INTERACTIVE CFD SIMULATIONS

Juan Fernando Duque Lombana

EAFIT UNIVERSITY

ENGINEERING SCHOOL

MECHANICAL ENGINEERING DEPARTMENT

MEDELLIN

2007

INTERACTIVE CFD SIMULATIONS

Graduation project for the degree of Mechanical Engineer

Advisor

Prof. Manuel Julio Garcia

EAFIT UNIVERSITY

ENGINEERING SCHOOL

MECHANICAL ENGINEERING DEPARTMENT

MEDELLIN

2007

Nota de Aceptación

_____

_____

_____

_____

President of the Jury

_____

Jury

_____

Jury

Medellín, February 2 Of 2007

Acknowledgments

“-A la patria nunca se llega- ... Pero cuando los caminos amigos se cruzan, todo el universo parece por un momento la patria anhelada”

-Demian, Hermann Hesse.

To my parents, Luis and Silvia, for all their invaluable love and support, this is for you and because of you.

There were many people involved on the development of this project and I would like to thank the following people: my advisor, Prof. Manuel J. Garcia, for all his support, uninterested help and specially for sharing his ideas about teaching, engineering and life. Eng. Miguel Henao, for his solid work and appreciations that helped me develop this study. Santiago Giraldo and Mario Gomez, for making any workspace a fun place. The dean of engineering Alberto Rodriguez and all the profs. at EAFIT University, for helping me to discover what engineering and teaching may be. To all the staff of the newly formed Applied Mechanics group, work has just begun!

I would like to thank my friends for all the good times that gave me energies to carry on during this years, you know who you are!

Abstract

This project is about the development of an implementable Interactive Computer Fluid Dynamics methodology. The range of this work begins with an overview of the current status of computational fluid dynamics simulation software and methodologies, continues with an introduction to what interactive and interactivity mean, develops an all original interactive CFD methodology to follow for the solution of fluid scenarios and finally, the description of the implementation of an interactive solver for CFD using the earlier developed methodology.

The project was developed entirely at the EAFIT University's Applied Mechanics Laboratory in Medellin and is part of a collaboration effort in companionship with the University of Aberta in Canada and Los Andes University in Bogota, Colombia.

Table of Contents

List of Tables

List of Figures

0 State of the Art

The following chapter presents some relevant information to this study, from which knowledge necessary for the development of the project can be obtained.

0.1 An introduction to Computer Fluid Dynamics (CFD).

0.1.1 What is CFD.

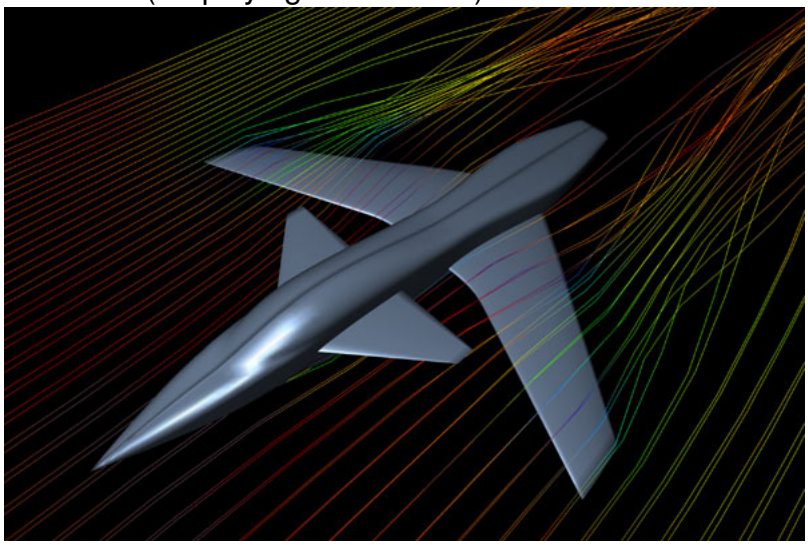
With the continuous development of computer science and hardware, problematics which solutions were once almost impossible to obtain or handle due to the large amount of data involved during the process, are now within the reach and capabilities of human beings.

Computer Fluid Dynamics (CFD) is the computer aided practice that deals with the calculation of different flow parameters in a fluid continuum (see Figures 1). The very support of CFD, is none other than the one proposed by Finite Element Analysis (FEA) (See (KIKUCHI, 1986) for an idea of this principles), and as such, the same difficulties that arise in both problem's definition and solution during a FEA analysis, arise during a CFD analysis.

The first question that has to be asked before starting with a CFD analysis is how the continuum is to be discretize and handled. Mainly, two possible answers might fit the question: to use a mesh dependent method (Eulerian view) or a meshless one (Lagrangian view) .

Mesh dependent methods, rely on the fact that a spatial domain Ω in R^3 might be discretized small cells to form a volume mesh that approximately represents the original Ω domain (See (KIKUCHI, 1986)). Once the discretized domain is set, a

Figure 1: CFD Results (Displaying streamlines).



FUEL TECH INC.@ (2004)

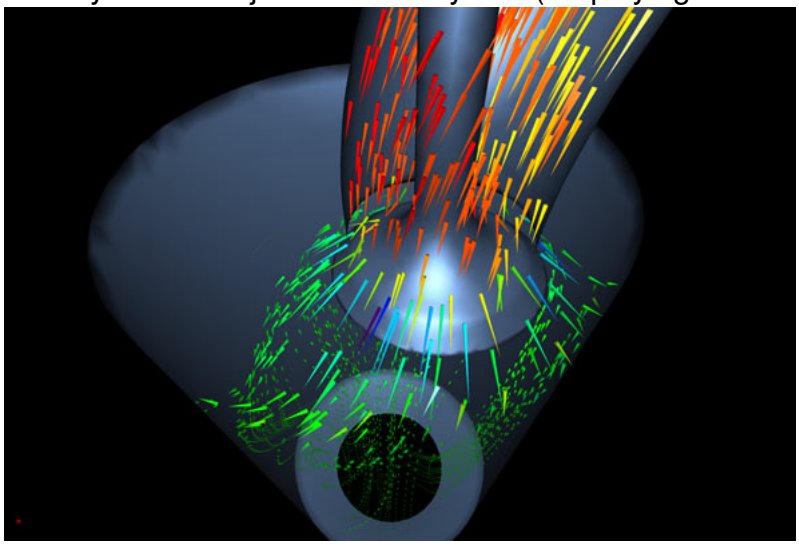
set of suitable algorithms is applied to solve the Navier-Stokes equations defined over the domain, obtaining a proper solution to both velocity and pressure variables.

The mesh format used to define Ω is flexible: can be either regular (hexaedrae) or irregular (tetrahedral), and if the problem is highly dynamic, the grid itself can be modified in time using adaptive mesh refinement methods.

Meshless methods represent a different alternative to the mesh dependent ones: Lattice Boltzmann methods simulating a mesoscopic system on a Cartesian space, Lagrangian viewpoint methods as smoothed particle hydrodynamics and spectral methods (where equations are projected over basis functions such as Chebyshev polynomials) are some methodologies to remark (WIKIPEDIA@, 2006b).

As a particular case, it is possible to directly solve set of Navier-Stokes equations for laminar flow cases, as well for turbulent flows in which all relevant length scales are contained within the grid. Other equations, such as the ones related to heat transfer

Figure 2: CFD Analysis for a injection valve system(Displaying streamlines).



FUEL TECH INC.@ (2004).

problems or chemical reactions can be solved simultaneously with the Navier-Stokes equations.

More complex CFD algorithms and programs allow the simulation of non-Newtonian fluids as blood and problematics including multiphase flows and interaction.

0.1.2 CFD state of the art and current limitants. CFD is undergoing a significant expansion in terms of both number of courses offered at universities and the number of researchers active in the field. There are a number of software packages available that solve fluid flow problems; the market is not as quite as large as the one for structural mechanics codes. The lag can be explained by the fact CFD problems are, in general, more difficult to solve. However, CFD codes are slowly being accepted as design tools by industrial users (FERZIGER y PERIC, 2002) .

There are several commercially software packages available, such as FLUENT (`http:`

//www.fluent.com), CFX (http://www-waterloo.ansys.com/) and STAR (http://www.cd-adapco.com/), and open source code packages under the GPL license such as OpenFOAM (www.opencfd.co.uk/openfoam/). These packages contained within their source codes routines and algorithms to solve incompressible, compressible and multiphase flows, direct numerical and large eddy simulation, combustion and heat transfers problems, but many particular cases and general problems are still impossible to solve with those softwares.

Open source packages as OpenFOAM, even though their learning curve is steep, represent an excellent solution for the dedicated engineer who wishes develop solutions for unsolved problems.

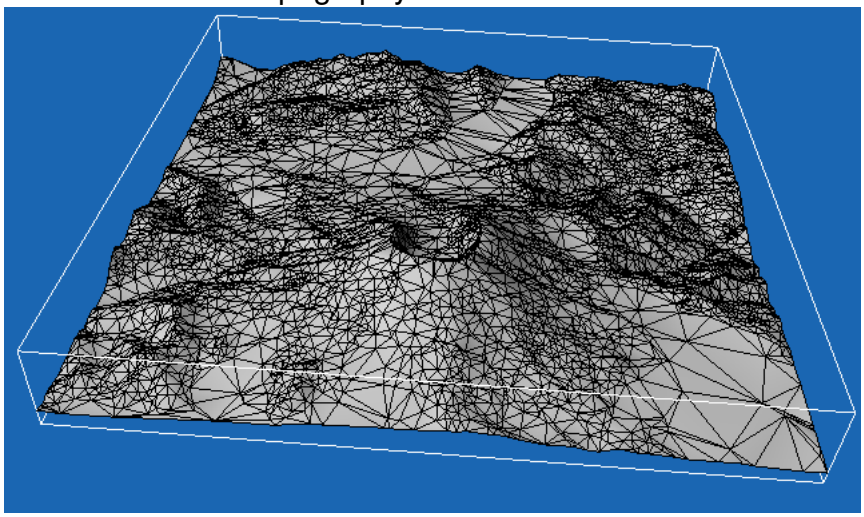
One of the problems faced by CFD developers and users is none other than the one of computational resources. CFD requires high amount of free memory to store up data related to both calculations and results, as well fast processing units are needed to perform the required operations, that could easily ascend to millions per iteration. Parallel processing computing has been the answer to this problematic and right now, big computer clusters are used by the market leaders to suffice their needs.

0.1.3 CFD from the inside, an overview. The CFD method may be subdivided into 3 large steps which cover the whole problem definition and solution process:

i. Pre-processing.

- Geometry definition. In this stage the physical bounds of the problem are defined. See Figure 3 for a graphical example.
- Geometry Discretization. The volume occupied by the fluid is discretized into finite cells (Figure 4); see DISCRETIZATION METHODS for more

Figure 3: Mount St. Helen's Topography.



GARCIA (2005)

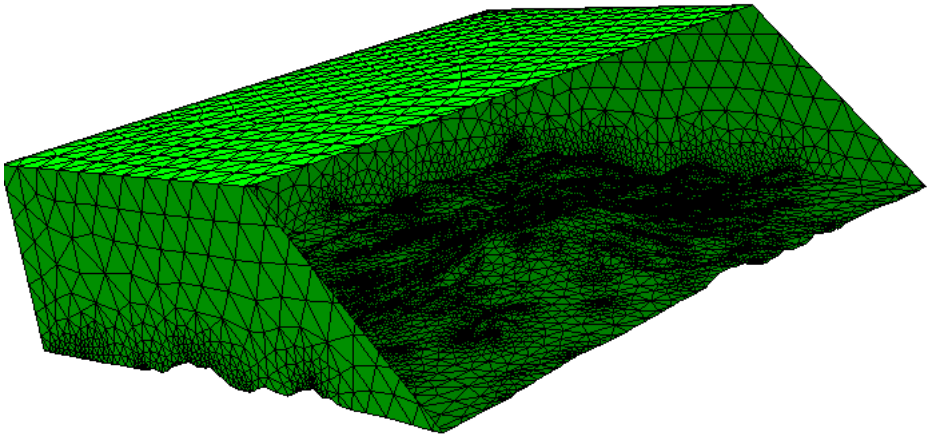
.

information about this topic.

- Physical model definition. Establishing the physical model that will describe the phenomena's behavior; e.g. the Navier-Stokes equations, heat transfer models, entropy equations, etc
- Boundary Conditions definition. After the physical-geometrical boundaries and models are defined, the fluid's behavior and properties at the boundaries of the model must be specified; for non-steady state problems (Transient) the initial conditions must be set as well.

ii. Solution. Once the preprocessing stage is done, a set of equations that define the model is generated given the geometry, boundary conditions and the physical model. This set of equations is solved iteratively (in most of the cases) for both steady state and transient problems. Diverse strategies are used to solve the huge equation systems involved during this step, such as Gauss Seidel, Successive overrelaxation or Krylov subspace methods (BARRET, 1994).

Figure 4: Mount St. Helen's CFD domain over the Topography.



GARCIA (2005).

iii. Post-processing. Post-processing is the final step of the CFD process. After solving the problem, visualizing the solution is an important matter that is vital to formulate a proper analysis and solution validation judgments. Stereoscopic view systems, high definition screens and immersive virtual reality environments are used for this purpose on high end systems and large scale projects.

0.1.4 Discretization Schemes. The selection of a suitable discretization scheme to solve a problem is one of the keys to success. For several problems, more than one discretization scheme is appropriate and right on this spot is where the criterion of the person who's facing the situation is definitive (solution times might vary from scheme to scheme exponentially for a given problem). Numerical stability is one of the main goals one must pursue, and in most of the cases the discretization scheme chosen is the one that fits best this condition. Some of the most used discretization methods being used are:

Finite Volumes Method. This method responds to the principle that the governing equations for the domain can be solved on discrete control volumes that represent the original domain; The integral approach for this method can be seen on the following equation:

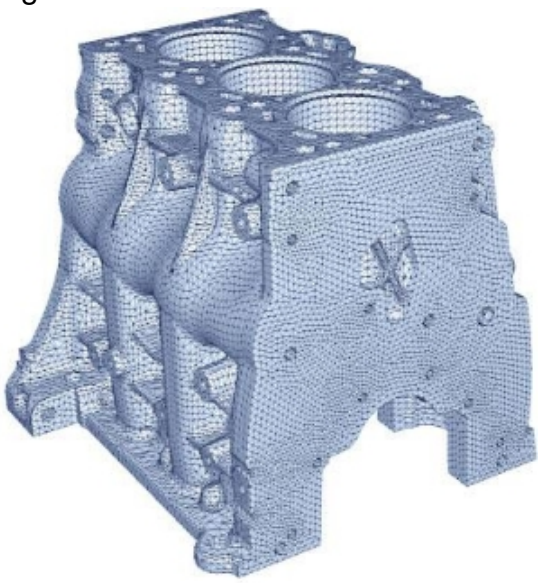
$$\frac{\partial}{\partial t}\int Q\,dV + \int F\,dA = 0 \tag{1}$$

Where Q is the vector of conserved variables, dV is a differential of Volume, F is the vector of fluxes and dA is the boundary of each differential of volume. It is evident that this method is inherently conservative. This is a very popular methodology on both commercial and open source codes.

Finite difference method. This method has historical importance and is simple to program. It is currently only used in few specialized codes. The main disadvantage is that it requires structured meshes, and coordinate transformations for complicated geometries (WIKIPEDIA@, 2006b).

Finite Elements Method. The development of this methodology can be tracked back to the 1940's. The main idea of this approach is the possibility to divide a continuous domain using a mesh discretization, returning a set of discrete sub domains.(See 5) The method was provided with a rigorous mathematical foundation in 1973 with the publication of Strang and Fix's *An Analysis of The Finite Element Method*, and has since been generalized into a branch of applied mathematics for numerical modeling of physical systems in a wide variety of engineering disciplines, e.g., electromagnetics and fluid dynamics (WIKIPEDIA@, 2006c). Although this methodology is used mainly for structural and mechanical analysis, it is also used for fluids; However, this formulation requires a special care for transient situations to ensure conservative solutions.

