Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXIV, págs. 1245-1258 (artículo completo)
Sebastián Giusti, Martín Pucheta y Mario Storti (Eds.)
Córdoba, 8-11 Noviembre 2016

AERODYNAMIC FLIGTH COEFFICIENTS USING PARALLEL COMPUTING TECHNIQUES

Nicolas Trivisonno, Luciano Garelli, Mario Storti and Gustavo R. Rodriguez

CIMEC Centro de Investigación de Métodos Computacionales, UNL, CONICET, FICH, Col. Ruta 168 s/n, Predio Conicet "Dr Alberto Cassano", 3000 Santa Fe, Argentina, nicolas.trivisonno@gmail.com, http://www.cimec.org.ar

Keywords: Aerodynamics coefficients, Computing Fluid Dynamics, Finite Volume Method, Code Saturne, Exterior Ballistics, HPC, Parallel Computing, MPI, OpenMP.

Abstract. The need of ensuring the trajectory of a projectile so as to achieve a target has always been a challenge for exterior ballistic engineers. One of the most suitable techniques consists in simulate the projectile at different angles of attack with Computational Fluid Dynamic (CFD) software to obtain the corresponding aerodynamic coefficients. This paper aims at showing the advantages of utilizing parallel computing for obtaining the aerodynamic coefficients utilizing the CFD software Code Saturne. As a first step, a benchmark and a mesh convergence analysis is carried out to validate employed methodology. Then, several simulations are performed with a mesh convergence analysis in order to obtain the aerodynamic coefficients. Also, a scalability analysis is performed in a Beowulf cluster to measure the performance of the software.

1 INTRODUCTION

The main purpose of the external ballistics is to predict the trajectory of a projectile so as to achieve a target. Even though several factors are involved in the flight of the projectile, wind load is dominant among all the loads exerted on the projectile, indeed the accuracy calculation of it, is critical. Such aerodynamic data have traditionally been provided by wind tunnels testing or flight testing; however, these techniques can be very expensive as well as limited. Computing Fluid Dynamics (CFD) has arisen in the last decades providing a way to supplement wind tunnels and flight test data due to its simplicity and low cost. Nevertheless, a huge complexity and also a high computing resource are required due to the extension of CFD problems.

Fortunately, nowadays improved computer technology and the state-of-the-art numerical procedures enable solutions to complex problems. A complex CFD problem needs to be tackled and the most profitable way to tackle it is to use a divide and conquer strategy; where computers unite their efforts to stand up to the challenge. This is the main idea of parallel computing which is defined as a set of processors that are able to work cooperatively to solve a computation problem. This formal definition holds a lot of intricacies inside, such as the program has to have instructions to guide it to run in parallel, as well as, issues related to data structure, processor to processor communication, parallel efficiency, and assessment of run time improvement must be taken under consideration.

2 BENCHMARK

Our scope on this paper was concerned about obtaining the aerodynamics coefficients of a rotating projectile. Due to the complexity of this calculation, High Performance Computing techniques had to be employed. Furthermore the benefits of utilizing parallel computing for this calculation were assessed.
In order to further validate the algorithms developed to calculate the drag coefficients and the respective forces and moments, a benchmark study was performed for four different flow regimes that were compared to different simulations. Furthermore the efficiency of the cluster utilized was assessed by performing an scalability analysis.

2.1 Computational Grid

Structural meshes were created using Salome [1] available grid generation software. The meshes were constructed by creating a sphere of $1\ [\mathrm{m}]$ of diameter at the origin and then enclosing it in a larger sized cube which length was $10 \times 8 \times 8\ [\mathrm{m}]$. A small cube was then inserted inside the sphere and then six hexagons were created by connecting the corner points of the inner cube to the corner points of the outer cube. All redundant geometrical entities were then deleted and only the sphere face retained, see Figure 1.

These six hexagons split the sphere in faces, and each edge generated on the surface of the sphere was split in 50 elements, generating elements of $12.3\ [\mathrm{mm}]$ length that have an equidistant distribution. Also the segments that connect the inner cube with the control volume, were split in segments whose length changes in a geometric progression $Lk = Lk - 1 \cdot d$. The progression started with a segment length of $15\ [\mathrm{mm}]$ and further segments having an expansion ratio of 1.08, Figure 1.

