INCOMPRESSIBLE NAVIER STOKES EQUATIONS SOLUTION USING BLOCK NESTED CARTESIAN GRID

Chr.G. Georgantopoulou•, G.A.Georgantopoulos•† and S. Tsangaris

Fluids Section, School of Mechanical Engineering
National Technical University of Athens
Heroon Polytechniou 9, 15783, Zografou-Athens, Greece
e-mail: christine@fluid.mech.ntua.gr

†Aerodynamic Lab
Hellenic Air Force Academy
Dekeleia Attikis, Greece
e-mail: ggeorga@hafa.gr

Keywords: *Cartesian grids, incompressible flows, grid generation, mesh block refinement*

Abstract

A method for the solution of the incompressible Navier-Stokes equation for the prediction of flows inside domains of arbitrary shaped bounds by the use of Cartesian grids with block-refinement in space is presented. In order to avoid the complexity of the body fitted numerical grid generation procedure, we use a saw tooth method for the curvilinear geometry approximation. The refinement method is based on the use of a sequence of nested rectangular meshes in which numerical simulation is taking place. The method is applied for laminar flows and based on a cell-centre approximation projection. We present the numerical simulation of both an internal and an external flow, about the fluid flow inside a stenosed tube and around a symmetric airfoil respectively. The utility of the algorithm is tested by comparing the convergence characteristics and accuracy to those of the standard single grid algorithm. The Cartesian block refinement algorithm can be used in any complex curvilinear geometry simulation, to accomplish a reduction in memory requirements and the computational time effort.

1 Introduction

The use of Cartesian grids in solving fluid flow problems started decades ago. In fact it was almost abandoned when the body-fitted structured curvilinear (BFC) grid approach came in, because the boundary surface is fitted with a new co-ordinate line based on the body contour, [1]. The main problem is that if you have to simulate a complex multiply connected domain with sharp boundaries it is difficult to automatically generate a grid of good quality. Current algorithms in BFC are still strongly dependent on the problem to be solved and required a lot of computational and human time effort. So in order to avoid these partial problems, recently has become a great development of Cartesian grids. As a matter of fact there are a lot of reasons to prefer this kind of grid generation applied to curvilinear geometries, against to the body-fitted grid methods. At first, the specification of the geometry description needed is easier than the other methods because it involves only a set of cells of co-dimension one with respect to the problem domain. In Cartesian grid methods the numerical grid is generated automatically containing simplified data

structures and formulations for the numerical fluxes. The Cartesian grid generation was used by Clarke [2] and Falle and Giddings [3] to calculate steady compressible flows [4]. Coirier and Powell [5] used a Cartesian methodology for steady transonic solutions Euler's equations and in [6] performed accuracy and efficiency assessments of the method. It's a cell-centred method with an interesting treatment of boundary conditions. Smith and Johnston [7] develop a grid generation procedure that uses Cartesian embedded unstructured approach for complex geometries.

Adaptive mesh refinement algorithms have been used extensively to solve a variety of problems in hypervolic conservation lows and have more recently been extended to incompressible flows [8, 9, 10]. Wang [11] develops a quadtree-based adaptive Cartesian/Quadrilateral grid generator and flow solver based on cell cutting-[12, 13, 14], and Deister [15] presents a refined Cartesian grid based in octa-tree.

In the present paper we present a Cartesian grid approach based on a saw-tooth method for the curvilinear geometries bounds approximation. This technique is based on Chen, Lee and Patakar [16], where they present the saw-tooth Cartesian method for heat transfer problem on a complex geometry. The emphasis is placed to present an improved accurate Cartesian approximation of curvilinear geometries and the corresponding fluid solution in comparison with those of the body fitted structured curvilinear method. We apply a nested refinement algorithm based on that of Jesse [17], Martin and Collela [12], and Berger and Collela [18], in which refined regions are organized into unions of a small number of nested rectangular blocks. Refinement is performed in space and the method is cell-centred finite volume, which allows the use of a single set of cell-centred solvers. The block refinement is automatic and it can be applied in any complex curvilinear geometry. It's applied to steady, incompressible flow fields for Navier Stokes numerical simulation [19]. The flow solver is based on a pseudo-compressibility technique Pappou and Tsangaris [20].

2 Domain Discretization

The main problem in Cartesian grid generation for a curvilinear geometry is that we have to use a technique to create an approximate Cartesian bound as close to the initial curvilinear one as possible. The new approximate bound are parted only by the use of grid lines, on x or z-axis either. The method is used, called saw-tooth and is has been chosen as the most appropriate for the finite volume cell centered numerical simulation of flow fields. This method provides independence and automation of grid generation for problems with complex boundaries, with or without existence of an analytical function. The main advantage of saw-tooth is that you can create any approximate Cartesian geometry, if you only have a set of data points, and so you can simulate any flow field even its geometry analytical function is unknown.