Figure 5: Mesh of an engine block used for FEM.



MACHINE DESIGN y MEDIA@ (2005).

Boundary Element Method. Conceptually, this formulation works by constructing a mesh over the modeled boundary. This formulation has become more popular since the 1980's. Because it requires calculating only boundary values, rather than values throughout the space defined by a partial differential equation. It is significantly more efficient in terms of computational resources for problems where there is a small surface/volume ratio (WIKIPEDIA@, 2006a).

0.1.5 CFD Post-processing. Flow simulation and visualization has been the center of research topics for the last 30 years. The governing equation that describe the fluid mechanics are the Navier-Stokes equations. Numerical solution to these equations is accomplished by Volume Element, Finite Element, Boundary Element and meshless methods among others (C. y TAYLOR, 2000).

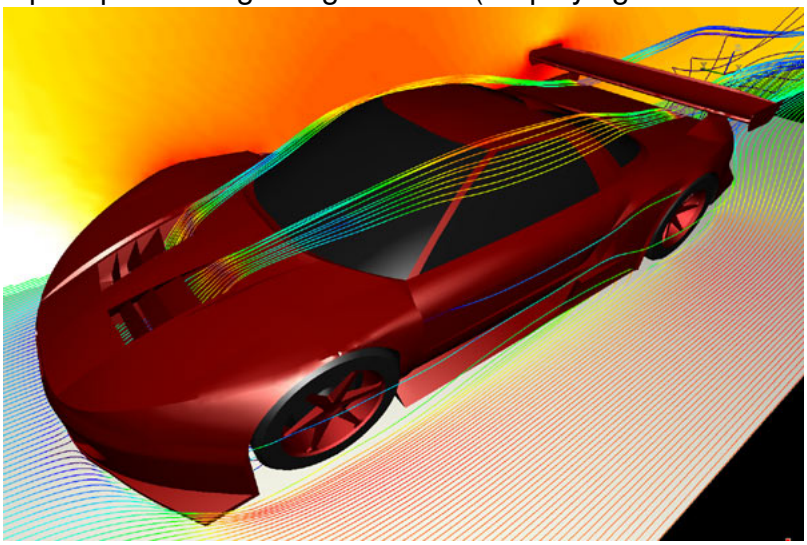
Due to the large simulation times visualization of the data is usually accomplished later stage in the design process. Any posteriori change in the design will have to be feed into the simulator data for a later visualization. As oppose to this model "Interactive design" allows engineers to design as they analyze. They will feel the fluid as if they were immerse in a real wind or current of water. Previous related works with similar objectives include:

- The virtual Wind Tunnel by the NASA Ames Research Center in which pre-computed solutions to the unsteady Navier-Stokes equations of fluid motion are visualized through a virtual environment. The scene is rendered from the point of view that tracks the head of the user. Visualization tools include particle paths, tuffs streak-lines and streamlines (BRYSON y LEVIT, 1992).

- The CAVEvis, a Distributed Real-Time Visualization of time-varying Scalar and Vector Fields using the CAVE Virtual Reality Theater at the NCSA National Center for Supercomputing Applications in Champaign, IL. (JASWAL, 1997) which is a interactive visualization tool for large data sets of scalar and vector fields. It allows streamlines, iso-surfaces and cut-planes. A use can intuitively select, move, manipulate, and interact in other ways with these domain space objects to do things such as emit sets of particles or probe for data values.

- Interactive Lens visualization techniques by Shaw et al which is an interactive glyph-based visualization system. The user may also explore the volume by using "lens", a rectangle that slices through the 3D volume and displays scalar information on its surface. A lens allows the display of scalar data in the 3D volume using a 2D contour diagram, and a texture-based volume rendering. They also use pseudo transparency to globally visualize the field domain (SHAW et al., 1999).

Figure 6: CFD post-processing using ACUITV (Displaying Streamlines).



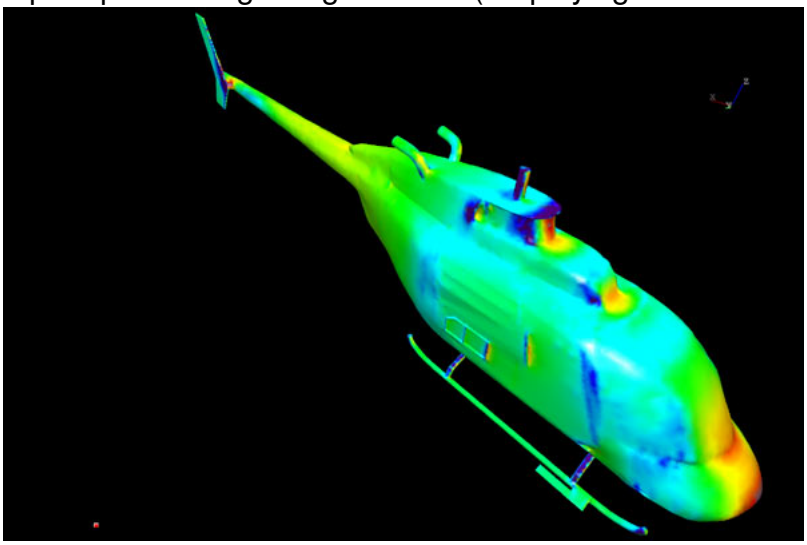
FUEL TECH INC.@ (2004)

- On the commercial side ACUITIV Software produces a visualization tool call ACUITIV that has the ability to read transient datasets as they are produced by a simulation software and offers an interactive tool suite for visualizing these complex datasets through the use of contours, iso-surfaces, streamlines, path-lines, streak-lines, time-lines and point queries (FUEL TECH INC.@, 2004). See Figure 6 and Figure 7.

0.2 Parallel computing

Parallel computing is based on the fact that the execution of a task, can be done simultaneously on multiple processors in order to obtain results faster; To achieve this goal, the task must be redefined and adapted, splitting it up in smaller tasks, which may be carried out by different processors with some coordination (WIKIPEDIA@, 2006e).

Figure 7: CFD post-processing using ACUITV (Displaying Pressure Map).



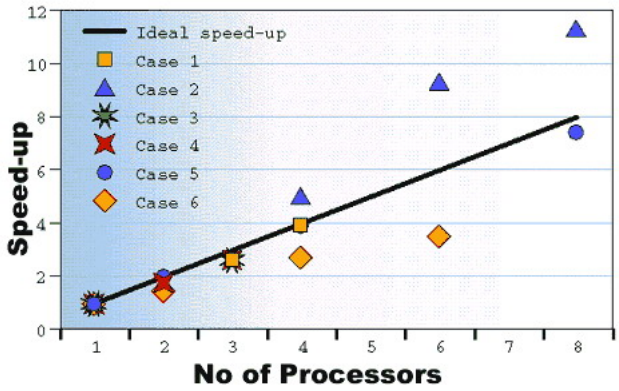
FUEL TECH INC.@ (2004).

An example of parallel computing results can be seen on figure 8. The plotted results were obtained using a commercial software (FLUENT). The diagram shows the time speed up present during the calculation for 6 different CFD scenarios, due to the use of n processors on the same computing task (using domain decomposition technics).

The main characteristics of parallel computing are: Task division, inter-task communication, clear identification of the parallelism degree that can be reached on a given problem, task distribution and control mechanisms (JARAMILLO et al., 2006).

0.2.1 Parallel Computing Systems. The term parallel processor is sometimes used for a computer with more than one processor, available for parallel processing. Systems with thousands of such processors are known as massively parallel (WIKIPEDIA@, 2006e). The recent multicore processors are also ideal to build up parallel computing systems. Subsequently there are what are referred to as “Large Grain” vs “Small

Figure 8: Typical parallel processing performance results for various system architectures. 3D laminar and turbulent test problems were used with meshes from 25,000 to 100,000 nodes.



FLUENT INC.@ (2006).

Grain" parallel processors. This refers to the size of the processor, eg. a PC based parallel system would be considered a small grain system.

There are many different kinds of parallel computers (or "parallel processors"). They are distinguished by the kind of interconnection between processors (known as "processing elements" or PEs) and between processors and memories. Flynn's taxonomy also classifies parallel (and serial) computers according to whether all processors execute the same instructions at the same time (single instruction/multiple data – SIMD) or each processor executes different instructions (multiple instruction/multiple data – MIMD). Parallel processor machines are also divided into symmetric and asymmetric multiprocessors, depending on whether all the processors are capable of running all the operating system code and, say, accessing I/O devices or if some processors are more or less privileged.

There are also a variety of architectures that have been developed to accomplish parallel processing. One such example would be a "Ring Architecture" where pro-

cessors are linked by a ring structure, allowing all processors to read and write data simultaneously, therefore accomplishing a true parallel processing environment. Using such an architecture with a large grain set of processors, has demonstrated processing speed improvements of 10 times or more over classical computing.

0.2.2 Parallel Computing Algorithms. It should not be imagined that successful parallel computing is a matter of obtaining the required hardware and connecting it suitably. The difficulty of cooperative problem solving is aptly demonstrated by the following dubious reasoning:

If it takes one man one minute to dig a post-hole then sixty men can dig it in one second.

In practice, linear speedup (i.e., speedup proportional to the number of processors) is very difficult to achieve. This is because many algorithms are essentially sequential in nature (Amdahl's law states this more formally).

Up to a certain point, certain workloads can benefit from pipeline parallelism when extra processors are added. This uses a factory assembly line approach to divide the work. If the work can be divided into n stages where a discrete deliverable is passed from stage to stage, then up to n processors can be used. However, the slowest stage will hold up the other stages so it is rare to be able to fully use *n* processors.

Most algorithms must be redesigned in order to make effective use of parallel hardware. Programs which work correctly in a single CPU system may not do so in a parallel environment. This is because multiple copies of the same program may interfere with each other, for instance by accessing the same memory location at the same time. Therefore, careful programming is required in a parallel system.

Superlinear speedup - the effect of an *n*-processor machine completing a task more than *n* times faster than a machine with a single processor similar to that in the multiprocessor has at times been a controversial issue (and lead to much benchmarking) but can be brought about by such effects as the multiprocessor machine having not just *n* times the processing power but also *n* times cache and memory thus flattening the cache-memory-disk hierarchy, more efficient use of memory by the individual processors due to partitioning of the problem and a number of other effects. Similar boosted efficiency claims are sometimes aired for the use of a cluster of cheap computers as a replacement of a large multiprocessor, but again the actual results depend much on the problem at hand and the ability to partition the problem in a way that is conductive to clustering.

0.2.3 Inter-thread communication. Parallel computers are theoretically modeled as Parallel Random Access Machines (PRAMs). The PRAM model ignores the cost of interconnection between the constituent computing units, but is nevertheless very useful in providing upper bounds on the parallel solvability of many problems. In reality the interconnection plays a significant role.

The processors may either communicate in order to be able to cooperate in solving a problem or they may run completely independently, possibly under the control of another processor which distributes work to the others and collects results from them (a “processor farm”).

Processors in a parallel computer may communicate with each other in a number of ways, including shared (either multiported or multiplexed) memory, a crossbar, a shared bus or an interconnect network of a myriad of topologies including star, ring, tree, hypercube, fat hypercube (a hypercube with more than one processor at

a node), an *n*-dimensional mesh, etc. Parallel computers based on interconnect network need to employ some kind of routing to enable passing of messages between nodes that are not directly connected. The communication medium used for communication between the processors is likely to be hierarchical in large multiprocessor machines. Similarly, memory may be either private to the processor, shared between a number of processors, or globally shared. Systolic array is an example of a multiprocessor with fixed function nodes, local-only memory and no message routing.

Approaches to parallel computers include:

- Multiprocessing.
- Computer cluster.
- Parallel supercomputers.
- Distributed computing.
- NUMA vs. SMP vs. massively parallel computer systems.
- Grid computing.

0.2.4 Parallel programming model. A parallel programming model is a set of software technologies to express parallel algorithms and match applications with the underlying parallel systems. It encloses the areas of applications, languages, compilers, libraries, communication systems, and parallel I/O. People have to choose a proper parallel programming model or a form of mixture of them to develop their parallel applications on a particular platform.

Parallel models are implemented in several ways: as libraries invoked from traditional sequential languages, as language extensions, or complete new execution models. They are also roughly categorized for two kinds of systems: shared memory systems and distributed memory systems, though the lines between them are largely blurred nowadays.

Currently main-stream parallel programming models are:

- PVM.
- MPI.
- OpenMP.
- Global Arrays.
- Co-Array Fortran.
- UPC.
- HPF.
- SHMEM.

0.3 Real time programming

In computer science real-time computing (RTC) is the study of hardware and software systems which are subject to a real-time constraint (operational deadlines from event to system response) (WIKIPEDIA@, 2006f). A non-real-time system is one for which there is no deadline, even if fast response or high performance is desired or even preferred.

Real time systems in most of the cases are the ones in which its main application is considered mission critical. It can be said that Real time calculations have failed their main objective if they are not completed in the time-period established as deadline before an event, when deadlines are independent of system load.

0.3.1 Hard and Soft real time systems. Real time systems (RTS) can be divided into two groups: one in which time constraints failures are critical for the performance (Hard or Immediate real-time) and other where those failures are not critical (Soft real-time) (WIKIPEDIA@, 2006f).

Hard real-time systems are typically those systems that have low level interactions with the physical hardware (WIKIPEDIA@, 2006f), e.g. car engine's control systems, pacemakers and industrial process controllers; delayed signals on any of those low-level interactions may easily cause catastrophic system failures.

Soft real-time systems are typically those used where there is some issue of concurrent access and the need to keep a number of connected systems up to date with changing situations, e.g. software used to update and maintain live video systems (WIKIPEDIA@, 2006f) . This kind of systems can operate at very low latencies, but violations of time constraints result in degraded quality, even though the system may continue to operate in a proper way.

0.3.2 Real time and high performance. Real-Time Computing is sometimes misunderstood with high performance computing, e.g. a massive supercomputer running an engineering simulation may show impressive performance, yet it is not executing a real-time computation. Conversely, once the hardware and software for an

anti-lock braking system has been designed to meet its required deadlines, no further performance gains are necessary. Furthermore, if a network server is highly loaded with network traffic, its response time may be slower but will (in most cases) still succeed. Hence, such a network server would not be considered an RTC system: Temporal failures (delays, time-outs, etc.) are typically small and compartmentalized but are not catastrophic failures

In an RTC system, a slow-down beyond limits would often be considered catastrophic for its application context.

Some kinds of software, such as many chess-playing programs, can fall into either category. For instance, a chess program designed to play in a tournament with a clock will need to decide on a move before a certain deadline or lose the game, and is therefore a real-time computation, but a chess program that is allowed to run indefinitely before moving is not. In both of these cases, however, high performance is desirable: the more work a tournament chess program can do in the allotted time, the better its moves will be, and the faster an unconstrained chess program runs, the sooner it will be able to move (WIKIPEDIA@, 2006f).

The essential difference between real-time computations and other kind of computations can be described as well with the following example : if a tournament chess program does not make a decision about its next move in its allotted time it loses the game - i.e. it fails as a real-time computation - while in the other scenario, meeting the deadline is assumed not to be necessary.

0.4 Problem contextualization

0.4.1 Context Since the building of the first wind tunnel by Francis H. Wenham in Great Britain in 1871, the study of fluid flow problems over complex topologies have developed immensely. Wind tunnels are used by engineers and scientists to simulate air-flow conditions in the laboratory. They consist of carefully designed ducts through which a stream of air is driven at controlled velocities and uniform conditions (BRITANNICA@, 2006).

Wind tunnels are designed to simulate factors such as speed, fluid pressure on the various surfaces of a model, and temperature during different fluid flow conditions. Scale models are cheaper, take less time to build, and generally allow more extensive instrumentation than full-scale testing, testing that sometimes is even impossible. Thus all flight vehicles are first tested in a wind tunnel before the design is finalized (BRITANNICA@, 2006).