This methodology employed was in order to avoid the configuration of a boundary layer. Several meshes were developed varying its configuration so as perform the simulations with

[1] GNU LGLP License

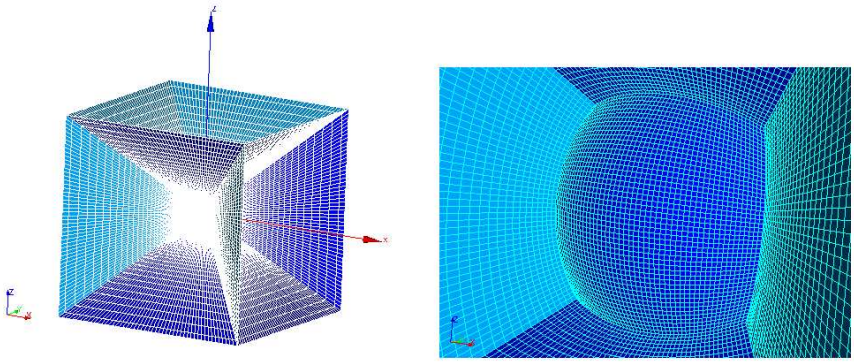

Figure 1: Mesh on the sphere

different mesh sizes, which are detailed in the table below, Table 2.1:

Description	Number of elements [elem]	Number of elements on the segment of the sphere [elem]	Geometric Progression Initial Length [mm]	Ratio
Mesh 4	905,248	50	15	1.080
Mesh 5	3,505,480	60	7.5	1.025
Mesh 6	2,505,832	60	2.5	1.050
Mesh 7	1,524,768	55	2.5	1.075

Table 1: Description of different meshes employed in the sphere

2.2 Results of the Benchmark

The simulations were performed for a laminar flow at *Re=300*, turbulent flow with laminar boundary layers at *Re=10e4* and turbulent flow with turbulent boundary layer at *=10e6*. These simulations were run utilizing CodeSaturne[2] finite volume approach software package.

The results of the simulations performed were found to be in excellent agreement, what means that the algorithms employed to calculate the coefficients, forces and moments were perfectly validated.

2.2.1 Laminar flow at Reynolds 300

The simulations were performed for a laminar flow, whose inlet velocity was 0.0147 $[\mathrm{m/seg}]$ and the time step was 2.5 $[\mathrm{seg}]$ for the mesh 4 and 1 $[\mathrm{seg}]$ for the mesh 5. In order to set the conditions of this regimen and also the following, the Reynolds number was adapted by varying its viscosity, due to the fact that if the velocity of the flow is varied, then the Courant number will be affected.

$$Re = \frac{v \cdot \phi}{\nu} \tag{1}$$

[2]GNU GLP Licence

$$C = \frac{\Delta t \cdot v}{\Delta x} \tag{2}$$

It was found that a considerable period of time was required to elapse before the motion settle into a regular periodic pattern. Figure 2 show the plot of the drag coefficient as a function of time between 3300 and 4100 [seg]. The mean drag coefficient (Cd) is 0.665 for the mesh 4, which was in agreement with Giacobello (2005) having a discrepancy of 1.05%. Moreover Kim et al. (2001) has obtained 0.657, Johnson and Patel (1999) and Constantinescu and Squires (2003) 0.656 and finally Jones and Clarke (2008) 0.661.

It should be note that Johnson and Patel (1999) and Jones and Clarke (2008) used a very similar meshing scheme.

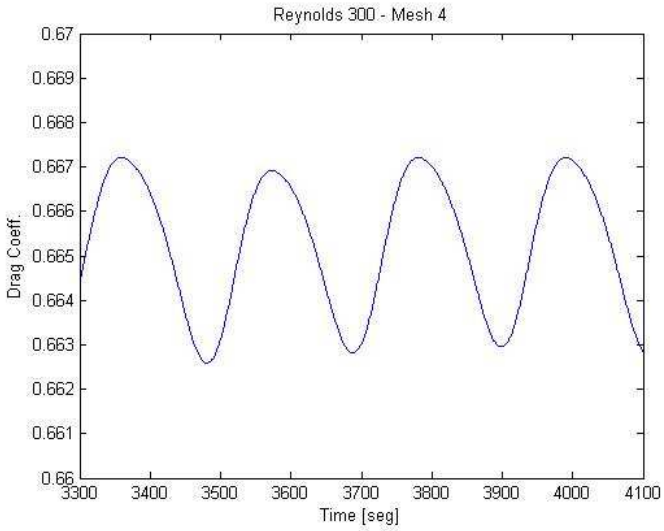

Figure 2: Periodic patter of the Drag Coefficient

Furthermore, the Strouhal number calculated from Figure 2 was 0.136, exactly the same value that was obtained by Constantinescu and Squires (2003). While Giacobello (2005) calculated a value of 0.134 the same as Kim et al. (2001).

