.9161 kg/s | 0.8837 kg/s |

A look at the table shows that although the mass flow in the chosen case shows maximum improvement over the baseline case, the total pressure rise shows a small decline. A possible reason behind this can be that the new blade shape has a longer blade-wise chord length. The pressure difference between the two sides of the blades is almost similar, thus indicating no significant change in the loading on the blade due to the change in shape.

# Conclusive Notes and Future Work

The objective of the study was to determine the optimum aerodynamic shape for the shape of a given axial fan. This was achieved using a partial parametric optimization approach with the optimization parameters being the shape variables for the shape of the fan blade. The optimization was done to maximize the mass flow through the fan, because a higher mass glow through the fan means a higher total thrust for the engine.

In the previous sections we have discussed the methodology and procedure followed for the study. The baseline mesh is first generated from the given fan geometry and a CFD analysis done on it to record the benchmark results which are to be improved upon by changing the shape of the blade passage. For the purpose of changing the shape, the leading edge of the blade is translated and rotated by a certain magnitude in space by means of a morphing algorithm. Each of these shapes is then solved for results through a CFD analysis similar to the baseline shape and the results compared thereafter. The primary parameter compared is the response parameter for optimization, or mass flow through the fan.

The best case i.e. the case with maximum mass flow rate among the tested cases, was observed to be case no translation of the nodes but their rotation by +10 degrees about x axis. The improvement in mass flow was observed to be ~4% over the baseline mass flow. However, in the chosen case, the total pressure rise through the fan decreased by ~2%. It was also attempted to determine a correlation between the shape variables and the response variable, but a good correlation was not found to exist. This can be, in part, because of uncertainty in CFD results. Since the test object was not available, and neither were any experimental results, or another set of results from an independent study, available (even for the baseline case), it was difficult to determine the accuracy of the CFD results. However, since the study was an optimization study and the emphasis was on comparing the results among themselves, it was taken care to follow the best CFD practices and guidelines regarding the processes followed during meshing, preprocessing and setting up the CFD solution.

There still might be, however, scope for further increasing the reliability of this study. One of the ways this can be done is by further refinement of the mesh, especially within the blade passage zone. While this may lead to an increase in computation time and morphing time, a better resolved mesh almost always leads to a more accurate solution in CFD. Another option that can be looked after in a future study is to increase the number of levels that will be tested for each variable for the optimization study. A more number of levels will yield more solution points in the matrix for comparison of the shapes. Though again, this leads to an increase in number of total computation time. However, using morphing, the time is significantly less than the equal number of test cases carried out independently of each other. Secondly, as the total number of cases increase, more experiment designs can be looked into, some of which do not require all the possible cases to be run but select a few cases from the large matrix to produce the results which can be extrapolated to fill the remaining empty cells.