To guarantee that the results of scale tests are directly proportional to full-scale models, dynamic similarity must be preserved, this implies that Mach and Reynolds numbers are to be the same as those present in full scale conditions. Mach number is the ratio of the air velocity to the velocity of sound (STREETER, 1966) and the Reynolds number reflects the ratio of inertial forces to viscous forces (STREETER, 1966) allowing a proper prediction of the present drag forces.

During the past century, the development of numerical methods and computers have opened the doors for physics, mathematicians and consequently to engineers to approximately solve physical problems once thought impossible without the proper tools: problems like the one seen on wind tunnels.

The establishment of physical model constitutes one of the main difficulties for the

development of computational tools and mathematical modelation. The presence of partial differential equations on most of the available CFD models, have made numerical methods and analysis a must have to solve the simultaneous equation systems that represent the model and it's constrains. Choosing a suitable numerical methods to solve any system, must be a really well thought decision: numerical stability, convergence and how susceptible is the system (and method) to any change on the boundary conditions is to be taken into account (SALEH, 2002).

Many interactive computer fluid dynamic efforts have been done in the past: CHAM (Concentration, Heat and Momentum) Limited (`http://www.cham.co.uk/`) with their virtual wind tunnel system (CHAM@, 2005) , MIT's David Oh with his Java virtual wind tunnel (OH@, 2001) and NASA Virtual Wind tunnel at Ames research center (BRYSON y LEVIT, 1992) are just a few to mention. One of the current lacks on many interactive CFD projects, is that precomputed solutions are the ones used for display, condition that constrains the design process both in time and flexibility.

0.4.2 Integrated mechanical design problem and Virtual Wind Tunnel (VWT) Mechanical design can be seen as an iterative process in which the shape of mechanical artifacts is to be redefined repeatedly until some design requirements are met, or vice versa, See (ROOZENBURG y EEKELS, 1995). The design requirements nature varies from sources as clear as economical resources to some so abstract as the needs and wishes from the users, and even though needs are more often placed over wishes in a hierarchical structure, the power of wish is still easily what makes the difference between the success of one design solution or another.

It can be said, that a complete cycle of design starts from the identification of the needs, wishes and desires (being those either shape or function oriented) and ends

with a mechanical analysis that checks if the final solution complies with all the basic design requirements stated in the initial phase.

How should be developed the steps in between the start and the end of the designing process is still subject of debate. Design methodologies as the ones proposed by Hansen & Andreasen (HANSEN y ANDREASEN, 2002) tend to present a global and integrated view of the initial conceptualization phase, which they split in 2 parts: the idea “with”(describes the functionality through the user perspective) and the idea “in”(describes the functionality through the engineering perspective) the product. This approach helps to maintain the relation between need, satisfaction and functionality. After the conceptualization phase, they propose the elaboration of an initial solution (form follows function or function follows form, if it is the case) given the previous abstraction of the design requirements.

Once an initial solution is achieved, an iterative process to obtain an optimized solution begins. Currently, with the aid of CAD tools and FEA systems, this iterative process has gained lots of agility and speed compared to an all human analysis process. Shape redefinition and material selection analysis on products are the ones that take most of the time because of the difficulties that represent the FEA preprocessing (mainly, human guided procedures) and shape optimization (even though proposed, automatic shape optimizing algorithms are still on early stages of development).

In recent years, industrial and academic efforts have been carried out towards establishing methodologies and software tools that allow to face the design task as an integrated task. Product Data Management tools integrated to CAD systems, allow the designers to handle information in an organized ways since the early stages of the design, and efforts like the ones from Kela and Perucchio (KELA et al., 1986),

Barthelemy (BARTHELEMY et al., 1991) or Garcia (GARCIA, 1999), have given initial steps forward an interactive, integrated design process, where the development of the solution turns around rather on the designing problematic than on the tools used to solve them.

The CFD analysis represents major tool for Naval and Aeronautic designers. Outer shape on most of their designs is directly related with the engineering requirements a product must suffice and so, fast, reliable and accurate tools to study the impact of a given shape (specially during the early stages of the design) are a must during this times, where time represents more than money: opportunities.

The University of Alberta in Canada, EAFIT and Los Andes universities in Colombia, after realizing the magnitude of the need inside industry and academy for an interactive module to study the movement of fluids, started an initiative to board the interactive VWT problem without precomputed solutions. The need to establish a proper abstract methodology to board the problem, to select the CFD approach that fit best the needs and desires of the project are a task that must be faced before any software or hardware implementation. In the following sections, an initial methodology to solve the VWT is presented to the reader with the objective of setting an initial debate (solution) around this problem, to be optimized during future work (...as any design problem).

1 Computers, Interactivity and Interactive Design

1.1 Interaction, Interactive and Interactivity

Before starting to study the interaction between at least 2 participants (e.g. Man-Man, Man-Machine), conventions about terms interaction, interactive and interactivity are to be set.

As addressed by Svanaes SVANAES (2000), an interaction is a process that involves at least two participants to complete another process. In the context of human-computer interaction, the human is interacting with the computer. An artifact is cataloged as interactive if it allows the user to have any level of interaction. Interactivity denotes the quality of the interactive aspects of an artifact (levels and forms of interaction).

Svanaes uses a simple example to establish the relation between interactivity and interactiveSVANAES (2000), being it metaphorically the same as the one between radioactivity and radioactive: Uranium is radioactive, Madame Curie studied radioactivity; Modern computers are interactive, the current study is about interactivity. From this example, Svanaes also concludes that interactivity can both be used as a noun to signify a general phenomenon, or to signify a property, as in “the interactivity of the modern computer”. During the following chapters, interactivity will be addressed as noun or property.

1.2 Multimedia-Software context

The design task has become over the past decades a game of exploring and analyzing several geometric configurations of a product quickly. The interaction with the machines and software that allow the designer to accomplish this task is by this means a crucial spot to guarantee the efficiency in terms of time and effort of this process.

Interactivity in learning as addressed by Barker is “a necessary and fundamental mechanism for knowledge acquisition and the development of both cognitive and physical skills” BARKER (1994), hence guaranteeing the desired interactivity level into any knowledge producing process is a mean of both controlling its stability and flow.

To guarantee any level of software interactivity, one must have a really deep understanding of the learner (user of the tool), an appreciation of software engineering capabilities, the importance of rigorous software design and the application of appropriate graphical user interfaces SIMS@ (1997). The proper apprehension and application of those practices leads to the following question: what makes an interactive application, instructional and effective?.

Boarding the Design task as an interactivity issue, the manipulation of the graphical and geometrical representations of the environment and the product is a prime target to achieve fast results that assure the continuity and flow of process. The actual multimedia capabilities of a computer rely mainly on 3 aspects for I/O: audio, text and visuals, and for the interactive design task, visuals (information and instructions I/O) and text are basically the ones more often relied on.

1.3 Levels of interactivity

The degree of interactivity an application presents or must have is inherent to the main goal and the methodology the designer desires to implement for the knowledge producing process. Various have been the distinctions and classifications throughout the years made for the level of interactivity a program can reach, here follows a small review of some relevant ones:

Rhodes & Azbell in 1985 RHODES y AZBELL (1985) identified 3 sequential degrees of of interactivity:

Reactive: where little user control takes place over the content's structure with program directed options and feedback

Co active: providing user control for sequence, timings and style.

Proactive: where the user controls both structure and content at most of the levels.

It can be seen in the last scale that the level of interactivity was perceived as directly proportional to the level of control and immersivity the user has related to the application (Figure 1).

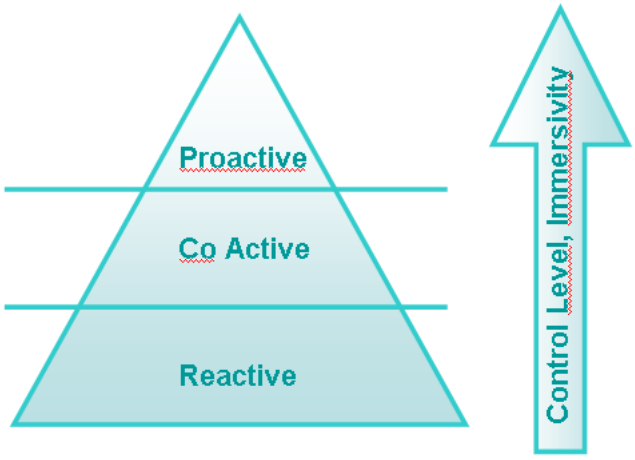
Jonassen in 1988 identified five levels of interactivity which focused more on the user's involvement with the application and the subsequent effect on learningSIMS@ (1997). Those levels are not exclusive between them, and are defined as:

i. The modality of the learner's response.

ii. The nature of the task.

iii. The level of processing.

Figure 1: Rhodes & Azbell's Degrees of Interactivity



iv. The type of program.

v. The level of intelligence in design.

In relation to the analysis of these five levels, Jonassen also suggests evaluating how deep does each level of interactivity of the application affects the learning process for the userJONASSEN (1988).

For Schwier & Misanchuk (1993), not only establishing the level of interactivity of an application helps to understand how does it affects the learning process. They introduced a detailed taxonomy of interactivity based on three dimensions:

i. Levels: This dimension points to how immersive and deep an interactive application may be. The 3 basic categories they propose are:

   (a) Reactive: Basic user response to interactions.

   (b) Proactive: User constructive and knowledge generative activities.

(c) Mutual: Artificial or virtual reality designs, where the user becomes a fully franchised citizen in the environment.

ii. Functions: This dimension revolves around the classification of the actions and activities an interactive application may present to the user. The proposed actions are:

(a) Confirmation: Verification of learning (previous and in development).

(b) Pacing: User control of timings and flow of the learning process.

(c) Navigation: Instructional control to play certain role on the application's environment.

(d) Inquiry: User interrogation and performance support.

(e) Elaboration: New user knowledge construction.

iii. Transactions: The way I/O instructions are passed from-to the user is basic. It is mainly dependant on the I/O devices used for the application, such as:

(a) Keyboard.

(b) Touch Screen.

(c) Mouse.

(d) Voice.

(e) Other I/O devices.

The "levels of interaction are based on the instructional quality of the interaction" SCHWIER (1993), which reinforces the idea introduced by Rhodes & Azbell that the higher the level, the better the instruction an deeper the interactivity. The critical factor Developers of interactive applications must always take into account, is that

even though the level of interaction is directly related to the involvement with the material to be studied, the more complexity of the interaction does not necessarily benefits the learning process, because it is still unknown where and how does the cognitive process takes place. SIMS@ (1997).

1.4 Design methodologies for Interactive Design

A number of diverse methodologies outlining techniques for human-computer interaction design have emerged since the rise of the field in the 1980s. Most design methodologies stem from a model for how users, designers, and technical systems interactWIKIPEDIA@ (2006d). Early methodologies, treated the user's cognitive processes as predictable (linearizable at the very end) and quantifiable, encouraging design practitioners to look to cognitive science results in areas such as memory and attention for GUI'S and ways of interacting. Modern models tend to focus on a constant feedback and conversation between users, designers, and engineers and push for technical systems to be wrapped around the types of experiences users want to have, rather than wrapping user experience around a completed system.

User-centered design (UCD) is a modern, widely practiced design philosophy rooted in the idea that users must take center-stage in the design of any computer system. Users, designers, and technical practitioners work together to articulate the desires, needs, and limitations of the user and create a system that addresses these elements. Often, user-centered design projects are informed by ethnographic studies of the environments in which users will be interacting with the system WIKIPEDIA@ (2006d).

When UCD is applied to more than single user interactions, it is often referred to

as user experience. A user experience comprises a number of separate interfaces, human-to-human contacts, transactions and conceptual architectures. It is not enough to have the separate interactions that comprise an experience being usable. The goal is that each interaction should integrate with every other interaction that forms a part of a single experience. In this way, the experience as a whole is rendered usable WIKIPEDIA@ (2006g).

The basis of UCD is to place the person in the center of the design stage (opposed to the product), focusing on cognitive factors as perception, memory, etc., as they come to play during people's interaction with the artifact and it's surroundings KATZ-HAAS@ (1999). To properly position the user in the middle of the process, UCD designers should focus on questions of the typeKATZ-HAAS (1998):

- Who are the users of this product?
- What are the users tasks and goals?
- What are the users experience levels with this product, and product like it?
- What functions do the users need from this product?
- What information might the users need, and in what form do they need it?
- How do users think this product should work?
- How can the design of this product facilitate users' cognitive processes?

This questions don't have to be answered explicitally during the design process, but implicitly the whole product design process should be present an answer to each one of them. In the following chapters, a methodology to perform interactive CFD

simulations will be developed. The reader is encouraged to keep those questions in mind while reading to get the most out of this study.

2 Interactive CFD Methodology

"The concept phase shall clarify the need/market aspect of the design solution" (Claus Thorp Hansen & Mogens Myrup Andreasen) HANSEN y ANDREASEN (2005).

The very first task to fulfill following Hansen & Andreasen integrated view of the design process is none other than establishing the object or product for the design task. In this case, to obtain a methodology suited to produce Interactive CFD simulations is our main goal, and a Virtual Wind Tunnel will be the designing exercise (final product). The objective of establishing this Methodology is to obtain an abstraction of the problematic and in terms of operations (not of computational tools), polish and extend its applicability to any Interactive CFD simulation process one wishes to simulate, and the objective of the VWT as a design exercise is none other than test the methodology in development.

Domain theory, as a way of understanding the design process will be a valuable tool during proper definition and abstraction of the methodology. Domain theoryHANSEN y ANDREASEN (2002) starts from the decomposition of the object of design into three domains, that can be described asHANSEN y ANDREASEN (2002):

Transformation Domain Where the focus is the pourpose of how and where data, matter or energy flows are transformed and interact with the artefact.

Organ Domain This domain compromises how do the active elements of the product create effects and their mode of action.

Part Domain The competence of this domain is the study of how are the organs (See above) distributed inside the boundaries of the system, how can they be

produced and assembled so every part solves its tasks.

After the domain decomposition of the object of design, the study of the functional synthesis inside each domain and the functional relationships between them is to be done via Hubka's function-mean law ANDREASEN (1980).

Being the CFD interactive simulations methodology still in an abstract state, to properly define the black box model of the whole system, items such as the basic need, task, user and ideas related to the object are to be studied and defined. During this chapter, this whole design exercise will be developed in those terms aided by domain theory.

2.1 Design exercise definition.

To properly reference this design exercise, the primitive design constrains for an interactive CFD methodology are to be defined in terms of Need, Task and User/Consumer. Need refers to the present unsatisfied state that the product will satisfy, the Task refers to the way the product will perform to satisfy the present need and the User/Consumer item to who will use with the product in question.

For the interactive CFD methodology case need, task and user will be:

- Need: A set of tools that allow the designer to test interactively the behavior of a design under flow conditions.
- Task: The main task performed by a CFD interactive methodology can be defined as describing a way for solving simulations of fluid flow, given a possible scenario defined by the user and finally allowing him to get feedback from the results.

- User/Consumer: The target market is none other than designers interested in testing their design's behavior under fluid flow conditions. Academic and Industrial means can be achieved thought this testing.

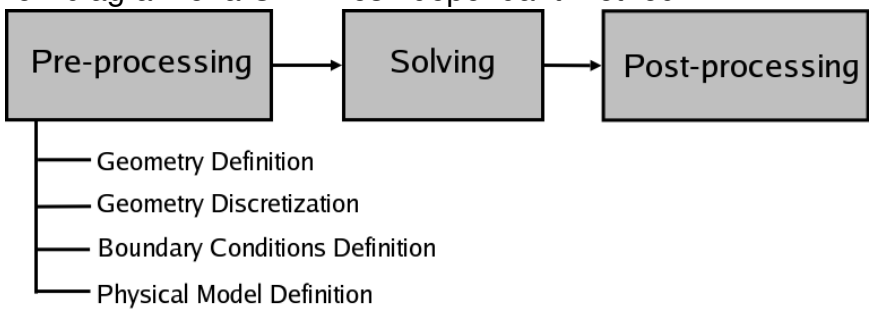
Given need, task and user, the idea "in" and "with" an interactive CFD methodology can be described as:

- The idea "in": To apply the CFD techniques to produce solutions for possible fluid flow scenarios, usefull for user intended feedback.
- The idea "with": To easily simulate fluid flow scenarios that can be used to feedback the shape designing process, boundary conditions definition and simulation variables tuning.