The periodicity of the flow *Re=300* is due to the regular shedding of vorticity. As described by Thompson and Le Gal (2004), the wake consist of a series of interconnected vortex loops above *Re=290*, while Sakamoto and Haniu (1990) show that hairpin shaped vortices begin to shed periodically at *Re=300*. Identifying the appropriate method to determine the vortex in turbulent flow has always enthralled the research community. Jeong and Hussain (1995) has performed a lot of tests for vortex identification and has concluded that the most appropriate is the *Q method*, where Q is the positive second invariant of the deformation tensor. If a vortex exist, indeed $Q>0$.

$$Q = \frac{\Omega i_j \Omega i_j + S i_j S i_j}{2} \tag{3}$$

$$S_{i,j} = \frac{\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial u_i}}{2} \tag{4}$$

$$\Omega_{i,j} = \frac{\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial u_i}}{2} \tag{5}$$

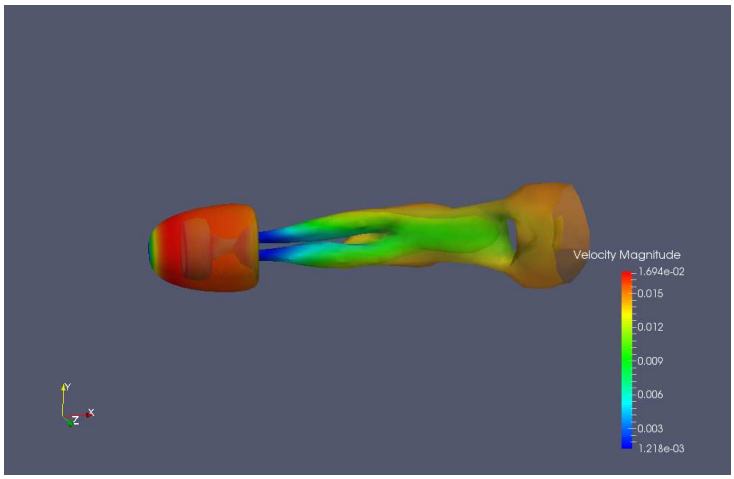

Figure 3: Isocontour velocity magnitude of the second invariant of the deformation tensor at Re=300

2.2.2 Turbulent flow Reynolds 10000

Due to the fact that the boundary layer is considerably thinner at this Reynolds regimes, the turbulent flow with laminar boundary layer has an increased resolution grid, the mesh 6. As the simulation is no longer a steady-state regime, a turbulence model has to be selected, in this case it was selected the $k - \epsilon$ model. The physical properties were the same that the previous situation, with the difference of the viscosity. The inlet velocity was 0.146 $[\mathrm{m/s}]$ and the time step was 0.137 $[\mathrm{seg}]$. The results obtained utilizing the mesh 5 are in agreement with the one obtained for Constantinescu and Squires (2003) and Jones and Clarke (2008) having a difference of 1.01% and 2.52% each. A better drag coefficient (Cd) was obtained by the mesh 4 due to more number of cells utilized.

3 COMPUTATIONAL METHODOLOGY

3.1 Parallel Programming

Foster (1995) has concluded that is possible to increase potential computing power more cost effectively by utilizing multiple, slower, less expensive components rather than a single, faster, more costly component. Therefore, the trend in computing hardware is towards multiple processors. Another trend that is affecting the way computing is being done, is the increase in network transfer rates. This allows physically separated resources to be utilized for solving a single problem.

These simultaneous use of more than one processor to execute a program is the basis of the *message passing* paradigm of parallel programming; where computers are a set of sequential processors, each with their owns memory, interconnected by some communication link, Foster (1995). Each processor executes a sequential set of instructions and communicates with other processors and accesses remote memory through the communication link. Distributed memory

Regimen	Mesh 4	Mesh 5	Mesh 6	Cd Ref	Referencies	Dif. % Mesh 4	Dif. % Mesh 5	Dif. % Mesh6
Reynolds 300	0.665	0.664	—	0.661	Jones and Clarke (2008)	0.60%	0.45%	—
			—	0.658	Giacobello (2005)	1.05%	0.90%	—
			—	0.657	Kim et al. (2001)	1.20%	1.05%	—
			—	0.656	Johnson and Patel (1999)	1.35%	1.20%	—
			—	0.656	Constantinescu and Squires (2003)	1.35%	1.20%	—
Reynolds 10000	—-	0.397	0.44	0.393	Constantinescu and Squires (2003)	1.01%	10.68%	—
				0.387	Jones and Clarke (2008)	2.52%	12.05%	—
				0.438	Kim et al. (2004)	-10.33%	0.45%	—
Reynolds 1000000	—-	—-	0.1408	0.800	Constantinescu and Squires (2004)	—	—	43.20%
				0.141	Jeong and Hussain (1995)	—	—	-0.11%
				0.139	Jones and Launder (1972)	—	—	1.31%
				0.104	Jones and Clarke (2008)	—	—	26.16%
				0.142	Jones and Launder (1972)	—	—	-0.82%

Table 2: Drag coefficients obtained by different simulations regimes

and shared memory systems fit this model. As the work is shared or distributed among different processors, data has to be exchanged.

