Blucher Mechanical Engineering Proceedings
May 2014, vol. 1 , num. 1
www.proceedings.blucher.com.br/evento/10wccm

10th World Congress on Computational Mechanics
8-13 July 2012 • São Paulo • Brazil

A MANUFACTURED SOLUTION TO TEST A CODE THAT SIMULATES A COMPRESSIBLE FLOW WITH SHOCK WAVES

J. L. Doricio[1], H. Ghioti da Silva[2]

[1] College of Integrated Science of Pontal (FACIP) (jldoricio@pontal.ufu.br)
[2] College of Integrated Science of Pontal (FACIP) (ghioti@pontal.ufu.br)
Federal University of Uberlândia - R. 20, n. 1600 - Bairro Tupã - Ituiutaba - MG - CEP 38304-402.

Abstract. *A numerical code aiming to simulate two-dimentional compressible flow, modeled by Euler equations with immersed boundaries, was submitted on a verification test using the Method of Manufactured Solutions. The code was developed to study computational aeroelasticity - flutter analysis by means of forced oscillations on the NACA 0012 airfoil section modeled by the Immersed Boundary Method. Finite Difference scheme using Steger-Warming method of second order of accuracy to the spatial discretization and a Runge-Kutta method of fourth order of accuracy to the time discretization were used. It was created a manufactured solution that mimics a compressible flow with presence of shock waves. The numerical code was submitted by a Mesh Refinement Test and a systematic analysis of actual order of accuracy was performed. The results indicated that the code was free of programming errors and the Method of Manufactured Solutions was a good tool for debugging the code.*

Keywords: *Method of Manufactured Solutions, shock waves, Immersed Boundary Method.*

1. INTRODUCTION

The method of manufactured solutions (MMS) [7] is a general and very powerful approach to code verification. Rather than trying to find an exact solution to a system of partial differential equations, the goal is to “manufacture” an exact solution to a slightly modified set of equations. For code verification purposes, it is not required (in fact, often not desirable) that the manufactured solution be related to a physically realistic problem; recall that verification deals only with the mathematics of a given problem. The general concept behind MMS is to choose the solution a priori, then operate the governing partial differential equations onto the chosen solution, thereby generating analytical source terms. The chosen (manufactured) solution is then the exact solution to the modified governing equations made up of the original equations plus the analytical source terms. Thus, MMS involves the solution to the backward problem: given an original set of equations and a chosen solution, find a modified set of equations that the chosen solution will satisfy. The initial and boundary conditions are then determined from the solution and can be useful if they are boundary conditions previously implemented in the code. In this work, the MMS is employed to perform a systematic analysis of actual order of accuracy of spatial discretization and to verify if the code was free of programming errors. The code tested simulates two-dimentional compressible flow, modeled by Euler equations with immersed boundaries. The code was developed to study computational aeroelasticity - flutter analysis by means of forced oscillations, on the NACA 0012 airfoil section modeled by the free-slip Immersed Boundary Method [4]. Finite Difference scheme using Steger-Warming method of second order of accuracy with MinMod flux limiter to spatial discretization and a Runge-Kutta method of fourth order of accuracy to time discretization was used. The Immersed Boundary Method (IBM) was developed by Peskin [6] to solve problems involving fluid-structure interaction. In his method, the domain is composed by an Eulerian mesh, used to represent the fluid domain, and a Lagrangian mesh, used to represent the elastic immersed boundary. The interaction between the elastic immersed boundary and the fluid is performed by a suavized Dirac delta function, which is the kernel of the IBM. The governing equations, in the IBM, are discretized in Cartesian computational meshes, and this is an advantage of the IBM because simplifies the mesh generation and reduces the complexity of the governing equations. Another advantage of this technique is that the Lagrangian mesh does not need to be align with the Eulerian mesh, this allows to simulate fluid flows with moving immersed boundaries, complex geometries or topological variations [10]. A fixed mesh can be used even for complex moving geometries. Mesh refinement will be required only if improvements in a local flow resolution is desired [5].

2. GOVERNING EQUATIONS

Consider flow of inviscid fluid in a rectangular bidimensional domain Ω with an immersed boundary represented by a closed curve Γ, described by $\mathbf{X}(s,t)$, with $0 \leq s \leq L_b$ and with $\mathbf{X}(0,t) = \mathbf{X}(L_b,t)$, where L_b is the Γ curve length. The Lagrangian variables are represented by upper case letters. The governing equations are given by:

$$\frac{\partial \mathbf{v}}{\partial t} + \frac{\partial \mathbf{s}}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} = \mathbf{h} \, , \tag{1}$$

where:

$$\mathbf{v} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho e \end{bmatrix} \quad \mathbf{s} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (\rho e + p)\, u \end{bmatrix} \quad \mathbf{g} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (\rho e + p)\, v \end{bmatrix} \quad \mathbf{h} = \begin{bmatrix} 0 \\ f_1 \\ f_2 \\ u f_1 + v f_2 \end{bmatrix} \tag{2}$$

$$p = (\gamma - 1)\left[\rho e - \frac{1}{2}\rho\left(u^2 + v^2\right)\right] \, , \tag{3}$$

$$\mathbf{f}(\mathbf{x},t) = \int_0^{L_b} \mathbf{F}(s,t)\delta^2(\mathbf{x} - \mathbf{X}(s,t))ds \, , \tag{4}$$

$$\frac{\partial \mathbf{X}(s,t)}{\partial t} = \mathbf{U}(\mathbf{X}(s,t),t) = \int_\Omega \mathbf{u}(\mathbf{x},t)\delta^2(\mathbf{x} - \mathbf{X}(s,t))d\mathbf{x} \, , \tag{5}$$