2.2 CFD User needs.

Basically, most of the CFD mesh dependant methods follow an almost linear process to perform an analysis, as shown in Figure 1:

Figure 1: Flow diagram of a CFD mesh dependant method



i. Pre-processing.

   - Geometry definition. In this stage the physical bounds of the problem are defined.
   - Geometry Discretization. The volume occupied by the fluid is discretized into finite cells.
   - Physical model definition. Establishing the physical model that will describe the phenomena's behavior; e.g. the Navier-Stokes equations, heat transfer models, entropy equations, etc.
   - Boundary Conditions definition. After the physical-geometrical boundaries and models are defined, the fluid's behavior and properties at the boundaries of the model must be specified; For non-steady state problems (transient) the initial conditions must be set as well.

ii. Solution. Once the preprocessing stage is done, a set of equations that define the model is generated given the geometry, boundary conditions and the physical model. This set of equations is solved iteratively (in most of the cases) for both steady state and transient problems. Diverse strategies are used to solve the huge equation systems involved during this step, such as Gauss-Seidel, Successive overrelaxation or Krylov subspace methods BARRET (1994).

iii. Post-processing. Post-processing is the final step of the CFD process. After solving the problem, visualizing the solution is an important matter that is vital to formulate a proper analysis and solution validation judgments. Stereoscopic view systems, high definition screens and immersive virtual reality environments are used for this purpose on high end systems and large scale projects.

Knowing this, the basic input information on each stage of the process needed to run a CFD analysis might be seen as (in no particular order):

i. Pre-processing.

- Geometry definition: Standard CAD models, e.g. IGES, STEP, VRML or STL models offer both portability and the accuracy needed for engineering analysis.
- Discretization Information: Primarily dependant to the kind of Pre-processor and processor (e.g Finite Volume, Boundary Elements or Finite Elements Methods oriented). Often, mesh type, mean cell size and type are the most widely used parameters to define the mesh.
- Boundary Conditions: Initial velocity profiles, solid-fluid interphase contact conditions, outlet pressures, temperature and heat transfer conditions, are the ones used more often on the CFD process.
- Selected physical model: to solve e.g. Turbulent, laminar, compressible, incompressible, etc...
- Selected numerical methods: To solve the equations that rule the chosen physical models, numerical methods are to be selected properly to guarantee accuracy, convergence and stability.
- Various solver constrains. The physical model solution is constrained by the accuracy of the numerical methods used to solve it and the correction methods set for it. The user must set the way the corrections and numerical methods behave so he can expect a certain behavior from the solver.

ii. Post-processing.

- Selected data for display: given the massive amount of information the user can access after solving an scenario, choosing which sets of data are to be and how will be analyzed is necessary.
- Viewpoint and Scales.

This basic data input is mainly related to a single case scenario. For an interactive scenario, the amount and nature of extra data input depends on the interactivity level and goals.

2.3 Interactive CFD Methodology Needs

The main idea behind most of the CFD simulations is to obtain a global and accurate idea of the behavior of the fluid flow in contact with other fluids or solids. This kind of simulations pretend to reproduce physical phenomenons present on the fluid flow and its implications, being this phenomenons from non-steady natures in most of cases.

To simulate non-steady phenomenons is one of the main ideas behind the CFD methodologies, therefore the interactive methodology in development is no different from the others. Non-steady simulations represent a mathematical and computational challenge not to be taken lightly and are one of the best ways to obtain the idea of the behavior of an specific design under load conditions before its construction.

Interactive scenarios revolve around possible changes. This changes might be classified into different classes due to the difference between their natures (basically, the data types and the quality of the information they hold) :

i. Geometrical: This changes are the ones that mainly relate to possible modifications on the geometrical aspects of the bodies that interact inside the simulation. The possibilities are:

   (a) Positioning: The relative position between two or more bodies inside a simulation may vary. The user might want complete control over this.

   (b) Orientation: The relative orientation of an object respect to any fixed or non fixed coordinate system might change.

   (c) Size: Scaling operations over a body respect to any coordinate system.

   (d) Shape modification of the elements.

   (e) Addition-Substraction of bodies.

ii. Physical: The nature of this changes is related to the possible modifications the user might want to do over over the physical constrains of the simulation. The possibilities are:

   (a) Fluid properties: The physical properties that define a fluid (Viscosity model, specific weight, specific heat, etc...).

   (b) Boundary conditions: The user might want to change the current boundary conditions to simulate different scenarios. Setting new boundary conditions might change the current flow's regime, so the solver must be aware that perhaps the current solving models might crash or simply not apply.

   (c) Timestep of the simulation.

iii. State of the Simulation: Setting Start-Stop flags is necessary to control the continuity of the process.

Basically, all the changes listed above are modifications of the scenario the user is intended to interact with, not over the application that is being used to solve the scenario.

Most of this changes require almost the performing of the whole pre-processing stage again. Most of the Geometrical changes force the recalculation of the mesh used for the computations and may as well induce changes on the scenario of other kinds, like changing the boundary conditions of the domain (e.g. during the substraction of a solid body immersed on the fluid). The possible effects of performing a change over the CFD scenario are portrayed on Table 2.1, as well as the possible chain of events that one change might cause.

Table 2.1: Changes on the CFD Scenario and their respective effects

Type	Change	Direct Effects	Possible Induced Changes
Geometric	Positioning	Mesh-Recalculation	Boundary Conditions Timestep
	Orientation	Mesh-Recalculation	Boundary Conditions Timestep
	Size	Mesh-Recalculation	Boundary Conditions Timestep
	Shape Modifications	Mesh-Recalculation	Boundary Conditions Timestep
	Addition-Subtraction	Mesh-Recalculation Boundary Conditions	Timestep
Physical	Fluid Properties		Timestep Mesh-Recalculation Fluid flow Regime Change
	Boundary Conditions		Timestep Fluid flow Regime Change Mesh-Recalculation
	Timestep		Mesh-Recalculation
State of the Simulation	Continue the simulation?	Start-Stop	

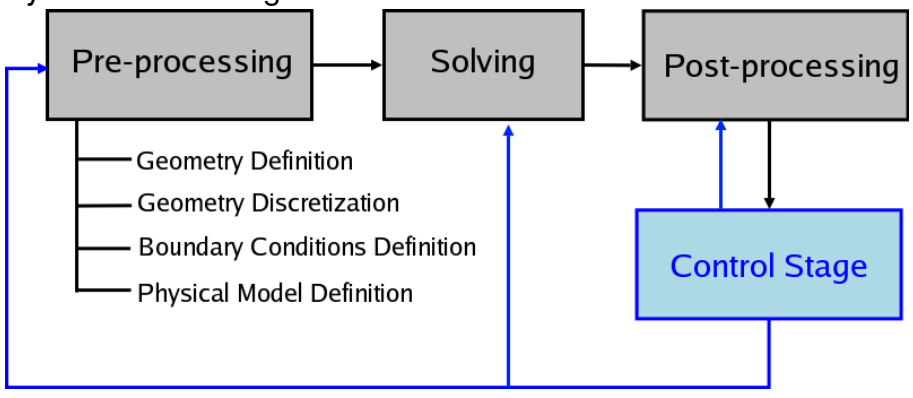
As shown in Table 2.1, a single change may induce a chain reaction that might produce a significantly greater computational cost than expected, e.g. If a body is added to the scenario, the whole mesh is to be recalculated, new boundary conditions are to be set over the body (no-slip wall conditions for example) and in the worst case, the timestep of the simulation is to be changed as well (generating new possible

computational costs). Bigger computational costs might be caused by the change on the fluid properties, being the worst case scenario one on which the mesh and the solution model have to be changed due to a change on the flow regime (Laminar to Turbulent).

The prime needs a totally interactive CFD methodology might have, are the same as the ones of a non-interactive methodology plus the ability to control the changes shown in Table 2.1. The level of immersivity the user might desire will only generate new needs and wishes mainly related to the sensitive aspects of the interaction, but not with the core interaction with the CFD scenario to be simulated.

Being the post-processing stage of the simulation an interactive stage, a new instance called Control (Parallel to the Post-Processing instance, and being fed by the Post-Process partial results) is added to the workflow diagram shown in Figure 1. The Workflow diagram for an interactive CFD simulation-methodology is shown on Figure 2, and might be described as follows:

Figure 2: Flow diagram of a CFD mesh Interactive method, Blue lines show the feedback by the Control Stage



i. Pre-processing.

   - Geometry definition. In this stage the physical bounds of the problem are defined.
   - Geometry Discretization. The volume occupied by the fluid is discretized into finite cells.
   - Physical model definition. Establishing the physical model that will describe the phenomena's behavior; e.g. the Navier-Stokes equations, heat transfer models, entropy equations, etc.
   - Boundary Conditions definition. After the physical-geometrical boundaries and models are defined, the fluid's behavior and properties at the boundaries of the model must be specified; For non-steady state problems (transient) the initial conditions must be set as well.

ii. Solution. Once the preprocessing stage is done, a set of equations that define the model is generated given the geometry, boundary conditions and the physical model. This set of equations is solved iteratively (in most of the cases) for both steady state and transient problems. Diverse strategies are used to solve the huge equation systems involved during this step, such as Gauss-Seidel, Successive overrelaxation or Krylov subspace methods BARRET (1994).

iii. Post-processing. After solving the problem, visualizing the solution is an important matter that is vital to formulate a proper analysis and solution validation judgments. Stereoscopic view systems, high definition screens and immersive virtual reality environments are used for this purpose on high end systems and large scale projects.

iv. Control stage of the simulation. Parallel to the Post-processing stage, the con-

trol stage is fed from the results shown during post-processing and is the stage where the user analyses if any change over the current CFD scenario is to be made. If the CFD scenario changes somehow, the control structures, organs or the user itself must perform a feedback process to all the other stages (Pre-process, Solution and Post-process) of the CFD simulation providing the necessary data to continue the simulation.

The possibility to perform any of the changes over the CFD scenario shown in Table 2.1 suggest that the list of needs that arise from an interactive CFD methodology might be grouped into 2 sets:

i. Non Interactive CFD needs Set. This set of needs are mainly related to the setup of the initial scenario to be simulated (See Section 2.2 for a complete list).

ii. Interactive CFD needs Set. This set of needs arises from the quality of being interactive during the post-processing via the control stage. This set of needs go hand by hand with the possibilities of changes over the CFD scenario described in Table 2.1.

    (a) Geometry redefinition. This subset of needs is mainly related with the topologycal and geometrical changes a CFD scenario might suffer during the simulation. Five basic specific needs compose it:

        i. Body Re-positioning control: To have control over the relative position between two coordinate systems attached to the bodies involved during the simulation.

        ii. Body Re-orientation control: To have control over the relative orientation between two coordinate systems attached to the bodies involved

during the simulation.

iii. Body Re-sizing control: To have control over the possible scalability of the bodies involved during the simulation.

iv. Body addition-subtraction: To have control over the possible addition or subtraction of new/old bodies involved on future stages of the simulation, which will be positioned, oriented and scaled via the controls on the items (i),(ii) and (iii).

(b) Physical variables redefinition. This subset of needs is mainly related with the changes of the physical constrains in a CFD scenario during the simulation. Three basic specific needs compose it:

i. Fluid properties control: To have control over the physical properties that define the fluid (Viscosity,Specific weight and heat, etc...).

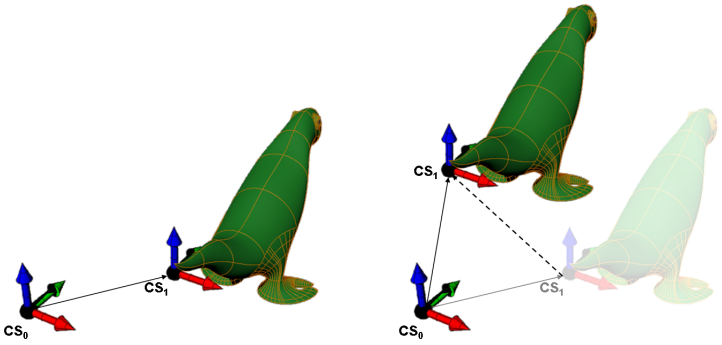
ii. Timestep control: To have control over the timesteps used to solve the scenario.

iii. Boundary conditions control: To have the ability to modify the present boundary conditions.

(c) Change the current state of the simulation: To have control over the fact of stopping or continuing the simulation process.

2.4 Methodology's main & Specific Objectives

Having studied the problem, being the eye of the beholder the one of a designer, the methodology's main and specific objectives can be defined as:



(a) Original coordinate systems positioning

(b) Relative re-positioning between 2 coordinate systems

Figure 3: Relative Positioning

2.4.1 Main Objective: To suggest a set of steps to follow for properly performing accurate and fast interactive CFD simulations.

2.4.2 Specific Objectives:

i. To define a proper methodology to integrate the initial pre-processing stage of the CFD methodology with the interactive CFD process, given the minimum user interaction as an objective.

ii. To establish the basic information needed by the solver to process a single timestep instance of the CFD simulation.

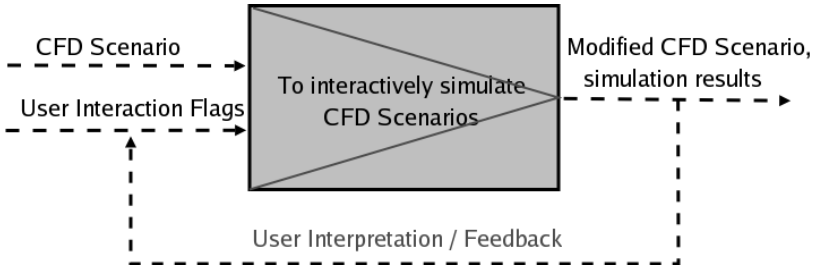
iii. To establish a proper methodology to integrate post-processing and interaction processes, minimizing the data streams between this two processes.

iv. To establish an organized process for controlling and validating the possible changes over the CFD simulation scenario during time.

2.5 Main Function and Flows Abstraction (Black Box Model)

Having an integrated idea of the design problem (Idea in, Idea with, see Section 2.1), a Need specifications definition for the product and newly defined objectives for the product, it is now possible to visualize a possible black box model definition as shown in Figure 4. The presented black box intentionally does not complies with all the proper conventions of conceptual design methodologies, in hopes of clarity.

Figure 4: Black Box Definition.



The Black box definition for this methodology features two inputs and one output flows, being:

i. Inputs:

   (a) CFD Scenario: This data input contains all the information related to properly define a case scenario as shown in Section 2.2 (Geometry definition, discretization information, boundary conditions, physical model, numerical methods, solver constrains).

   (b) User I/O Flags: All the information associated with Start-Stop signals and CFD scenario modifications.

ii. Outputs:

(a) CFD Simulation Results: The post-processed results of the scenario

The main function abstracted for the whole process is “To Simulate”, contextually rewritten as “To interactively simulate a CFD Scenario”. This function, as Figure 4 shows, integrates the two input data flows into a single output via the simulation (Basically, a transformation) process.

This black box abstraction also exemplifies a finite state machine control structure, due to it’s cyclic lifecycle. After an initial change of state (establishing an initial CFD scenario and feeding the user’s START flag to the system), the simulation process takes place and then the results are presented as both output to the user and a feedback to the next timestep of the simulation. The user may interpret the outputs and then either change the scenario, feed a STOP flag or simply let the simulation process continue indefinitely.