The most predominant message passing data is the Message Passing Information (MPI), which defines a set of library routines that can be called from C and Fortran programs. The programming style for MPI has changed over the years, which in turn has continued to spur the evolution of the MPI standard (Hempel, 1994). MPI is expected to provide better performance on large massively parallel processors (MPP) but does not provide for heterogeneous distributed computing and lacks many task management functions, McBryan (1994). Likewise, other models are available for parallel programming, the most popular is the share memory programming model Pthreads McBryan (1994). Pthreads is a standard implementation for shared memory programming using *threads*. A thread is a light-weigh process that shares memory with other threads, but has it owns program counter, registers and stack that each thread can execute a different part of the code. OpenMP is an alternative library of Pthreads that attempts to avoid the low-level programming constructs required by Pthreads.

The processor of a share memory system have a more efficient way of accessing remote memory than do the processors of a distributed memory. A very effective way to improve our productivity is utilizing both models, so as to make the most profitable parallel algorithms. Indeed, MPI and OpenMP are combined together to take advantages of both programming models when clusters of shared memory multiprocessor (SMP) machines are linked. Hybrid codes became more interesting for compute nodes with the growing number of cores in a common shared memory domain. Each MPI spawns several threads and we have the logical processors.

$$\#LogicalProcessor = \#ranks \cdot \#threads \tag{6}$$

Logical processors are the available cores and the goal of this methodology is to maximize the number of threads and minimizing the numbers of MPI ranks so as to reduce the overhead that grows with the number of ranks. This overhead is due to the memory consumption of internal buffers and limited scaling potential of collective MPI operations, Jeffers and Reinders (2013).

3.2 Cluster Employed

The Supercomputer that was employed for running the simulations is at the Research Centre of Computing Mechanics (CIMEC) which is located on the city of Santa Fe, Argentina. This Centre works together with the Argentinean Council for Scientific Research (CONICET), the Argentinean National Agency for Technological and Scientific Promotion (ANPCyT) and the National University of the Litoral (UNL).

The supercomputer is called *Seshat* and its equipment is integrated by:

- Equipment:
  - Head-Node
    - Processor: Intel(R) Xeon(R) CPU E5-2640 v2 @ 2.00GHz (2 CPU x 8 cores)
    - Motherboar: Intel S2600CP
    - Memory: 96 GB (12) 8GB - DDR3 - 1600 Mhz
  - Calculation Available Nodes ($69\ nodes$)
    - Processor: Intel(R) Xeon(R) CPU E5-1620 v2 @ 3.70GHz (1 CPU x 4 cores)
    - Mother-board: Supermicro X9SRL-F
    - Memory: 16 GB (2) 8GB - DDR3 - 1600 Mhz
  - Total Memory: 1.2 TB
  - Total Cores: 292 Cores
- Network
  - Ethernet
    - Switches: (2) Supermicro - 1Gpbs - 48 ports
  - Infiniband
    - QDR 40 Gbps

This computer-networking communications standard used in high-performance computing, features very high throughput and very low latency. The theorical peak performance of this supercomputer is:

$$TPP = 69\ [\text{nodes}] \cdot 1\ [\text{processor}] \cdot 4\ [\text{cores}] \cdot 2\ [\text{AVX units}] \cdot 4\ [\text{doubles}] \cdot 3.7\ [\text{GHz}] \quad (7)$$

$$TPP = 8,196\ [\text{GFLOPS}] \quad (8)$$

Furthermore, a real High Performance Linpack (HPL) was performed.

$$HPL = 6,927[\ \text{GFLOPS}] \quad (9)$$

$$\eta = \frac{HLP}{TPP} = 85\% \quad (10)$$

Nodes	Time	Speed Up	Efficiency
1	44.77	-	-
2	22.22	2.01	1.01
4	11.5	3.89	0.97
8	5.69	7.87	0.98
16	3.04	14.73	0.92
24	2.15	20.82	0.87
36	1.52	29.45	0.82
48	1.38	32.44	0.68
56	1.37	32.68	0.58

Table 3: Speed Up and Efficiency of the $Seshat$ cluster for 3 Million cells mesh

Nodes	Time	Speed Up	Efficiency
1	141	-	-
2	71.14	1.98	0.99
4	36.27	3.89	0.97
8	20.67	6.82	0.85
16	9.73	14.49	0.91
24	6.66	21.17	0.88
36	4.31	32.71	0.91
48	3.4	41.47	0.86
56	3.02	46.96	0.83
68	2.58	54.65	0.80

Table 4: Speed Up and Efficiency of the $Seshat$ cluster for 9 Million cells mesh

3.3 Scalability Analysis of the Cluster Employed

As it was mention before, dividing a problem and linking efforts of several computers to tackle it, is the main idea of parallel computing which is defined as a set of processors that are able to work cooperatively to solve a computation problem.

