COBEM2015

23rd ABCM International Congress of Mechanical Engineering
December 6-11, 2015, Rio de Janeiro, RJ, Brazil

Towards Gilbert-Johnson-Keerthi Distance Algorithm Applied to Porosity Distributed Resistance for Fluid Flow Numerical Analysis

Victor Damatto Moreira [1] **In memoriam**
Rogerio Gonçalves dos Santos
Faculty of Mechanical Engineering - FEM-UNICAMP
vdamatto@gmail.com, roger7@fem.unicamp.br

Tatiele Dalfior Ferreira
Sávio S. V. Vianna
Faculty of Chemical Engineering - FEQ-UNICAMP
tatieledf@gmail.com, savio@feq.unicamp.br

Abstract. *The porosity parametrization of complex geometry is very common in explosion simulation. This project proposes a novel approach to calculate the mesh porosity (volume and area) based on the Gilbert–Johnson–Keerthi (GJK) distance algorithm. The GJK algorithm can be used to check the collision between convex objects. This concept is used in the current work to check the collision between an element of the mesh and an object of the geometrical model. The approach also concerns the breaking of the element of the mesh recursively to obtain refined values of porosity. This novel approach has demonstrated that it can handle complex geometries using short computational time. The proposed model is implemented in an "in house" Euler solver. Analysis of the flow over a bluff body shows consistency with the expected results.*

Keywords: *Porosity Distributed Resistance - PDR, Gilbert–Johnson–Keerthi (GJK) distance algorithm, Computational Fluid Dynamics - CFD*

1. INTRODUCTION

The Porosity/Distributed Resistance (PDR) model is widely used to assess gas explosion in complex geometries (Bjørn H. Hjertager, 1992; Vianna and Cant, 2010; Savill and Solberg, 1994; Forthergill *et al.*, 2003). It is applied to model the geometry in CFD simulations and thus assess a great level of geometrical details. By using this method, the large-scale geometry is fully represented and resolved and the small-scale objects are taken together locally and approximated in terms of their effective porosity and resistance to the flow.

The concept of Porosity Distributed Resistance (PDR) was introduced by Patankar & Spalding to handle multi obstacles flows. Initially, the method was implemented in heat exchanger geometries and it was progressively applied for complex geometries (Bjørn H. Hjertager, 1992; Vianna and Cant, 2010).

The main idea of PDR is to consider the small scales objects, which are not solved by the computational mesh, as a porous media. This medium would offer a resistance to the fluid flow (Savill and Solberg, 1994). This technique allows the use of a much coarser computational mesh than that required to resolve the fine scales of the geometry, which in turn permits the use of a less detailed specification of the geometry.

This work presents a method to represent different geometry types and sizes as a porous media in a mesh calculation to be used in CFD simulations. The method has been implemented in the "in house" code developed by the University of Campinas. Preliminary results for a cold flow across a cubic geometry are presented.

2. THE PDR METHOD

In the PDR approach, due to the presence of small unresolvable obstacles, only part of each computational cell volume is available for the flow. It also considers that such objects offer additional resistance to the flow and are responsible for increasing the production of turbulence. As consequence, the transport governing equations are modified.

Considering a typical control volume shown in Fig. 1, the porosity or volume fraction occupied by the fluid can be defined as:

$$\beta_v = \frac{V_f}{V_f + V_s} = \frac{V_f}{\triangle x \triangle y \triangle z} \tag{1}$$

Here, V_f is the volume of fluid and V_s is the volume of solid obstruction. In the same way, the area porosity for each

of the three directions (x, y and z) can be defined. The porosity area in the x direction, for example, can be expressed as:

$$\beta_x = \frac{\int_{surface} dydz}{\triangle y \triangle z} \tag{2}$$

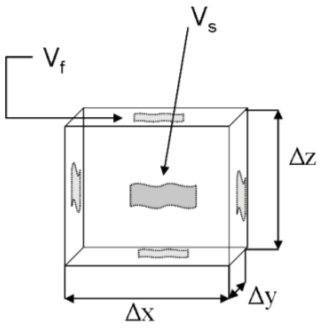


Figure 1: Typical computational cell showing a part occupied by solid objects and the area projected in the faces (Vianna and Cant, 2010).

The values of volume and area porosities range from 0.0 (when the computational cell is completely blocked) to 1.0 (the computational cell is completely open).