The abstracted main function (to Simulate, a transformation process at the very end!) when read in context, implies Interactivity (a term previously developed in this document, see Chapter 1). The nature of the User feedback might be the same no matter the level of immersion/interactivity present on the methodology or application, but the degree of knowledge construction will sensibly change due to the possible user sensitive-cognitive processes.

The methodology in development is intended to be generic but even though interactivity is a main requirement, it’s level might vary depending on the desired application of the methodology, impplying possible positive/negative effects on the effectiveness of the Interactive CFD simulations. Being those effects function of the user cognitive, sensitive and technical abilities, further studies are to be conducted to see how those variables affect the level of perception of the obtained results during post-processing.

The resolution of the results is as well a variable to be taken into account during those studies, but being this a problem of a more implementation-based nature for the current study, it will be addressed as such on this document.

2.6 Functional Synthesys

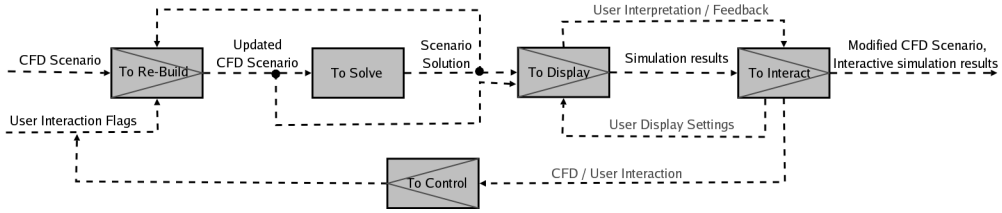
Following the Domain Theory as reviewed by Hansen & Andreasen HANSEN y ANDREASEN (2002), every product is to be decomposed into 3 layered domains (transformation, Organ and Part) following Hubka's function-mean law.

Being the methodology in development an abstraction of the processes and relations that take place during an interactive CFD simulation, the Transformation and Organ domains are generic for any application. The Part domain holds similarities between different applications of the methodology no matter the depthness of the interactive experience, but as detailed as it is, it is of no use to establish right now a single general abstraction of it. The currently in-development VWT implementation will be used for this means.

2.6.1 Transformation Domain

First level of function segregation. As mentioned in Section 2.5, the main function for this methodology is "To Simulate". The functional boundaries of the system contain the whole data transformation process and border with the user feedback of the scenario the designer is interacting with. Starting from the black box diagram shown in Figure 4 the first level of the functional segregation is shown in Figure 5.

Figure 5: First level of function segregation



The first functional level is composed by 5 main functions that can be described as follows:

i. To Rebuild (the mesh & CFD scenario). This function receives as an input a properly defined “CFD scenario” suited for the solver being used. It contains all the information needed for the pre-processing stage of the simulation. During the pre-processing stage the geometry is fed to the meshing algorithms, boundary conditions are applied and the physic constrains of the system are set. This initial CFD scenario contains all the basic information for the solver to properly work, also this information can be modified via the second input, “User interaction Flags”. As the simulation process develops, the solution of the current state of the simulation is applied to the current CFD scenario as an initial guess of what the next timestep solution might be (for geometrical modifications of the CFD scenario, this can considerably minimize the “numerical shock” during the calculation of the proper solution).As output of this function, an “Updated CFD Scenario” is given to the next phases of the simulation. This updated scenario contains a set of valid boundary conditions, geometry (mesh), physical constrains (fluid properties, timestep, etc...), an initial guess of what the solution of the flow might be and information about the flow model used for solving it for

the next timestep.

ii. To Solve (the current CFD scenario). The solver takes all the definitions inside the CFD scenario and uses them to solve via various methodologies (Finite Volume, FEM, etc... See subsection 0.1.4) the differential equations that represent the fluid flow properties one wishes to calculate. The results obtained from the solution of those equations is passed on to the “Display” phase of the simulation (See Figure 5) and to the “Rebuild” phase for generating a next simulation scenario to be solved for a single timestep.

iii. To Display (the CFD scenario and it’s solution). Having a proper representation of the geometry and a solution to the current CFD scenario, an intgrated way to display both is required. Visual, auditive or haptical means are usefull for this purpose. The “Display” and “Interact” functions are intrinsically related and through the means of the Display, the desired level of immersivity inside of the current CFD scenario can be reached (from only text displays, to fully immersive virtual reality scenarios, depending on the design requirements). Setting visualization options such as camera angle, color scale, zoom, perspective-orthographic type of view, etc.. are mainly dependant of the kind of display method used.

iv. To Interact (with the displayed simulation results). Given the information (simulation results, see Figure 5) on the “Display” phase, performing an interaction given the interpretation of the results is the next step to follow. The interaction can be performed at two levels,with the CFD scenario or with the display settings used for the simulation results. To perform the interaction many means may be used (the same used for display) depending on the level of immersivity desired. The user interaction produces a set of proposed modifications to

the CFD scenario that might not be valid (E.g. changing the values of certain boundary conditions might induce changes on the flow regime) that will be passed on to the “Control” phase for validation.

v. To Control (validation and adaptation of the modifications to the CFD scenario proposed by the user). After the “Interaction” phase the set of proposed modifications to the CFD scenario is checked and modified to be acceptable in terms of the pre-processor and solver (Guaranteeing the simulation stability and that the proper models are used to solve the possible flow conditions). Once the modifications are modified and validated, they pass to the pre-processor in the “Rebuild” stage to restart the cycle (See 5).

Second level of function segregation. Each of the 5 main functions can be described on terms of their respective subfunctions.

Figure 6: Second level of function segregation (Rebuild Function)

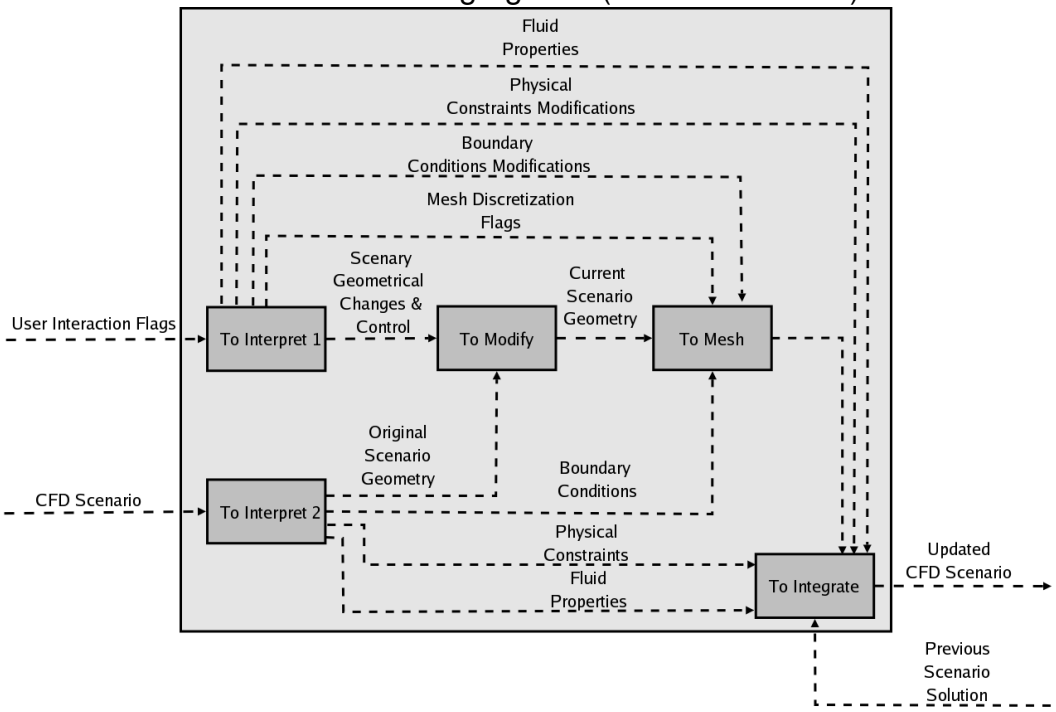


Figure 7: Second level of function segregation (Solve Function)

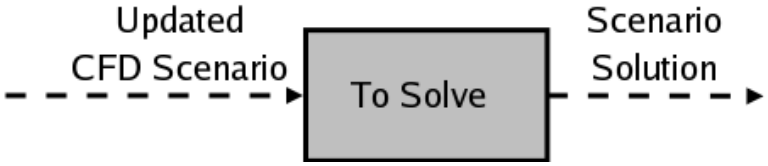


Figure 8: Second level of function segregation (Display Function)

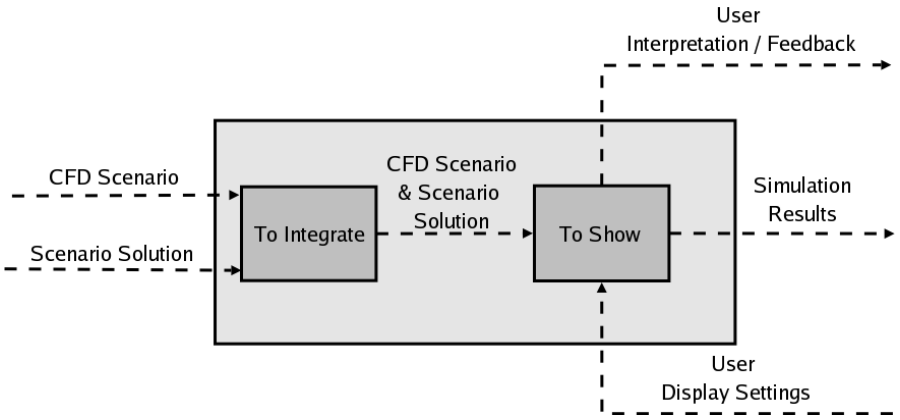


Figure 9: Second level of function segregation (Interact Function)

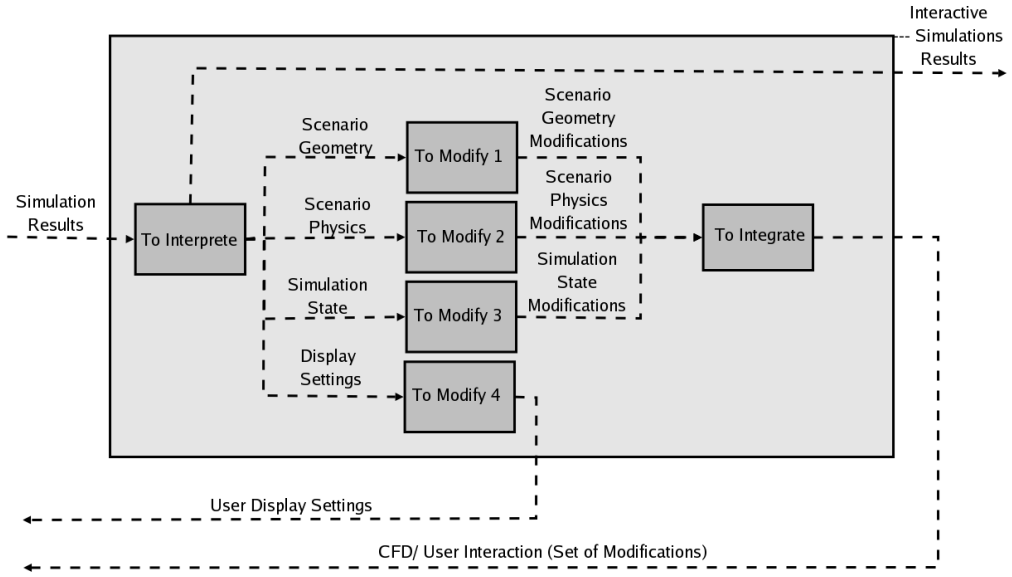
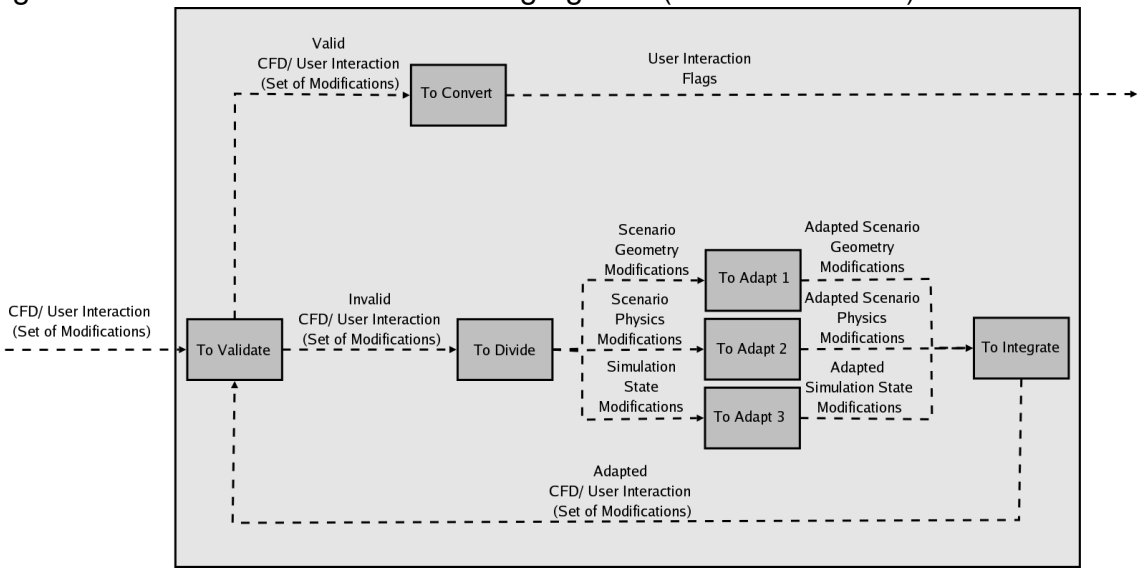


Figure 10: Second level of function segregation (Control Function)



i. Rebuild Function. The rebuild function can be segregated into the following set of subfunctions. The big grey square that surrounds all the diagram (Figure 6), represents the domain of this function.

   (a) To Interpret (the User Interaction Flags). Reading a valid user input for the interaction flags and classifying it into the proper groups of information to be passed on to the different stages of the pre-processing phase.

   (b) To Interpret (the CFD scenario). Reading a valid user input for the CFD scenario and grouping all the information contained inside it to be fed to the next stages of the pre-processing phase.

   (c) To Modify (the CFD scenario geometry, given the geometrical user interaction flags). With the gometrical information of the initial CFD scenario and the possible modifications that it is subject to, the geometry is modified to the point it is needed to be fed in later stages to the Meshing algorithms.

(d) To Mesh (the modified CFD scenario geometry). The geometry is discretized following the mesh discretization flags that were input by the user.

(e) To Integrate (the modified CFD scenario Mesh, the physical contrains, fluid properties & the lastest solution). The whole discretized scenario is integrated into a single packet to be solved in later stages. The boundary conditions are applied over the discretized domain, the fluid properties are set, the physical constrains are induced and the solution of the latest solved scenario is mapped on the current domain to continue with the simulation.

ii. To Solve.(Figure 7)

(a) To Solve (the current CFD scenario). The CFD scenario is solved for the next timestep using the properly selected solution model inside the solver for the current flow regime.

iii. To Display.(Figure 8)

(a) To Integrate (the current CFD scenario geometry & solution). The computed solution from the “Solve” phase is integrated with the current scenario geometry in a way that it is possible to be shown in later stages.

(b) To Show (the integrated CFD scenario geometry & solution). The integrated information from the previous stage is shown in a human understandable way, so the user can infer from it all the information he is supposed to.

iv. To Interact. (Figure 9)

(a) To Interpret (the simulation results). Given the simulation results, the information the user may abstract from is divided in 4 groups to easily modify

the current CFD scenario settings.

(b) To Modify 1 (the CFD scenario geometry). From the inferences made during the “Display” stage, the user is able to suggest a possible modification to the current scenario geometry.