Scalability relates to the behavior of an algorithms in terms of parallel efficiency or speed up as a function of processor count. As it was mentioned before, in order to perform the most profit parallel algorithms, MPI and OpenMP were combined together.

Code Saturne CFD includes the ability to perform scalar and parallel simulations with great ease. The load balancing for parallel version is performed by partitioning. By default, Code Saturne CFD employs *scratch partitioning*, whereas, for parallel computing it employs *parmetis*.

In order to asses the scalability of the Cluster Seshat the benchmark mentioned before was run varying the number of logical processors. In each node one process was executed and each process had four threads.

Both the speed up and the efficiency were assessed for the three following mesh size: 3, 9 and 19 millions cells. All the systems showed good speed up until reaching a threshold number of processors, after that the run-time remains steady or even increase. The threshold is the number of processors at which the *processor message passing time* equals the *processor computation time*. From Figure 4 it could be seen that for the 3 millions mesh simulation up to 50 processors the speed up is increasing, whereas it remain steady from this quantity. In fact, if we increase

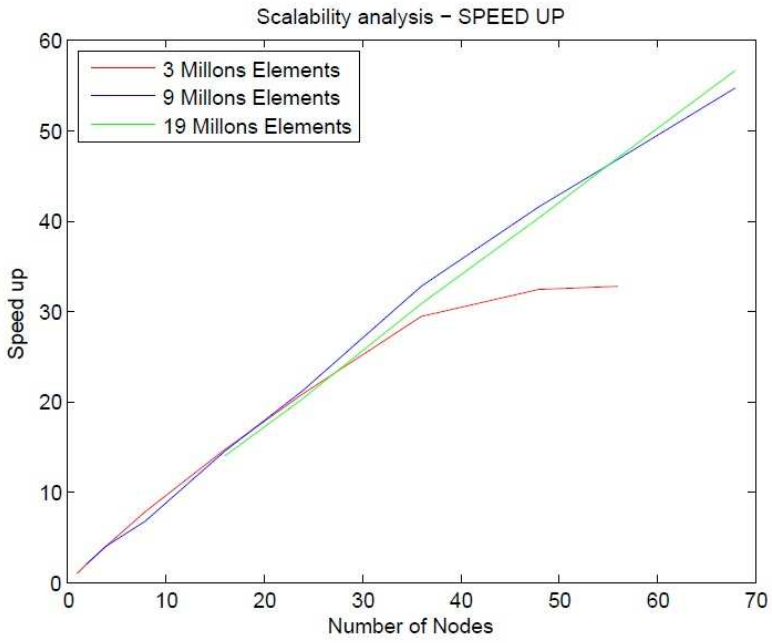

Figure 4: Figure of the Speed Up at the Cluster Seshat

Nodes	Time	Speed Up	Efficiency
1	303	-	-
16	21.84	13.87	0.87
24	14.96	20.25	0.84
36	9.85	30.76	0.85
48	7.54	40.19	0.84
56	6.45	46.98	0.84
68	5.35	56.64	0.83

Table 5: Speed Up and Efficiency of the $Seshat$ cluster for 19 Million cells mesh

the number of nodes, no gain will be obtained, which means that the threshold for 3 millions cell is 50 nodes. Nevertheless, for the other two situations no thresholds are evidenced on the Figure 4, due to the fact that we are still located at the *speed up* period. Likewise, these situations also have a threshold which is absolutely more than 70 processors.

In the same way, the Figure 5 shows the parallel performance drops some for increasing the number of nodes. Although, at grater number of nodes employed, the efficiency remains steady for the biggest mesh, what means the parallel algorithm of 19 millions cell has a satisfactory scalability.

4 MODEL

4.1 Model Geometry

The model adopted on the simulations was the bullet provided by the Ministry of Defense of Argentinian Government. An schematic diagram of the basic shape is shown on the Figure 6. The maximus diameter and the length are $41,3$ and $73,34$ $[\mathrm{mm}]$ each.