2.1 Modified Transport Equations

The application of the conservation concepts in the control volume shown in Fig.1 leads to the following modified transport equations for mass (Eq. 3) and momentum (Eq. 4):

$$\frac{\partial}{\partial t}(\beta_v \rho) + \frac{\partial}{\partial x_i}(\beta_i \rho U_i) = 0 \tag{3}$$

$$\frac{\partial}{\partial t}(\beta_v \rho U_i) + \frac{\partial}{\partial x_j}(\beta_j \rho U_j U_i) = -\beta_v \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}(\beta_j \sigma_{ij}) + \beta_v \rho g_i + R_i \tag{4}$$

U_i is the velocity component in the x_i direction; ρ is the density; p is the pressure; g_i is the gravitational acceleration; σ_{ij} is the turbulent flux of momentum and R_i is the additional resistance caused by the obstacles (Vianna and Cant, 2010, 2012; Bjørn H. Hjertager, 1992).

It is important to bear in mind that rather than the additional resistance term in the momentum equation, all terms carry an aditional β variable in the modelling. β represents the porosity value. It comes in two ways. The first, β_j, concerns the area porosity and it is used to correct the flux through the computational cell face. The second, namely β_V, deals with all quantities in the equation which are per unit volume.

The current PDR approach was implemented in a finite-volume code that uses the PDR concepts to represent the geometry for reacting flow. At this stage the Euler equation is considered, therefore equation 4 becomes;

$$\frac{\partial}{\partial t}(\beta_v \rho U_i) + \frac{\partial}{\partial x_j}(\beta_j \rho U_j U_i) = -\beta_v \frac{\partial p}{\partial x_i} + \beta_v \rho g_i + R_i \tag{5}$$

3. METHODOLOGY

The developed approach relies on the Minkowski Addition and Gilbert-Johnson-Keerthi Distance Algorithm concepts to calculate the porosity values which parameterizes the real 3D geometry.

3.1 Minkowski Addition

Given two sets $A, B \in \mathbb{R}^n$ of position vectors in the euclidean space, the Minkowski addition is formed by adding every vector in A to every vector in B:

$$A + B = \{a + b \mid a \in A,\ b \in B\} \tag{6}$$

Similarly, the Minkowski difference is formed by subtracting ever vector in B from every vector in A, equation 7.

$$A - B = \{a - b \mid a \in A,\ b \in B\} \tag{7}$$

One important property of the Minkowski difference is that if the two sets are colliding, the resulting difference will contain the zero vector.

3.2 Gilbert-Johnson-Keerthi Distance Algorithm Implementation

The Gilbert-Johnson-Keerthi Distance algorithm (GJK algorithm), (Gilbert and Foo, 1989; Gilbert *et al.*, 1988), is a method for obtaining the minimum distance between two convex objects based on the concept of the Minkowski difference. This algorithm can also be used to check the collision between two convex objects.

The main advantages of the GJK algorithm is the stability and speed. The speed is of the order $O(n)$ where n is the number of points in the Minkowski Difference. For this reason, it is extensively used to check for collision detection in video games (Ericson, 2004a,b; van den Bergen *et al.*, 1999). The algorithm also relies on a support function that allows the checking for collisions between any kind of discrete and analytical objects, in any kind of format.

The porosity of a geometry can be obtained using the GJK algorithm to check for a collision between a cell of the mesh and each solid part of the geometry. This process is described in Figure 2 and it is implemented as follows:

- on step 1 a geometry is imported into a domain.
- on step 2 a mesh is created over the domain.
- on step 3 the GJK technique is used to calculate the porosity of the mesh.

In this approach, every cell fully occupied by a solid has a porosity value of 0.0, while every empty cell has a porosity value of 1.0. The porosity is directly proportional to the filled space, so if an element of the mesh is half full its value is 0.5.

