E3S Web of Conferences **7**, 04025 (2016) DOI: 10.1051/e3sconf/20160704025
FLOOD*risk* 2016 - 3rd European Conference on Flood Risk Management

Innovations in irregular meshing to improve the performance of 2D finite volume flood simulation

Sam Jamieson[1,a], Julien Lhomme[1] and David Fortune[1]

[1]*XPSolutions, Jacobs Well, Newbury, RG14 1BD, UK*

Abstract. There are now many successful applications of 2D finite difference flood simulations on rectangular grids. However, finite volume algorithms are now starting to take over as the preferred choice for simulating flood inundation. The use of finite volume algorithms allows the simulation to be based on irregular meshes. Yet many users are not convinced of the benefits of irregular meshing. This is partly because meshing adds a rather unwelcome extra process to the modelling. But it is more likely because the process of generating irregular meshes has been problematic and time consuming. In the rush to develop excellent finite volume flow engines, meshing has been rather left behind. Yet the quality of the flow modelling depends on the quality of the meshing. The authors have been working on innovative mesh generation techniques. The aim has been to come up with a faster, more reliable mesh generation process. But also to improve the resulting mesh in order to speed up the hydraulic simulation and to generate better quality flood inundation results. This paper discusses and illustrates these advances in meshing. Results are presented to show how well the meshing techniques deal with problematic boundaries. The resulting meshes are applied to test cases to show the impact of either grids or different qualities of mesh on flood inundation results for a variety of circumstances.

1 Introduction

In the field of hydraulic inundation modelling there is increasing research and industrial use of Finite Volume models due their enhanced robustness and natural treatment of super-critical flows and shock conservation. Finite Volume schemes, unlike Finite Difference schemes, naturally lend themselves to unstructured computational meshes which benefit from the flexibility of being able to easily vary cell size throughout the domain and conform to complex domain boundaries. The engineer will commonly require increased resolution in a location of interest, which can be difficult to achieve with a regular square grid model.

Generating an appropriate irregular mesh for hydraulic computations is no simple task, and there has been significant research into this field. The most commonly used method for generating a 2D mesh is Delaunay triangulation. This uses the Delaunay criterion that no node should exist in the circumcircle of another triangle in the mesh. In itself the Delaunay criteria is only a method for connecting existing points; a point insertion method is also required. Point insertion can be accomplished using a regular distribution of points, or by recursively adding points on the edges [1], at the centroids [2], at circumcentres [3-5], or at points along the Voronoi segments [6] on triangles from the initial triangulation.

An alternative approach is Advancing Front meshing. In this method triangles are progressively added to the domain boundary and active meshing front, until they have advanced across the whole domain and met somewhere in the middle [7-9]. As each new point is added it is connected to existing points for form new triangles according to the Delaunay criteria.

Most advances in irregular meshing has been developed in the field of CFD where the complexity of the domain geometry is of paramount importance. In inundation modelling however this is not always the case. The engineer will normally know the desired level of detail required for a project but may be hampered by input geometry of a smaller scale. For example when importing a large dataset of building boundaries from an external source the boundary complexity can be of higher resolution than the desired computational cell size. The engineer will want to represent the general impact (relative to the desired representation scale) of these buildings as blockages in the flow path, without wanting every minor detail of the building footprint to be represented. Similar cases can easily be imagined for all features in the domain which impact on the computational mesh.

Having these small-scale features in the mesh can have a large consequence. The Courant condition [10] states that water should not flow across more than one cell in a single time step. Therefore smaller cells typically require a smaller time step when they are wet. The most popular type of Finite Volume model, and those most commonly used for real industrial applications, have explicit time discretisation with a globally adaptive time step. Therefore a single small wet cell can require a much smaller time step than the rest of the model, greatly reducing the Courant

[a] Corresponding author: sam.jamieson@xpsolutions.com

number of the majority of the cells and increasing computational cost.