The main problem of the above method is that if you want to decrease the approximation error with the initial curvilinear geometry, you have to cluster the used uniform grid. In many cases a huge grid size is needed, and this is unproductive. In order to overcome this problem we create a block refinement grid wherever the flow domain demands. We define the block's bounds and we analyze the way of variable's value transfer between coarse and fine interfaces. It will be shown that the method is stable and accurate, because it provides satisfying results and minimizes the computational memory.

2.1 Saw tooth method

We present an automatic grid generation technique that can be used for any complex curvilinear geometry. At first we project the original contour of the curvilinear geometry onto a Cartesian grid. This complex contour is described by a set of data points on x or z-axis either. We have to control if the contour segment between two neighbour data points varies monotonically with respect to both x or z directions. If we discover that this rule doesn't occur we have to cluster the Cartesian grid. The second step of the procedure is the specification of the approximated Cartesian points for the representation of the geometry by using the saw tooth method

(figure 2). If an original data point is on x-axis, we calculate the distance between this and its neighbouring grid nodes in the same direction (x). According the smallest distance we choose the corresponding grid node as the Cartesian approximated point, (figure 1).

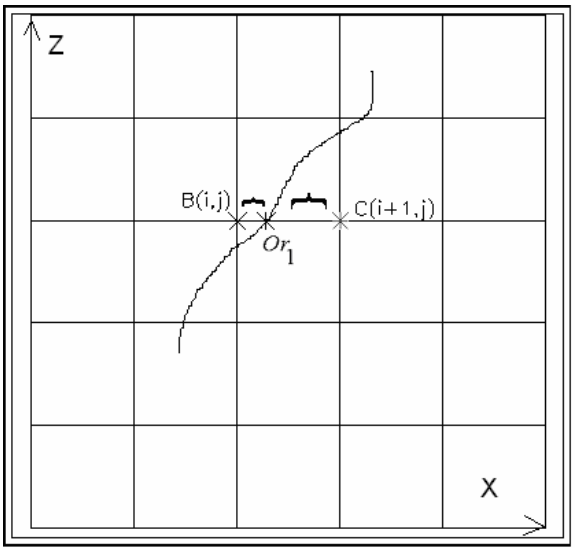


Fig. 1: Specification of approximate Cartesian points

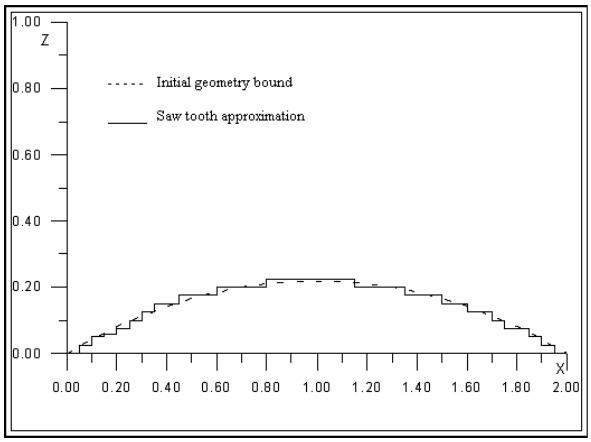
We create a contour by connecting these points and the complex bound has been transformed to a Cartesian geometry bound (figure 2). In this work we mainly use the rule of minimum distance in relation to the final grid and the resulting numerical simulation of the flow field.

2.2 Block - refinement mesh method

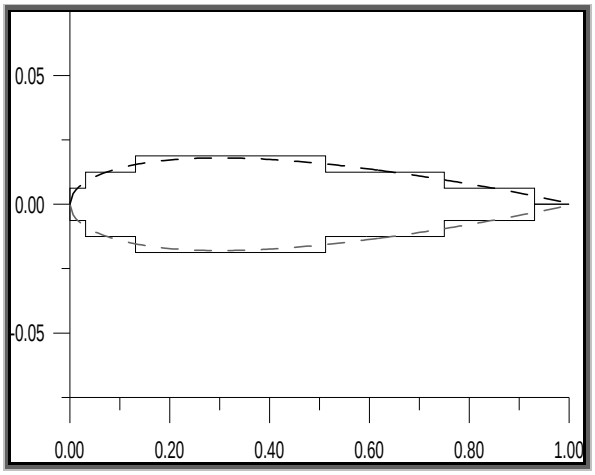
We choose a block refinement technique by the use of a hierarchical structured grid approach. The method is based on using a sequence of nested rectangular meshes in which numerical simulation is taking place (figure 3). The whole domain is a rectangle whose sides lie in the coordinate directions. We simulate the domain based in as many refine grids as we need. Although the discrete solution must be independent of how the refine grids are composed, we have to follow some criterion, in order to succeed grid hierarchy and properly nested grids:

a) A fine grid starts and ends at the corner of a cell in the next coarser grid [24].
b) All the sub-grids must be rectangular.
c) Numerical simulation fist started by the coarsest grid and follows to the next level.
d) The neighboring girds must be only one level up or down.