$$\mathbf{F}(s,t) = \mathbf{S}(\mathbf{X}(s,t),t) \, . \tag{6}$$

In equations (1) to (6), $\mathbf{x} = (x,\ y) \in \Omega$ is the position vector, $\mathbf{u}(\mathbf{x}, t) = (u(\mathbf{x}, t), v(\mathbf{x}, t))$ is the velocity vector, $p(\mathbf{x}, t)$ is the pressure, $\rho(\mathbf{x}, t)$ is the density and $e(\mathbf{x}, t)$ is the total energy, given by:

$$e = e_i + \frac{1}{2}\left(u^2 + v^2\right) , \tag{7}$$

where e_i is the internal energy. The Eulerian force is given by $\mathbf{f}(\mathbf{x}, t) = (f_1(\mathbf{x}, t), f_2(\mathbf{x}, t))$, and the Lagrangian force is given by $\mathbf{F}(s, t) = (F_1(s, t), F_2(s, t))$. Equation (3) is the constitutive equation for pressure with $\gamma = 1.4$. $\mathbf{S}(\mathbf{X}(s, t), t)$, in equation (6), is a function that express the elasticity of the structure, and represents a free-slip boundary.

3. NUMERICAL METHOD

The Immersed Boundary Method was implemented using the finite differences approach for Eulerian and Lagrangian meshes. Consider a computational domain $\Omega : [\ 0, L_1\ ]\times[\ 0, L_2\ ]$. The fluid variables are defined over an Eulerian mesh $M \times N$ with $\mathbf{x}_{ij} = (x_i, y_j) \in \Omega$ for $i = 0, 1, ..., M-1$ and $j = 0, 1, ..., N-1$, where Δx_{ij} and Δy_{ij} is the size of each mesh division at point $\mathbf{x}_{ij}$. Consider a curve $\Gamma : [\ 0,\ L_b\ ]$, the set of S Lagrangian points $\mathbf{X}_k = (X_k, Y_k) \in \Gamma$ with $k = 0, 1, ..., S - 1$ is used to discretize the immersed boundary, with initial displacement between the points set up as Δs_k.

The classical fourth order Runge-Kutta method [8] is employed to discretize the time variables, and the second order Steger-Warming method [9] with Min-Mod flux separator is employed to discretize the spatial variables. Each stage of the Runge-Kutta method is represented by $\wp$, and $t^{(n+1)} = t^{(n)} + \Delta t$ represents the instant of time and Δt represents the size of pass in time. According to [8], this numerical scheme is stable for Courant Friedrichs-Lévi (CFL) number of $2\sqrt{2}$. The numerical method is given by:

- **Preliminary stage of pass in time:**

  1. The Lagrangian variables, that represent the immersed boundary, are set up in $\wp = 0$ at time $t^{(n)}$ as:

  $$\mathbf{F}_{(0)}(\ s\ ) = \mathbf{F}^{(n)}(\ s\ ) , \quad \mathbf{U}_{(0)}(\ s\ ) = \mathbf{U}^{(n)}(\ s\ ) , \quad \mathbf{X}_{(0)}(\ s\ ) = \mathbf{X}^{(n)}(\ s\ ) .$$

  2. The Eulerian variables, that represent the fluid field, are set up at $\wp = 0$ as:

  $$\mathbf{f}_{(0)}(\mathbf{x}) = \mathbf{f}^{(n)}(\mathbf{x}) , \quad \mathbf{v}_{(0)}(\mathbf{x}) = \mathbf{v}^{(n)}(\mathbf{x}) .$$

- **Intermediate stages of pass in time for $\wp = 0, 1, 2$ and 3:**

  1. The Lagrangian force $\mathbf{F}_{(\wp+1)}(\ s\ )$ is calculated in the immersed boundary using the configuration of $\mathbf{X}_{(\wp)}(\ s\ )$, as follows:

  $$\mathbf{F}_{(\wp+1)}(\ s\ ) = \mathbf{S}(\mathbf{X}_{(\wp)}(s, t), t^{(n)}) , \tag{8}$$

  It is necessary that the elastic boundary stays closed to the original configuration of the structure with very small displacements and representing a free-slip boundary type. This is achieved adequately choosing a forcing term $\mathbf{S}(\mathbf{X}(s, t), t)$, as follows:

  $$\mathbf{S}(\mathbf{X}(s, t), t) = \kappa \operatorname{proj}_{\mathbf{n}(s,t)} (\mathbf{X}^e(s, t) - \mathbf{X}(s, t)) + \zeta \operatorname{proj}_{\mathbf{n}(s,t)} (\mathbf{U}^e(s, t) - \mathbf{U}(s, t)) , \tag{9}$$

where $\kappa \gg 1$ represents elasticity, $\zeta > 1$ represents damping, $\mathbf{n}(s,t)$ is the normal vector at the point $\mathbf{X}(s,t)$ of the immersed boundary and $\mathrm{proj}_{\mathbf{a}}\mathbf{b}$ is the orthogonal projection operator of vector $\mathbf{b}$ on vector $\mathbf{a}$. Equation (9) connects the points $\mathbf{X}(s,t)$ of the immersed boundary with the equilibrium points $\mathbf{X}^e(s,t)$ using a generalized Hooke law.