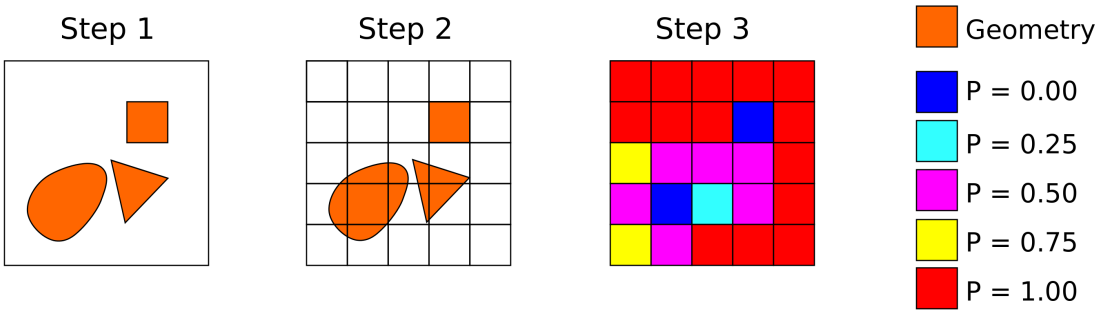


Figure 2: Details of the steps of the porosity generation. Coloured scheme on the right hand side identifies the regions in the computational cell which have different porosities values.

Each cell is assigned a volume porosity and an area porosity for each of its faces. Using a structured mesh, each cell will be an hexahedron. For the area porosity, first each solid of the geometry must be "sliced" on the required plane. A trivial slicing method, described by (Rodrigo M. M. H. Gregori and da Silva, 2014) is used. Using the structured mesh, the area porosity is calculated on the faces of the hexahedrons.

Because the GJK algorithm can only return true or false for the collision, an recursive division approach is used to obtain a refined value of the porosity. This technique is shown in Figure 3. For this particular case two cells of the mesh are shown. On the first step the GJK test shows that the top cell element is not in contact with the solid while the bottom cell is in contact with the object. Therefore the top cell element is immediately assigned a value of 1 to the porosity. The cell element that is in contact with the solid (bottom cell) requires refinement of the porosity calculation as previously discussed. On the next step, the bottom element is divided into k sub-elements and each sub-element is tested with the GJK algorithm again. For the example shown in Figure 3, $k = 4$. On the next step the sub-element that is in contact with the solid is divided again in k sub-elements and the GJK algorithm is applied once more. For this case, a maximum of three recursive steps is used. The porosity value is obtained by subtracting each sub-element from the main element according to its value. If it is not in contact with any solid geometry, the value of the part that is subtracted is zero, otherwise it is $\frac{1}{k^n}$, where n is the number of the iteractions.

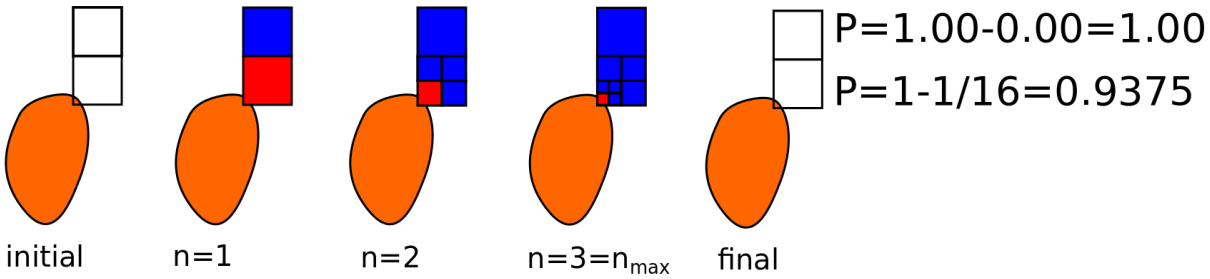


Figure 3: Porosity recursion. The 3 main iterations during the division the computational cells for accurate porosity calculation

The refined volume porosity is obtained breaking every element of the mesh (cube) into 8 equal elements, while the refined area porosity is obtained dividing each face of the mesh element (square) into 4 equal elements.

4. RESULTS

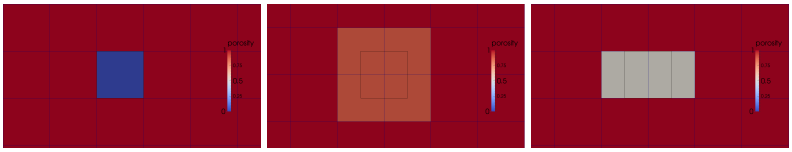
The modified GJK algorithm was implemented in a parallel code developed within the context of this work in order to evaluate the application of the method. Different geometries were considered. A chemical process plant was considered as an example of a real engineering case.