(c) To Modify 2 (the CFD scenario physics). From the inferences made during the “Display” stage, the user is able to suggest a possible modification to the current physical conditions present on the CFD scenario.

(d) To Modify 3 (the CFD simulation state). To modify the state of the simulation (to stop it).

(e) To Integrate (the proposed geometry,physics and simulation state modifications). To integrate the modifications proposed by the user into a single set of User interaction flags.

(f) To Modify 4 (the “Display” phase solution display settings). To modify the display settings used on the future “Display” phase of the simulation.

v. To Control. (Figure 10)

(a) To Validate (the set of modifications) The user proposed interaction flags might enter in conflict (some geometrical changes might induce global physical changes as well and vice versa), during this phase the interaction flags validity is checked and hence are classified into valid or not valid.

(b) To Interprete (the invalid User interaction flags) If the interaction flags are invalid, they are to be divided into the basic groups that compose them (Geometry, Physics, state of the simulation)

(c) To Adapt 1 (the CFD scenario geometry modifications) Given the values of the proposed physic modifications of the scenario, adapt the geometric

modifications to guarantee the stability of the following process (e.g. given a substantial diminution of the timestep value, the meshing parameters probably will have to change).

(d) To Adapt 2 (the CFD scenario physics modifications)

(e) To Adapt 3 (the CFD simulation state) Validating a stopping signal.

(f) To Integrate (the CFD scenario geometry, physics and state modifications) Grouping again all the groups of modifications into a single set.

(g) To Convert (a valid set of modifications) Given a valid set of modifications, to convert it into the same format as the one used for user interaction flags, to be able to feed the "Rebuild" phase of the methodology.

2.6.2 Organ Domain Given the data transformation and processes stated on subsection 2.6.1, five function carriers related to the main functions are estimated to create the desired effects from the initial information:

i. Pre-Processor: The pre-processing organ is an entity capable of handling the complete geometry definition, geometry discretization, boundary conditions settings and the physical model selection. The pre-processor should be able to receive the data on a given format and present the results on a format readable by the solver. This organ must as well have the ability to modify all the desired initial CFD scenario conditions stated on Section 2.3.

ii. Solver: This organ is intended for solving the CFD scenario given some particular flow conditions. This organ should be able to read the CFD scenario given by the pre-processor and write the results in a format readable for the post-processor organ.

iii. Post-Processor: This organ must be able to read the CFD scenario geometry written by the pre-processor, the solution of the CFD scenario obtained by the solver and present them in a integrated and human understandable way.

iv. Interaction organ: This organ must allow the user to interact with the CFD scenario presented by the post-processor organ. The interaction with the post-processor is vital for the interaction process, clearly stated rules for the data interchange process must be set, and depend on both the implementation and the level of immersivity present.

v. Control Organ: This organ is intended to validate or adapt the suggested modifications to the CFD scenario. It must be able to understand the CFD scenario modifications data given by the interaction organ, transform it and present it to the pre-processor organ in a way that the cfd scenario modifications are viable and therefore the process may continue.

2.6.3 Part Domain The design parameters of each organ are to be set depending on the particular implementation of the methodology. An example of a the part domain for an interactive CFD application will be described on the following chapters, given the fact that after proposing this methodology as a guideline for future interactive CFD applications, the next step of this study is to build a basic Virtual Wind Tunnel (VWT) simulator that will be presented during the following chapters.

3 Virtual Wind Tunnel application architecture

3.1 Presentation.

In this chapter, the user will find how to install the current developments of the virtual wind tunnel project, where the components are located and how to compile them. This chapter consists of basic notes about Netgen (NG) installing and it's requirements, OpenFOAM (OF) installing and finally ParaVoxel (PV) integration.

The present state of the virtualWindTunnel (VWT) application is to be cataloged as an interactive Computer Fluid Dynamics (CFD) interactive solver for incompressible laminar Navier-Stokes equations using the PISO algorithm. It is an adaptation of the original OF icoFoam solver Wiki (2006); the code is inherently transient, requiring an initial condition (such as zero velocity) and boundary conditions.

The VWT depends on a set of open source packages to perform the different operations required to solve both the CFD problem and all related geometrical problems resulting from the interactivity level of the application. In the following sections, each of the applications will be briefly described at two levels: how to install it and how does it interacts with the VWT application.

As an important note, the reader should take into account that when in this guide a reference to "the object" is made, it is referring to the solid model that is to be analyzed inside the virtual wind tunnel.

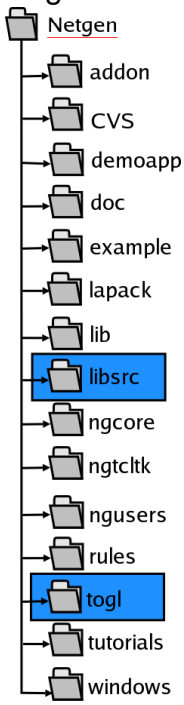
3.2 Netgen.

Inside the VWT application, NG is the one who deals with the proper acquisition of the object's boundary representation (B-REP) and in a later stage, the generation of a valid triangular surface mesh to be used as the inner object representation inside the CFD scenario.

The object's B-REP is to be acquired from a proper CAD file so the VWT application may work. Currently, valid ASCII Stereolitography (STL) and neutral files (IGS and STEP) are used for geometry acquisition, providing valid engineering tolerances for design process. The IGS and STEP file integration is done via the OpenCascade kernel with the NG source code. (See SCHOBERL (2006a) for information about this process).

The current version of NG source code used for this purpose is v4.4. Only two large parts of the source code are used in this application: the files related to the meshing and geometry acquisition algorithms (located inside the libscr directory) and the files related to the NG GUI widgets (located inside the togl directory, this files are maintained due to compilation issues). (See Figure 1).

Figure 1: Required portions of the Netgen code (Highlights)



This two directories of the NG source code are copied and translated into the root directory that contains all the VWT source code, as shown in Figure 2.

IMPORTANT: While compiling Netgen 4.4 source code, a proper installation of Tcl/Tk, OpenGl and Tix is needed. Beware that the availability of Tcl/Tk source code is a MUST to compile NG and its sources. See SCHOBERL (2006b),SCHOBERL (2006a),SCHOBERL (2005) and NG README file for more information.

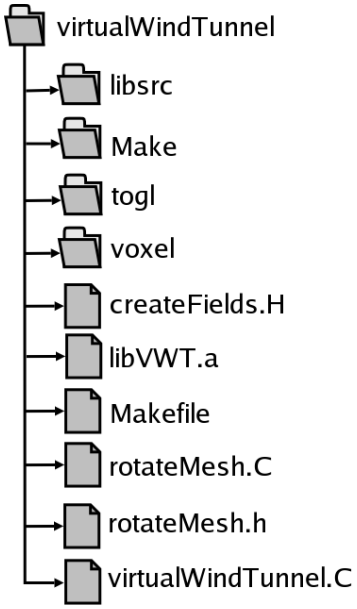
3.3 ParaVoxel

ParaVoxel is a third party Fixed Grid preprocessor, used inside the VWT application to discretize the CFD domain into Hexahedrons and Tetrahedrons. It is fully integrated with the VWT application and all its source code is located into the voxel

directory, inside the root directory that contains all the VWT source code, as shown in Figure 2.

There is not much information available about the ParaVoxel preprocessor. It's a meshing module-algorithm developed by Miguel Henao during a collaboration program between the University of Alberta and EAFIT. For more information see the comments inside the source code.

Figure 2: Final positioning of the unified source code of Netgen, ParaVoxel and VWT.



3.4 Current VWT data flow review.

The main idea of this application is to allow the user of the Virtual Wind Tunnel, to easily modify the geometrical orientation of the object who's aerodynamic behavior is studied. During the current phase of the project, given the fact that no Graphical User Interface is developed, the set of desired object orientations are to be input via a simple text file, later to be described in detail.

The flow of data inside the application on its current state can be described as shown in Figure 3. Four colors are present on the graphic:

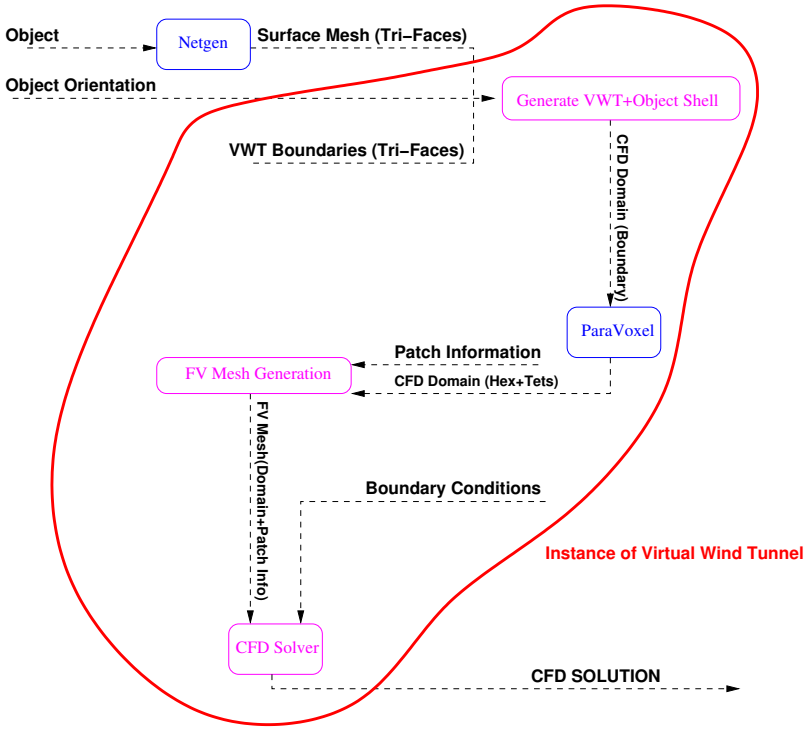
- The Red delimiter shows the abstract boundary for a single instance of solution of the VWT.
- The Blue delimiters show the instances where the external packages (NG and PV) take part into the solution process, mainly on meshing situations.
- The magenta delimiters refer to 3 stages of the solution process:
  - The integration of the geometries of VWT and the object to be analyzed.
  - The Finite volume mesh (fvMesh) generation and patch detection stage.
  - The solution stage of the CFD scenario.
- The black color identifies the different I/O data flows between stages.

It is important to remark that in Figure 3, the only data inputs required to solve a single instance of the VWT problem are the Object and the Object Orientation information, hence if the Object is to be the same, the next iteration of VWT has got only to deal from the generation of the boundary of the CFD domain.

3.5 Current VWT application compilation.

The root directory of the VWT application source code is currently to be located on the same directory where all the OF solvers for incompressible flow are positioned (icoFoam, boundaryFoam, nonNewtonianIcoFoam, etc...). This directory can be reached (after a proper OF installation and setting of system variables) at $FOAM_APP/solvers/incompressible.

Figure 3: Flow Diagram of a VWT instance



The main directory is called virtualWindTunnel, it contains a basic makefile (Makefile) that compiles the application's library (libVWT) that contains all the NG, PV and other objects of the VWT application, to be linked later with the other OF libraries using OpenFOAM's own wmake. All the required files for wmake compilation of the VWT application are located at:

$FOAM_APP/solvers/incompressible/virtualWindTunnel/Make

. For more information about how does wmake works, see WMA (2000-2006).

To compile the application, just type in the command line (while being in the $FOAM_APP/solvers directory) make. The application's executable will be placed on the same directory, under the name virtualWindTunnel.

3.6 Current VWT application code pointers and Hints.

The VWT source code lies mainly on 4 files, which contents and nature will be described as follows:

- virtualWindTunnel.C: This file contains the main routine of the application and it's the starting point of it. It contains all the "still in development" code and recent adaptations before being put on their respective endpoints.
- rotateMesh.C: This file contains all the functions that define and perform the rotation of any NG surface mesh, as well as the ones that merge NG's surface meshes. The functions defined here are used for both performing the rotation operations to the object (for properly defining it's orientation inside the VWT) and the definition of the boundary of the CFD domain.
- rotateMesh.H: Contains all the defines (Classes, constants and macros required in the rotateMesh.C file).
- icoFoamAdapt.H: Contains the portion of the icoFOAM code that given all the proper elements (Finite Volume Mesh (fvMesh), Boundary conditions and fluid definitions) carries on with the problem solving process. It was mainly detached from virtualWindTunnel.C given its similarity to icoFoam code during the development process.

The basic I/O procedures take place in the virtualWindTunnel.C file, and the hins to locate the lines corresponding to each, are:

- Object's BREP loading: The portion of the code that develops this stage invokes the function Ng_STL_LoadGeometry.

- Object's orientation definition: The orientation's definition is refereed to an instructions file, and it is loaded with the function readOperationsFile.
- Object's boundary mesh generation: Using NG, the triangular boundary mesh is defined invoking the set functions Ng_STL_InitSTLGeometry, Ng_STL_MakeEdges and Ng_STL_GenerateSurfaceMesh consecutively.
- VWT Boundary generation: it takes place when invoking the function genSimpleBoxForNetgen.

After performing the basic I/O, the procedures for each instance of the VWT take place in the lines:

- Object's mesh rotation and positioning: This stage is developed in the line that invokes the function rotateSet_quaternion.
- CFD Domain's Boundary generation: It takes place by merging the two Netgen Meshes; It happens in the line that contains the mergeMeshes function calling.
- CFD Domain's Discretization: This stage is developed by invoking the netgenToFoam_paraVoxel function. Still in development, an instance of the "fvMesh" class called "Mesh" has got to be the output of this function, so the icoFoamAdapt.H include has all the proper inputs.
- CFD problem solving: look for the inclusion of icoFoamAdapt.H.

The CFD Domain's Discretization stage is still under heavy development. Finding ways for fast volume-meshing and FOAM patch detection features are the current needs of the project. See the netgenToFoam_paraVoxel function for more information on the still-in-development code.

3.6.1 OPENFOAM patch detection over unstructured VWT meshes. During the construction of a fvMesh, the user must supply to OF a valid list of indexed boundary faces. Having explicit information about those boundary faces (Do they belong to the boundary with the object? or, Do they belong to the VWT boundary?), given the data inputs provided by ParaVoxel and the original I/O data is still an unknown.

After performing the generation of the *fvMesh* given the cell data (vertex, cell types and connectivity), OF catalogues all the un-neighbored faces of each cell by default as defaultFaces, and storages each face index into a list. As a post-condition, all the boundary faces of the VWT are stored into a single list, which only has information about indexes.

Extracting the boundary of the VWT is still insufficient to perform the CFD analysis: It is still needed information about where to establish the boundary conditions on the geometry. OF uses a figure, called patch to map the desired boundary conditions over grouped set of faces.

Given an OF polyMesh (or a fvMesh), automatically sort the set of boundary faces belonging to the VWT's CFD domain boundary into a set of patches, composed by:

- Object Boundary: This patch is composed by all the boundary faces that are in contact with the object of study.
- Inlet: This patch is composed by all the faces that represent the fluid inlet to the VWT; It's suggested to use one of the X-Axis orthogonal VWT set of faces.
- Outlet: This patch is composed by all the faces that represent the fluid outlet of the VWT; It's suggested to use the inlet's opposite X-Axis orthogonal VWT set of faces.

- Tunnel Walls: This patch is composed by all the faces that are neither Inlet, Outlet or Object Boundary.

Filtering the Inlet, Outlet and Tunnel Walls patches out of the unsorted set of boundary faces will leave only the Object Boundary patch inside that list. Information about the topological relationships and geometrical positioning of the Inlet,Outlet and Tunnel Walls patches is available since the beginning (the set of patches defined by the union of this three sets, corresponds to the bounding box of the CFD domain).

3.6.2 Boundary condition setting over unstructured VWT meshes. After filtering the OF patches, boundary conditions are to be set over them. The correspondant boundary conditions, as defined by OF, should be as follows:

i. Object Boundary: Fixed Walls.

ii. Inlet: Inlet (Velocity has got to be set).

iii. Outlet: Outlet (Fixed Pressure value).