2. The Lagrangian force $\mathbf{F}_{(\wp+1)}(\ s\ )$ is interpolated into the Eulerian field to determine the value of $\mathbf{f}_{(\wp+1)}(\mathbf{x})$, as follows:

$$\mathbf{f}_{(\wp+1)}(\mathbf{x}) = \sum_{s} \mathbf{F}_{(\wp+1)}(\ s\ )\delta_h^2(\mathbf{x} - \mathbf{X}_{(\wp)}(\ s\ ), \mathbf{n})\Delta s\ , \tag{10}$$

where the delta function is given by:

$$\delta_h^2(\mathbf{x},\mathbf{n}) = \frac{1}{\Delta x \Delta y}\mathfrak{d}\left(-\frac{n_y x}{\Delta x\,(n_y^2+n_x^2)} + \frac{n_x y}{\Delta x\,(n_y^2+n_x^2)},\quad -\frac{n_x x}{\Delta y\,(n_y^2+n_x^2)} - \frac{n_y y}{\Delta y\,(n_y^2+n_x^2)}\right)\ , \tag{11}$$

where $\mathbf{n} = (n_x,\ n_y)$ is the normal vector at the Lagrangian point $\mathbf{X}_{(\wp)}(\ s\ )$ of the immersed boundary and

$$\mathfrak{d}(\mathfrak{x},\mathfrak{z}) \equiv \begin{cases} \dfrac{25}{8}\,\dfrac{e^{(-\frac{25}{16}\mathfrak{x}^2 - \frac{25}{16}\mathfrak{z}^2)}}{\pi} & 0 \le \mathfrak{z} < 5 \\ 0 & \textit{otherwise} \end{cases}\ . \tag{12}$$

This delta function interpolates data from external region of the immersed structure. This gaussian function can be truncated with radius of five cells nodes. Doing this, the error between the integral and unit value is

$$\left|\, 1\ -\ \iint_{-5}^{5} \mathfrak{d}(\mathfrak{x},\mathfrak{z})\ d\mathfrak{x}\ d\mathfrak{z}\,\right| < 0.2 \times 10^{-17}\ .$$

then, truncating this function with radius of five cells nodes the errors is sufficiently small to avoid excessive roundoff and truncation errors and avoid unnecessary computational cost.

3. The Euler equations given by (1) are solved using the force term $\mathbf{f}_{(\wp+1)}(\mathbf{x})$ at stage $(\wp+1)$ of Runge-Kutta method and with the second order Steger-Warming scheme with Min-Mod flux separator. Consider the governing equation (1), that can be represented by:

$$\frac{\partial \mathbf{v}}{\partial t} + P(\mathbf{v}) = 0\ , \tag{13}$$

where

$$P(\mathbf{v}) \equiv \frac{\partial \mathbf{s}}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} - \mathbf{h}\ . \tag{14}$$

At stage $(\wp + 1)$, $\mathbf{v}_{(\wp+1)}$ is given by:

$$\begin{aligned}
\mathbf{v}_{(1)} &= \mathbf{v}_{(0)} - \frac{\Delta t}{2} P(\mathbf{v}_{(0)}) \\
\mathbf{v}_{(2)} &= \mathbf{v}_{(0)} - \frac{\Delta t}{2} P(\mathbf{v}_{(1)}) \\
\mathbf{v}_{(3)} &= \mathbf{v}_{(0)} - \Delta t P(\mathbf{v}_{(2)}) \\
\mathbf{v}_{(4)} &= \mathbf{v}_{(0)} - \frac{\Delta t}{6} \left[P(\mathbf{v}_{(0)}) + 2P(\mathbf{v}_{(1)}) + 2P(\mathbf{v}_{(2)}) + P(\mathbf{v}_{(3)}) \right]
\end{aligned} \tag{15}$$

where $\mathbf{v}_{(0)}$, $\mathbf{v}_{(1)}$, $\mathbf{v}_{(2)}$, $\mathbf{v}_{(3)}$ and $\mathbf{v}_{(4)}$ are the fluid variables defined by equation (2) in the intermediate stage of the Runge-Kutta time integration method.

The external boundaries of the Eulerian domain are treated with Rieman invariants representing open atmosphere to avoid reflection [1].

4. The Eulerian velocity $\mathbf{u}_{(\wp+1)}(\mathbf{x})$, calculated by the previous pass, is interpolated into the immersed boundary as follows:

$$\mathbf{U}_{(\wp+1)}(s) = \sum_{\mathbf{x}} \mathbf{u}_{(\wp+1)}(\mathbf{x}) \delta_h^2(\mathbf{x} - \mathbf{X}_{(\wp)}(s), \mathbf{n}) \Delta x \Delta y \, . \tag{16}$$

where $\delta_h^2(\mathbf{x}, \mathbf{n})$ is the delta function defined by equation (11).

5. The new position of the points that represents the immersed boundary, $\mathbf{X}_{(\wp+1)}(s)$, are updated using the Runge-Kutta method in the correspondent $(\wp + 1)$ pass:

$$\begin{aligned}
\mathbf{X}_{(1)} &= \mathbf{X}_{(0)} + \frac{\Delta t}{2} \phi(\mathbf{U}_{(0)}) \\
\mathbf{X}_{(2)} &= \mathbf{X}_{(0)} + \frac{\Delta t}{2} \phi(\mathbf{U}_{(1)}) \\
\mathbf{X}_{(3)} &= \mathbf{X}_{(0)} + \Delta t \phi(\mathbf{U}_{(2)}) \\
\mathbf{X}_{(4)} &= \mathbf{X}_{(0)} + \frac{\Delta t}{6} \left[\phi(\mathbf{U}_{(0)}) + 2\phi(\mathbf{U}_{(1)}) + 2\phi(\mathbf{U}_{(2)}) + \phi(\mathbf{U}_{(3)}) \right]
\end{aligned} \tag{17}$$

where $\phi(\mathbf{V}) \equiv \mathrm{proj}_{\mathbf{n}(s,t)} \mathbf{V}(s, t)$.