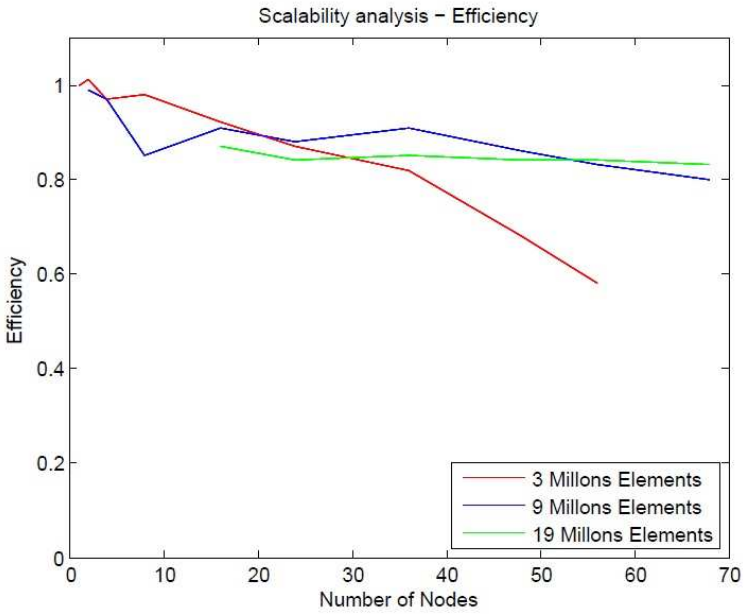

Figure 5: Figure of the Efficiency at the Cluster Seshat

Two different flight situations were assessed, in which both the velocity of the flow and the attack angle were switched. When a CFD simulation is performed, instead of setting a velocity on the projectile, the projectile remains steady and the velocity is set as a boundary condition of the inlet flow. The simulation were performed for the below flight conditions, where the pressure, density and temperature were those of the ambient air, i.e. $101325\ [\mathrm{Pa}]\ 1,225[\mathrm{kg/m^3}]$ and $288,15\ [\mathrm{K}]$.

- Situation 1, Angle of attack = 0 °:

$$Vx = -76\ [\mathrm{m/s}]\ Vy=0\ [\mathrm{m/s}]\ w=386,6\ [\mathrm{rad/seg}]$$

- Situation 2, Angle of attack = 10 °:

$$Vx= -75,06\ [\mathrm{m/s}]\ Vy=11.89\ [\mathrm{m/s}]\ w=386,6\ [\mathrm{rad/seg}]$$

Where w is the roll on clockwise direction, Vx and Vy are the velocities of the flow on the *"x"* and *"y"* direction.

4.2 Computational Grids

The bullet under study was immersed in control volume which size was $0.5 \times 0.25 \times 0.25\ [\mathrm{m}]$. The control volume extended from $0.1484[\mathrm{m}]$ upstream of the bullet and $0.2782\ [\mathrm{m}]$ downstream. So as to perform the mesh, the projectile was divided in 2 groups: the skin of the bullet and the front and back of it. The front and the back of the projectile were meshed with the *quadrangle mapping criteria* where the maximum and minimum size were set to $0.0005\ [\mathrm{m}]$. The skin mesh was generated also utilizing the *mapping criteria* that splits faces into quadrangular elements. However this mapping was integrated by the revolution profile and the different circumferences that were generated by this profile. Both revolution profile and the circumferences

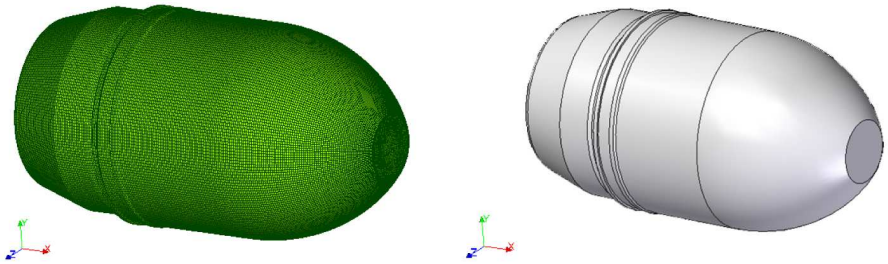

Figure 6: Geometry of the bullet under study

were mesh utilizing the *wire descritetization* criteria, where the edge is split into a number of mesh segments following the 1D hypothesis. For the revolution profile segment, the *circumference segments* hypothesis was utilized and 100 elements were equidistantly distributed (1D hypothesis) around the hole circumferences. On the other hand, for the revolution profile, a *long length* criteria was utilized, where this length was set in 0.0004 [m]. For this configuration a $14,363,803$ meshed elements were obtained. Another meshes were also generated varying the meshing parameters which were detailed before. This results are shown in Table 6 where the average length of the quadrangle cell is shown.

Description	Number of elements [elem]	Average Length [mm]	Boundary layer [elem]
Mesh 0°2M	1463382	0.5	0.16
Mesh 0°5M	4915990	0.486	0.16
Mesh 0°7M	76228841	0.307	0.16

Table 6: Description of different meshes employed in the bullet for angle of attack 0°

Likewise, to perform the study of the aerodynamics coefficients, forces and moments, the finer mesh was utilized.

5 RESULTS

Numerical simulations have been carried out utilizing Code Saturne CFD software in the same cluster located on the Research Centre of Computational Methods (CIMEC). The forces and moments for the two different angles of attack were obtained. The simulations performed were run utilizing LES turbulence model.