A physical domain's point can be contained in several grids. The solution of the variables in this point will be taken from the finest grid containing the point



(a)



(b)

Fig. 2: (a) (b) Saw - tooth geometry approximation

2.3 Block- nested refinement algorithm

The proposed nested algorithm contains several levels of grids. We create a coarse level at

the beginning and we solve the domain. We name this coarse level *m=0* and each next refine sub – grid is named *m+1*. The coarsest grid is uniform on x and z direction respectively. We define an integer refinement factor, like [9], $I = dx_m / dx_{m+1} = dz_m / dz_{m+1}$. For convenience the above factor should be a second power.

As we have created the coarse grid we simulate the flow field and calculate the variables. At this time the coarse-fine interfaces are neglected since no information from the finer level is available yet. Of course the geometry approximation error is quite big but this is not a problem, as we have just a prediction for the fluxes near the geometry bound. We have already defined the limits of the refinement levels and we proceed the calculation to the next refinement level. The sub-grids bounds must lie on a grid line of the previous level grid. As we use staggered grids and the variable values are expressed on the cell's centre, we consider pseudo – cells all around the physical domain and the sub – grids too. By this way we estimate the variables using interpolation between pseudo – cells and their neighbor cells. The pseudo-cells of each sub-grid *m* are lying on the level *m-1*. We continue this process for all the sub- grids. As we have fulfilled the simulation in all sub-grids and we have the flow field results at $m_{\max}$ level, we resolve the problem in the coarser levels again to ensure conservation. In this step of the procedure we have to be careful because we can apply the numerical simulation only in rectangular sub-grids. As we resolve in *m-1* levels, all of them have to be rectangular. We find a new solution, this time by the influence of the fine levels. In addition we must satisfy both Dirichlet and Neumann matching conditions along coarse-fine and fine- coarse interfaces. That's why we give the velocity values, but we solve for pressure. With nested grids, each grid is separately defined and has its own solution vector, so that a grid can be advanced independently of other grids, except for the determination of its boundary values. The information exchange between two successive levels is described in the next section.

The grid algorithm is comprised of multiple levels. As we have already created the cartesian approximate geometry bound, the grid generation and the numerical simulation procedure is as follows:

- Create a coarse Cartesian grid (level m=0), simulate, (imposition of proper boundary conditions) and solve the flow field.
- Transfer the solution to the next grid level (m+1) by using the appropriate boundary conditions.
- Solve the flow field on the new sub-domain.
- Transfer the solution to the next level (m+2) with new boundary conditions.
    - .
    - . (Repeat the procedure for all the levels)
    - .
- Simulate and solve the flow on the last sub-domain (level mmax).
- Transfer the solution to the coarser grid level (mmax-1) as its boundary conditions.
- Solve the sub domain with the influence of the refined grid results.
    - .
    - . (Repeat the procedure for all the levels)
    - .
- Solve the coarsest-initial sub domain (level m=0).
- Take the solution of the variables by the finest grid.

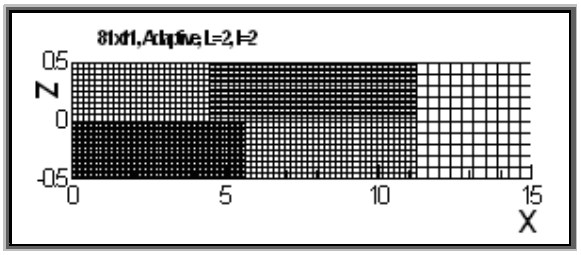


Fig. 3: Block refinement grids, I=2, m=2.

2.4 Boundary Conditions

If the grids are adjacent, the boundary conditions of one grid are provided by the other. If they aren't adjacent, the boundary conditions are established by either coarser level condition or by the physical boundary condition.