To avoid this unnecessary inefficiency, the engineer's desired scale of representation should be considered the primary constraint on meshing, overriding the complexity of the domain geometry, as appropriate. Instead the currently available mesh generation tools approach the problem from the opposite angle; they take the domain geometry as priority number one, and override the engineer's choice of resolution. This is not to say that the current mesh generation tools are inadequate *per se*; it is just that they were not designed for, and are therefore not ideally suited for inundation modelling.

This paper first considers a hypothetical meshing scenario designed to help demonstrate these considerations. It will then introduce a new meshing tool which aims to respect the engineer's desired scale of representation. Finally it will show the effectiveness of this new meshing tool on a real dataset, provided by the UK Environment Agency.

The new meshing tool is named 'xp2dfv meshing engine', because it complements the 'xp2dfv flow engine' as part of the new software package xpflood. Comparisons in the study are made against an industry standard Delaunay meshing tool, and are referred to as 'standard meshes'. These have been computed using 'Triangle' [5], and were produced using constrained conforming Delaunay triangulation with a minimum angle of 33°. Any comparisons are intended to highlight the difference to common triangulation approaches used in the industry, and not specifically to 'Triangle' itself.

2 Hypothetical meshing scenario

How to handle complex geometry at the domain boundaries is a complex problem for inundation simulations. This issue is exemplified here with a hypothetical case study involving a simple flat 50 m x 50 m domain containing three buildings (Figure 1). The buildings are considered high relative to the flow depth and impervious to flow, therefore they are to be treated as voids in the domain. But the issues here could be applied to any complex meshing geometry.

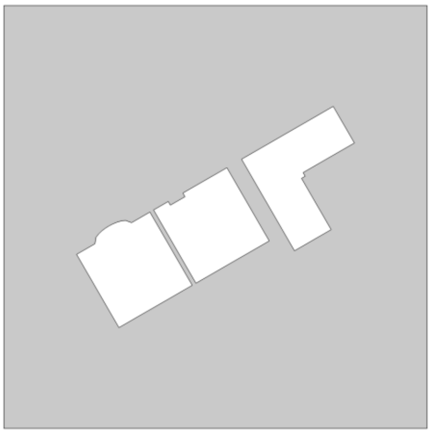
Figure 1. Hypothetical meshing domain with building voids.

In this scenario the engineer has already made a decision that a 2 m cell size provides an appropriate scale of representation for this analysis. There are two gaps between the three buildings; the left gap is 0.5 m wide and therefore does not provide a significant flow path for this scale of analysis. The right gap however is 2 m and so should be represented in the resultant mesh.

A regular square grid is applied to the domain with cell size of 2 m, which results in a 25x25 grid. Grid cells are considered void if their centroid falls within one of the buildings. This produces a grid of 551 squares with the general position of the buildings accounted for. But the geometrical shape of the buildings is not well represented, and the flow path on the right is not maintained, as water must flow through open cell faces (Figure 2).

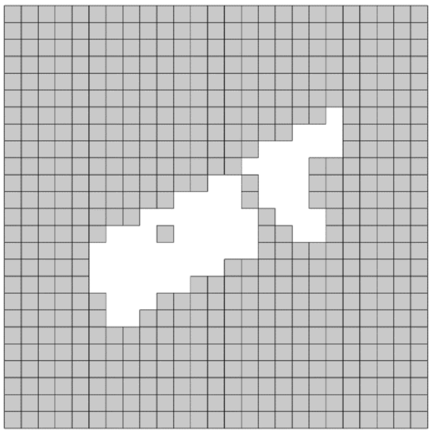
Figure 2. Regular square grid

Irregular meshing has the advantage that it can conform to domain geometry. The standard triangulation produces a good quality mesh that achieves this, using a max triangle size of 4 m^2 (Figure 3). But by conforming to the geometry it produces a large number of triangles that are significantly smaller than the chosen scale of representation. This happens in particular around the "bay window" of the left-most building, the "doorway alcove" of the centre building and the "corner pier" in the right-most building.

