7. ANKARA INTERNATIONAL AEROSPACE CONFERENCE
11-13 September 2013 - METU, Ankara TURKEY

AIAC-2013-075

DEVELOPMENT OF AN EXTERNAL BALLISTIC SIMULATION SOFTWARE

Serkan Berkay Körpe[1] and Mehmet Halûk Aksel[2]
Mechanical Engineering Department
Middle East Technical University
Ankara, Turkey

ABSTRACT

A fully automatic external ballistic CFD analysis-simulation software is developed in C++ for the purpose of acquiring instant aerodynamic properties, particularly pressure based drag coefficient during the free flight of any intended flat based or boat tailed projectile. To eliminate any dependency on other commercial or third party software, a three-dimensional Computational Fluid Dynamics (CFD) analysis code is developed for the entire analysis process including a three-dimensional geometry modeler and surface mesh generator, an adaptively-refined Cartesian volume mesh generator and an Euler solver.

INTRODUCTION

Starting from 1850's, external ballistic researches were continued by experimental studies mostly based on yaw cards [17]. In the first half of twentieth century, two World Wars occasioned substantial growth in these researches. After World War II, besides the development of supersonic wind tunnels and free flight spark photography ranges, computers started to be used for external ballistic applications. The first electronic general-purpose computer, ENIAC, was designed for solving external ballistic problems, during the World War II [16]. By 1970's empirical codes have being developed which have been based on the experimental data accumulated for decades [3][11][14][15]. On the other hand, the advancements in numerical integration methods directed researchers towards the solution of 6 degree of freedom motion equations and CFD applications [5][10][13]. This paper presents a three-dimensional external flow CFD software specialized for low caliber external ballistic applications.

GEOMETRY MODELLING AND SURFACE MESH GENERATION

An automatic three-dimensional geometry modeler is developed for the first step of pre-processing part of the analysis. Considering any flat based or boat tailed projectile, the 2-D surface is assembled by four line segments and a circular arc, as shown in Figure 1. These segments represent the kurtosis, ogive, cylinder, boat tail and base sections of the projectile, respectively.

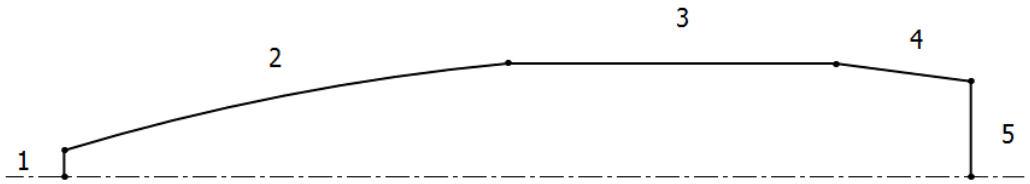

Figure 1: 2-D projectile schematic view

[1] M.S. Student in Mechanical Engineering Department, Email: sekorpe@gmail.com
[2] Prof. in Mechanical Engineering Department, Email: aksel@metu.edu.tr

Software user is supposed to specify the geometric features of the intended projectile as ogive length, ogive radius, ogive center offset, cylinder length, cylinder diameter and tail length in calibers and tail angle in degrees. Primarily, the specified geometric features are used for drawing 2-D outline of the projectile. Then by revolving the outline 360° around *x*-axis, the 3-D surface model of the projectile is obtained. Revolving process produces two circular surfaces at the nose and base, and three quasi-cylindrical surfaces covering the sides of the projectile (Figure 2a). In order to mesh these five generated surfaces, two different surface meshing algorithms are developed for circular and frusto-conical, cylindrical surfaces. The surface mesh generating algorithm has the ability of refining mesh on any intended surface (Figure 2b, 2c). Optimum mesh intensity is essential for catching the inclination on curved surfaces and reducing time consumption at Cartesian volume mesh generation process.

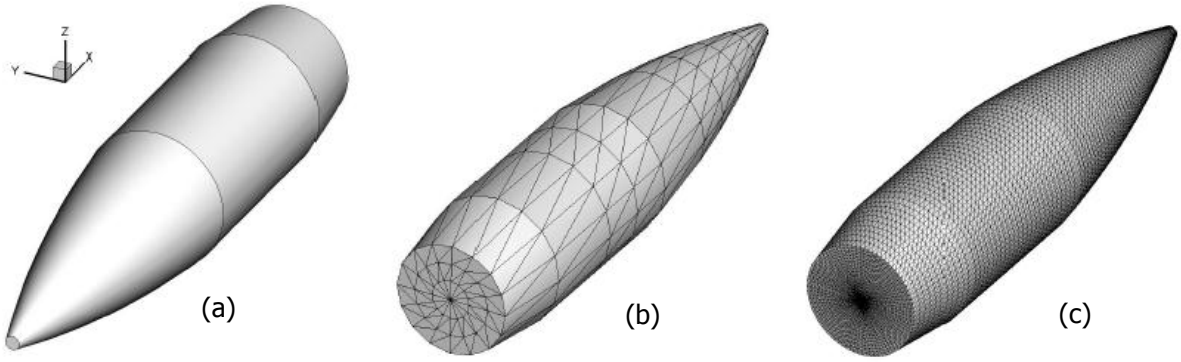

Figure 2: (a) 3-D surface model, (b) coarse and (c) fine surface meshes of G_7 type projectile

CARTESIAN VOLUME MESH GENERATION

Cartesian mesh is a type of unstructured mesh which is very attractive and popular recently. Several advantages of this mesh provide this popularity and widely usage. First of all, Cartesian mesh uses the advanced and complicated data structures as Quadtree in two-dimensional and Octree in three-dimensional space. These data structures make mesh generation very easy and flexible even for complex three-dimensional geometries. Flexibility in mesh generation leads to a great advantage for mesh refinement, coarsening and multigrid applications. Another advantage of the Cartesian mesh is the monotomy of the cells. Monotonous cells provide convenience to the area, volume and flux calculations and eliminate the skewness problems as seen in structured and unstructured meshes.