For a grid level m, the bordering cell values are provided using values from adjacent level, where they are available, or from physical boundary conditions. The data transfer can be done either to a coarse –fine interface, either to a fine-coarse one. For both of these cases, we can linearly interpolate or bilinearly interpolate. In the present paper we linearly interpolate as described below. As we have already mentioned, the sub-grid bounds are absolutely adjacent. The pseudo-cell of each sub-grid belongs to the boundary cells of the previous grid level. So when we solve in a refined level (m) we neglect the 'pseudo-cells' of the coarse level (m-1), and we use for the refined boundary transfer, the boundary cells by level m-1. That's very important because any other option will provide inaccurate solutions at whole flow field.

Let's consider that we have already solved into the initial coarse grid and we have to continue the numerical simulation into a sub-grid. In order to specify the boundary conditions at coarse grid and sub-grid interfaces, we represent $u^{m+1}(i,k)$ and $w^{m+1}(i,k)$, the values of the velocity components on the sub-grid pseudo-cells. The $u^m(l,n)$ and $w^m(l,n)$ are the corresponding coarse grid values into the physical domain. Every interpolation takes place either on x either on y-axis. If we consider that we apply the new velocity values on x-axis, (figure3), interpolation is applied as follows:

$$u^{m+1}(i,k) = \frac{u^m(l,n) + u^m(l+1,n)}{2}$$

and

$$w^{m+1}(i,k) = \frac{w^m(l,n) + w^m(l+1,n)}{2} \quad (1)$$

Also,

$$u^{m+1}(i,k) = u^{m+1}(i+1,k) = ... = u^{m+1}(i+I-1,k) \quad (2)$$

Therefore, if the refinement factor is set to be equal 2, (I=2), the above relation becomes as below:

$$u^{m+1}(i,k) = u^{m+1}(i+1,k) \quad (3)$$

The relation between i and l is:

$$l = 2 * i - 1 \quad (4)$$

As we have assigned the velocity values on the boundary bounds, we must apply a condition for the pressure. Assuming that we simulate for an axisymmetric flow, the pressure vertical derivative at the interface is estimated as follows:

$$\frac{\partial p}{\partial n} = n_x \left[\frac{1}{\mathrm{Re}} \left(\frac{\partial^2 u}{\partial y^2} + \frac{1}{y} \cdot \frac{\partial u}{\partial y} + \frac{\partial^2 u}{\partial x^2} \right) - u \cdot \frac{\partial u}{\partial x} - v \cdot \frac{\partial u}{\partial y} \right] + \quad (5)$$

$$+ n_y \left[\frac{1}{\mathrm{Re}} \left(\frac{\partial^2 v}{\partial y^2} + \frac{1}{y} \cdot \frac{\partial v}{\partial y} + \frac{\partial^2 v}{\partial x^2} - \frac{v}{y^2} \right) - u \cdot \frac{\partial v}{\partial x} - v \cdot \frac{\partial v}{\partial y} \right]$$

where, $\frac{\partial p}{\partial n}$ is the pressure vertical derivative, Re the Reynolds number, n_x and n_y the components of the unit normal vector, u and v the axial and the vertical velocity components respectively.

The derivatives discretization is applied by one-sided difference formula, either forward or backward. It depends on the position of each sub-grid in relation with the previous level one.

In order to transfer the boundary values through a fine – coarse interface, we once more apply interpolation and we estimate the pressure vertical derivative as above. With the same symbols, interpolation between the velocity values is:

$$u^m(l,n) = \frac{u^{m+1}(i,k) + u^{m+1}(i+1,k) + ... + u^{m+1}(i+I-1,k)}{I} \quad (6)$$

where, I is the refinement factor.

So, we interpolate for the velocity components and we solve for pressure. Although, this isn't necessary, we prefer it because we want to maintain accurate and stable solutions. We agree with Collela [12], that if you want to obtain a robust algorithm solving for pressure is needed. By the way the results both of the ways of simulation are good enough.

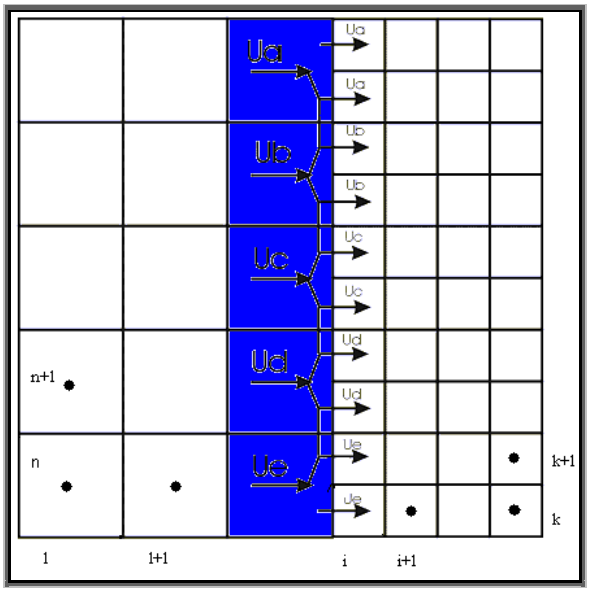


Fig. 4: Linear interpolation in order to transfer the velocity values to a coarse- fine interface.