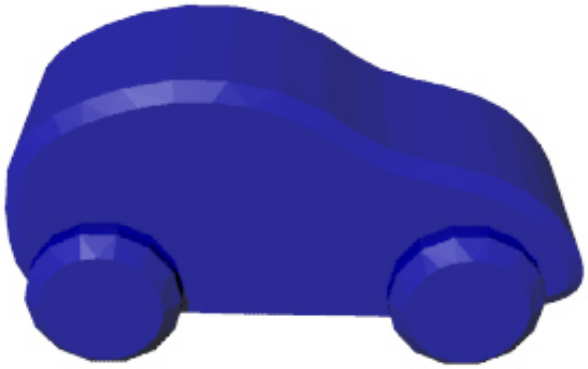
iv. Tunnel Walls: Fixed Walls.

4 Examples and Results

A single CFD scenario will be used for comparative tests in order to prove the stability and validate the methodology developed in the present work (Virtual Wind Tunnel).

For testing purposes, the geometry of a toy car modeled in a standard CAD software and later exported to STL format will be used as the test specimen inside the virtual wind tunnel. The model consists of 812 triangular facets (See Figure 1).

Figure 1: STL Model of Toy Car.



The object was positioned inside a rectangular box that simulates the boundaries of the VWT and sets the boundaries of the CFD domain. For boundary conditions, a single Inlet normal to the X axis of the world coordinate system (WCS) is set, and as inlet will be used the geometrically opposed face. The remaining faces of the box and the ones belonging to the object will be set as no-slip wall conditions. See Figure 2 for a graphical display of the CFD scenario. A summary of the boundary conditions applied can be seen on Table 4.1.

Figure 2: Boundary Conditions over the CFD Domain

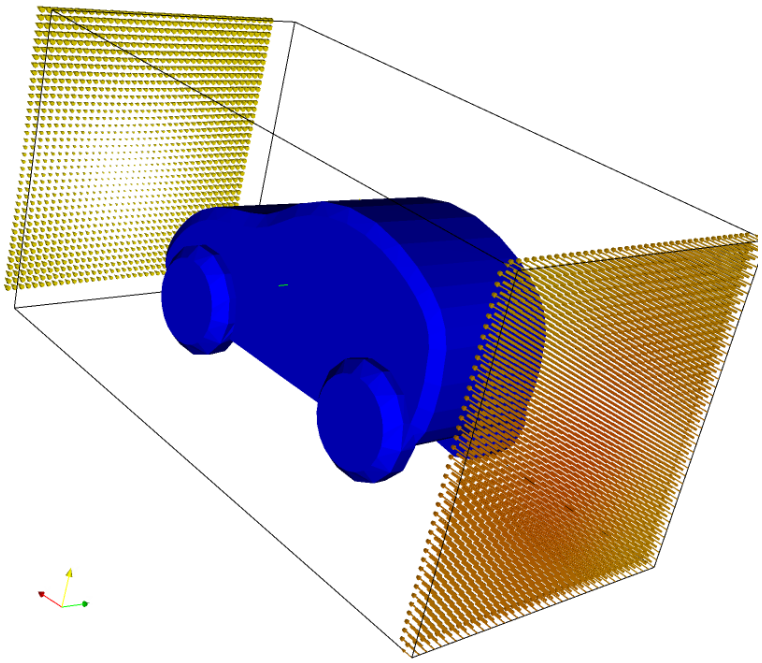


Table 4.1: Boundary conditions summary

Region	Boundary Condition
Inlet	U=5 un/sec
Outlet	P=0
Outer walls	U=0
Object	U=0

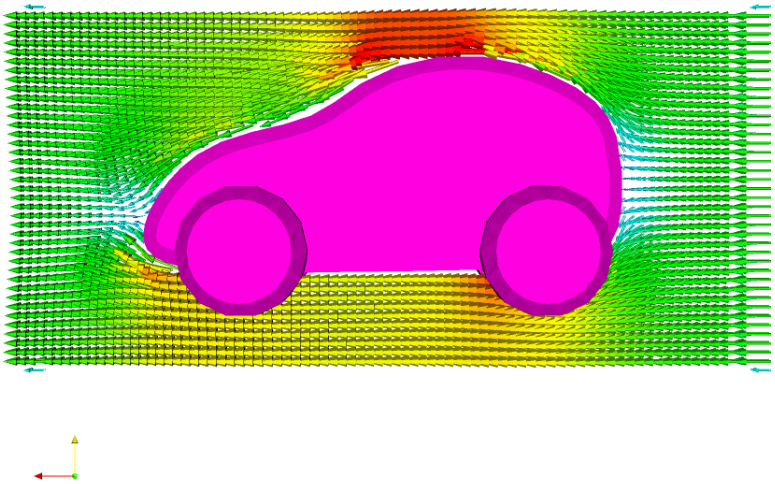
Three different tests will be performed using the same CFD scenario. Each test will have a constant mesh, and either the object orientation or the simulation timestep will be interactively controlled. Test 1 and test 3 use the same timestep and rotation angles (across the same axis), but the mesh in Test 1 is coarser than the one used for Test 3 (See Table 4.2 and Table 4.6 for detailed information).

4.1 Test 1

Test 1 is intended to prove the stability of the software-methodology during a simulation in case a *geometrical* condition is changed. During this test, consecutive rotations along the X axis were performed over the object after 2.25 seconds of simulated time, with angle-steps of 15° between them.

This test was performed on a single core Pentium IV machine @ 2.80GHz with 512 kB cache size, 1GB RAM and 128 MB nVidia Geforce fx 5500 video card.

Figure 3: Results example for Test 1



The Table 4.2 contains all the fixed grid meshing related data and results. The reader should take into account that given the fact the mesher implemented uses fixed grid algorithms, the meshing times are slightly the same, no matter the rotation performed

over the object.

The Table 4.3 groups the state of some indicators that give an idea of the state of the simulation. The reader should check that no matter the rotation performed, the Courant numbers (Mean and Maximum) stay into acceptable ranges, with slight variations between them.

Table 4.2: Test 1 Meshing times and information

Degrees	0	15	30	45
Netgen Mesh Reading time	0.035506	0.038138	0.037565	0.035298
Facet-ray classification time	0.053609	0.057974	0.055696	0.051179
Node classification time	0.100273	0.095389	0.098756	0.095456
Element classification time	0.012593	0.011137	0.011853	0.011535
dt	0.250	0.250	0.250	0.250
FX	80	80	80	80
FY	40	40	40	40
FZ	40	40	40	40
Mesh Writing time	6	6	6	6

Table 4.3: Test 1 Stability Indicators

Degrees	Time	Mean Courant	Max. Courant	Exec Time	Delta
0	0.25	0.000000	0.250000	15.92	
	0.50	0.056469	0.269171	24.31	8.39
	0.75	0.056475	0.268952	32.12	7.81
	1.00	0.056481	0.268701	39.58	7.46
	1.25	0.056489	0.268435	46.92	7.34
	1.50	0.056497	0.268163	54.18	7.26
	1.75	0.267888	0.056506	60.49	6.31
	2.00	0.056516	0.267613	67.41	6.92
	2.25	0.056528	0.267339	74.52	7.11
15	2.50	0.054928	0.267092	16.35	
	2.75	0.057805	0.269435	24.03	7.68
	3.00	0.057812	0.269172	30.96	6.93
	3.25	0.057824	0.269481	38.38	7.42
	3.50	0.057837	0.269719	44.59	6.21
	3.75	0.057851	0.269992	51.89	7.30
	4.00	0.057867	0.270064	58.17	6.28
	4.25	0.057883	0.269820	64.26	6.09
	4.50	0.057900	0.269246	69.68	5.42
30	4.75	0.056207	0.267403	15.57	
	5.00	0.058952	0.281698	22.97	7.40
	5.25	0.058963	0.281633	29.60	6.63
	5.50	0.058980	0.281294	35.44	5.84
	5.75	0.058998	0.281281	41.30	5.86
	6.00	0.059010	0.281631	46.94	5.64
	6.25	0.059042	0.282287	52.58	5.64
	6.50	0.059065	0.283128	58.31	5.73
	6.75	0.059089	0.284085	63.44	5.13
45	7.00	0.057446	0.274999	15.14	
	7.25	0.059962	0.317584	22.63	7.49
	7.50	0.059980	0.323540	28.75	6.12
	7.75	0.060003	0.328442	34.53	5.78
	8.00	0.060030	0.332648	40.38	5.85
	8.25	0.060058	0.335957	45.95	5.57
	8.50	0.060088	0.338278	51.51	5.56
	8.75	0.060119	0.339767	56.67	5.16
	9.00	0.060152	0.340469	61.61	4.94

4.2 Test 2

Test 2 is intended to prove the stability of the software-methodology during a simulation in case a physical condition is changed, in this case the simulation timestep. During this test, the position of the object is the same for all timesteps, but subsequently the timesteps will be changed after short periods to test the stability of the application.

This test was performed on a single core Pentium IV machine @ 2.80GHz with 512 kB cache size, 1GB RAM and 128 MB nVidia Geforce fx 5500 video card.

The Table 4.4, contains all the fixed grid meshing related data and results. Table 4.5 groups the state of some indicators that give an idea of the state of the simulation.

Table 4.4: Test 2 Meshing times and information with dT=0.25 s.

Netgen Mesh Reading time:	0.029355
Facet-ray classification time:	0.072997
Node classification time:	0.119648
Element classification time:	0.010811
DT	0.25
FX	80
FY	40
FZ	40
Mesh Writing time	6

Table 4.5: Test 2 Stability Indicators

dt	Time (s)	Mean Courant	Max. Courant	Exec. Time (s)	Calculation Time (s)
0.250000	0.25	0.000000	0.250000	14.32	
	0.50	0.056469	0.269171	21.02	6.70
	0.75	0.056475	0.268952	27.29	6.27
	1.00	0.056481	0.268701	33.31	6.02
	1.25	0.056489	0.268435	39.25	5.94
	1.50	0.056497	0.268163	45.41	6.16
0.2	1.70	0.045073	0.214350	15.19	
	1.90	0.045208	0.214397	21.65	6.46
	2.10	0.045215	0.214203	27.50	5.85
	2.30	0.045222	0.214012	33.25	5.75
	2.50	0.045230	0.213823	38.70	5.45
	2.70	0.045237	0.213638	44.26	5.56
	2.90	0.045246	0.213463	49.82	5.56
	3.10	0.045254	0.213292	54.93	5.11
	3.30	0.045263	0.213122	59.99	5.06
	3.50	0.045272	0.212955	64.85	4.86
0.1	3.60	0.022633	0.106388	14.58	
	3.70	0.022645	0.106367	22.59	8.01
	3.80	0.022647	0.106327	28.47	5.88
	3.90	0.022649	0.106289	33.07	4.60
	4.00	0.022652	0.106250	37.67	4.60
	4.10	0.022654	0.106212	42.60	4.93
	4.20	0.022656	0.106175	47.26	4.66
	4.30	0.022658	0.106137	52.21	4.95
	4.40	0.022660	0.106100	56.87	4.66
	4.50	0.022663	0.106064	61.50	4.63
	4.60	0.022665	0.106027	65.86	4.36
	4.70	0.022667	0.105991	70.52	4.66
	4.80	0.022669	0.105956	74.88	4.36
	4.90	0.022672	0.105920	79.25	4.37
	5.00	0.105885	0.022674	83.58	4.33
0.5	5.50	0.113348	0.529226	14.85	
	6.00	0.113434	0.528496	21.24	6.39
	6.50	0.113487	0.527716	26.51	5.27
	7.00	0.113538	0.526963	32.44	5.93
	7.50	0.113588	0.526243	37.76	5.32
	8.00	0.113639	0.525559	43.58	5.82
	8.50	0.113691	0.524913	49.49	5.91
	9.00	0.113743	0.524304	55.33	5.84
	9.50	0.113794	0.523731	61.55	6.22
0.05	9.55	0.011383	0.052315	14.50	
	9.60	0.011387	0.052320	20.71	6.21
	9.65	0.011387	0.052312	26.53	5.82
	9.70	0.011388	0.052306	32.10	5.57
	9.75	0.011388	0.052300	37.02	4.92
	9.80	0.011389	0.052294	42.06	5.04
	9.85	0.011389	0.052289	46.77	4.71

4.3 Test 3

Test 3 is basically the same as Test 1 but instead of using a Fixed Grid of 80x40x40 elements, a more refined mesh of 80x80x80 elements was be used. During this test, rotation operations along the X axis were performed over the object after one second of simulated time, with anglesteps of 15° between them.

This test was performed on a single core Pentium IV machine @ 2.80GHz with 512 kB cache size, 1GB RAM and 128 MB nVidia Geforce fx 5500 video card.

The Table 4.6 contains all the fixed grid meshing related data and results. The reader should take into account that given the fact the mesher implemented uses fixed grid algorithms, the meshing times are slightly the same, no matter the rotation performed over the object.

The Table 4.7 groups the state of some indicators that give an idea of the state of the simulation. The reader should check that no matter the rotation performed (and compared to Test 1, using a finer grid), the Courant numbers (Mean and Maximum) stay into acceptable ranges, with slight variations between them.

Table 4.6: Test 3 Meshing times and information

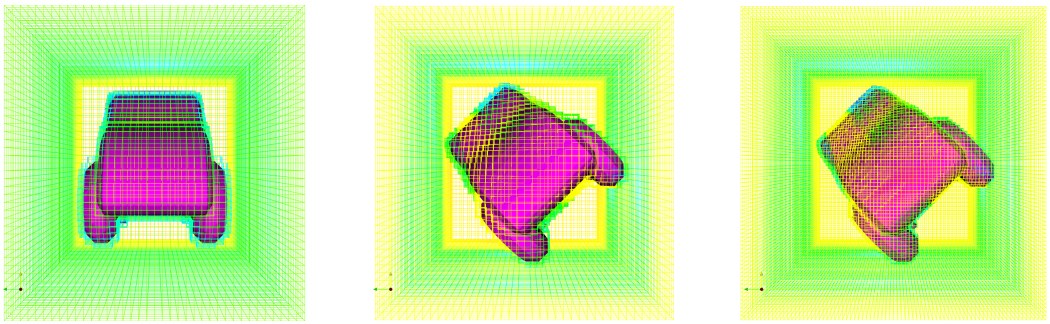
Degrees	0	15	30	45
Netgen Mesh Reading time	0.034898	0.036115	0.035067	0.034809
Facet-ray classification time	0.096860	0.101890	0.098497	0.096012
Node classification time	0.231206	0.236420	0.242625	0.244444
Element classification time	0.045652	0.043130	0.047725	0.047323
DT	0.250	0.250	0.250	0.250
FX	80	80	80	80
FY	80	80	80	80
FZ	80	80	80	80
Mesh Writing time	24	24	23	23

Table 4.7: Test 3 Indicators

Degrees	Time (s)	Mean Courant	Max. Courant	Exec. Time (s)	Calculation Time (s)
0	0.25	0.000000	0.250000	68.68	
	0.50	0.042542	0.330600	113.43	44.75
	0.75	0.042555	0.331445	154.30	40.87
	1.00	0.042569	0.333884	193.13	38.83
15	1.25	0.000000	0.250000	70.21	
	1.50	0.042943	0.320238	113.87	43.66
	1.75	0.042956	0.320942	152.42	38.55
	2.00	0.042969	0.322796	189.46	37.04
30	2.25	0.041541	0.300704	77.43	
	2.50	0.045148	0.339218	121.86	44.43
	2.75	0.045164	0.341569	157.91	36.05
	3.00	0.045185	0.341682	190.81	32.90
45	2.25	0.043735	0.327505	78.53	
	2.50	0.046945	0.353855	123.72	45.19
	2.75	0.046968	0.353989	162.02	38.30
	3.00	0.046998	0.352301	195.88	33.86

4.4 Results analysis

No matter the rotation operations performed on Tests 1 and 3 (See Figure 4, the Courant numbers stayed inside acceptable levels (< 0.7) during all the simulation stages. The stability was never compromised, and as the Figure 5 and Figure 6 show, after the initial numerical shock induced by the geometry modifications, the solution quickly converged to similar computation times.