- **Final stage: the fluid variables are updated at time $t^{(n+1)}$:**

1. The Lagrangian variables are set up at $t^{(n+1)}$ as follows:

$$\mathbf{F}^{(n+1)}(s) = \mathbf{F}_{(4)}(s) \, , \quad \mathbf{U}^{(n+1)}(s) = \mathbf{U}_{(4)}(s) \, , \quad \mathbf{X}^{(n+1)}(s) = \mathbf{X}_{(4)}(s) \, .$$

2. The Eulerian variables are set up at $t^{(n+1)}$ as follows:

$$\mathbf{f}^{(n+1)}(\mathbf{x}) = \mathbf{f}_{(4)}(\mathbf{x}) \, , \quad \mathbf{v}^{(n+1)}(\mathbf{x}) = \mathbf{v}_{(4)}(\mathbf{x}) \, .$$

4. CODE VERIFICATION

The code developed using the numerical algorithm presented in this work was implemented using C^{++} language and verified using the Method of Manufactured Solutions, [2, 3, 7]. One manufactured solution was constructed by the authors for density ρ, velocity components u, v and pressure p as follows:

$$\rho(\mathbf{x},0) = \frac{\eta}{800}x^3 + \frac{\eta}{800}y^3 + \frac{3\eta}{4} \qquad where \qquad \eta = \begin{cases} 1 & if x \le \frac{3}{2} \\ 1.1 & if x > \frac{3}{2} \end{cases}, \tag{18}$$

$$u(\mathbf{x},0) = \frac{\eta}{3}\sin(\frac{1}{5}y) + \frac{\eta}{3}\sin(\frac{1}{5}x) + \frac{\eta}{2} \qquad where \qquad \eta = \begin{cases} 1 & if x \le \frac{3}{2} \\ 1.1 & if x > \frac{3}{2} \end{cases}, \tag{19}$$

$$v(\mathbf{x},0) = \frac{\eta}{4}\cos(\frac{1}{5}y) + \frac{\eta}{4}\cos(\frac{1}{5}x) + \frac{4\eta}{7} \qquad where \qquad \eta = \begin{cases} 1 & if x \le \frac{3}{2} \\ 1.1 & if x > \frac{3}{2} \end{cases}, \tag{20}$$

$$p(\mathbf{x},0) = \frac{\eta}{7}\sin(\frac{1}{5}x) + \frac{\eta}{7}e^{(1/4y)} \qquad where \qquad \eta = \begin{cases} 1 & if x \le \frac{3}{2} \\ 1.1 & if x > \frac{3}{2} \end{cases}. \tag{21}$$

Although the manufactured solution involves the soft functions $sin(x)$, $cos(x)$ and $exp(x)$, it is interesting for this proposal because of the high derivative terms presented in the discretization terms from the numerical solution are not nulls. It allows a systematic study of order of accuracy of the code via a mesh refinement test [3]. Figures 1 show the plot of the manufactured solutions of equations (18)-(21). The source terms are given by:

$$\mathscr{F} = \frac{\partial \mathbf{s}}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} - \mathbf{h}, \tag{22}$$

where:

$$\mathscr{F} = \begin{bmatrix} \mathscr{F}_1 \\ \mathscr{F}_2 \\ \mathscr{F}_3 \\ \mathscr{F}_4 \end{bmatrix} \quad \mathbf{s} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (\rho e + p)\, u \end{bmatrix} \quad \mathbf{g} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (\rho e + p)\, v \end{bmatrix} \quad \mathbf{h} = \begin{bmatrix} 0 \\ f_1 \\ f_2 \\ uf_1 + vf_2 \end{bmatrix} \tag{23}$$

$$e = \frac{p}{\rho\,(\gamma - 1)} + \frac{1}{2}\left(u^2 + v^2\right), \tag{24}$$

The values of $\mathscr{F}_1$, $\mathscr{F}_2$, $\mathscr{F}_3$ and $\mathscr{F}_4$ are determined inserting the values of ρ, u, v and p given by equations (18) to (21) in equation (22). The source terms are obtained using the software of algebra manipulation Maple V. Figures 2 show the plot of the source terms.

The code verification was performed using seven cartesian meshes, with $\Delta x = \Delta y = 0.4$, $\Delta x = \Delta y = 0.2$, $\Delta x = \Delta y = 0.1$, $\Delta x = \Delta y = 0.05$, $\Delta x = \Delta y = 0.025$, $\Delta x = \Delta y = 0.0125$ and $\Delta x = \Delta y = 0.00625$, denoted by Mesh 1, Mesh 2, ..., and Mesh 7 in a rectangular domain 3×3. The numerical error analysis in these meshes was performed using the mean norm:

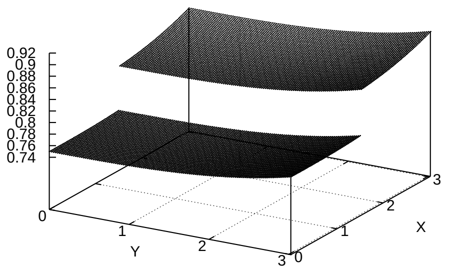
$$||\mathfrak{H}||_{L_1} \equiv \frac{1}{MN} \sum_{0 \le i,j < M,N} \mathfrak{H}_{ij}.$$

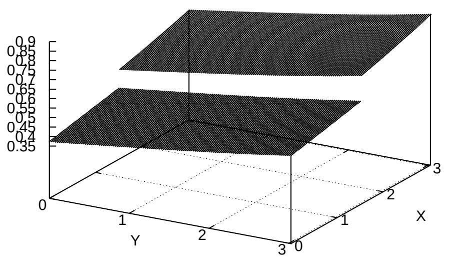
and the maximum norm:

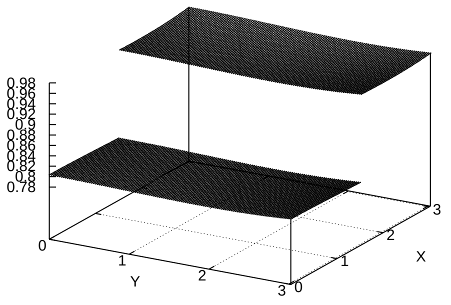
$$||\mathfrak{H}||_{\infty} \equiv \max_{0 \le i,j < M,N} |\mathfrak{H}_{ij}|.$$

according to [3].

