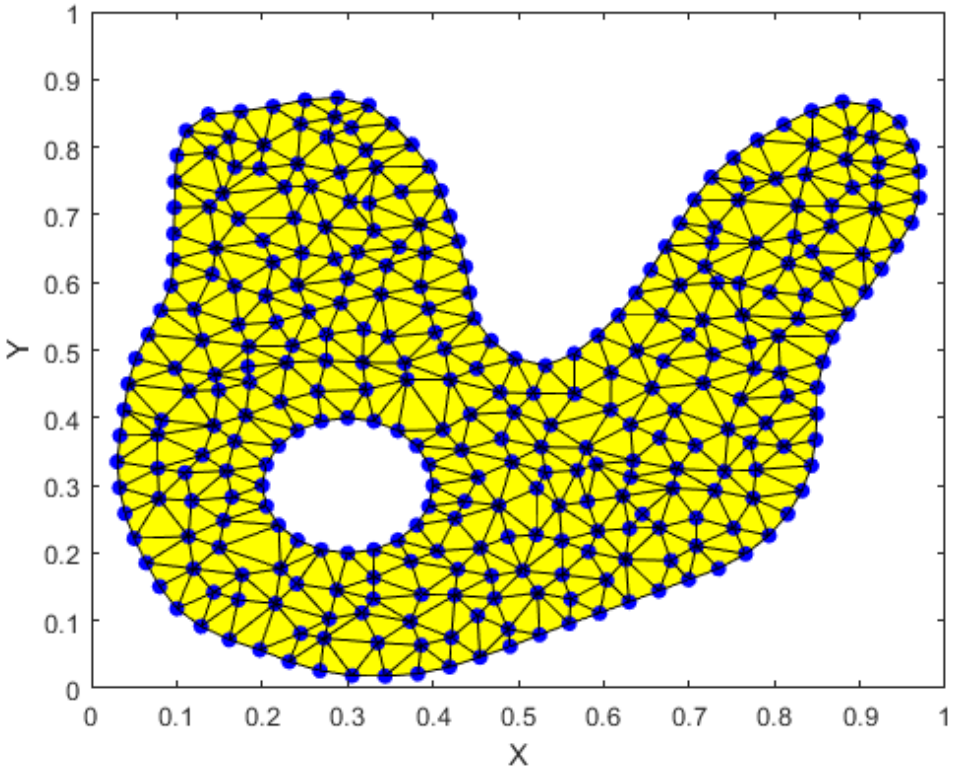
Implementation of self-organizing maps for mesh generation

Mahdi Karami



Abstract

The implementation of the self-organizing maps (SOMs) for mesh generation is investigated for 2-D geometries. For the SOM implementation, the positions of the winner neurons are modified continuously over iterations based on their lateral coefficient (related to how much they are close to the introduced datapoint) as well as the learning rate which is gradually decreasing during the iterations. For rectangular domains, two independent standard variables are engaged for mesh generation. The boundary nodes are fixed while the positions of the internal neurons are modified based on the selected random points from the domain introduced to SOM at each iteration. The nonuniform mesh distribution is obtained by applying some mathematical modifications on the probability functions. For domains with an arbitrary shape, the datapoint are selected from the domain by picking them from a pre-defined probability distribution function, which is achievable by either inverse method or rejection method. Some samples for uniform and nonuniform mesh distributions are provided for simply connected geometries as well a domain with a hole inside to show the capability of SOM for mesh generation.

Table of Contents

Table of Figures

1. Introduction

Most of the real-world applications in physics and engineering problems require solving a set of governing equations in a given physical domain. The governing equations of the systems are usually described by a few partial differential equations (i.e., PDEs), which are sometimes nonlinear and couped that make them almost impossible to be solved analytically. The numerical methods have then been developed to solve such complicated problems by discretizing the PDEs over the computational domain and change the system of nonlinear differential equations into a system of linear equations. The so-called 'mesh generation' is a process that divides the computational domain into a reasonable number of computational cells which will be used in discretized equations to obtain the distribution of the quantities of interest over the domain. Some discretization techniques like finite element method (FEM, used in solid mechanics) are based on the computational nodes (or vertices). The other type of discretization which is commonly used in computational fluid dynamics (CFD) is called finite volume method (FVM) which is a cell-based approach. The computational grid can be structured or unstructured. There are plenty of approaches used for mesh generation [1]. The present project aims to investigate the feasibility of implementing self-organizing maps (SOMs) in mesh generation process.