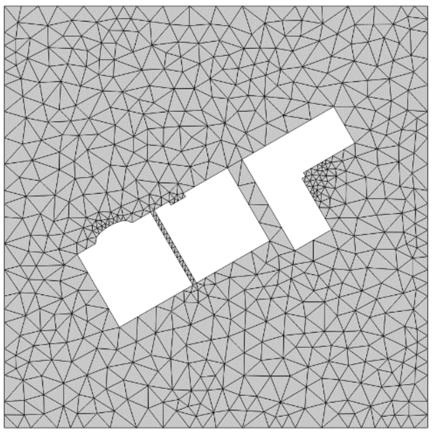
Figure 3. Traditional meshing of domain respecting the building geometry.

The standard mesh has 1,144 triangles, with a minimum triangle size of 0.0985 m^2. This will have a large impact on computational performance because not only are there significantly more calculation cells, but the time-step will also have to be severely curtailed due to the Courant condition. Of course if the engineer requires this level of detail there is no option but to use a higher resolution with corresponding higher computational cost. But when the engineer considers that level of detail to be below the desired scale of representation this cost is a pure loss.

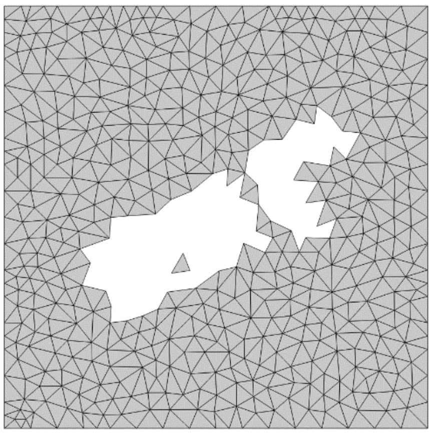
Figure 4. Traditional meshing of domain with buildings represented by null triangles from domain sampling

If the engineer decides to not conform to the building outlines (Figure 4) then the resultant mesh has 908 triangles and a minimum triangle size of 0.62 m^2. This is a large improvement on the minimum triangle size, and therefore the computation time step, but as with the regular square grid there is no cell-face flow path between the buildings and the right-hand flow path will be ignored. This mesh has the negatives of irregular meshing with none of the positives. If the modelling decision has been taken to use irregular meshing then is normally preferably to conform to the domain geometry.

The solution left to the engineer is to export the domain geometry to an external package, such as GIS software, and try to perform simplification and generalisation algorithms on the domain. This can be time consuming and difficult, without guaranteed results.

3 Description of xp2dfv meshing engine

A new meshing algorithm has been created for the xp2dfv flow engine. Other meshing tools start from the geometry as the primary constraint and then work to produce the mesh. The xp2dfv meshing engine instead starts from the user's desired triangle size (DTS), then handles fitting the triangles to the domain geometry accordingly. The aim is to produce a set of triangles with a mean triangle size as close to the DTS as possible, with a narrow range of triangle sizes. In particular it aims to maximise the minimum triangle size, knowing that this will greatly reduce computational cost in subsequent flow simulation.

The meshing engine uses an advancing front approach to place the triangles sequentially from the domain boundaries, with special care taken to ensure each triangle is strictly Delaunay. If there are no complex boundaries in the domain then the algorithm will reduce to a scheme analogous to [9].

When there are complex boundaries the meshing algorithm undertakes an innovative triangle placement scheme to 'best-fit' the triangles to the boundary. It takes an ideal equilateral triangle matching the DTS and attempts to find an optimal location for this on the boundary. It is very rare that this can be perfectly achieved, and so the application of the geometrical constraints contorts the triangles to non-equilateral, but still always Delaunay.

The algorithm takes the DTS as a proxy for an appropriate scale of representation, and then makes the following considerations:

- Triangles should be placed on features to maintain the general feature shape.
- Spaces between features which are smaller than the DTS should be ignored, with the effect that (very) proximal features can become merged.
- If features are too close to fit the DTS, but far enough apart to constitute a significant flow path, then triangles are forced into the gap, with the effect that a "minimum width corridor" is enforced between them.
- Very small features are completely ignored.