3. Numerical Simulation

The incompressible equations after the addition of the pseudocompressibility term, take on a hyperbolic character with pseudo-pressure waves propagating with finite speed. In such types of problems "the information" inside the flow field is transmitted along its characteristic curves. In this sense we can relate the sign of eigenvalues with the upwind representation of the flow variables at the cell faces. The upwinding of the inviscid fluxes gives more freedom in devising implicit algorithms (Steger and Kutler [21] and Thomas and Walters [22]), since it loads up the diagonals of the implicit factors. Upwind differencing (Hartwich et al. [23]), also, alleviates the necessity to add and to tune the numerical dissipation for numerical stability and accuracy as the schemes with central differencing that belong to the family of Beam and Warming Schemes (Beam and Warming [24]).

The upwind scheme of the hyperbolic problem, in this paper, is based on the extended by the method of pseudocompressibility Flux Vector Splitting method. FVS is a shock-capturing upwind method, well known for solving compressible high speed (transonic, supersonic and hypersonic) flows. Here, we extend FVS method of Steger and Warming for solving incompressible flow fields implicitly [20]. In such flow fields the splitting of the convective flux vectors has to change sense because of their non-homogeneous property. This is a very important element of the present study. The values of the flux vectors at the cell faces are approached by upwind schemes up to third order of accuracy. The unfactored discretized Navier-Stokes equations are solved by an implicit second order accurate in time scheme, using Gauss-Seidel relaxation technique.

4. Results

In order to examine the accuracy of the above method, we present the numerical simulation of flow field inside a stenosed tube and around a symmetric airfoil NACA0012. We chase out the accuracy of the results comparing them with the correspondence of Cartesian uniform grid, with the same base grid size.

4.1 Steady flow inside a stenosed tube.

In stenosed tubes, the appearance of large recirculating zones, the high gradients of the wall shear stresses and the strong increase of the pressure drop are the main effects depending on the severity of the stenosis.

Table 1: Boundary Conditions

Lower bound:	$\frac{\partial u}{\partial y}=0, w=0, \frac{\partial p}{\partial y}=0$	*Inlet:*	$u=1-y^2, w=0 \frac{\partial p}{\partial x}=0$
Upper bound:	$u=w=0, \frac{\partial p}{\partial y}=0$	*Outlet*	$\frac{\partial u}{\partial x}=0, w=0, p=const$

The grid generation and the numerical method that was described above were used for the calculation of steady flow inside a stenosed tube. The stenotic area is the 0.25% of the inlet area. The used numerical refinement grid is level=1 and I=2, (base grid: 401x26).

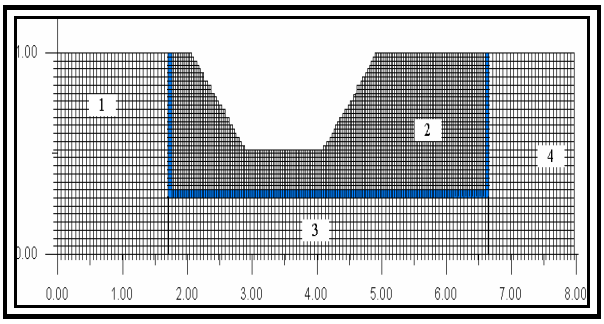


Fig. 5: Part of the used block nested numerical grid 401x26, L=1, I=2

The Re number, that was based on the maximum inlet velocity and the radius of the inlet, was set equal to 100. The boundary conditions are summarized as above, at table I. In order to control the accuracy of the proposed method, we simulated the current flow field by the use of two uniform grids: a curvilinear one sized 501x31 and a uniform Cartesian grid, 401x26 too. A part of the block nested numerical grid is presented in figure (5), where is picturized that the whole domain is parted to 4 sub grids. Four velocity profiles along the flow field are presented in figure (6) and the pressure distribution along the stenosed tube is presented in figure (7).