SOM is an unsupervised approach in machine learning used for clustering unlabeled datapoints by building a self-organizing network of neurons. In clustering, a huge amount of datapoints is divided into limited number of clusters based on their similarity and proximity [2,3]. Indeed, each cluster is a representative of multiple datapoints with similar characteristics. The mesh generation process can be considered as a clustering problem since each computational cell (i.e., cluster) is a representative of many physical points (or infinite number of points due to the continuum paradigm) that exist in it. Accordingly, the SOM approach has the potential to generate appropriate grid for a computational domain by clustering it into a limited number of computational cells. The unstructured grid with triangular cells is considered in the current project.

2. Mesh generation requirements

some characteristics of an appropriate mesh are listed below.

- Compatibility with the computational domain and its boundaries. It means that the union of all computational cells must cover all the domain and precisely match the boundary. For real-world complicated geometries, it is always a big challenge to create such an accurate mesh.
- Control over the number of nodes/cells. The computational cost of a numerical simulation is exponentially a function of the grid size. Moreover, in some applications like turbulence modelling in fluid dynamics, the size of the computational cells affects the performance of the models used to solve the equations. Hence, we always need to have a control on the size of the cells as well as the number of grids.
- Control over grid density. In most of applications of numerical simulations, the nonuniform grids are preferred since they provide the ability to spend the computational cost on the more important regions of the computational domain where the gradient of physical quantities is higher. For instance, the so-called 'boundary layer' region is always one of the most important parts of the geometry in fluid mechanics problems which requires a remarkably higher mesh density than the rest of the domain.

3. Methodology

the proposed method for generating appropriate grid for a given computational domain includes the following steps.

1- Initialize internal neurons. Number of neurons is selected based on the requirement of the problem as well as the computational resources. The initial location of neurons cane be set randomly or based on a pre-defined grid.

2- Initialize boundary neurons. There can be two types of boundary neurons including fixed or constrained ones. Fixed neurons are preferred for sharp corners of the domain while the constrained ones can be used for smooth boundaries. The latter neuron type is enforced to only move along the boundary and is restricted in the normal direction. The boundary neurons form the overall shape of the domain and ensure the compatibility of the grid with the boundaries of the domain.

3- Loop:

3-1 Pick a random point from the domain and introduce it to the SOM. It should be mentioned that the picked points are selected from infinite number of physical points in the domain. Thus, an appropriate probability distribution function (PDF) is needed [4]. For complex geometries, it is difficult to define such distribution function over the domain. The problem becomes more challenging when the nonuniform mesh is desired. To have a nonuniform mesh, we need to pick points in the region of interest with more probability than the other ones. Accordingly, generating a desired PDF function for an arbitrary geometry is one of the biggest challenges of the project.

3-2 Find the winner neuron (i.e., the closest one to the given datapoint) and its neighbors whose locations will be updated according to a function for lateral feedback weights. The set of updated neurons can either be defined by a fixed number of closest ones, or by setting a certain vicinity to the given datapoint. The learning rate and size of updated neighborhood should be managed in such a way that they are large in the beginning of the solution and get smaller as the solution converges. The location of updated neurons is given by Eq. (1), where X_n is the location of neuron and X_p is the location of the current introduced point from domain. $g(t)$ and f_{mn} are respectively the learning rate and the coefficient of the lateral feedback weights.

$$X_n(new) = X_n(old) + g(t) \times f_{mn} \times (X_p - X_n(old)) \quad (1)$$

3-3 The boundary neurons actively participate in the SOM process, except the last step where their updated locations are subjected to their constrains. Fixed boundary neurons are not allowed to move while the constrained neurons are prevented to move in the direction normal to boundary. Thus, the correcting vector for boundary neurons (i.e., the second term in R.H.S. of Eq. (1)) is replaced with its projection on the boundary path.

3-4 For each neuron, repel all the neighbor neurons which are too close. This step is necessary to avoid the situation that some clusters have a very small distance from each other that is a common situation in SOM clustering.

4- Implement the Delaunay triangulation method to convert the resulted network into computational mesh.

4. Simple Domains

Some primary results of simple geometries are illustrated in this section which have been caried out by implementing SOM methodology described in the last section.