On the other hand, the main disadvantage of Cartesian meshes is handling the cells which are interacted with the surface of the geometry. Forming these irregular cut and split cells can be compelling and troublesome. The other disadvantage is, after the creation of cut and split cells, the volume of these irregular cells can be extremely small with respect to outer cells. The hazards of this kind of problem can be eliminated by cell merging techniques. Cartesian mesh is not allso sufficient for viscous flow solutions but can be used by hybridization with structured boundary layer mesh [1].

Octree Data Structure and Root Cell

Dynamic Octree data structure is used for the developed three-dimensional Cartesian mesh generator and Euler solver. Dynamic data structure allocates memory depending on the situational needs and removes unnecessary data from the memory during the execution of the software. This type of data structure makes mesh refinement and coarsening very practical since the refinement and coarsening processes are extremely solution dependent. Dynamic allocation brings great utility to the developed code by increasing the efficiency of the memory usage.

Octree is a tree-type data structure and mostly used to discretize the three-dimensional space into eight octants. It starts with the undivided cubic space which is called the root cell and it subdivides into eight equal volumes of cubes for multiple times. Figure 3 represents Octree data structure used for the Cartesian mesh. Subdivision process of the root cell continues until the desired level of divisions is reached and the flow variable computations take place in the highest level divisions of the root cell. Therefore, if a cell is subdivided into eight octants, it is no longer flagged as computational cell and it acts as a bridge between its children, parent and neighbor cells.

The most important attribution of Octree data structure is the relativity between all cells, from the root cell to computational cells. Connectivity between cells is carried out by parent, neighbor and child relativity variables. The bonds between Octree elements are crucial for safety of the mesh generation

and the information transfer between neighboring cells for two or more order solutions. Besides the flow variable information, cells in Octree data structure hold information of cell relativity and mesh properties. These variables that are kept within a cell structure are listed on Table 1.

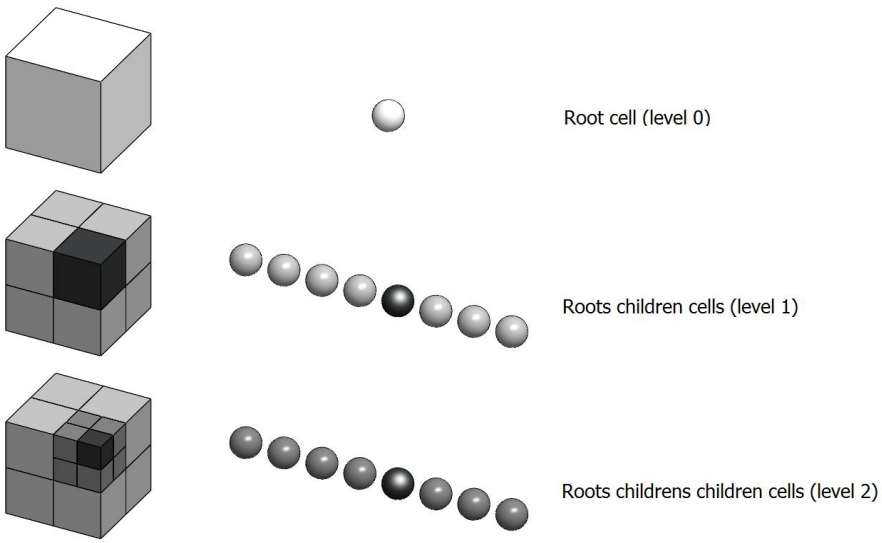

Figure 3: Octree data structure

Since the root cell is the undivided initial cell, it also defines the flow domain around the projectile. Placing the projectile at the center of the root cell, edge length of the root cell is chosen 20 times larger than the projectiles chord length. For the external flow CFD analysis for projectiles, flow domain boundaries must be far enough from geometry surface so that disturbances in flow variables do not or weakly reach to the domain boundaries for the safe application of the far-field boundary condition.

Uniform Mesh Generation

Uniform mesh generation is the subdivision of the root cell to desired level of computational cells. After the subdivision of the root cell, eight children cells are formed and these cells are connected to the root cell by parent-child relationship. Child cell formation process starts by creating eight new cell structures and defining their corner and center coordinates. Indexing these new eight cells is very important for setting neighbor relationships as presented (Figure 4b).

Table 1. Octree variable list for a Cartesian cell

Number of variables	Representation of variables
8	Corner coordinates
1	Center coordinate
1	Division level
1	Computational cell flag
1	Parent cell
6	Neighbor cells
8	Child cells
1	Type

Neighboring relationships for each created child cell are set during the uniform mesh generation process. Since the child cell indexes are fixed within a parent cell, neighbor indexes are fixed as well. To give an example, in Figure 4c, east, south and bottom neighbors of 6^{th} child cell are 5^{th}, 7^{th} and 2^{nd} child cells of the same parent cell, respectively. But the west, north and top neighbors are not child cells of the same parent cell. These cells are connected to each other by neighboring relationships of

their parents. 5th child cell of the west cell of the parent of the 6th cell is always west neighbor of this cell and the same procedure is followed for the north and top neighbors.

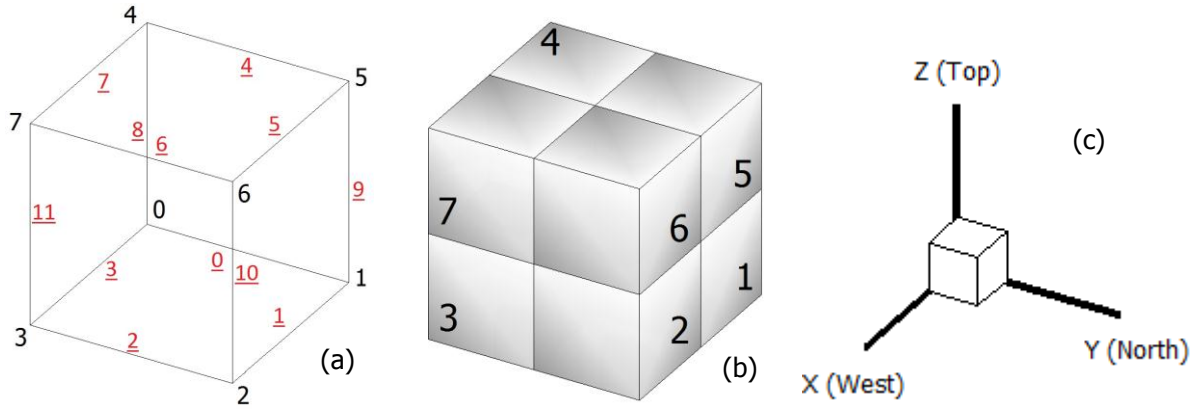

Figure 4: (a) Corner and edge, (b) child cell and (c) neighbor indexing

Geometry Fitting to Uniform Mesh

At the end of uniform mesh generation process, the projectile geometry must be fit into the uniform mesh. This fitting procedure is the most complicated part of the Cartesian mesh generation. The procedure is basically, extracting the solid volumes within the Cartesian cells which are interacted by the projectile geometry and changing their cubic shapes, volumes and center coordinates, creating new cut surfaces.