When applied to the hypothetical meshing scenario above the xp2dfv meshing engine produces a high quality mesh with only 615 triangles (Figure 5), which is fewer than the non-constrained Delaunay mesh. Most importantly the minimum triangle size is 1.54 m^2, which is 2.5 times the standard non-conforming mesh and >15 times the standard conforming mesh.

Note that the significant flow path on the right is represented whilst the insignificant flow path on the left has been disregarded. The overall shape of the buildings is still well maintained, including the protrusion of the "bay window" in the left-most building.

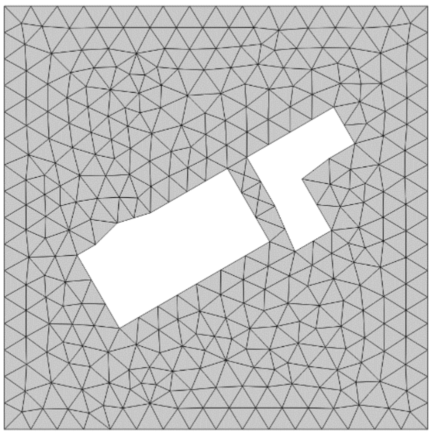
Figure 5. xp2dfv meshing using boundary best-fit.

The xp2dfv meshing engine produces this meshing with its default configuration without the need for user interaction. This greatly simplifies the job of the engineer. However there are cases where it is necessary to change the DTS across the modelled domain, for example in a localised impact assessment, or if a particular flow path is known to be significant even though it is smaller than the global DTS. In these cases the xp2dfv meshing engine will use user-defined mesh refinement areas. Other user configurations include the min and max triangle angles and parameters associated with the boundary fitting, but these are beyond the scope of this paper.

4 Real world example

The UK Environment Agency produced a series of 2D hydraulic benchmark tests [11] to test the performance of different hydraulic models. Test 8B of these tests includes a building dataset with a good degree of complexity (Figure 6), and is therefore a good dataset for testing the performance of the new meshing engine.

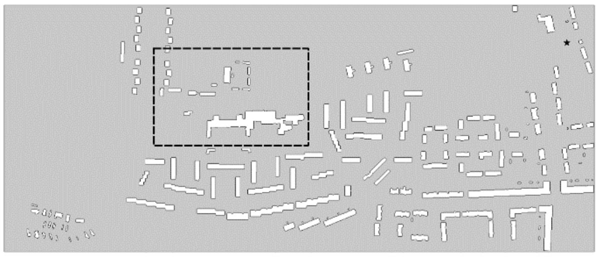
Figure 6. T8B domain with voids for buildings. Thick dotted line shows location of subsequent images. Black star in top-right corner is inflow location.

4.1 Meshing

The scale of representation for this domain is taken as 16 m^2. The standard meshing produces a mesh with 130,028 triangles, with very small triangles around all the buildings (Figure 7).

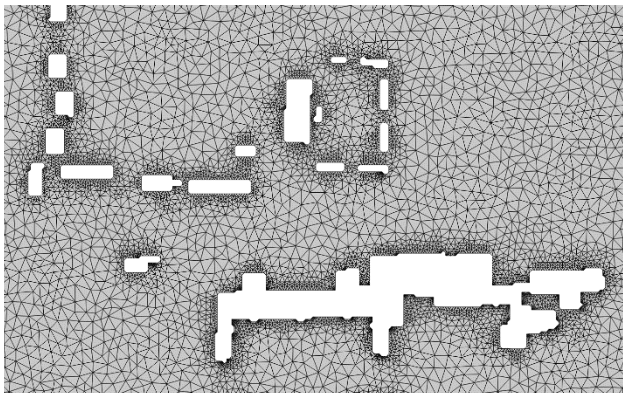
Figure 7. T8B domain with standard meshing.

The average triangle size is 2.67 m^2, significantly below the desired size. The minimum triangle size is 2.95E^{-14} m^2. Clearly it is unfeasible to undertake a simulation using this initial mesh. The user is forced to manually attempt simplification of the domain geometry.