The performance of the code is also discussed. The tests were conducted using a structured hexaedrical mesh. A value of $n_{max} = 3$ was used for both porosities (area and volume porosity). The volume step size for $k = 8$ subdivisions is $\frac{1}{8^3} = 0.001953125$, the area step size for $k = 4$ subdivisions $\frac{1}{4^3} = 0.015625$. The last part of this section addresses a cold flow around a squared geometry.

4.1 Validation of the developed code for porosity calculation based on modified GJK algorithm

In order to validate the model and the developed computer program a simple cubic geometry that fits precisely in the computational cell was considered. The cube was placed at different positions, as shown in Figure 4.

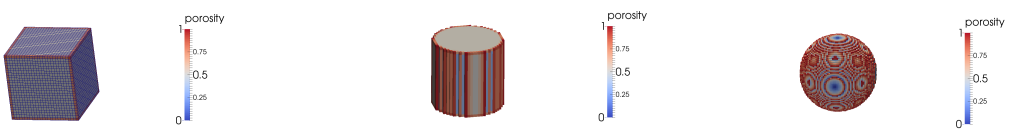
Analysis of Figure 4c shows that the porosity is zero for the case where the cube fits exactly the mesh. For the other two configuration, shown in Figure fig:val3 (b) and Figure fig:val3 (c), the value of the porosity is calculated in accordance with the empty space.



(a) Cube: Porosity 0.00 (b) Cube: Porosity 0.25 (c) Cube: Porosity 0.50

Figure 4: Porosity values considering the cube (a) in the centre of the computational cell, (b) at the centre of coincident lines of 4 cells (cell vertex) and at (c) half way between two cells.

The calculation of the porosity was also investigated in primary geometrical shapes commonly used in engineering design. Figure 5 shows the volume porosity results (for basic geometrical shapes) obtained by the developed computational code. Analysis of Figure 5 shows that the code is able to mimic the original geometrical shapes in a proper manner.



(a) Cube (b) Cylinder (c) Sphere

Figure 5: Various porosity results for different basic shapes obtained by the developed code (values > 0.99 not shown)

4.2 Performance Analysis

A set of tests was conducted in order to test the speed of the method and code. The tests parameters were:

- number of threads (1, 2, 3, 4, 5, 6)
- number of points of the geometry (20, 40, 360)
- number of geometries (1, 2, 4, 8, 16)
- mesh size (10x10x10 and 100x100x100).

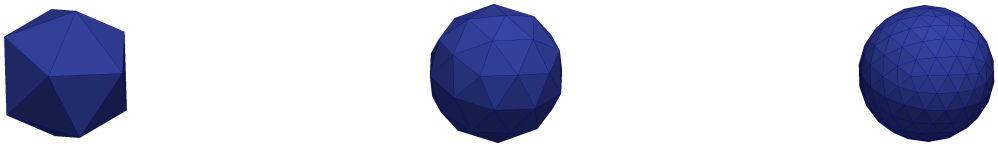
Figure 6 shows the different number of points considered in the spherical geometry. The sensitivity analysis of the performance of the code is presented in Table 1 and 2. The number of geometries (1,2,4,8 and 16) is presented in Figure 7

Detailed description of the number of points previously mentioned is present in Figures 6 and 7. The maximum computing time was $\boldsymbol{t_{max}} = 20.120s$ for the case considering 1 thread, 360 points and 16 objects in the whole geometry. This particular case considered a mesh size of 10^6 cells. The extensive list of results is presented in Table 1 and 2. It is important to note that the time is normalised with the maximum computational time.

It is possible to note that the time does not have a linear relation with the number of threads. This independency is due to the time required by the process of importing the STL object and writing off the porosity data. This part of the code was not parallelised yet.

Although the time was expected to increase with the number of points and number of objects, the final result shows that it had an erratic behaviour. This behaviour can be explained by the optimisation technique inserted in the code, and the "chance" of convergence of the GJK algorithm.

Comparing the elapsed time for the 10x10x10 mesh and the time for the 100x100x100 mesh it is possible to conclude that although the mesh is 1000 larger the time only increased by a factor of 10^2.



(a) 20 points (b) 80 points (c) 360 points

Figure 6: Spheric geometry parametrised by different number of points as coded .

(a) 1 sphere (b) 2 spheres (c) 4 spheres

(d) 8 spheres (e) 16 spheres

Figure 7: Number of spheres considered in the analysis.