Cell Type Specification

Geometry fitting into generated uniform mesh starts by identifying types of the computational cells by ray-casting method. There are three types of cells defined for the Cartesian mesh. If all corners of a cell are inside the projectile surface, the cell is referred as an inside cell, if all corners are outside of the body than the cell is called an outside cell and if some corners are inside and some of them outside than this type of cell is called a cut cell. For a Cartesian mesh, there is an extra cell type called as a split cell which is a very complex form of the cut cell and difficult to handle in three-dimensions. However, simplicity of the projectile shapes prevents this type of cell formation. Example configurations of cut and split cells are given in Figure 5.

To identify the types of Cartesian cells in uniform mesh, ray-casting method is used for determining if the cell corners are inside or outside of the projectiles volume. Considering each corner of a cell as ray sources, rays are sent in *+x*, *+y*, *+z* and *-x*, *-y*, *-z* directions, reaching to boundaries of the flow domain. The aim of this procedure is the count of the ray-triangle intersection for each ray which specifies inside-outside properties of the corners by ray-triangle intersection algorithms in three-dimensional space [6][9]. Figure 6 shows ray-casting method on a coarse surface mesh of G_7 type projectile.

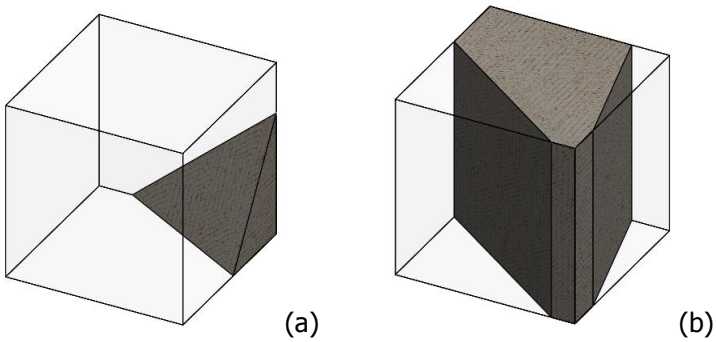

Figure 5: (a) Cut and (b) split cell examples

Considering P1, P2 and P3 points as the corner points of any Cartesian cell, blue rays starting from these points and continuing along *-y* direction intersect 0, 2 and 1 surface triangles, respectively. If a

ray source is outside of the projectile, than the rays emitting from it, must intersect 0 or 2 surface triangles as P1 or P2 and if the source is inside of the projectile than the rays must intersect 1 surface triangle. After specifying inside-outside properties of all corners of all Cartesian cells, all Cartesian cells are categorized as inside, outside or cut cell.

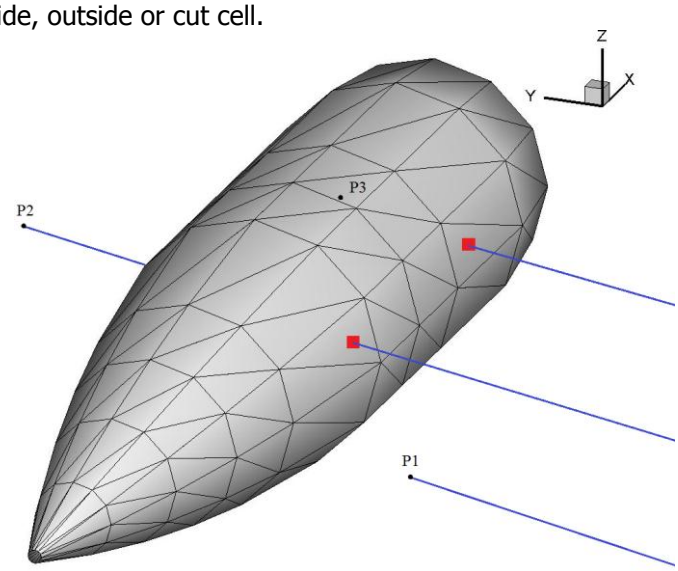

Figure 6: Ray-casting method on a coarse surface mesh

Cut Cell Creation

For each identified cut cell, creating cut surfaces resulting from surface intersections is the main part of the geometry fitting into a uniform Cartesian mesh. Several methods can be applied to achieve this such as marching cubes [18], marching tetrahedrals or exact fitting [8]. For the developed code, marching cubes method is used for cut cell surface and volume creations.

First step of the Marching Cubes algorithm is calculating *cube indexes* of all cut cells. *Cube index* is an integer value used for Marching Cubes table which reserves the information of which edges of the cell are cut, how many cut surface triangles describe projectile surface and location of corners of these triangles. *Cube index* of a cut cell is calculated by using inside corner indexes by the pseudo code given below.

```
cube_index=0
for corners from 0 to 7
cube_index=cube_index+2^index_of_inside_corner
```

For the cut cell configuration given in Figure 5a, since the only inside corner is the corner indexed as 1, *cube index* of this type of cut configuration is 2. In Table 2, first 6 out of 256 lines of the Marching Cubes table are presented. Following the third line (table starts from 0th line), up to the first -1 value, total count of non -1 integer values divided by 3 gives the triangle count representing projectile surface inside cut cell and for this case, it is 1. These three non -1 values indicate cut cell edges which cut cell triangle corners stand. The information that Marching Cubes table provides is also used for volume and centroid calculations of cut cells and also flux calculations from cut surfaces.