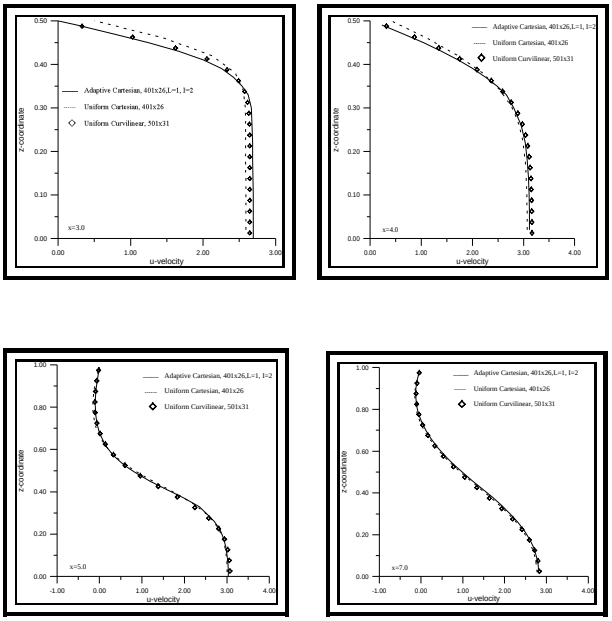


Fig. 6: Velocity profiles along flow field

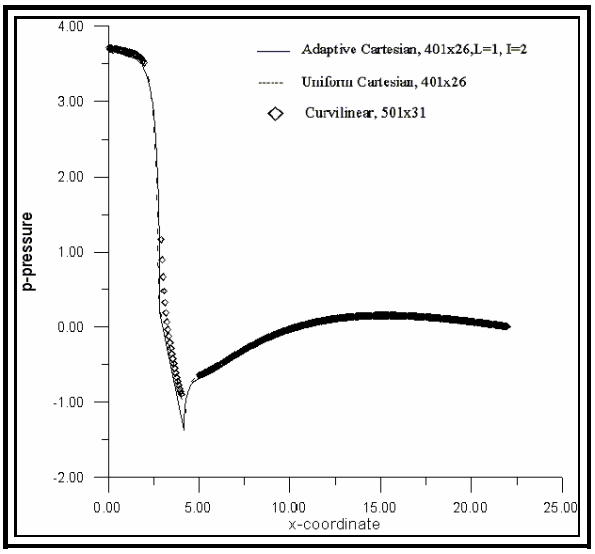


Fig. 7: Pressure distribution along the stenosed tube

It seems that the convergence between block nested algorithm results and curvilinear grid' s is very satisfied, while the Cartesian uniform grid' s results present greater relative error. It's important to be mentioned that we managed to approve the numerical results, by the use of only one sub – grid with a refinement factor equal to two, around the stenotic area.

4.2 Flow around airfoil NACA0012

In the second test case we simulate the flow field around a symmetric airfoil, (figure 8). It is impossible to solve the airfoil domain using a uniform Cartesian grid for, because in order to achieve the desired geometry bound approximation a huge number of Cartesian grid cells would be needed. In order to avoid the above memory problem, we apply the block nested algorithm of two grid levels (m=2), while the integer refinement factor is equal to 4 (I=4). The base grid size is 61x51 and the whole computational domain consists of 25586 cells.

The corresponding uniform Cartesian grid comprises 796416 cells and its use for the airfoil numerical simulation is time-consuming and unprofitable. Regardless of the time problem we solved the fluid flow around the airfoil using a 320x315 uniform Cartesian grid and we realized

that the use of the 61x51 refine grid decreases the CPU time by 93%.

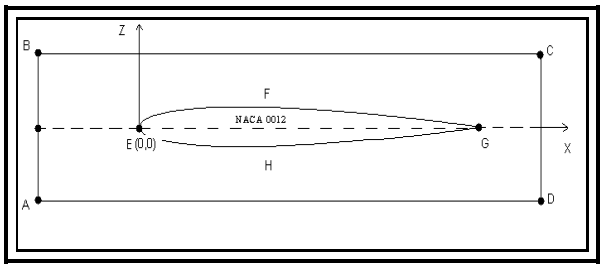


Fig. 8: Physical domain around airfoil NACA0012

We applied free stream conditions and we gave the velocity value for all the boundary limits except of the [CD] limit (figure 8), where we the pressure value is given.

The angle of attack is equal to 0 and the Reynolds number is 100. We present two axial velocity profiles along the flow field, (figure 9). The comparison took place by corresponding results by bibliography [28].