Angle of attack = 0°	Fx [N]	Fy [N]	Fz [N]	Mx [Nm]	My [Nm]	Mz [Nm]
2 Millions	-1.05	-0.0914	0.148	-4.02e-4	-6.3e-4	-2.69e-4
5 Millions	-0.83	-0.0198	-0.028	-2.73e-4	-4.92e-4	-3.5e-4
7 Millions	-0.82	0.042	-0.025	-2.68e-4	1.33e-4	-6.6e-5

Table 7: Forces and Moments for 0°of angle of attack

Angle of attack = 10°	Fx [N]	Fy [N]	Fz [N]	Mx [Nm]	My [Nm]	Mz [Nm]
7 Millions	-1.25	0.116	1.08	-4.8e-3	-4.1e-2	4.36e-4

Table 8: Forces and Moments for 10°of angle of attack

Both on the Figures 7 and 8 the streamlines at the front and back of the projectile. On the first one, also the pressure, the velocity and the strain efforts are shown, whereas on the second one, just the velocity. Also the shear stress is detailed on the Figure 9. Moreover, *Q criterion* is shown on the Figure 10, where the vortex shedding are perfectly visualized.

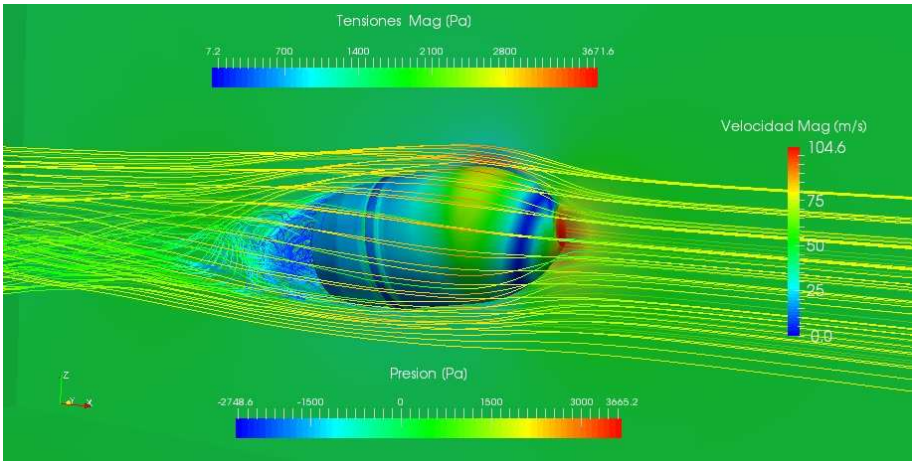

Figure 7: Streamlines and Effort on the bullet, angle of attack 10°

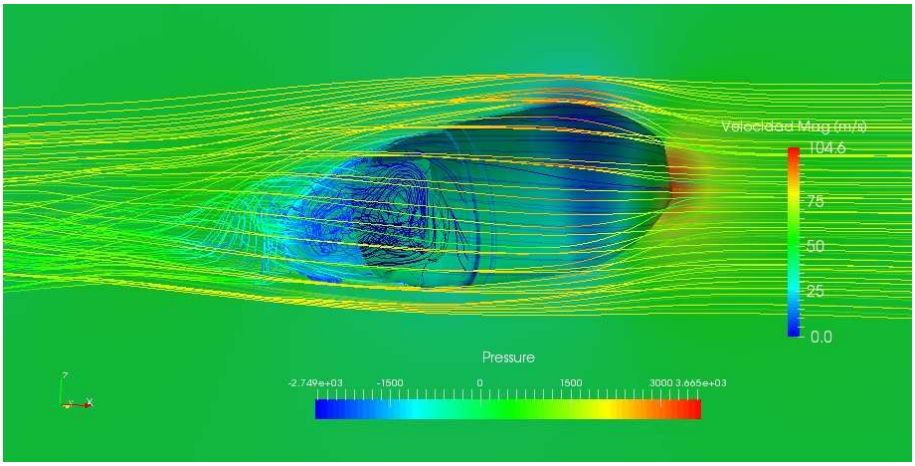

Figure 8: Streamlines and Effort on the bullet, angle of attack 10°

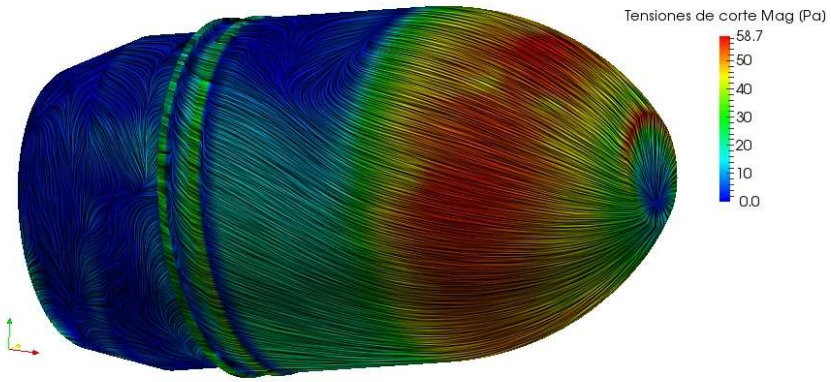

Figure 9: Shear efforts on the bullet, angle of attack 10°

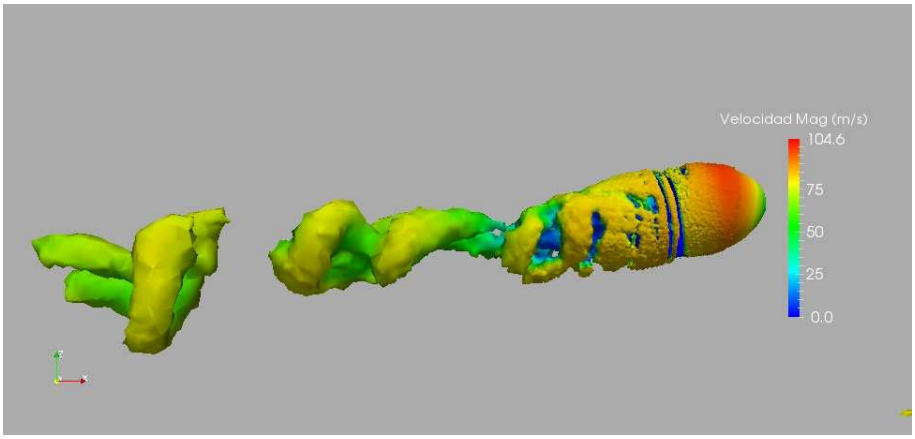

Figure 10: Iso-surfaces of the Q Criterion, angle of attack 10°

ACKNOWLEDGMENT

The authors wish to thank the support given by the following research projects:

- Argentinean Council for Scientific Research (CONICET projects PIP 112-20111-00978, and 11220150100588CO).
- Argentinean National Agency for Technological and Scientific Promotion, ANPCyT, (grants PICT-E-2014-0191, PICT-2014-0191, PICT-2015-2904);
- Universidad Nacional del Litoral, Argentina (grant CAI+D-501-201101-00233-LI);
- Santa Fe Science Technology and Innovation Agency (grant ASACTEI-010-18-2014); and
- European Research Council (ERC) Advanced Grant Real Time Computational Mechanics Techniques for Multi-Fluid Problems (REALTIME, Reference: ERC-2009-AdG).
- Secretaría de Ciencia, Tecnología y Producción para la Defensa. Programa de Investigación y Desarrollo para la Defensa. PIDDEF PIDDEF-4/14.