# References

1. NASA page on Turbofan Engine description, [*http://www.grc.nasa.gov/WWW/k-12/airplane/aturbf.html*](http://www.grc.nasa.gov/WWW/k-12/airplane/aturbf.html)
2. NASA Turbofan Thrust page, [*http://www.grc.nasa.gov/WWW/k-12/airplane/turbfan.html*](http://www.grc.nasa.gov/WWW/k-12/airplane/turbfan.html)
3. W N Dawes; *Towards a fully parallel integrated geometry kernel, mesh generator, flow solver & post-processor*; AIAA-2006-0942, 44 th AIAA Aerospace Sciences Meeting & Exhibit, 9-12 January 2006, Reno, NV
4. WN Dawes, CF Favaretto, SA Harvey, S Fellows, GA Richardson; *Towards topology-free optimization – an application to turbine internal cooling geometries*; AIAA-2008-925, 46 th AIAA Aerospace Sciences Meeting & Exhibit, 7-10 January 2008, Reno, NV
5. CF Favaretto, WN Dawes; *Enhancing the Productivity and Quality of the CFD Process in Turbomachinery Design*; IGTC2007-ABS-13
6. WN Dawes, SA Harvey, S Fellows, N.Eccles, D Jaeggi, WP Kellar; *A practical demonstration of scalable, parallel mesh generation*; AIAA-2009-0981, 47 th AIAA Aerospace Sciences Meeting & Exhibit, 5-8 January 2009, Orlando FL
7. W.N.Dawes, W.P.Kellar, N.C.Eccles & S.A.Harvey; *Automated Meshing for Aero-Thermal Analysis of Complex Automotive Geometries*; SAE Technical Paper 2011-01-05232011-01-0523, SAE World Congress 2011
8. Stefan Harries; *Practical Shape Optimization using CFD*; Whitepaper, November 2014 (Version 1.0)
9. Whitepaper; *CFD Optimization with Altair HyperWorks*; May 2007
10. Dominique Thévenin, Gábor Janiga; *Optimization and Computational Fluid Dynamics*; Springer publication, ISBN: 978-3-540-72152-9
11. Giovanni Lombardi, Marco Maganzi; *Use of the CFD for the Aerodynamic Optimization of the Car Shape: Problems and Application*; EASC 2009, 4th European Automotive Simulation Conference, Munich, Germany, 6-7 July 2009
12. Rupak Biswas, Roger C. Strawn; *Tetrahedral and hexahedral mesh adaptation for CFD problems*; Applied Numerical Mathematics, Volume 26, Issues 1–2, January 1998, pp. 135-151
13. Matthew L. Staten, Steven J. Owen, Suzanne M. Shontz, Andrew G., Salinger, Todd S. Coffey; *A Comparison of Mesh Morphing Methods for 3D Shape Optimization*; Proceedings of the 20th International Meshing Roundtable, 2012, pp. 293-311
14. David R. White, Lai Mingwu, Steven E. Benzley, Gregory D. Sjaardema; *Automated Hexahedral Mesh Generation by Virtual Decomposition*; Proceedings of the 4th International Meshing Roundtable, 1995, pp. 165-176
15. Y. Lu, R. Gadh, T.J. Tautges; *Feature based hex meshing methodology: feature recognition and volume decomposition*; Computer-Aided Design 33 (2001), pp. 221-232
16. Marc Alexa; *Recent Advances in Mesh Morphing*; Computer Graphics Forum Volume 21, Issue 2, June 2002, pp. 173-198
17. Rolf Dornberger, Dirk Büche, Peter Stoll; *Multidisciplinary Optimization In Turbomachinery Design*; European Congress on Computational Methods in Applied Sciences and Engineering ECCOMAS 2000, Barcelona, 11-14 September 2000
18. Altair Hyperworks *User Manual* 13.0 release

# Appendix

## Solver input file

# +----------------------------------------------------------------------+

# | This is an AcuSolve input file. |

# | Problem: OnePassage |

# | Generated by: acuConsole |

# | Version: 12.0.310 |

# | Release date: July 10, 2013 |

# | Date: Wed Jan 21 17:43:44 IST 2015 |

# +----------------------------------------------------------------------+

# +----------------------------------------------------------------------+

# | Problem Description |

# +----------------------------------------------------------------------+

ANALYSIS {

title = "GENX5 Scaled Mesh"

sub\_title = "Steady State"

input\_version = "12.0.310"

type = steady

}

EQUATION {

flow = navier\_stokes

absolute\_pressure\_offset = 101325.0 # N/m2

temperature = none

species\_transport = none

turbulence = spalart\_allmaras

mesh = eulerian

external\_code = off

particle\_trace = off

running\_average = off

}

# +----------------------------------------------------------------------+

# | Auto Solution Strategy |

# +----------------------------------------------------------------------+

AUTO\_SOLUTION\_STRATEGY {

max\_time\_steps = 300

initial\_time\_increment = 10000000000.0 # sec

convergence\_tolerance = 0.0001

num\_krylov\_vectors = 10

relaxation\_factor = 0.5

flow = on

turbulence = on

}

STAGGER( "flow" ) {

max\_linear\_solver\_iterations = 10000

}

STAGGER( "turbulence" ) {

max\_linear\_solver\_iterations = 10000

}

# +----------------------------------------------------------------------+

# | Material Model: Air |

# +----------------------------------------------------------------------+

MATERIAL\_MODEL( "Air" ) {

Type = "fluid"

density\_model = "Air"

viscosity\_model = "Air"

porosity\_model = "Air"