A number of simplification techniques are considered in this study:

A. QGIS ‘Simplify Geometries’ [12] (modified Douglas-Peucker algorithm) with a tolerance of 0.1.
B. QGIS ‘Simplify Geometries’ with a tolerance of 1.
C. GRASS ‘v.generalize’ [13] using the Douglas-Peucker algorithm with a tolerance of 1.0.
D. GRASS ‘v.generalize’ using Vertex Reduction algorithm with tolerance of 6.0787 (length of edge of DTS).
E. GRASS ‘v.generalize’ using Cubic Hermite Splines with tolerance of 6.0787.

The simplification algorithms greatly reduce the number of points along the building boundaries, but they are not designed with the intention of maintaining an appropriate separation distance between adjacent points, as would be required by the standard meshing engine if it were to avoid small triangles. In addition the GRASS routines do not make any simplification between the first and last points of a polygon, which can leave these two points very close together. Finally the GRASS and QGIS routines consider each polygon in isolation from the others, so they cannot cope with buildings close to each other. The buildings in this dataset are generally not too close together, but part of this effect is still visible in some locations, see for example Figure 8.

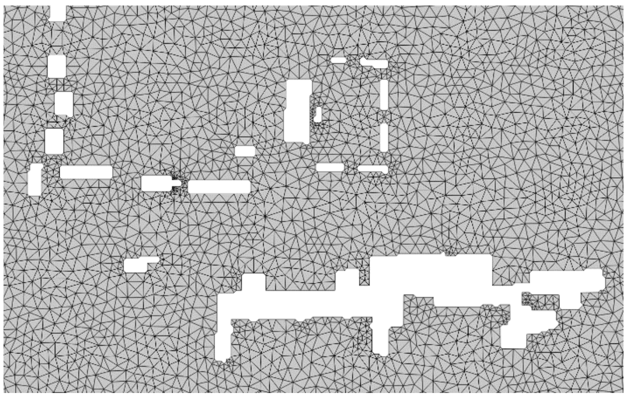
Figure 8. T8B domain using simplification method ‘A’ and the standard meshing algorithm.

The mesh produced with simplification method ‘A’ has a significantly reduced triangle count at 47,666, but the smallest triangle size is still very small, at 0.016 m^2 (Figure 8). The other simplification routines perform slightly better in terms of increasing the median triangle size, but none of them manage to increase the minimum triangle size acceptably; the largest minimum triangle size is 0.39 m^2, achieved by method B, whilst the smallest is from method E with only 0.0004 m^2.

The new xp2dfv meshing engine produces a mesh with only 24,754 triangles and a minimum triangle size of 5.38 m^2 (Figure 9). Notice that some buildings are considered too small or too narrow (relative to the DTS) and have therefore been disregarded. Of the remaining buildings, the overall shape is well maintained given the triangle size.

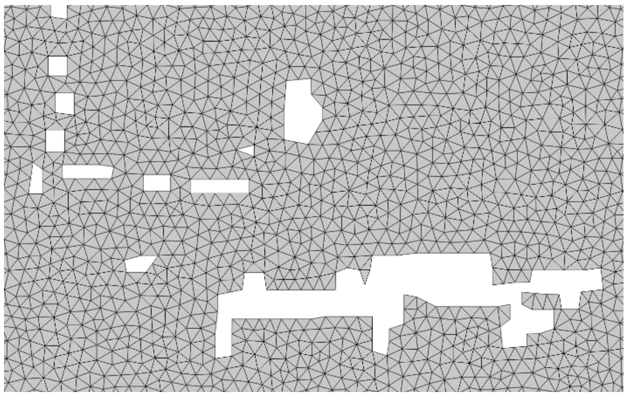
Figure 9. T8B domain using xp2dfv meshing engine.

A comparison of the range of triangle sizes is shown in the boxplots in Figure 10. Even though the simplification routines greatly increase the triangles sizes of the three quartiles, they all retain very small minimum size.