The authors made extensive use of *Free Software* as GNU/Linux OS, GCC/G++ compilers, Octave, and *Open Source* software as VTK, ParaView, Salome and Code Saturne among many others.

REFERENCES

Constantinescu G. and Squires K. Numerical investigations of flow over a sphere in the subcritical and supercritical regimes. *Physics of Fluids (1994-present)*, 16(5):1449–1466, 2004.

Constantinescu G.S. and Squires K.D. Les and des investigations of turbulent flow over a sphere at re= 10,000. *Flow, Turbulence and Combustion*, 70(1-4):267–298, 2003.

Foster I. Designing and building parallel programs. 1995.

Giacobello M. Wake structure of a transversely rotating sphere at moderate reynolds numbers. 2005.

Hempel R. The mpi standard for message passing. In *International Conference on High-Performance Computing and Networking*, pages 247–252. Springer, 1994.

Jeffers J. and Reinders J. *Intel Xeon Phi coprocessor high-performance programming*. Newnes, 2013.

Jeong J. and Hussain F. On the identification of a vortex. *Journal of fluid mechanics*, 285:69–94, 1995.

Johnson T. and Patel V. Flow past a sphere up to a reynolds number of 300. *Journal of Fluid Mechanics*, 378:19–70, 1999.

Jones D. and Clarke D. Simulation of the flow past a sphere using the fluent code. 2008.

Jones W. and Launder B. The prediction of laminarization with a two-equation model of turbulence. *International journal of heat and mass transfer*, 15(2):301–314, 1972.

Kim J., Kim D., and Choi H. An immersed-boundary finite-volume method for simulations of flow in complex geometries. *Journal of Computational Physics*, 171(1):132–150, 2001.

Kim S.E. et al. Large eddy simulation using unstructured meshes and dynamic subgrid-scale turbulence models. *AIAA paper*, 2548:2004, 2004.

McBryan O.A. An overview of message passing environments. *Parallel Computing*, 20(4):417–444, 1994.

Sakamoto H. and Haniu H. A study on vortex shedding from spheres in a uniform flow. *Journal of Fluids Engineering*, 112(4):386–392, 1990.

Thompson M.C. and Le Gal P. The stuart–landau model applied to wake transition revisited. *European Journal of Mechanics-B/Fluids*, 23(1):219–228, 2004.