4.1. Results of uniform mesh

Figure 1 depicts the results of SOM methodology for generating mesh on a simple 1×1 square. The final locations of neurons are illustrated by red asterisks, while the blue lines represent the mesh edges resulted from Delaunay triangulation. Due to the lack of boundary neurons, the resulted grid does not coincide the domain and its boundaries well. One can see some grid cells with poor quality (i.e., triangles with a very wide angle) in some regions on the domain.

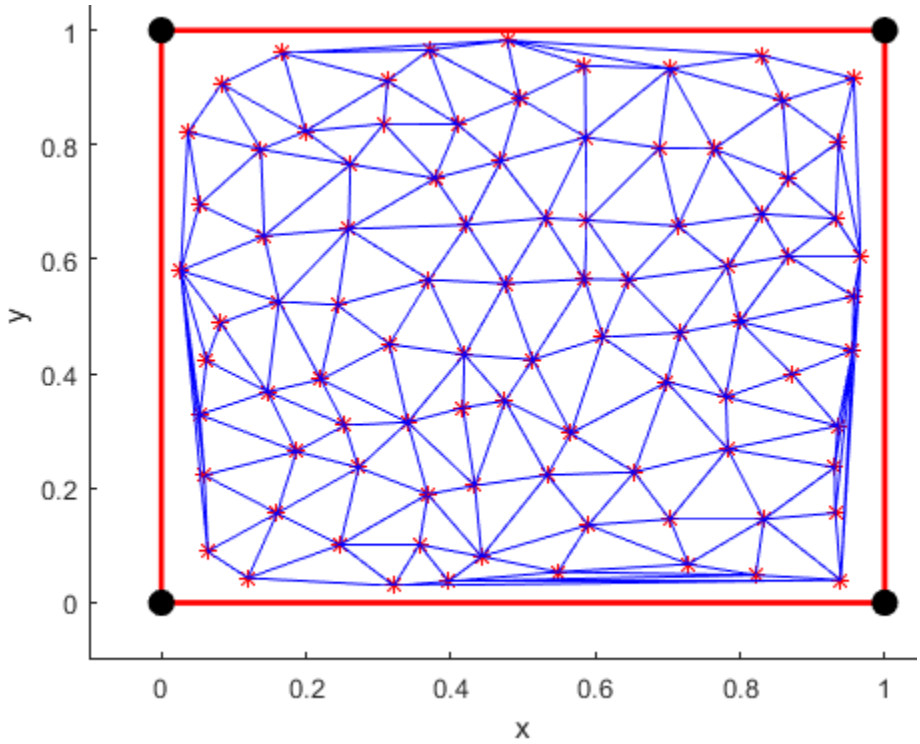


Figure 1: SOM mesh generation result for a 1×1 square geometry without boundary neurons.

Figure 2 displays the SOM mesh generation results where four fixed neurons at the square corners are engaged. It can be seen that the achieved mesh is compatible with the boundary although the mesh quality is still poor in some regions. Figure 3 illustrates the SOM mesh generation results for 80 internal neuron and 7 fixed neurons on each edge of the domain. The quality of the resulted mesh is acceptable.