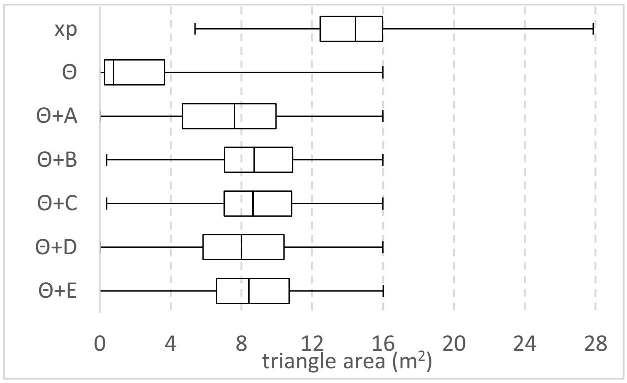

Figure 10. Boxplots of triangle areas for T8B domain. 'xp' = xp2dfv meshing engine, Θ = standard meshing engine, and Θ+X is the standard meshing using simplification routine X.

The distribution of triangle sizes is shown in Figure 11. For clarity only simplification routine B is shown in the graph (arguably the most effective simplification routine here). Notice that for xp2dfv the distribution of triangle sizes is well grouped around the DTS of 16 m^2, whereas for the standard meshing algorithm this is an absolute maximum. In the work-flow of the hydraulic engineer, if they want a particular triangle size on average they would have to use trial-and-error to find an appropriate maximum size that will produce a mean triangle size close to the DTS.

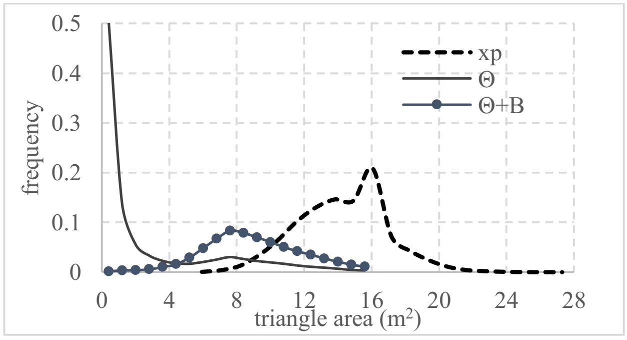

Figure 11. Frequency plot of triangle sizes for xp2dfv meshing engine (xp), standard meshing engine (Θ) and the standard meshing using simplification routine B (Θ+B).

4.2 Flow simulation

A simple simulation was conducted using a point inflow (shown as a black star in Figure 6) with the hydrograph given by the EA Benchmark test 8A; a bell shaped hydrograph with peak flow of 5 cumecs, lasting a total of 30 minutes. The simulation length was 5 hours. xp2dfv was used for the simulations; a 2D finite volume full shallow water equation model. It was run using the 1st-order scheme, Open-MP parallelisation and an adaptive time step with target Courant number of unity. The hardware was a dual-core i7 processor with 2.1 GHz and 8 GB RAM. Reported runtimes are median values of three runs.

Mesh	Min Δt (s)	# time steps (k)	CPU mins
xp	0.334	45	2.1
Θ	0.012	1,262	263.4
Θ+A	0.037	382	26.8
Θ+B	0.242	66	3.2
Θ+C	0.236	54	3.8
Θ+D	0.202	61	3.3
Θ+E	0.013	901	53.1

Table 1. Simulation run times

The simulation results show that, as expected, mesh configuration can have a significant impact on simulation efficiency. The xp2dfv mesh allowed for a simulation with the largest time step, and therefore the fewest iterations and shortest run-time. This effect is likely to be minimised here because the Open-MP parallelisation is less efficient with less computational cells, and because the simulation flow path happened to avoid the locations of smallest triangles in the other grids. However the exact speed-up is not of great concern, because clearly this will be highly dependent on the modelling scenario and complexity of input terrain. The xp2dfv meshing tool has produced a mesh that respects the user's chosen scale of representation, which results in an efficient simulation. This has been achieved without any user-interaction, at the first attempt.