Table 2**:** First six rows of triangle table [2]

-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1
0,	8,	3,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1
1,	9,	0,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1
1,	8,	3,	9,	8,	1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1
2,	10,	1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1
0,	8,	3,	1,	2,	10,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1,	-1

Geometric Adaptations

Uniform mesh generation is the first set of divisions for creating a coarse mesh and adapting it to the projectile geometry. For avoiding cells with unnecessarily large volumes which are also close to flow domain boundaries, uniform mesh is limited to a maximum of three or four levels. However, after the third or fourth division, the cell resolution close to external flow geometry is comparably low to get precise results. Mesh intensity is increased for intended flow regions by geometric adaptations during the mesh generation and solution adaptation during the flow solution processes.

Essentially, mesh adaptation is the refinement of Cartesian cells, increasing their division levels by subdividing them into their child cells. Two types of geometric refinements are applied in the code. First adaptation is the box adaptation which is the method of increasing division level of cells to an intended value, within a specified box volume. The boundaries of this box are properly selected for external flow analysis. Second adaptation is called as the cut cell adaptation, which is achieved by subdividing cut cells until the desired level of computational cells are obtained. Figure 7 shows the applied forms of both geometric adaptations on projectile for supersonic flow analysis.

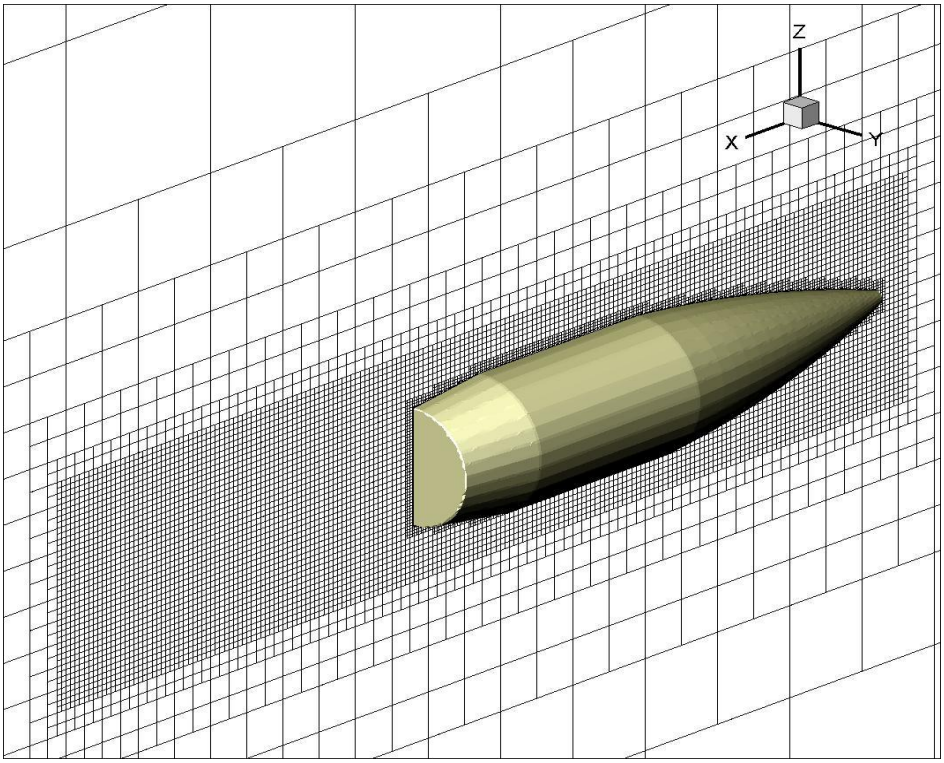

Figure 7**:** Geometric adaptations on *G_7 type* projectile.

FLOW SOLVER

Three-dimensional, steady-state, inviscid and compressible flow solver (Euler Solver) is developed for the solution of Euler equations. Equations are discretized spatially by using cell centered finite volume method. Liou's Advection Upstream Splitting Method (AUSM) is applied for flux calculations between neighboring Cartesian cells and first-order spatial accuracy is used for conserved variables in the flux calculations. Solution adaptation is used for capturing shock waves, expansion waves and wake region where gradients of primitive flow variables are relatively high.

Euler Equations

Three-dimensional compressible Euler equations can be represented in the conserved form as

$$\frac{\partial}{\partial t}\int_V \mathbf{U}dV + \oint_A \mathbf{F}\bullet\vec{n}dA = 0 \qquad (1)$$

where $\mathbf{U}$ is the vector of conserved variables and $\mathbf{F} \bullet \vec{n}$ is the vector of fluxes calculated throughout cell surfaces. A and V are the area and the volume of related surface and the cell, respectively. Considering the surface normal vector as $\vec{n} = n_x\vec{i} + n_y\vec{j} + n_z\vec{k}$,

$$\mathbf{U} = \left(\rho, \rho u, \rho v, \rho w, \rho E\right)^T \tag{2}$$

$$\mathbf{F}_x = \left(\rho u, \rho u^2 + p, \rho uv, \rho uw, \rho uH\right)^T \tag{3}$$

$$\mathbf{F}_y = \left(\rho v, \rho uv, \rho v^2 + p, \rho vw, \rho vH\right)^T \tag{4}$$

$$\mathbf{F}_z = \left(\rho w, \rho uw, \rho wv, \rho w^2 + p, \rho wH\right)^T \tag{5}$$

where, ρ is the density, u , v and w are the velocity components in x , y and z directions, respectively, E is the specific total energy, H is the total specific enthalpy and p is the static pressure of the air. In these equations, E and H are functions of u , v , w , ρ and p and thermodynamically related to these parameters.

Spatial Discretization of Euler Equations

The methodology used for solving integral forms of the Euler equations is cell centered finite volume method. Taking the integral of the $\frac{\partial}{\partial t}\int_V \mathbf{U}dV$ term in Equation (1)

$$\frac{\partial}{\partial t}\int_V \mathbf{U}dV = V\frac{\partial \mathbf{U}}{\partial t} \tag{6}$$

and integrated from of the $\mathbf{F} \bullet \vec{n}$ term in Equation (1) is

$$\oint_A \mathbf{F} \bullet \vec{n}dA = \sum_{i=1}^{nFace} \hat{\mathbf{F}}_i A_i \tag{7}$$

where, $\hat{\mathbf{F}}$ term is the fluxes perpendicular to the surface. Combining Equations (6) and (7)

$$\frac{\partial \mathbf{U}}{\partial t} = -\frac{1}{V}\sum_{i=1}^{nFace} \hat{\mathbf{F}}_i A_i \tag{8}$$

where the term $\sum_{i=1}^{nFace} \hat{\mathbf{F}}_i A_i$ is referred to as residuals ($\mathbf{R}(\mathbf{U})$) of the cell and they are functions of conservative variables, $\mathbf{U}$.