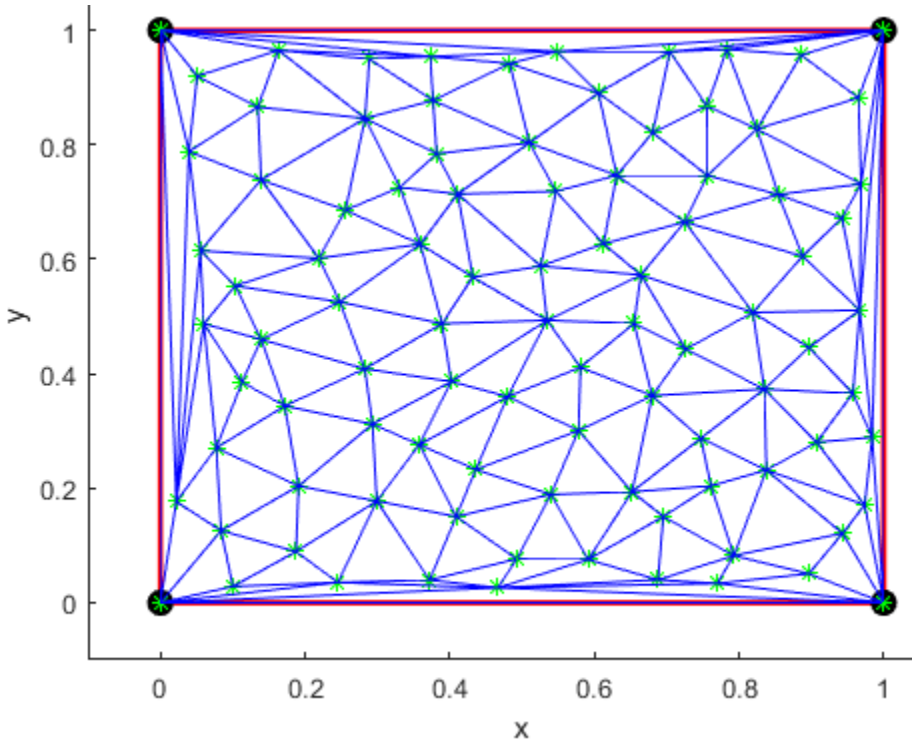


Figure 2: SOM mesh generation result for a 1×1 square geometry with four fixed neurons at the corners.

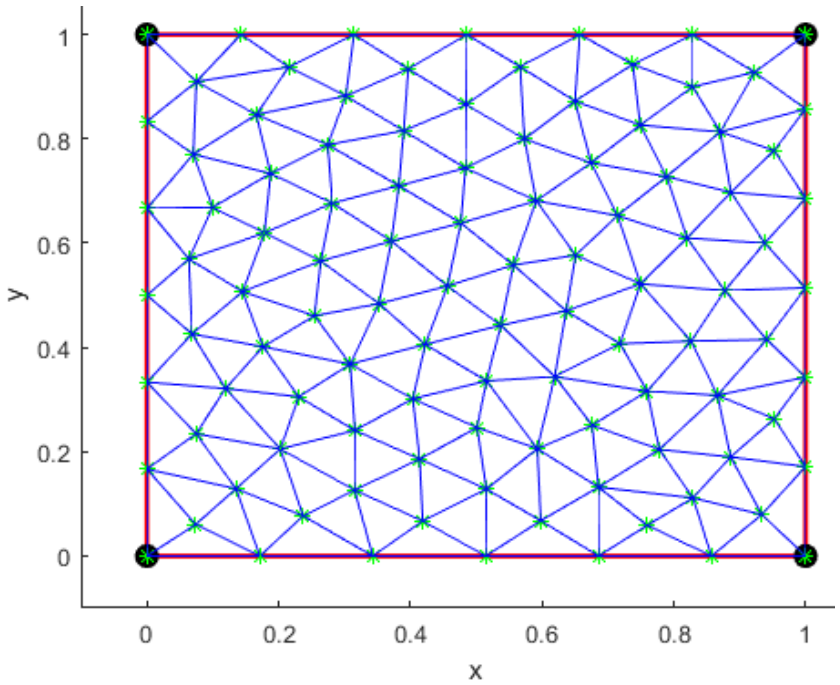


Figure 3: SOM mesh generation result for a 1×1 square geometry with 80 internal neurons and 7 fixed neurons on each edge of the domain.

A finer grid can be obtained from SOM by engaging more neurons, as depicted in Fig. 4 with 250 internal neurons.