}

DENSITY\_MODEL( "Air" ) {

type = constant

density = 1.225 # kg/m3

isothermal\_compressibility = 0.0 # m2/N

}

VISCOSITY\_MODEL( "Air" ) {

type = constant

viscosity = 1.781e-005

}

POROSITY\_MODEL( "Air" ) {

type = none

}

# +----------------------------------------------------------------------+

# | Reference Frame |

# +----------------------------------------------------------------------+

REFERENCE\_FRAME( "Rotation" ) {

centrifugal = on

coriolis = on

angular\_acceleration = on

rotation\_center = { 0.0, 0.0, 0.0; }

angular\_velocity = { 113.097335529, 0.0, 0.0; }

# { 1080.0, 0.0, 0.0, } RPM

}

# +----------------------------------------------------------------------+

# | Nodal Output |

# +----------------------------------------------------------------------+

NODAL\_OUTPUT {

output\_frequency = 50

output\_time\_interval = 0.0 # sec

output\_initial\_condition = off

continuous\_output = off

num\_saved\_states = 2

}

# +----------------------------------------------------------------------+

# | Nodal Initial Condition |

# +----------------------------------------------------------------------+

NODAL\_INITIAL\_CONDITION( pressure ) {

default\_value = 0.0 # N/m2

satisfy\_boundary\_condition = off

}

NODAL\_INITIAL\_CONDITION( velocity ) {

default\_values = { 10.0, 0.0, 0.0; }

satisfy\_boundary\_condition = off

}

NODAL\_INITIAL\_CONDITION( eddy\_viscosity ) {

default\_value = 1e-005 # m2/sec

satisfy\_boundary\_condition = off

}

# +----------------------------------------------------------------------+

# | Coordinates |

# +----------------------------------------------------------------------+

COORDINATE {

coordinates = Read( "MESH.DIR/OnePassage.crd" )

}

# +----------------------------------------------------------------------+

# | Element Sets |

# +----------------------------------------------------------------------+

ELEMENT\_SET( "Inlet\_Volume" ) {

elements = Read( "MESH.DIR/OnePassage.Inlet\_Volume.hex8.cnn" )

shape = eight\_node\_brick

medium = fluid

quadrature = full

material\_model = "Air"

body\_force = "none"

mesh\_motion = "none"

reference\_frame = "none"

residual\_control = on

oscillation\_control = on

}

# +----------------------------------------------------------------------+

# | Element Sets |

# +----------------------------------------------------------------------+

ELEMENT\_SET( "Outlet\_Volume" ) {

elements = Read( "MESH.DIR/OnePassage.Outlet\_Volume.hex8.cnn" )

shape = eight\_node\_brick

medium = fluid

quadrature = full

material\_model = "Air"

body\_force = "none"

mesh\_motion = "none"

reference\_frame = "none"

residual\_control = on

oscillation\_control = on

}

# +----------------------------------------------------------------------+

# | Element Sets |

# +----------------------------------------------------------------------+

ELEMENT\_SET( "Passage" ) {

elements = Read( "MESH.DIR/OnePassage.Passage.hex8.cnn" )

shape = eight\_node\_brick

medium = fluid

quadrature = full

material\_model = "Air"

body\_force = "none"

mesh\_motion = "none"

reference\_frame = "Rotation"