In this work, tests involving analysis of Euler and Lagrange forces are not performed. The proposal of this study was to reach reliability that programming errors were negligible. Study of performance and/or ability of this code to simulate immersed boundary could be a good test and the authors classify it as solution verification test and should be performed after a code verification test [7]. Further, test of the choice of the meshes and preliminar tests to the problem of interest can be found in [4].


a)


b)


c)

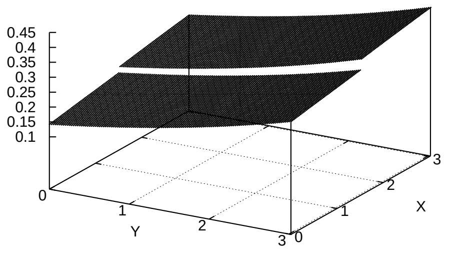
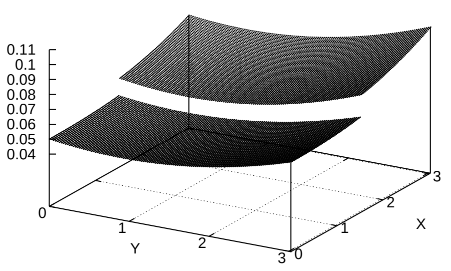
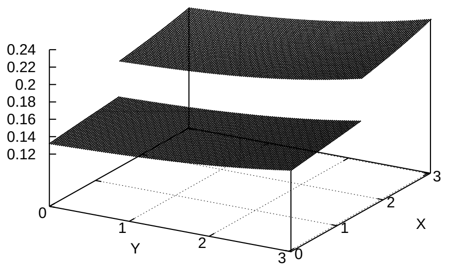
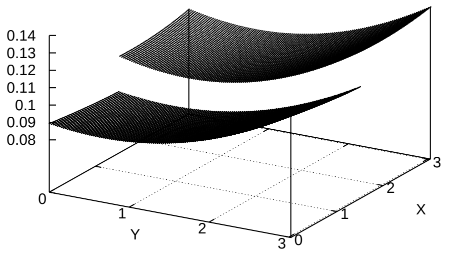

d)

Figure 1. Analitical solutions, ρ, u, v, p.


a)


b)


c)

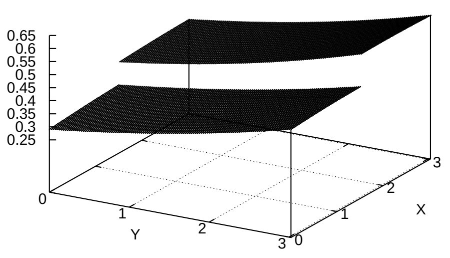

d)

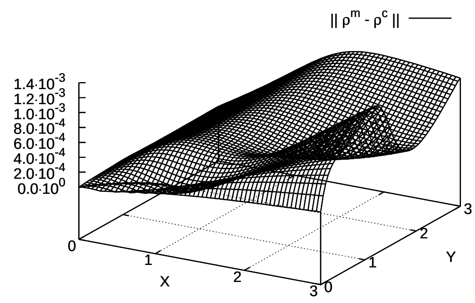
Figure 2. Analitical forcing functions $\mathscr{F}_1, \mathscr{F}_2, \mathscr{F}_3, \mathscr{F}_4$.

5. RESULTS

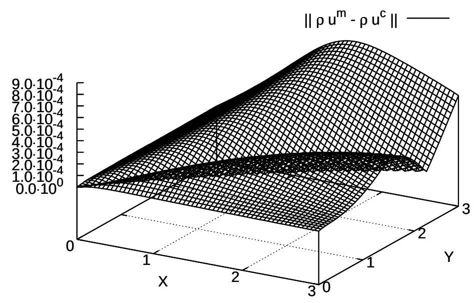
The main interest in this test was to verify the implementations of the equations attempting to eliminate implementation errors in the code. In fact, the chosen manufactured solution can be manipulated to perform a good test of boundary condition implementations and evaluation of discretization error through of mesh refinement. The computational code using the forcing terms $\mathscr{F}_1$, $\mathscr{F}_2$, $\mathscr{F}_3$ and $\mathscr{F}_4$ was executed until the convergency stop criterion $||\xi^{(k)} - \xi^{(k-1)}||_\infty \leq 10^{-12}$ has been reached. Where ξ represents the fluid variables and k represents the present time step. Figures 6a to 6d show the convergency for meshes 1 to 4 using the code executed with order one and two with the descontinuity not present (i.e., $\eta = 1$, for $x \in \mathbb{R}$), and with MinMod flux limiter with descontinuity present (i.e., $\eta = 1$, for $x \leq \frac{3}{2}$ and $\eta = 1.1$ otherwise). In all cases, the convergency is achieved. It is important to note that the convergency for order 2 code is more faster than for order 1 code because of the discretization errors and the velocity of convergency with MinMod flux limiter remained between order 1 and order 2, as expected. The aim of the flux limiter is decrease the code order from two to one in the regions where there is high gradient of the fluid variables, like at shock waves, to ensure that the solution is physically consistent.