Temporal Discretization of Euler Equations

For the temporal discretization of Euler equations, 2nd stage Runge-Kutta multi-stage time stepping method is applied to the Equation (8).

$$\mathbf{U}^0 = \mathbf{U}^n \tag{9}$$

$$\mathbf{U}^1 = \mathbf{U}^0 - \beta_1 \frac{\Delta t}{V}\mathbf{R}\left(\mathbf{U}^0\right) \tag{10}$$

$$\mathbf{U}^2 = \mathbf{U}^1 - \beta_2 \frac{\Delta t}{V}\mathbf{R}\left(\mathbf{U}^1\right) \tag{11}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^2 \tag{12}$$

In Equations (10) and (11), Δt is the time step, β's are the stage coefficients and selected as $\beta_1 = 0.4361$ and $\beta_2 = 1.0$ [2]. Explicit time integration formulation is used, Equations (9) and (10) point that the *Residual* term is calculated using the previous steps conserved variables.

AUSM Flux Vector Splitting Scheme

The AUSM scheme uses the split form of Mach number and pressure terms to calculate the fluxes throughout cell surfaces [12].

$$\hat{\mathbf{F}}(\mathbf{U}) = \frac{1}{2}\left[M_{1/2}\left(\mathbf{F}'(\mathbf{U_L}) + \mathbf{F}'(\mathbf{U_R})\right) - \left|M_{1/2}\right|\left(\mathbf{F}'(\mathbf{U_R}) - \mathbf{F}'(\mathbf{U_L})\right)\right] + p_{1/2} \tag{13}$$

In Equation (13), $M_{1/2}$ and $p_{1/2}$ are the split Mach number and pressure term and they are functions of left and right side values.

$$\mathbf{F}' = (\rho c, \rho u c, \rho v c, \rho w c, \rho H c)^T \tag{14}$$

where, c is the speed of sound and

$$M_{1/2} = \frac{1}{2}\left(M_L^+ + M_R^-\right) \tag{15}$$

$$p_{1/2} = \frac{1}{2}\left(p_L^+ + p_R^-\right) \tag{16}$$

where

$$M^{\pm} = \begin{cases} \pm\frac{1}{4}(M \pm 1)^2 & |M| \le 1 \\ \frac{1}{2}\left(M \pm |M|\right) & |M| > 1 \end{cases} \tag{17}$$

$$p^{\pm} = pM^{\pm} \cdot \begin{cases} \pm 2 - M & |M| \le 1 \\ \frac{1}{M} & |M| > 1 \end{cases} \tag{18}$$

Boundary Conditions

Ghost cell method is used for the application of two different boundary conditions. Ghost cell is a mirror-image cell with equal volumes and the symmetry surface is the boundary. For the case of external flow of a projectile, the first boundary condition that is applied to the boundary faces of the root cell is far-field boundary condition. During the uniform mesh generation process, the cells that have one or more face on root cell face were flagged as cells at far-field boundary. The second boundary condition is the wall boundary condition which is applied to the cut cells. Figure 8 shows schematic view of wall boundary condition application by using a ghost cell. For the far-field boundary condition, variables defined at ghost cell center are the free stream values of the projectile at any certain time after the projectile leaves the gun. Moreover, for the wall boundary condition, these variables are exactly the same as the neighboring cut cell values expect the normal component of the velocity vector, which is defined as the same magnitude and opposite direction at the ghost cell center. This definition leads to the normal component of the fluid velocity to be zero at the projectile surface.

Solution Refinement

Mentioned in the previous sections, solution refinement is one of the important features of the Cartesian mesh. High definition solutions can be achieved by the fine initial mesh structure but this approach greatly decreases the efficiency of the analysis, additionally, excessively time and memory consumption. On the other hand solution refinement approach is basically subdividing computational cells at a certain time of solution, when the gradients of flow variables are large.

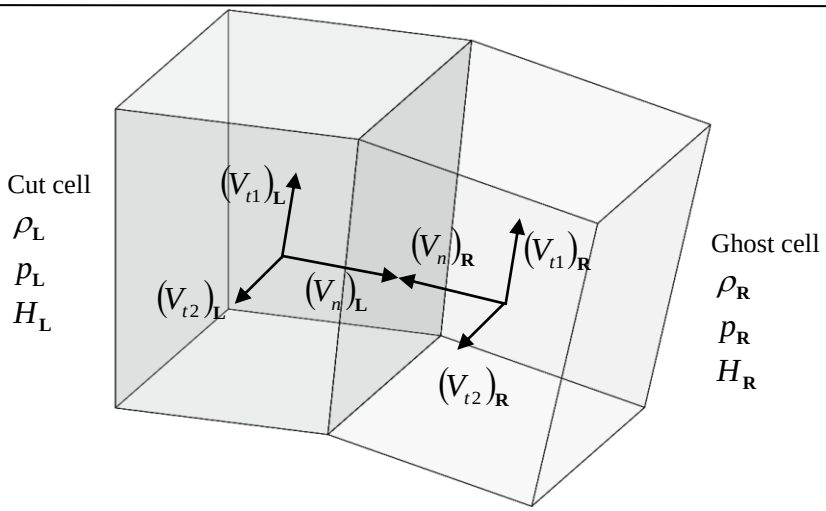

Figure 8: Ghost cell wall boundary condition

The criteria used for the developed code is based on the curl of the velocity for detecting shear layers and the gradient of the velocity for detecting shock waves [4]. These criteria which also depend on the volume of cell are given by

$$\tau_{Div} = (\nabla \bullet \mathbf{V}) V^{0.5} \tag{19}$$

$$\tau_{Curl} = (\nabla \times \mathbf{V}) V^{0.5} \tag{20}$$

where V is the volume of a cell. When the standard deviations of divergence and curl variables exceeds the predefined values, the cells having these deviations are flagged to be refined before moving to next time step. The flagged cells are refined and the projectile geometry is fitted to the refined Cartesian mesh structure.