residual\_control = on

oscillation\_control = on

}

# +----------------------------------------------------------------------+

# | Simple Boundary Condition |

# +----------------------------------------------------------------------+

SIMPLE\_BOUNDARY\_CONDITION( "Axis" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Axis.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

type = wall

active\_type = all

precedence = 1

reference\_frame = "Rotation"

wall\_velocity\_type = match\_mesh\_velocity

temperature\_type = flux

heat\_flux = 0.0

convective\_heat\_coefficient = 0.0

convective\_heat\_reference\_temperature = 0.0

turbulence\_wall\_type = wall\_function

roughness\_height = 0.0 # m

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Axis" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Axis.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Blade\_Downstream" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Blade\_Downstream.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Simple Boundary Condition |

# +----------------------------------------------------------------------+

SIMPLE\_BOUNDARY\_CONDITION( "Blade\_Edges" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Blade\_Edges.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

type = wall

active\_type = all

precedence = 3

reference\_frame = "Rotation"

wall\_velocity\_type = match\_mesh\_velocity

temperature\_type = flux

heat\_flux = 0.0

convective\_heat\_coefficient = 0.0

convective\_heat\_reference\_temperature = 0.0

turbulence\_wall\_type = wall\_function

roughness\_height = 0.0 # m

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Blade\_Edges" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Blade\_Edges.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Blade\_Upstream" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Blade\_Upstream.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Simple Boundary Condition |

# +----------------------------------------------------------------------+

SIMPLE\_BOUNDARY\_CONDITION( "Casing" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Casing.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

type = slip

active\_type = all

precedence = 1

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Casing" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Casing.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Simple Boundary Condition |

# +----------------------------------------------------------------------+

SIMPLE\_BOUNDARY\_CONDITION( "Inlet" ) {

surfaces = Read( "MESH.DIR/OnePassage.Inlet\_Volume.hex8.Inlet.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Inlet\_Volume"

type = inflow

inflow\_type = stagnation\_pressure

active\_type = all

precedence = 2

reference\_frame = "none"

hydrostatic\_pressure = off

stagnation\_pressure = 0.0 # N/m2

eddy\_viscosity = 1e-005 # m2/sec

non\_reflecting\_factor = 0.0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Inlet" ) {

surfaces = Read( "MESH.DIR/OnePassage.Inlet\_Volume.hex8.Inlet.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Inlet\_Volume"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Nodal Boundary Condition |

# +----------------------------------------------------------------------+

NODAL\_BOUNDARY\_CONDITION( "Inlet y\_velocity" ) {

nodes = Read( "MESH.DIR/OnePassage.Inlet\_0\_quad4\_0\_Inlet\_Volume\_0\_hex8.nbc" )

variable = "y\_velocity"

type = zero

precedence = 3

reference\_frame = "none"

active\_type = all

}

NODAL\_BOUNDARY\_CONDITION( "Inlet z\_velocity" ) {

nodes = Read( "MESH.DIR/OnePassage.Inlet\_0\_quad4\_0\_Inlet\_Volume\_0\_hex8.nbc" )

variable = "z\_velocity"

type = zero

precedence = 3

reference\_frame = "none"

active\_type = all

}

# +----------------------------------------------------------------------+

# | Simple Boundary Condition |

# +----------------------------------------------------------------------+

SIMPLE\_BOUNDARY\_CONDITION( "Inlet\_Axis" ) {

surfaces = Read( "MESH.DIR/OnePassage.Inlet\_Volume.hex8.Inlet\_Axis.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Inlet\_Volume"

type = slip

active\_type = all

precedence = 1

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Inlet\_Axis" ) {

surfaces = Read( "MESH.DIR/OnePassage.Inlet\_Volume.hex8.Inlet\_Axis.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Inlet\_Volume"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Inlet\_Interface" ) {

surfaces = Read( "MESH.DIR/OnePassage.Inlet\_Volume.hex8.Inlet\_Interface.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Inlet\_Volume"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Inlet\_Periodic1" ) {

surfaces = Read( "MESH.DIR/OnePassage.Inlet\_Volume.hex8.Inlet\_Periodic1.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Inlet\_Volume"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Inlet\_Periodic2" ) {

surfaces = Read( "MESH.DIR/OnePassage.Inlet\_Volume.hex8.Inlet\_Periodic2.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Inlet\_Volume"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Simple Boundary Condition |