Figures 3 to 5 show the errors in the solution for the fluid variables ρ, ρu, ρv and p using the codes of order 1 and 2 with $\eta = 1$, for $x \in \mathbb{R}$, and with MinMod flux limiter using $\eta = 1$, for $x \leq \frac{3}{2}$ and $\eta = 1.1$ otherwise at meshes 1 to 4. These figures show that the magnitude of the error in the fluid variables reduces with the mesh refinement, this pattern is necessary so that the code is free of implementation errors. In addition, the magnitude of the error in the code of order 1 is greater than the magnitude of the error in the code of order 2. This pattern is also expected. The complete code with flux limiter presented errors with magnitude close to magnitude obtained for code of order 2. It can be observed that the errors distribution is higher at regions close to descontinuity where the order 1 is applied by the flux limiter. Using these error, shown by figures 3 to 5 in all meshes simulated, the numerical code orders for ρ, ρu, ρv and p variables using the mean norm executed with order 1, 2 and with flux limiter codes was calculated and them is shown by figure 7, and using the maximum norm is shown by figure 8.

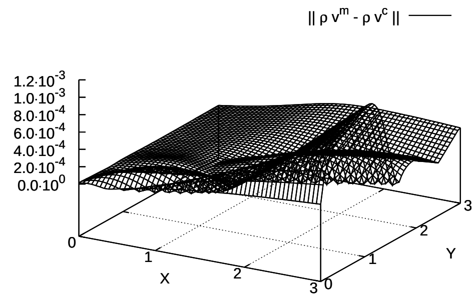
Interestingly, in most cases, for the coarser mesh, the error behaves with higher order than the expected theoretical order, but in some cases the order is less, see figures 7d and 8d for order 2 code. However, in all cases, the decaiment of the error tends to the theoretical order of the numerical discretization. According to these figures, the code using the flux limiter presents errors higher than the errors obtained by the code of order 2 and lower then order 1. But is interesting to note, that the errors in the code with flux limiter decay at order 2. The orders obtained by use of mean norm are consistent with use of the maximum norm. But in some cases, the results obtained by applying these norms may be different due to discretization errors prevalent in some regions of the domain, like walls or descontinuities, according to [3]. In this case, the correct order is obtained using the maximum norm. That difference in the results has not happened in the tests performed in this work.



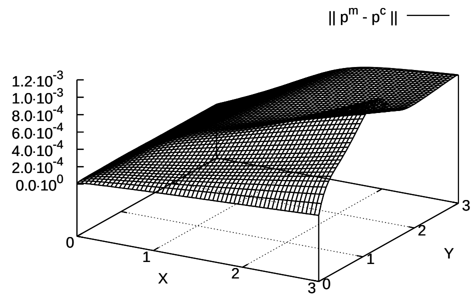
a)



b)

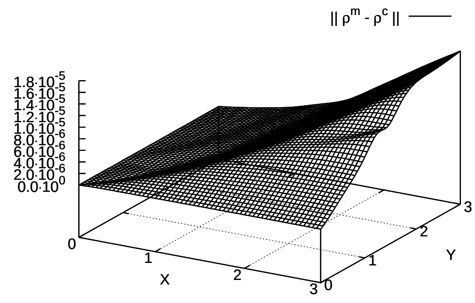


c)

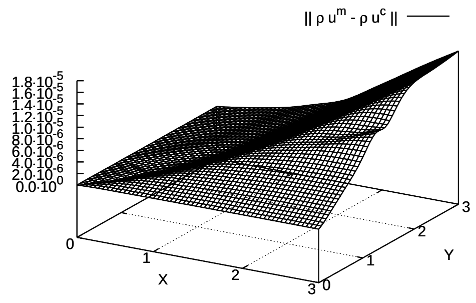


d)

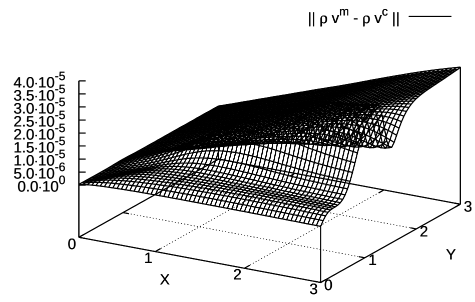
Figure 3. Errors using the order 1 code for mesh 4.



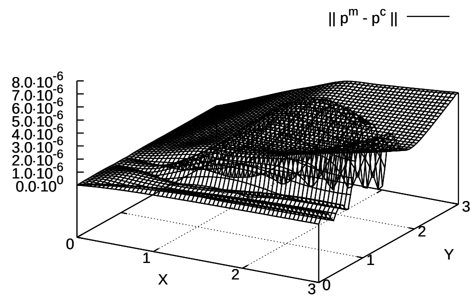
a)



b)

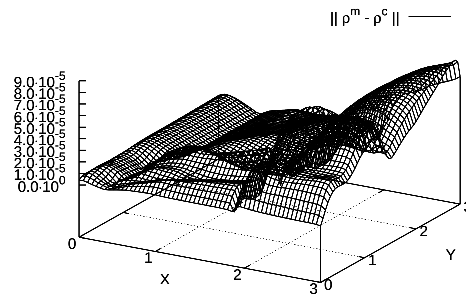


c)

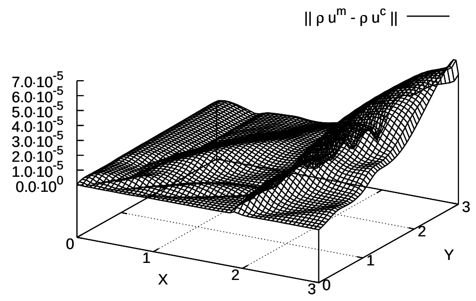


d)

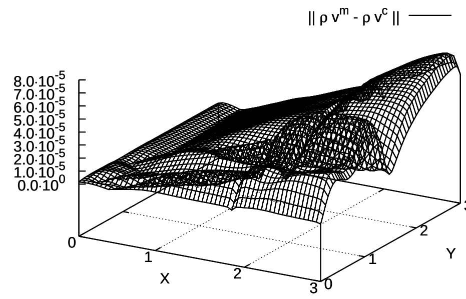
Figure 4. Errors using the order 2 code for mesh 4.