RESULTS

The aim of developing this CFD tool is to obtain the pressure based drag force and drag coefficient which have the greatest impact on reducing the flight speed of any intended projectile for direct shooting. Considering that, a low caliber projectile leaves the gun barrel at a Mach number about 3, the most of speed reduction occurs at supersonic speeds. Therefore, after the convergence of the numerical solution of Euler equations, code is specialized for computing the pressure based force acting in the longitude direction of the projectile. Using the reference area of the intended projectile and free stream density, axial drag coefficients are calculated for G_7, G_3 types and 105mm projectiles at different Mach numbers and compared with experimental results available in the literature. Convergence criterion is selected as 10^{-7} for all tests and C_D convergence is also ensured.

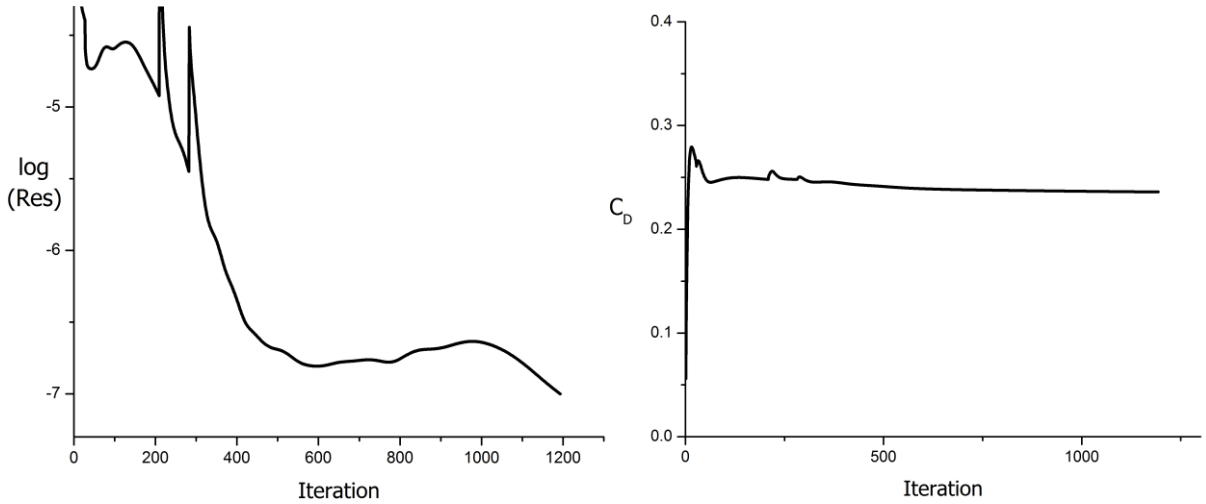

Figure 9: Residual and C_D convergence histories of G_7 type projectile for $M_\infty = 2.90$

Figure 9 shows the convergence histories of average of residuals and C_D at a free stream Mach number of 2.90. Figures 10 and 11 show the solution adapted mesh with Mach contours and pressure contours on the projectile surface at $M_\infty = 2.90$, respectively.

Figures 12, 13 and 14 compare the C_D values which are obtained numerically by the developed software and experimental data for G_7, 105mm and G_3 respectively. Experimental values for 105mm and G_3 cases are supplied by The Machinery and Chemical Industry Institution (MKEK), Ankara, Turkey.

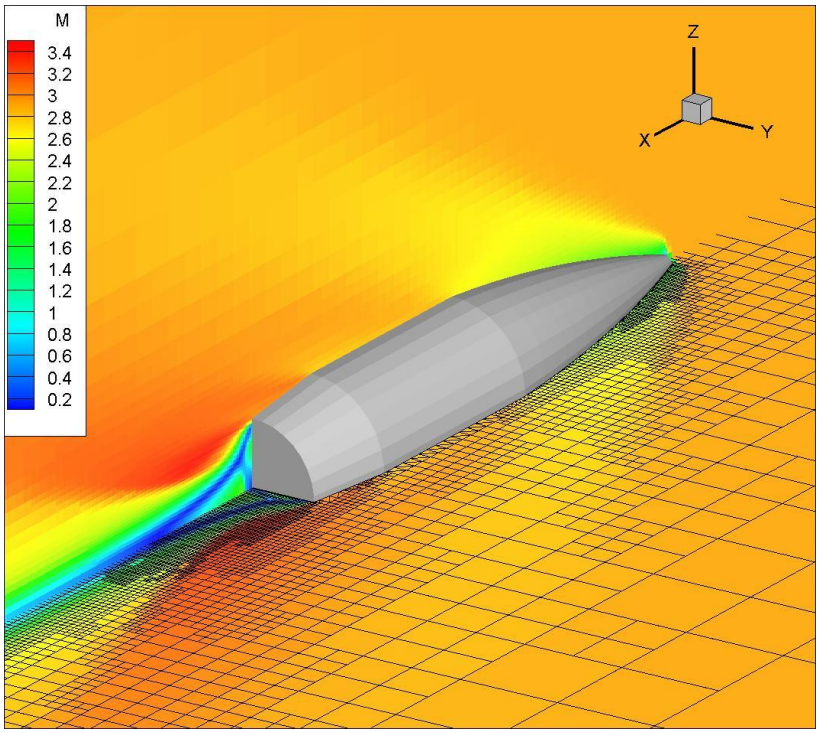

Figure 10: Mach contours and solution adapted mesh of G_7 type at $M_\infty = 2.90$