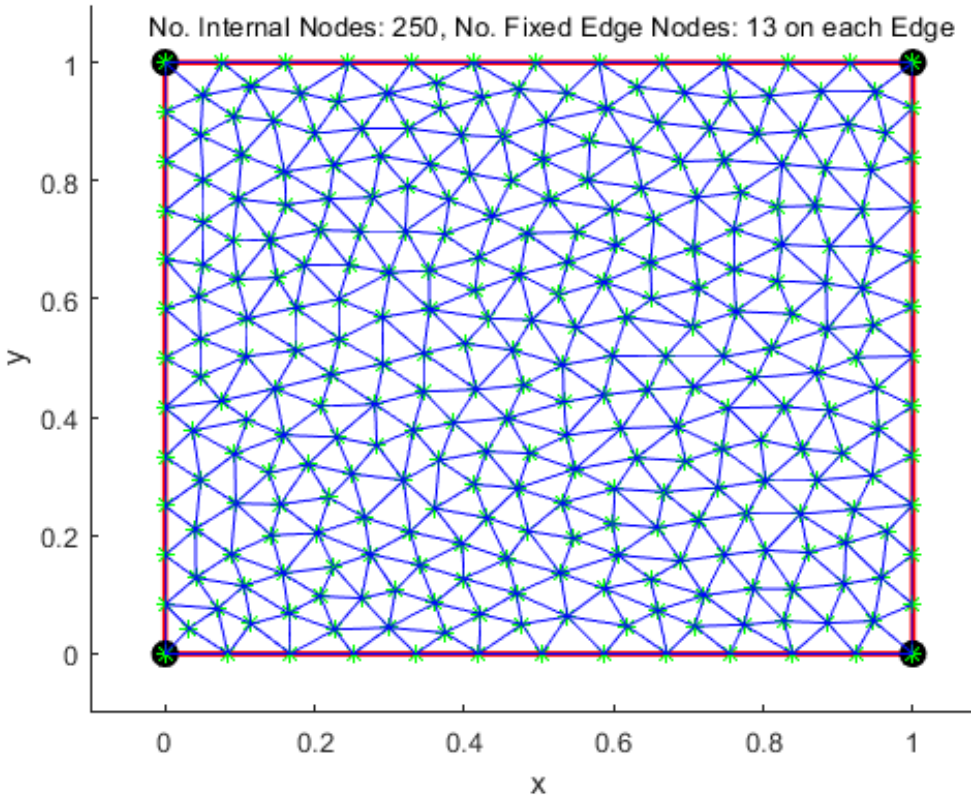


Figure 4: SOM mesh generation result for a 1×1 square geometry with 250 internal neurons and 13 fixed neurons on each edge of the domain.

4.2. Results of non-uniform mesh

To generate nonuniform mesh, we need to pick the points of the domain by applying a probability distribution function. The uniform distribution has been used for simulations resulting in Figures 1 to 4. The histogram of such distribution (for either X or Y coordinates) is depicted in Fig. 5. A combined PDF can be used to provide a mesh with higher density in a specific region. Figure 6 depicts the histogram of a random variable to provide higher resolution in the range (0.4,0.6) comparing to the rest of the region.

To obtain a random distribution whose histogram is depicted in Fig. 6, the following calculations are given. The PDF of such distribution is given by Fig. 7. The black rectangle is a uniform distribution over (0,1) range. We want to increase the probability of the points in range (p,q) in such a way that their chance to be selected is a times greater than the rest of the domain. The aim is to obtain the desired distribution by using a uniform random variable. The total area under the curve is calculated as

$$A_{tot} = 1 + (q - p)(a - 1) \tag{2}$$

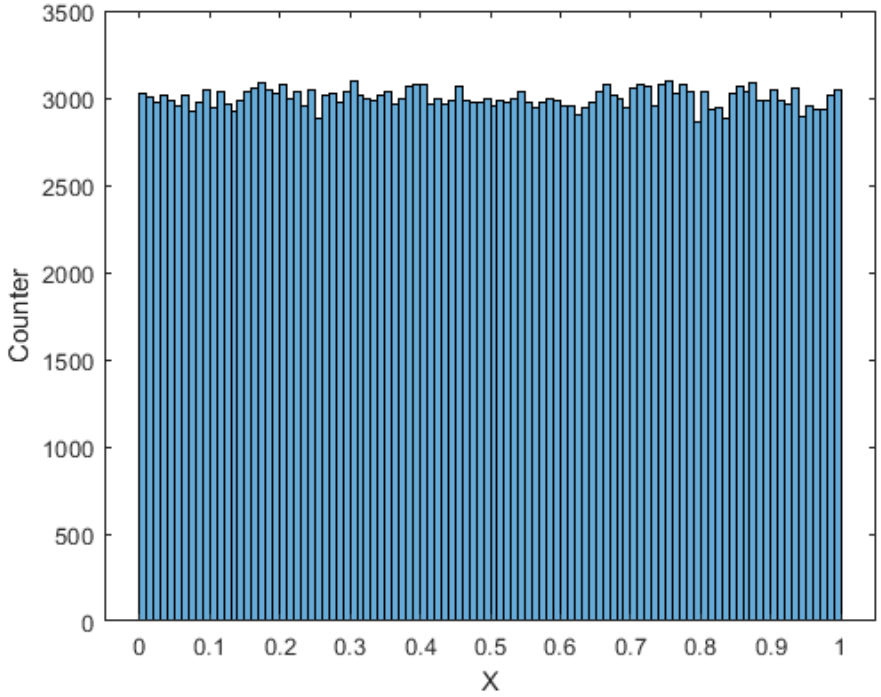


Figure 5: Histogram for uniformly distributed random variable used for SOM simulations of Figure 1 to 4.