Table 1: Performance Test 1

Threads	**Points**	**Number**	**Mesh**	t/t_{max}	**Threads**	**Points**	**Number**	**Mesh**	t/t_{max}	**Threads**	**Points**	**Number**	**Mesh**	t/t_{max}
1	20	1	10	0.001	2	20	1	10	0.001	3	20	1	10	0.000
			100	0.740				100	0.440				100	0.339
		2	10	0.001			2	10	0.001			2	10	0.001
			100	0.691				100	0.455				100	0.360
		4	10	0.001			4	10	0.001			4	10	0.001
			100	0.724				100	0.418				100	0.344
		8	10	0.001			8	10	0.001			8	10	0.001
			100	0.738				100	0.470				100	0.351
		16	10	0.002			16	10	0.001			16	10	0.001
			100	0.744				100	0.423				100	0.351
	80	1	10	0.001		80	1	10	0.001		80	1	10	0.001
			100	0.720				100	0.493				100	0.318
		2	10	0.001			2	10	0.001			2	10	0.001
			100	0.734				100	0.459				100	0.369
		4	10	0.001			4	10	0.001			4	10	0.001
			100	0.782				100	0.401				100	0.374
		8	10	0.002			8	10	0.001			8	10	0.001
			100	0.762				100	0.500				100	0.369
		16	10	0.006			16	10	0.003			16	10	0.002
			100	0.851				100	0.502				100	0.400
	360	1	10	0.003		360	1	10	0.001		360	1	10	0.001
			100	0.843				100	0.426				100	0.378
		2	10	0.002			2	10	0.001			2	10	0.001
			100	0.783				100	0.489				100	0.364
		4	10	0.006			4	10	0.002			4	10	0.002
			100	0.846				100	0.509				100	0.382
		8	10	0.006			8	10	0.003			8	10	0.002
			100	0.871				100	0.469				100	0.441
		16	10	0.011			16	10	0.006			16	10	0.005
			100	1.000				100	0.646				100	0.518

Table 2: Performance Test 2

Threads	**Points**	**Number**	**Mesh**	t/t_{max}	**Threads**	**Points**	**Number**	**Mesh**	t/t_{max}	**Threads**	**Points**	**Number**	**Mesh**	t/t_{max}
4	1	1	10	0.000	5	20	1	10	0.000	6	20	1	10	0.000
			100	0.293				100	0.223				100	0.188
		2	10	0.000			2	10	0.000			2	10	0.001
			100	0.285				100	0.217				100	0.183
		4	10	0.001			4	10	0.001			4	10	0.001
			100	0.305				100	0.222				100	0.190
		8	10	0.001			8	10	0.001			8	10	0.001
			100	0.312				100	0.244				100	0.196
		16	10	0.001			16	10	0.001			16	10	0.001
			100	0.311				100	0.237				100	0.196
	2	1	10	0.001		80	1	10	0.001		80	1	10	0.000
			100	0.283				100	0.223				100	0.182
		2	10	0.001			2	10	0.001			2	10	0.001
			100	0.311				100	0.226				100	0.196
		4	10	0.001			4	10	0.001			4	10	0.001
			100	0.295				100	0.224				100	0.180
		8	10	0.001			8	10	0.001			8	10	0.001
			100	0.288				100	0.236				100	0.191
		16	10	0.001			16	10	0.001			16	10	0.001
			100	0.328				100	0.270				100	0.210
	3	1	10	0.001		360	1	10	0.001		360	1	10	0.001
			100	0.293				100	0.230				100	0.175
		2	10	0.001			2	10	0.001			2	10	0.001
			100	0.296				100	0.229				100	0.186
		4	10	0.001			4	10	0.001			4	10	0.001
			100	0.313				100	0.246				100	0.210
		8	10	0.002			8	10	0.002			8	10	0.002
			100	0.327				100	0.268				100	0.230
		16	10	0.004			16	10	0.004			16	10	0.004
			100	0.379				100	0.330				100	0.277