# +----------------------------------------------------------------------+

SIMPLE\_BOUNDARY\_CONDITION( "Inlet\_Slipwall" ) {

surfaces = Read( "MESH.DIR/OnePassage.Inlet\_Volume.hex8.Inlet\_Slipwall.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Inlet\_Volume"

type = slip

active\_type = all

precedence = 1

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Inlet\_Slipwall" ) {

surfaces = Read( "MESH.DIR/OnePassage.Inlet\_Volume.hex8.Inlet\_Slipwall.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Inlet\_Volume"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Interface\_In" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Interface\_In.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Interface\_Out" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Interface\_Out.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Simple Boundary Condition |

# +----------------------------------------------------------------------+

SIMPLE\_BOUNDARY\_CONDITION( "Outlet" ) {

surfaces = Read( "MESH.DIR/OnePassage.Outlet\_Volume.hex8.Outlet.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Outlet\_Volume"

type = outflow

active\_type = all

precedence = 1

pressure = 0.0 # N/m2

pressure\_loss\_factor = 0.0

hydrostatic\_pressure = off

back\_flow\_conditions = on

eddy\_viscosity\_back\_flow\_type = area\_average

non\_reflecting\_factor = 0.0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Outlet" ) {

surfaces = Read( "MESH.DIR/OnePassage.Outlet\_Volume.hex8.Outlet.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Outlet\_Volume"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Simple Boundary Condition |

# +----------------------------------------------------------------------+

SIMPLE\_BOUNDARY\_CONDITION( "Outlet\_Axis" ) {

surfaces = Read( "MESH.DIR/OnePassage.Outlet\_Volume.hex8.Outlet\_Axis.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Outlet\_Volume"

type = slip

active\_type = all

precedence = 1

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Outlet\_Axis" ) {

surfaces = Read( "MESH.DIR/OnePassage.Outlet\_Volume.hex8.Outlet\_Axis.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Outlet\_Volume"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Outlet\_Interface" ) {

surfaces = Read( "MESH.DIR/OnePassage.Outlet\_Volume.hex8.Outlet\_Interface.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Outlet\_Volume"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Outlet\_Periodic1" ) {

surfaces = Read( "MESH.DIR/OnePassage.Outlet\_Volume.hex8.Outlet\_Periodic1.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Outlet\_Volume"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Outlet\_Periodic2" ) {

surfaces = Read( "MESH.DIR/OnePassage.Outlet\_Volume.hex8.Outlet\_Periodic2.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Outlet\_Volume"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Simple Boundary Condition |

# +----------------------------------------------------------------------+

SIMPLE\_BOUNDARY\_CONDITION( "Outlet\_Slipwall" ) {

surfaces = Read( "MESH.DIR/OnePassage.Outlet\_Volume.hex8.Outlet\_Slipwall.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Outlet\_Volume"

type = slip

active\_type = all

precedence = 1

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Outlet\_Slipwall" ) {

surfaces = Read( "MESH.DIR/OnePassage.Outlet\_Volume.hex8.Outlet\_Slipwall.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Outlet\_Volume"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Simple Boundary Condition |

# +----------------------------------------------------------------------+

SIMPLE\_BOUNDARY\_CONDITION( "Passage\_Axis" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Passage\_Axis.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

type = slip

active\_type = all

precedence = 1

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Passage\_Axis" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Passage\_Axis.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Periodic1" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Periodic1.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Periodic2" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Periodic2.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Simple Boundary Condition |