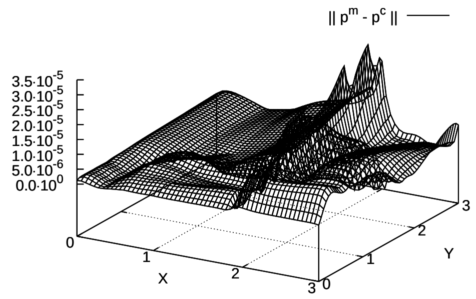
a)



b)

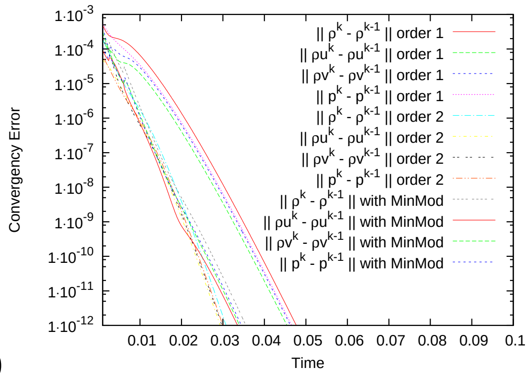


c)

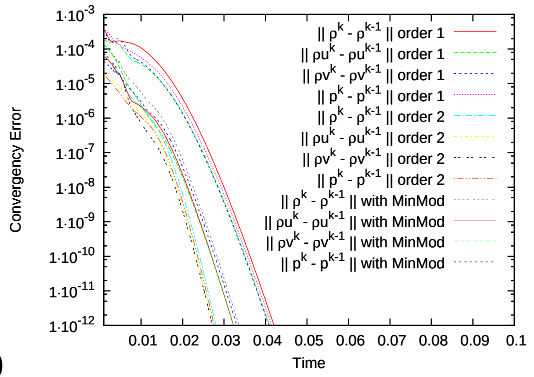


d)

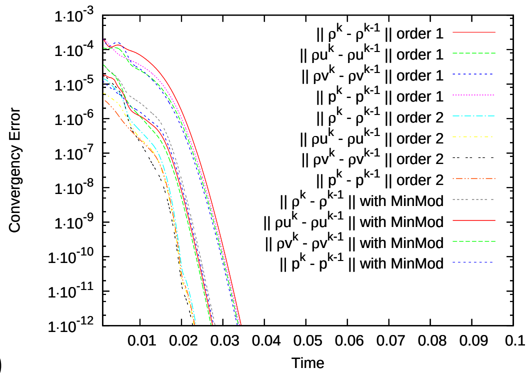
Figure 5. Errors using the MinMod flux limiter code for mesh 4.



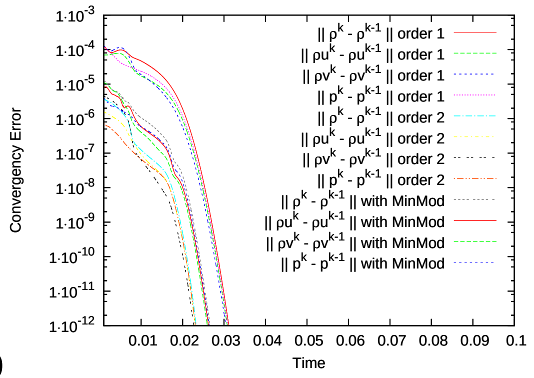
a)



b)



c)



d)

Figure 6. Convergency for mesh 1 to mesh 4.