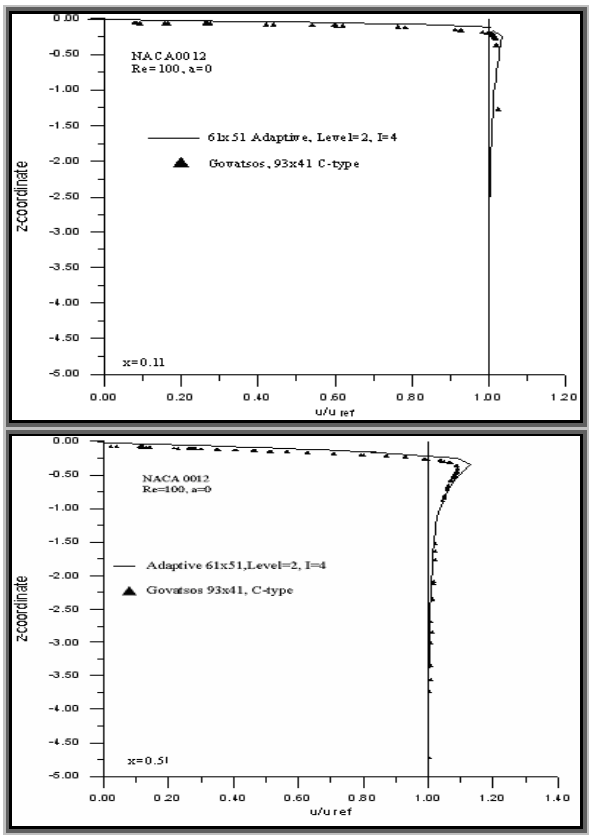


Fig. 9: Axial velocity profiles on two different positions of fluid flow around airfoil, Re=100

5. Conclusions

This paper proposes a method for the approximation of complex curvilinear geometries by using Cartesian co-ordinates only. In order to succeed the best geometry approximation close to the initial curvilinear bound we apply saw-tooth method in combination with a grid block refinement technique. We use a cell center discretization and the boundary transfer is demonstrated in the interfaces by the use of interpolation.

We presented the numerical simulation of two flow fields: both inside a stenosed tube and around a symmetric airfoil. We created the approximate Cartesian geometry by a saw-tooth method and we applied a block-nested grid in order to achieve an approximate Cartesian bound close enough to the initial curvilinear bound using a lesser number of Cartesian cells. At the stenosed tube numerical simulation, we created the approximate Cartesian geometry by a saw-tooth method and we applied a block-nested grid with one level and refinement factor equal to 2. We solved the incompressible navier stokes equations into four separate sub – grids using linear interpolation in order to transfer the boundary conditions, (figure 4). A comparison of the axial velocity results took place, between block Cartesian grid, uniform Cartesian grid and curvilinear grid, with satisfied results. By the use of the block nested grid we succeed to improve the result's accuracy toward the corresponded of uniform Cartesian grid.

On the second test case we examined the fluid flow around a NACA0012. Airfoils are 'thin' bodies and a use of a uniform Cartesian grid is very unprofitable and some times the algorithm is impossible to converge. So the use of the block nested grid is necessary and provides a lot of advantages according to the CPU memory and converging time. A comparison of the axial velocity results took place, between block Cartesian grid and bibliography results. The differences appearing between the profiles due to the different simulation methods, types of grid, residuals and certainly to digitization error. By the use of the block nested grid we succeed in improving the converging time results and sometimes decreasing them over 90%. It's

important to be mentioned that the flow rate is concerned, in both of the above test cases, in spite of the differences depicted in the above velocity profiles.

The above numerical solution proves that the Cartesian block refinement method is stable and accurate enough, regardless of the produced Cartesian bound is less accurate than the curvilinear one. The block Cartesian method is simple and gives a convergent and grid independent solution for complex curvilinear geometries, accomplishing also to reduce CPU memory and the simulation's computing time effort. With appropriate choice of local block refinement multilevel solutions computed with this algorithm can attain the accuracy of the equivalent uniform fine grid at less computational cost.

References

[1] Thompson, J.F., Thames, F.C. and Mastin, C.W. "*Automatic numerical generation of body fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies*", J. of Computational Physics, Vol. 15, pp. 299-319, 1974

[2] Clark, D.K., Salas, M.D. and Hassan, H.A., "*Euler calculations for multielement airfoils using Cartesian grids*", AIAA J., Vol. 24, pp.353-356, 1986

[3] Falle, S. and Giddings, J., "*An adaptive multigrid applied to supersonic blunt body flow*", Numerical Methods in Fluid Dynamics, 1988

[4] Pember, R.B., Bell, J.B., Collela, P., Cruthhfield, W.Y. and Welcome, M.L., "*An adaptive Cartesian grid method for unsteady compressible flow in irregular regions*", J. of Computational Physics, Vol. 120, pp. 278-304, 1995

[5] Coirier, W.J. and Powell, K.G., "*Solution-Adaptive Cartesian cell approach for viscous and inviscid flows*", AIAA J., Vol. 34, pp. 938-945, 1996

[6] Coirier, W.J. and Powell, K.G., "*An accuracy assessment of Cartesian-mesh approaches for the Euler equations*", J. of Computational Physics, Vol. 117, pp. 121-131, 1995