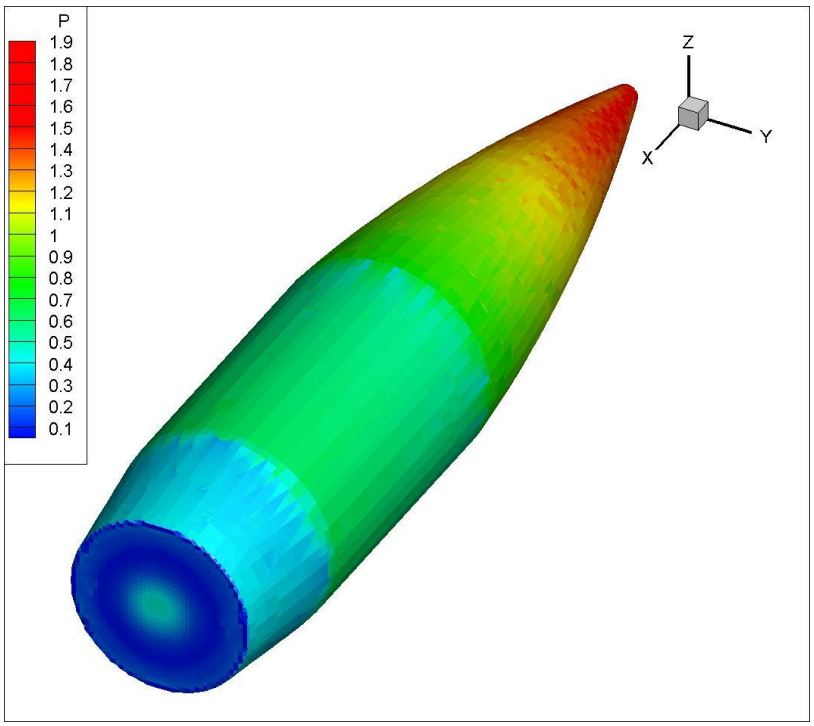

Figure 11: Pressure contours on G_7 type projectile surface at $M_\infty = 2.90$

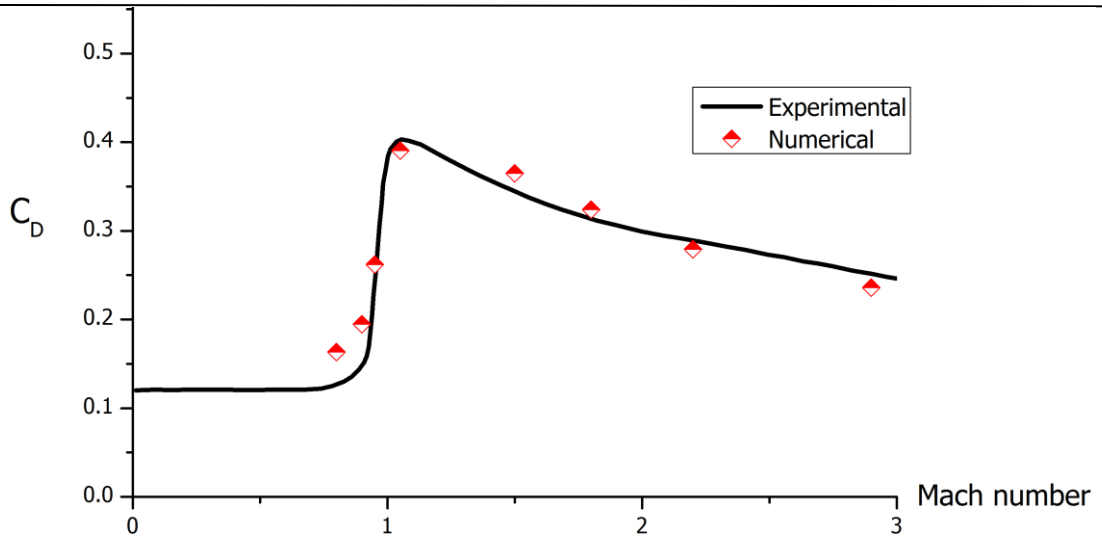

Figure 12: Drag coefficient numerical and experimental values for G_7 type projectile [7]

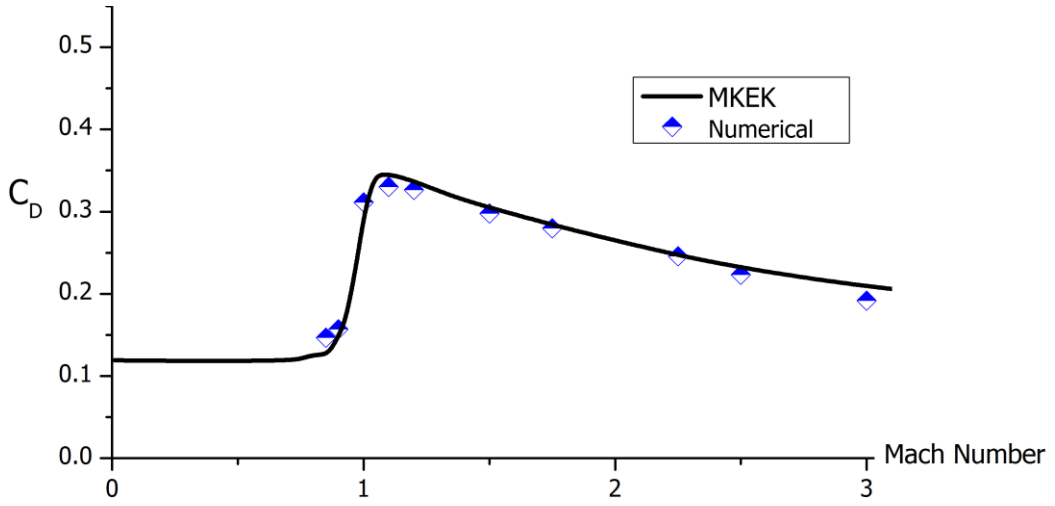

Figure 13: Drag coefficients for a 105mm projectile of the present study compared with the data supplied by MKEK