# +----------------------------------------------------------------------+

SIMPLE\_BOUNDARY\_CONDITION( "Pressure\_Side" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Pressure\_Side.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

type = wall

active\_type = all

precedence = 3

reference\_frame = "Rotation"

wall\_velocity\_type = match\_mesh\_velocity

temperature\_type = flux

heat\_flux = 0.0

convective\_heat\_coefficient = 0.0

convective\_heat\_reference\_temperature = 0.0

turbulence\_wall\_type = wall\_function

roughness\_height = 0.0 # m

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Pressure\_Side" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Pressure\_Side.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Span15" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Span15.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Span33" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Span33.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Span50" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Span50.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Span66" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Span66.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Span85" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Span85.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Stream\_surf quad4 Inlet\_Volume brick8" ) {

surfaces = Read( "MESH.DIR/OnePassage.Inlet\_Volume.hex8.Stream\_surf.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Inlet\_Volume"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Stream\_surf quad4 Outlet\_Volume brick8" ) {

surfaces = Read( "MESH.DIR/OnePassage.Outlet\_Volume.hex8.Stream\_surf.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Outlet\_Volume"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Stream\_surf quad4 Passage brick8" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Stream\_surf.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Simple Boundary Condition |

# +----------------------------------------------------------------------+

SIMPLE\_BOUNDARY\_CONDITION( "Suction\_Side" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Suction\_Side.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

type = wall

active\_type = all

precedence = 3

reference\_frame = "Rotation"

wall\_velocity\_type = match\_mesh\_velocity

temperature\_type = flux

heat\_flux = 0.0

convective\_heat\_coefficient = 0.0

convective\_heat\_reference\_temperature = 0.0

turbulence\_wall\_type = wall\_function

roughness\_height = 0.0 # m

}

# +----------------------------------------------------------------------+

# | Surface Output |

# +----------------------------------------------------------------------+

SURFACE\_OUTPUT( "Suction\_Side" ) {

surfaces = Read( "MESH.DIR/OnePassage.Passage.hex8.Suction\_Side.quad4.ebc" )

shape = four\_node\_quad

element\_set = "Passage"

integrated\_output\_frequency = 1

integrated\_output\_time\_interval = 0.0

statistics\_output\_frequency = 1

statistics\_output\_time\_interval = 0.0

nodal\_output\_frequency = 0

nodal\_output\_time\_interval = 0.0

num\_saved\_states = 0

}

# +----------------------------------------------------------------------+

# | Periodic Boundary Condition |

# +----------------------------------------------------------------------+

PERIODIC\_BOUNDARY\_CONDITION( "Inlet all" ) {

nodal\_pairs = Read( "MESH.DIR/OnePassage.Inlet.pbc" )

variable = all

type = axisymmetric

precedence = 1

rotation\_axis = { 0.0, 0.0, 0.0;

1.0, 0.0, 0.0; }

active\_type = all

}

# +----------------------------------------------------------------------+

# | Periodic Boundary Condition |

# +----------------------------------------------------------------------+

PERIODIC\_BOUNDARY\_CONDITION( "Outlet all" ) {

nodal\_pairs = Read( "MESH.DIR/OnePassage.Outlet.pbc" )

variable = all

type = axisymmetric

precedence = 1

rotation\_axis = { 0.0, 0.0, 0.0;

1.0, 0.0, 0.0; }

active\_type = all

}

# +----------------------------------------------------------------------+

# | Periodic Boundary Condition |

# +----------------------------------------------------------------------+

PERIODIC\_BOUNDARY\_CONDITION( "Passage all" ) {

nodal\_pairs = Read( "MESH.DIR/OnePassage.Passage.pbc" )

variable = all

type = axisymmetric

precedence = 1

rotation\_axis = { 0.0, 0.0, 0.0;

1.0, 0.0, 0.0; }

active\_type = all

}

# +----------------------------------------------------------------------+

# | RUN |

# +----------------------------------------------------------------------+

RUN {

}