4.3 Practical Cases

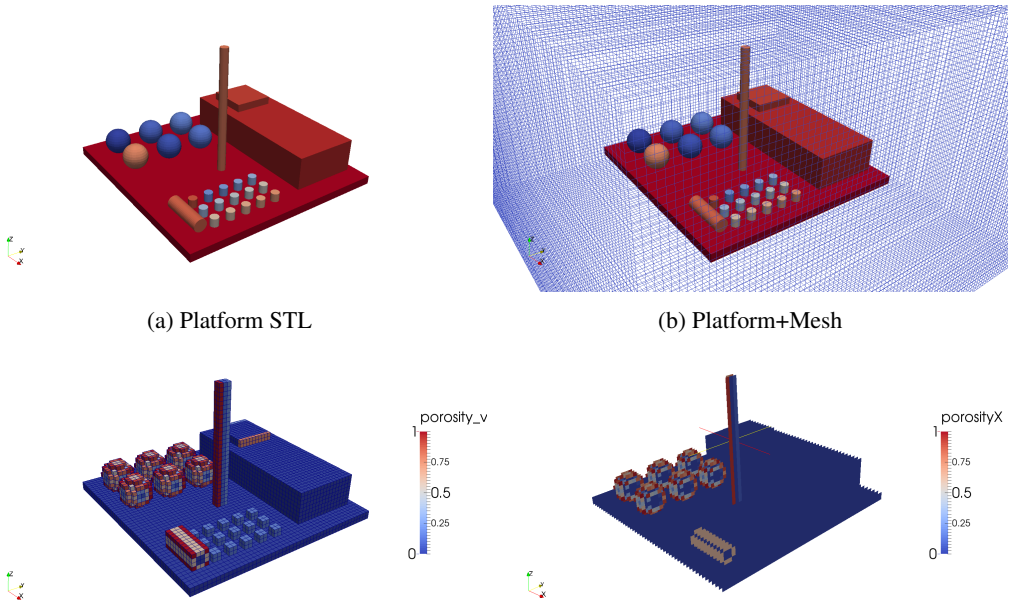
The evaluation of the proposed approach as well as the code developed was also tested in a geometrical model that resembles a real offshore platform deck. The process deck is 100 metres long by 106 metres wide. The accommodation module is 10 metres high. Basic geometrical shapes are placed on the process deck mimicking typical process equipment commonly found in offshore oil rigs.

Eight colours were considered to represent the STL geometry as a porous media. The maximum number of iteration considered was 3 ($n_{max} = 3$). It took approximately 10 seconds to complete the parametrisation of the oil platform previously mentioned.

Figure 8 shows the combination of the three stages of the generation of the parametrised version of the geometrical model. Figure 8 (a) shows the CAD (Computer Aided Design) file imported into the developed code. The geometry is match with the computational mesh shown in Figure 8 (b). Figure 8 (c) and (d) show the porosity model for the volume porosity and area porosity, respectively.

In figure 8b it is possible to observe that the STL geometry fits into the hexahedral mesh with size of 99x99x63 (617463 cells). Such configuration comprises the first step of the porosity code where the location of the solid geometry will be compared with the location of the computational cell.

Analysis of Figure 8 shows that it is possible to represent the STL geometry by the porous regions given by the modified GJK code (Figures 8c and 8d).



(a) Platform STL (b) Platform+Mesh

(c) Platform Porosity Volume (d) Platform Porosity Area (x direction)

Figure 8: Porosity applied to a simple oil platform model

The porosity distributed resistance modelling implemented in the Euler solver as described in section 2.1 is discussed here.

The porosity file mimicking the geometry is read by the in house Euler solver and a simple flow over a bluff body is conducted.

The squared shape geometry is 1 metre long by 1 metre wide and 1 metre high. It is placed in the centre of the computational domain that measures 10 metres long by 3 metres wide and 3 metres high. The velocity is prescribed at the left hand side of the computational domain. At the upper and lower bonds the net flux is zero. The boundary condition downstream the obstacle is set to have zero gradiente (see Figure 9)

For the computational cells in the region occupied by the box the values of volume/area porosities were evaluated by the modified GJK code. Figure 9 shows the result obtained by the Euler code. Analysis of Figure 9 shows that the flow is reduced at the wake of the obstacle due to the resistance to the flow and obstruction caused by the box. The region of the computational where the velocity is zero matches the spacial coordinates of the box.