Many research publications will attribute considerable importance to the simulation efficiency; this is clearly important, but it must not be forgotten that the running length of the simulation is only one aspect of the project costs, and the engineer's (hydraulic modeller's) time can be more costly. Using some of the current meshing tools available the engineer may have to, after computing an initial mesh, export the data from the meshing application, import into a GIS package, investigate and choose an appropriate simplification routine, trial several different parameter combinations, then re-import to their meshing application and compute another mesh. This procedure may be iterative as the domain geometry is fine-tuned. In addition there will be some aspects of the geometry which cannot be naturally simplified, such as two buildings which are too close together, closer than the desired scale of representation. To the author's knowledge, the simplification routines available in GIS packages have no automated method for dealing with these issues. Accordingly handling the domain geometry and producing a satisfactory mesh can become a significant task in a flood modelling study, requiring considerable modeller interaction.

5 Conclusions

The xp2dfv meshing tool available in the new software package xpflood is designed specifically for flood modelling applications. It aims to automate the difficult task of generating a suitable computational mesh, leaving the engineer free to concentrate on the appropriate scale of representation and other important modelling decisions, rather than the complexity of simplifying geometries to achieve their goal. The xp2dfv meshing tool will automatically apply a best-fit approach to triangle placement, which respects the user's decisions about desired scale of representation. In many applications this could save significant time and project costs just in terms of user time. It also has the effect of reducing simulation times by maximising the minimum triangle size, with a result better than what could be achieved through manual use of GIS simplification routines. This comparison exercise demonstrates the performance of the xp2dfv meshing tool and shows how a meshing approach tailored to flood modelling applications can lead to time savings, both in terms of simulation runtime and modellers time.

6 References

1. Borouchaki, H., Hecht, F., Saltel, E. and George, P. L. (1995) Reasonably Efficient Delaunay Based Mesh Generator in 3 Dimensions. *Proceedings 4th International Meshing Roundtable, October 1995*, 3-14.
2. Weatherill, N. P. and Hassan, O. (1994). Efficient Three-dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints, *International Journal for Numerical Methods in Engineering*, **37**, 2005-2039.
3. Chew, P. L. (1989). Guaranteed-Quality Triangular Meshes. TR 89-983, Department of Computer Science, Cornell University, Ithaca, NY.
4. Ruppert, J. (1992). A New and Simple Algorithm for Quality 2-Dimensional Mesh Generation. Technical Report UCB/CSD 92/694, University of California at Berkely.
5. Shewchuk, J. R. (1996). Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. *First Workshop on Applied Computational Geometry, ACM, May 1996*, 203-222.
6. Rebay, S. (1993). Efficient Unstructured Mesh Generation by Means of Delaunay Triangulation and Bowyer-Watson Algorithm. *Journal Of Computational Physics*, **106**, 25-138.
7. Lo, S. H. (1991). Volume Discretization into Tetrahedra - II. 3D Triangulation by Advancing Front Approach. *Computers and Structures*, **39(5)**, 501-511.
8. Lohner, R. (1996). Progress in Grid Generation via the Advancing Front Technique. *Engineering with Computers*, **12**, 186-210.
9. Mavriplis, D. J. (1995) An Advancing Front Delaunay Triangulation Algorithm Designed for Robustness. *Journal of Computational Physics*, **117(1)**, 90-101.
10. Courant, R., Friedrichs, K., and Lewy, H. (1928). Über die partiellen differenzgleichungen der mathematischen physik. *Mathematische Annalen*, **100(1)**, 32-74.
11. Néelz, S. and Pender, G. (2010). Benchmarking of 2D hydraulic modelling packages. Technical Report SC080035/SR2, Environment Agency of England and Wales.
12. QGIS Development Team (2015). QGIS 2.8 Geographic Information System User Guide. Open Source Geospatial Foundation Project. Available at: http://docs.qgis.org/2.8/en/docs/user_manual/.
13. GRASS Development Team (2012). GRASS 6.4 Users Manual. Open Source Geospatial Foundation, USA. Available at: https://grass.osgeo.org/grass64/manuals/.