[7] Smith, R.J. and Johnston, L.J., "*A novel approach to engineering computations for complex aerodynamic flows*", Proceedings of the 4th International Conference on Numerical grid Generation in Computational Fluid Dynamics and related Fields, pp. 271-285, 1994

[8] Almgren, A.S., Bell J.B., Collela P., Howell L.H. and Welcome M.L., "*A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations*", J. of Computational Physics, Vol. 142, pp. 1-46, 1998

[9] Martin D. and Collela P., "*A cell-centered adaptive projection method for the incompressible Euler equations*.", J. of Computational Physics, Vol. 163, pp. 271-312, 2000

[10] Howell L.H. and Bell J.B., "*An adaptive mesh projection method for viscous incompressible flow*", SIAM J. Sci. Comput., Vol. 18, pp. 996-1007, 1997

[11] Wang, Z.J., "*A Quadtree-based adaptive Cartesian/Quad grid flow solver for Navier-Stokes equation*", Computers and Fluids, Vol. 27, pp.529-549, 1998

[12] Tuncer, I.H., "*Two-dimensional unsteady Navier-Stokes solution method with moving overset grids*", AIAA J., Vol. 35, pp. 471-476, 1997

[13] Agresar, G., Linderman J.J., Tryggvason, G. and Powell, K.G., "*An adaptive, Cartesian, front-tracking method for the motion, deformation and adhesion of circulating cells*", J. of Computational Physics, Vol. 143, pp. 346-380, 1998

[14] Udaykumar, H.S., Kan, H.C., Shyy, W. and Tran-Son-Tray, R., "*Multiphase dynamics in arbitrary geometries on fixed Cartesian grids*", J. of Computational Physics, Vol. 137, pp.366-405, 1997

[15] Deister F., Rocher D., Hirschel E.H. and Monnoyer F., "*Adaptively refined Cartesian grid generation and Euler flow solutions for arbitrary geometries*", Notes on numerical Fluid Dynamics, Vieweg, Braunschweig/Wiesbaden, 1998

[16] Chai, J.C., Lee, H.S. and Patakar, S.V., , "*Treatment of irregular geometries using a Cartesian coordinates finite-volume radiation heat transfer procedure*", Numer. Heat Transfer, Vol.26, pp.179-197, 1994

[17] Jesse J.P., Fiveland W.A., Howell L.H., Collela P. and Pember R.B., "*An adaptive mesh refinement algorithm for the radiative transport equation*", J. of the Comput. Physics, Vol. 139, pp. 380-398, 1998.

[18] Berger M.J. and Collela P., "*Local adaptive mesh refinement for shock hydrodynamics*", J. of Comput. Physics, Vol. 83, pp.64-84, 1989.

[19] Georgantopoulou Chr.G., Pappou Th.J., Tsaggaris S.G., "*Cartesian grid generator for N-S numerical simulation of flow fields in curvilinear geometries*", Proceedings of the 4th GRACM congress on comput. Mechanics, pp. 526-534, 2002

[20] Pappou, Th. and Tsangaris, S., "*Development of an artificial compressibility methodology using Flux Vector Splitting*", International J. for Numerical Methods in Fluid, Vol. 25, pp.523-545, 1997

[24] Steger, L. and Kulter, P. "*Implicit finite-difference procedures for the computation of vortex wakes*", AIAA J, Vol.15, pp. 581-590, 1977

[25] Thomas, L. and Walters, R.W., "*Upwind relaxation algorithms for the Navier-Stokes equations*", AIAA paper, 85-1501-CP, 1999

[26] Hartwich, M., Hsu, C.H. and Liu, C.H., "*Vectorizable implicit algorithms for the flux-difference split, three-dimensional Navier-Stokes equations*", J. of Fluid Engineering, Vol. 110, pp. 297-305, 1988

[27] Beam, R.M. and Warming, R.F., "*An implicit finite-difference algorithm for hyperbolic system of conservation-law form*", International J. of Computational Physics, Vol. 22, pp. 87-110, 1976

[28] Ãåù ñãáí õï ðï ýëï õ, Ã.×., Ãåù ñãáí ôüðï õëï ò, Á.Ã., ÐÜðõï õ, ÉÈ. êáé Ôóáããáñçò, Ó., «*Åöáñìï ãÞ êáñôåóáíþ í ðëåãìÜù í óôçí åðßëõóç ðåäßù í ñïÞò ãýñù áðü êáì ðõëüãñáì ì åò ãåù ì åôñßåò*», Recent advances in mechanics and the related fields, Vol.1, pp.76, 2003