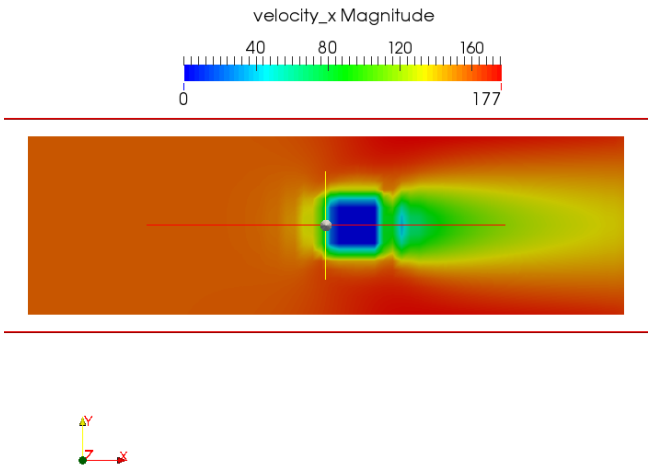


Figure 9: Velocity profile of a air flow around a box into a rectangular computational domain.

5. CONCLUSION

The application of the Gilbert–Johnson–Keerthi (GJK) distance algorithm for porosity calculation is proposed. The porosity calculation is tested for various basic geometries and the results obtained are satisfactory. The methodology is also tested in an oil platform CAD file that mimics a real engineering geometry. The approach discussed in this paper proved to parametrise the geometry properly.

The technique was validated and proved to be fast. It is well within the time scale required by engineering design.

The disadvantage of the GJK algorithm is that it is not able to handle non-convex objects. It can, however, be overcome by any convex decomposition techniques available.

The porosity was implemented in the PDR (Porosity Distributed Resistance) formulation of the Euler flow and the preliminary result of a flow over a bluff body is very promising.

Future work will assess how the approach behaves in turbulent reacting flows for explosion modelling purposes.

6. REFERENCES

Bjørn H. Hjertager, Tron Solberg, K.O.N., 1992. "Computer modelling of gas explosion propagation in offshore modules". *Loss Prevention in the Process Industries, IEEE Journal of*, Vol. 5, No. 3, pp. 165–174.

Ericson, C., 2004a. "Collision detection in interaction 3d environments". *SIGGRAPH Presentation*.

Ericson, C., 2004b. *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology)*. CRC Press, har/cdr edition. ISBN 9781558607323.

Forthergill, C., Chynowet, S., Roberts, P. and Packwood, A., 2003. "Evaluation a cfd porous model for calculating ventilation in explosion hazard assessments". *Loss Prevention in the Process Industries, IEEE Journal of*, Vol. 4, No. 16, pp. 341–347.

Gilbert, E. and Foo, C., 1989. "Computing the distance between smooth objects in three dimensional space". In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*. pp. 158–163 vol.1. doi: 10.1109/ROBOT.1989.99983.

Gilbert, E., Johnson, D. and Keerthi, S., 1988. "A fast procedure for computing the distance between complex objects in three-dimensional space". *Robotics and Automation, IEEE Journal of*, Vol. 4, No. 2, pp. 193–203. ISSN 0882-4967. doi:10.1109/56.2083.

Rodrigo M. M. H. Gregori, Neri Volpato, R.M. and da Silva, M.V.G., 2014. "Slicing triangle meshes: An asymptotically optimal algorithm". *Proceedings of 14th Inter. Conference on Computational Science and Its Applications*.

Savill, A. and Solberg, T., 1994. "Some improvements to pdr/$k-\varepsilon$ model predictions for explosions in confined geometries". In *Procedings ER-COFTA/IMA Confined Flow Dispersion Through Groups Obstacles, Cambridge, UK, 1994*. pp. 227–247.

van den Bergen, G., Van, G. and Bergen, D., 1999. "A fast and robust gjk implementation for collision detection of convex objects". *Journal of Graphics Tools*, pp. 7–25.

Vianna, S. and Cant, R., 2010. "Modified porosity approach and laminar flamelet modelling for advanced simulation of accidental explosion". *Loss Prevention in the Process Industries, IEEE Journal of*, , No. 23, pp. 3–14.

Vianna, S.S. and Cant, R.S., 2012. "Explosion pressure prediction via polynomial mathematical correlation based on advanced {CFD} modelling". *Journal of Loss Prevention in the Process Industries*, Vol. 25, No. 1, pp. 81 – 89. ISSN 0950-4230. doi:http://dx.doi.org/10.1016/j.jlp.2011.07.005. URL `http://www.sciencedirect.com/science/article/pii/S095042301100115X`.

7. ACKNOWLEDGEMENT

The authors would like to thank the National Council for Scientific and Technological Development (CNPQ) and the Coordination for the Improvement of Higher Education Personnel (CAPES).

8. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.