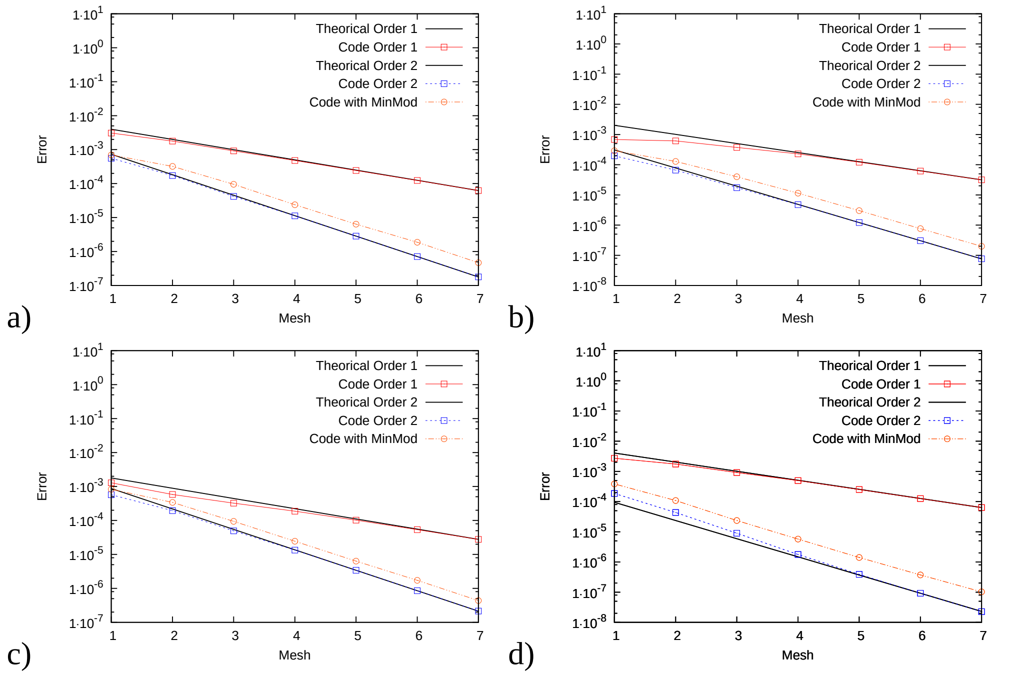


Figure 7. Code order for ρ, ρu, ρv and p fluid variables using the mean norm.

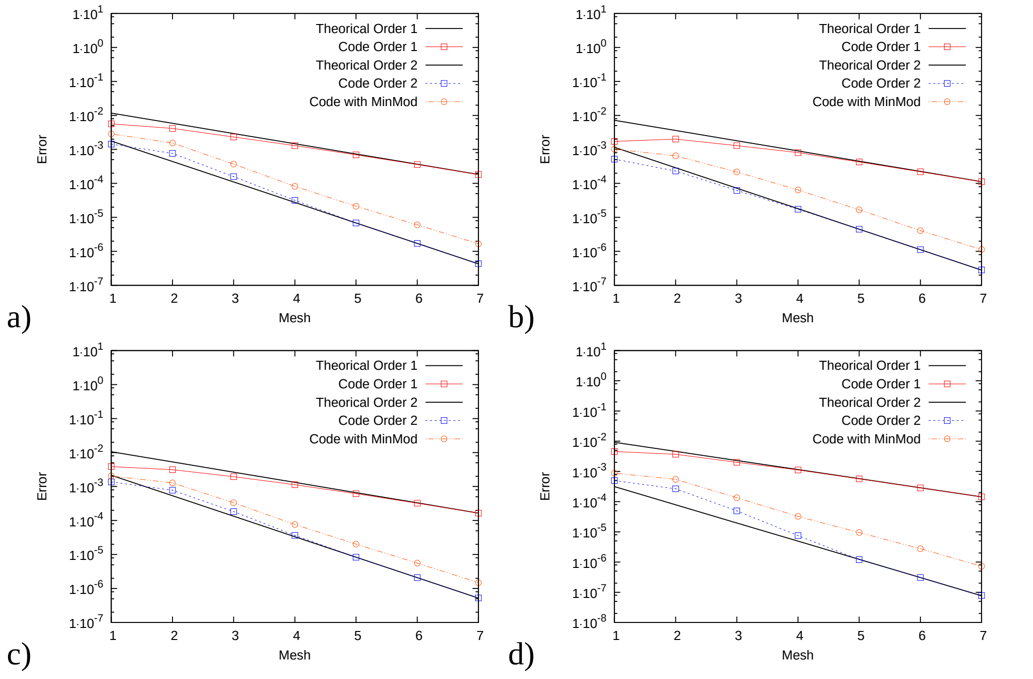


Figure 8. Code order for ρ, ρu, ρv and p fluid variables using the maximum norm.

6. CONCLUSIONS

In this work, the code that simulates compressible flow modeled by Euler equations with a descontinuity that simulates a shock wave was tested by means of the manufactured solutions method. It was shown that the theoretical order of the numerical method employed for the discretization of spatial variables is consistent with the order calculated numerically. It was also shown that the orders obtained by the use of mean norm are consistent with the use of the maximum norm. The results obtained indicate that the code is free of programming errors for the problem simulated and the order of the numerical method tends to theoretical order and is very close after mesh 4.

References

[1] Cesar Augusto Buonomo. Técnica de fronteiras imersas com formulação viscosa e compressível. Master's thesis, Instituto Tecnológico de Aeronáutica, 2004.

[2] Clarence O. E. Burg and Vasanth K. Murali. Efficient code verification using the residual formulation of the method of manufactured solutions. pages 1–13. 34th AIAA Fluid Dynamics Conference, 2004.

[3] H. G. da Silva, L. F. Souza, and M. A. F. Medeiros. Verification of a mixed high-order accurate dns code for laminar turbulent trasition by the method of manufactured solutions. *International Journal for Numerical Methods in Fluids*, 64:336–354, 2010.

[4] José Larcio Doricio. *Estudo da aplicabilidade do método de fronteira imersa no cálculo de derivadas de Flutter com as equações de Euler para fluxo compressível*. PhD thesis, Escola de Engenharia de So Carlos - USP, 2009.

[5] M. N. Linnick and H. F. Fasel. A high-order immersed boundary method for unsteady incompressible flow calculations. *AIAA 2003-1124*, 2003.

[6] Charles S. Peskin. *Flow Patterns around Heart Valves: A digital computer method to solve the equations of motion*. PhD thesis, Albert Eistein College of Medicine, 1972.

[7] Christopher J. Roy. Review of code and solution verification procedures for computational simulation. *Journal of Computational Physics*, 205:131–156, 2005.

[8] Stefan Schreier. *Compressible Flow*. John Wiley & Sons, 1982.

[9] J. L. Steger and R. F. Warming. Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods. *Journal of Computational Physics*, 40(2):263–293, 1981.

[10] T. Ye, R. Mittal, H. S. Udaykumar, and W. Shyy. Incompressible flows with complex immersed boundaries. *Journal of Computational Physics*, 156:209–240, 1999.