(a) Original Orientation. (b) Test 1 after performing 45° rotation (c) Test 3 after performing 45° rotation

Figure 4: Example of rotation operations

Figure 5: Test 1 Results

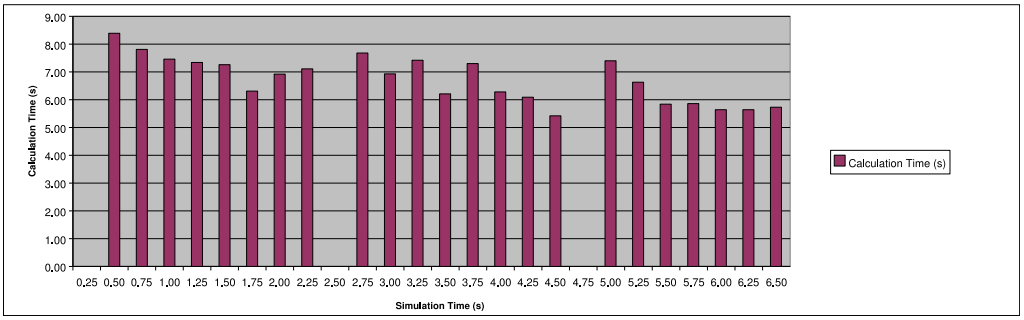
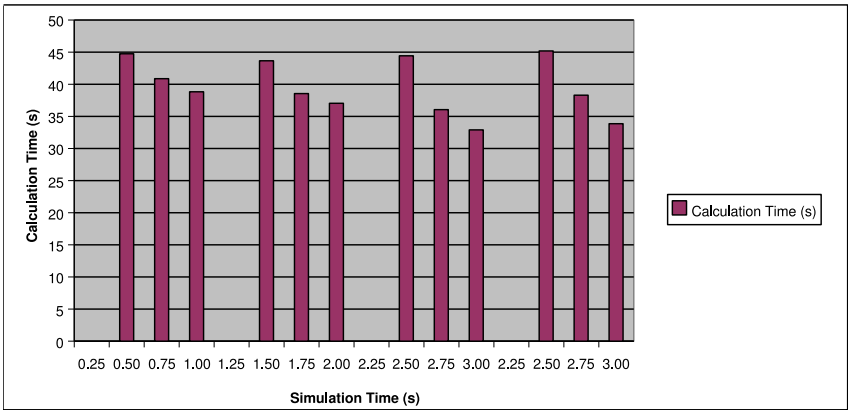
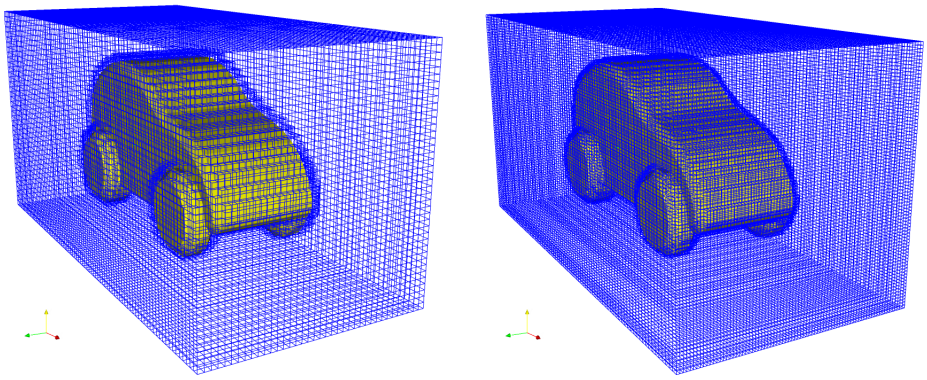


Figure 6: Test 3 Results



After comparing Test 1 and Test 3 results (Figure 5 and Figure 6), it is evident to conclude that for the same boundary conditions, the same fluid flow model and timestep, the coarser the fixed grid (See Figure 7) mesh, the faster the simulation will perform. The drawback of coarser meshes is the possible loss of solution accuracy given the actual defeaturing produced by the fixed grid representation. The changes on the density of the grid did not compromise the stability of the application/methodology and the geometrical changes were successfully performed during runtime as the methodology proposed.

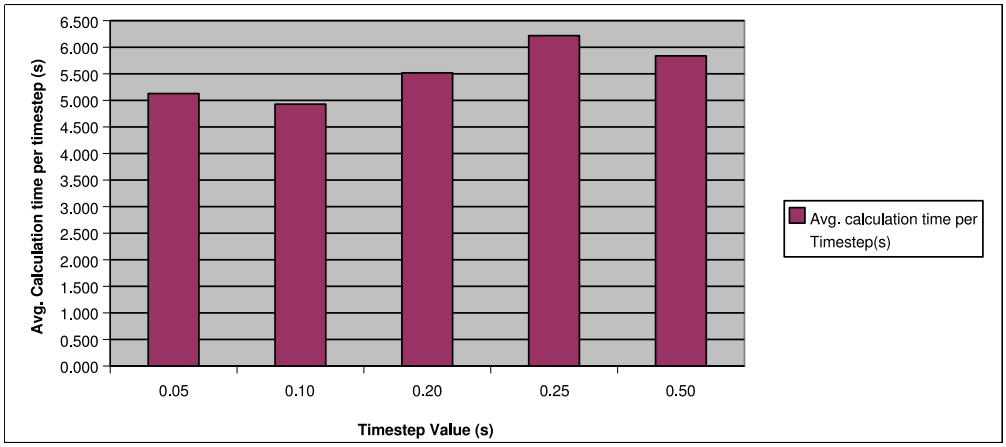


(a) Test 1 Initial Mesh, FG=(80,40,40) (b) Test 3 Initial Mesh, FG=(80,80,80)

Figure 7: Different mesh densities for the same CFD scenario

The test 2 results show that for a same mesh and orientation, changing the simulation timesteps slightly influences the computation times between calculations with almost no effect over the convergence. In case a simulation of long-lasting phenomenons is desired, the user should use bigger timesteps to cover more ground with less calculations. Automatic timestep calculations for a simulation are possible, relating the velocity, cell width and the Courant number.

Figure 8: Test 2 Results



5 Conclusions and future work

5.1 Conclusions

A new methodology for the solution of interactive CFD simulations is presented on this work. The CFD methodologies used in this project for solving a CFD scenario are based on discretization methods for the solution of the partial differential equations.

The solvers used during the testing of the developed methodology are based on the finite volume theory. It was observed that the quality of the results of this work is directly related with the quality of the mesh used for the calculations, hence the meshing stage of any interactive or non interactive CFD simulation is critical to achieve a good performance of the solvers. The use of fixed grid meshing algorithms provides reduced meshing times compared to delone algorithms, with the drawback of possible defeaturing of the part that is being meshed.

Interactive CFD simulations are proven possible with the presented methodology, being the methodology independent from the level of immersivity and the implementation.

Some of the present problems (oportunities?) for interactive CFD simulations are:

The actual hard disk writing capabilities present on low-end computers. Writing the fixed grid mesh on disk takes much more time than calculating it, and for display purposes and later review of the results it is necessary to write it. This is one of the bottlenecks present on the current VWT implementation.

The speed of meshing algorithms is still far from the one required for real time implementations. Delone triangulation based algorithms present a more accurate rep-

resentation of the CFD domain with the drawback of a difficult mesh tunning for CFD simulations. Fixed grid algorithms present faster meshing times and meshes that behave well in finite volume solvers, with the drawback of a possible part defeaturing, hence diminishing the compatibility of the simulation results and the real phenomena. In some cases, where preliminary shape design tests are needed, defeaturing may present an advantage for both speed and for providing hints about what the shape might become.

The neutral formats (IGS, STEP) even though functional, are still interpreted in different ways by various CAD kernels. Geometry inputs may vary from CAD to CAD depending on the way a neutral format is read.

Complex turbulence models are available but are still very user dependant. Too much initial information is needed, making an interactive simulation still hard to accomplish. New parameters of different natures appear with every different model, hence the information the user must supply or interact with is much more complex given the fact that slight variations on the parameters may induce big changes on the simulation results.

Given the nature of the data obtained from the simulation results to be displayed, powerfull Graphical Processing Units (GPU'S) are needed. The better the GPU, the better the performance for the display modules, hence guaranteeing the diminishing of displaying times and inducing a faster user response.

5.2 Future Work

The VWT implementation after proven possible with a single core CPU, is going to be improved by adding an interactive-collaborative module and parallel computing

technics. The main goal of those future improvements is to achieve a soft-real time quality during the simulation and in future stages a possible hard real time quality.

“Solo el pensamiento vivido tiene valor”.

-Demian, Hermann Hesse.

Bibliography

OpenFOAM 1.3 User Guide. OpenCFD Ltd, 2000-2006. 3.2 Compiling applications and libraries, Available with OpenFOAM.

ANDREASEN, Mogens Myrup. Machine design methods based on a systematic approach - contribution to design theory, 1980. Dissertation, Department of machine design, Lund Institute of Technology, Sweden.

ATLURI, Satya N. Methods of computer modeling in engineering the sciences tomo 1.

BARKER, P. Design and Production of Multimedia and Simulation-based Learning Material 1 ed. Kluwer Academic Publishers, 1994.

BARRET, R. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods 2 ed. SIAM, Philadelphia, PA, 1994.

BARTHELEMY, B.; HAFTKA, R.; MADAPURQ, U. y SANKARANARAYANAN, S. Integrated structural analysis and design using three-dimensional finite elements. En : American Institute of Aeronautics and Astronautics tomo 29, no 5, p. 791–797, 1991.

BRITANNICA@, ENCICLOPEDIA. Wind Tunnels. Web, 2006. Available at: `http://www.britannica.com/ebi/article-9277765`.

BRYSON, Steve y LEVIT, Creon. The Virtual Wind Tunnel. tomo 4. 1992.

C., ZIENKIEWICZ O. y TAYLOR, R. L. Finite Element Method: Volume 3, Fluid Dynamics 5 ed. tomo 3. Butterworth-Heinemann, 2000.

CHAM@, limited. CHAM's Virtual Wind Tunnel. Web, 2005. Http://www.cham.co.uk/phoenicsvwt/autovwt.htm.

FERZIGER, J.H. y PERIC, M. Computational Methods for Fluid Dynamics 3 ed. Springer, 2002.

FLUENT INC.@, . Parallel Processing. Web, 2006. Available at: http://www.fluent.com/software/fidap/parallel.htm.

FUEL TECH INC.@, . ACUITIV Software post-processor for CFD. Web, 2004. Available at: http://www.acuitiv.com/.

GARCIA, Manuel. Fixed Grid Finite Element Analysis in Structural Design and Optimisation. Tesis Doctoral Department of Aeronautical Engineering, The University of Sydney, 1999.

————. Virtual Wind Tunnel Talk. University of Alberta, 2005. Talk conducted to professors and students at the University of Alberta.

HANSEN, Claus Thorp. y ANDREASEN, Mogens Myrup. Engineering Design Synthesis 1 ed. Springer-Verlag, 2002. Chapter 6, Two approaches to synthesis based on the domain theory.

————. On the content of a product idea. 2005. Talk conducted to professors and students at Melbourne University.

JARAMILLO, Juan David; VIDAL, Antonio M. y CORREA, Francisco Jose. Metodos directos para la solucion de sistemas de ecuaciones lineales simetricos, indefinidos, edispersos y de gran dimension. Investigation Booklet, DOCUMENT 40-022006, 2006. Chapter 5, Parallel Computing.

JASWAL, Vijendra. CAVEvis: Distributed Real-Time Visualization of Time-Varing Scalar and Vector Fields Using the CAVE Virtual Reality theater. IEEE Visualization Conference, 1997. Proceedings of the 8th IEEE Visualization Conference, Phoenix, AZ. 301–308.

JONASSEN, D. Instructional Designs for Microcomputer Courseware, 1988. Hillsdale, NJ: Lawrence Erlbaum.

KATZ-HAAS, Rasa. Ten Guidelines for User-Centered Web Design. En : Usability Interface tomo 5, no 1, 1998.

KATZ-HAAS@, Rasa. User-Centered Design and Web Development. Web, 1999. Available at: http://www.stcsig.org/usability/topics/articles/ucd\%20_web_devel.html.

KELA, A.; PERUCCHIO, R. y VOELCKER, H. Towards an automatic finite element analyis. En : Computers in Mechanical Engineering p. 55–71, 1986.

KIKUCHI, Noboru. Finite Element Methods in Mechanics 1 ed. Cambridge University Press, 1986.

MACHINE DESIGN, Magazine y MEDIA@, PENTON. A better mesher for FEA. Web, 2005. Available at: http://www.machinedesign.com/ASP/viewSelectedArticle.asp?strArticleId=58645&strSite=MDSite&catId=0.

OH@, David. The Java Virtual Wind Tunnel -A Two Dimensional Computational Fluid Dynamics Simulation-. Web, 2001. Http://raphael.mit.edu/Java/.

RHODES, D.M. y AZBELL, J.W. Designing interactive video instruction professionally. En : Training and Development Journal tomo 39, no 12, p. 31–33, 1985.

ROOZENBURG, N. y EEKELS, J. Product Design: Fundamentals and Methods 1 ed. John Wiley and Sons Inc, 1995. Chapter 4, What is Design?

SALEH, Jamal. Fluid flow handbook 1 ed. McGraw-Hill, 2002 530 - 554 p.

SCHOBERL, Joachim. NETGEN 4.4 User Guide. Available with Netgen Distribution, 2005.

————. NETGEN - automatic mesh generator. Web, 2006a. http://www.hpfem.jku.at/netgen/.

————. NETGEN - troubleshooting. Web, 2006b. http://www.hpfem.jku.at/netgen/.

SCHWIER, E., R.A.and MISANCHUK. Interactive Multimedia Instruction, 1993. Nglewood CLiffs, NJ: Educational Technology Publications.

SHAW, Christopher D.; HALL, James A.; EBERT, David S. y ROBERTS, D. Aaron. Interactive lens visualization techniques. p. 155–160. IEEE Visualization Conference, 1999.

SIMS@, Roderick. Interactivity: A Forgotten Art? Web, 1997. Available at: http://intro.base.org/docs/interact/.

STREETER, Victor L. Fluid Mechanics 4 ed. McGraw-Hill Book Company, 1966.

STRUMOLO, Gary S. The virtual aerodynamic/aeroacoustic wind tunnel. En : Journal of Engineering Mathematics VAWT tomo 43, no 2-4, p. 173–187, 2002.

SVANAES, Dag. Understanding Interactivity: Steps to a Phenomenology of Human-Computer Interaction, 2000. Monograph,ISBN 82-471-5201-0.

Wiki, OpenFOAM. icoFoam. Web, 2006. http://gcdw05.unileoben.ac.at/non_cdl/OpenFOAMWiki/index.php/IcoFoam.

WIKIPEDIA@, The free encyclopedia. Boundary element method. Web, 2006a. Available at: http://en.wikipedia.org/wiki/Boundary_Element_Method.

————. Computational fluid dynamics. Web, 2006b. http://en.wikipedia.org/wiki/Computational_fluid_dynamics.

————. Finite element method. Web, 2006c. Available at: http://en.wikipedia.org/wiki/FEM.

————. Human-Computer Interaction. Web, 2006d. http://en.wikipedia.org/wiki/Human-computer_interaction.

————. Parallel computing. Web, 2006e. Available at: http://en.wikipedia.org/wiki/Parallel_computing.

————. Real time programming. Web, 2006f. Available at: http://en.wikipedia.org/wiki/Real-time.

————. User-centered design. Web, 2006g. http://en.wikipedia.org/wiki/User-centered_design.