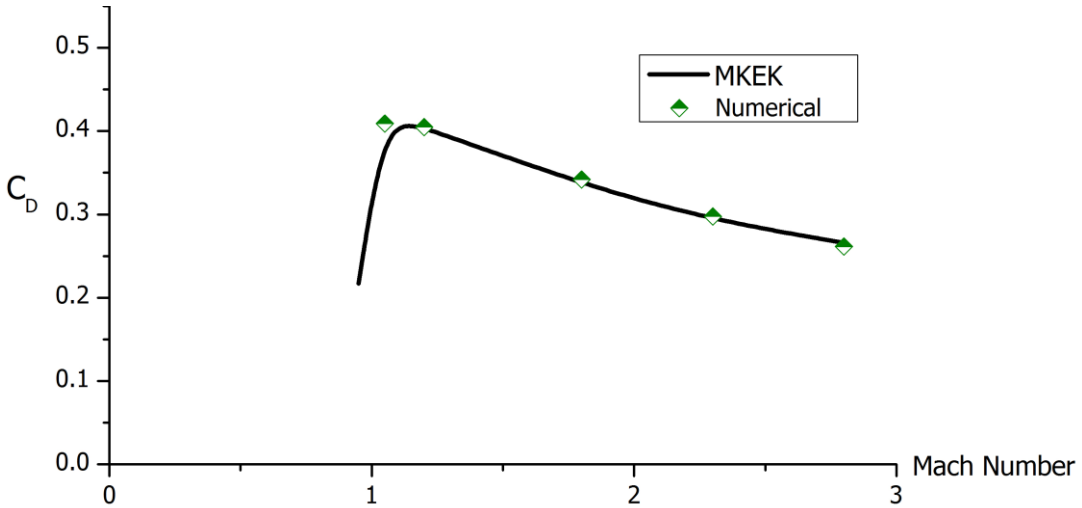

Figure 14: Drag coefficients for a G_3 type projectile of the present study compared with the data supplied by MKEK

CONCLUSION

Reviewing the result for three different projectiles, it can be concluded that the Euler solver gives accurate pressure based zero yaw drag coefficient for different flight Mach numbers in the range of leaving barrel and hitting target. For supersonic and some of transonic flow cases, the highest error is below 4% and mostly around 2%. It is also observed that getting closer to the Mach number of 3, error increases due to the change in the flight conditions towards the hypersonic flow. On the other hand, reaching to subsonic region also increases the error. Since the boundary layer and viscous effects scale up and flow considered as incompressible in this region, analysis exceeds the capacity of the developed compressible, inviscid flow solver. For the flow regimes around a Mach number of 0.80, the highest error observed to be about 12%.

This CFD software, from pre-processing to post-processing, has been developed and specialized for low caliber direct shooting applications. Recently, one of the most popular methods, Cartesian mesh structure, is used and software has great potential for future developments. More advanced geometry fitting procedures can be applied to the developed Cartesian mesh generator to enhance its geometry complexity scale and involve more complicated, rocket engine systems included external ballistics applications. Additionally, incompressible or compressible, viscous or hypersonic flow solvers can be included by using Cartesian mesh technology basement.

References

[1] Şahin, M.S. (2011) *Development of a Two-Dimensional Navier-Stokes Solver for Laminar Flows Using Cartesian Grids,* MS Thesis in the Middle East Technical University, 2011.

[2] Çakmak, M. (2009) *Development of a Multigrid Accelerated Euler Solver on Adaptively Refined Two- and Three-Dimensional Cartesian Grids*, MS Thesis in Middle East Technical University, 2009.

[3] Moore, F. G., Moore, L. Y. (2008) *The 2009 Version of the Aeroprediction Code: The AP09*, Aeroprediction, Inc., API Report No. 3, 2008.

[4] Siyahhan, B. (2008) *Two Dimensional Euler Flow Solver on Adaptive Cartesian Grids, MS Thesis in the Middle East Technical University*, 2008.

[5] DeSpirito, J., Heavey, K. R. (2006) *CFD Computation of Magnus Moment and Roll Damping Moment of a Spinning Projectile*, Army Research Laboratory, ARL-RP-131, 2006.

[6] Dan Sunday (2001) http://geomalgorithms.com/a06- intersect-2.html, softSurfer, copyright 2001.

[7] McCoy, R.L. (1999) *Modern Exterior Ballistics*, Schiffer Publishing Ltd., ISBN: 0-7643-0720-7, 1999.

[8] Aftosmis, M.J. (1997) *Solution Adaptive Cartesian Grid Methods for Aerodynamic Flows with Complex Geometries*, 28th Computational Flud Dynamic Lecture Series, Von Karman Institute for Fluid Dynamics, Lecture Series 1997-02.

[9] Möller T. and Trumbore B. (1997) *Fast, Minimum Storage Ray/Triangle Intersection*, Journal of Graphics, gpu and Game Tools, Vol. 2, pp. 21-28, 1997.

[10] Patel, N., Edge, H., Clarke, J. (1995) *Three-Dimensional (3D) Large Fluid Flow Computations for U.S. Army Applications on KSR-1, CM-200, CM-5 and Cray C-90*, Army Research Laboratory, ARL-TR-712, 1995.

[11] Burns, K. A., Deters, K. J., Stoy, S. L., Vukelich, S. R., Blake, W. B. (1993) *MISSLE DATCOM Users's Manual - Revision 6/93*, McDonnel Douglas Aerospace, Wright Laboratory Report WL-TR-93-3043, 1993.

[12] Liou, M.S. and Steffen, C. J. (1993) *A New Flux Splitting Scheme*, Journal of Computational Physics, Vol. 107, p: 23-39, 1993

[13] Sturek, W. B., Nietubicz, C. J., Sahu, J., Weinacht, P. (1992) *Recent Applications of CFD to the Aerodynamics of Army Projectiles*, Army Research Laboratory, ARL-TR-22, 1992.

[14] McCoy, R. L. (1981) *MCDRAG - A Computer Program for Estimating the Drag Coefficients of Projectiles*, Ballistic Research Laboratory Technical Report ARBRL-TR-02293, 1981.

[15] Whyte, R. H. (1973) *Spin-73, An Updated Version of the SPINNER Computer Program*, Picatinny Arsenal, Technical Report 4588, 1973.

[16] Weik, M. H. (1961) *The ENIAC Story*, Ordnance Ballistic Researh Laboratories, Revised 1961

[17] Mann, F. W. (1909) *The Bullet's Flight from Powder to Target*, Munn & Company, New York, 1909.

[18] Shu, R., Zhou, C. and Kankanhalli, M.S., *Adaptive Marching Cubes*, Institute of Systems Science, Nationaal University of Singapore.