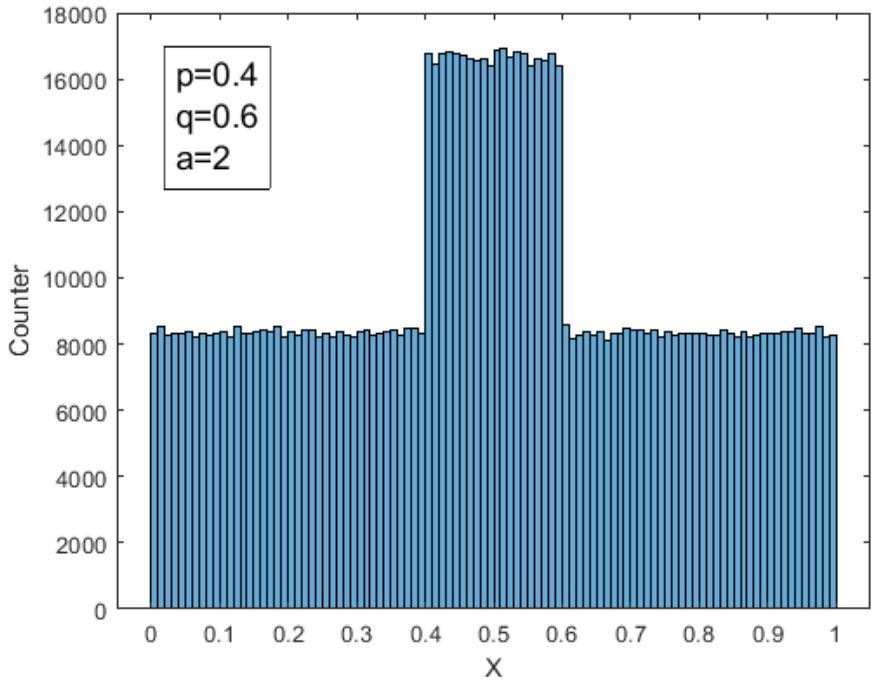


Figure 6: Histogram for the random variable with greater probability in specified range in the middle (according to Eq. (2)).

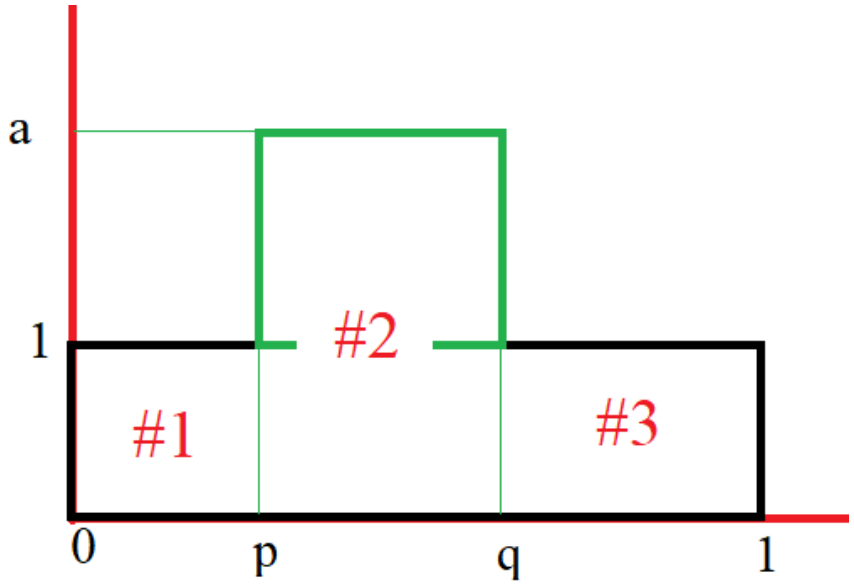


Figure 7: PDF for random variable used in Figure 6.

There are three distinct regions, shown by their number. The probability of a point picked from any region (x_i) is calculated by dividing the area of that region over the total area.

$$\begin{cases} x_1 = \dfrac{p}{A_{tot}} \\ x_2 = x_1 + \dfrac{a \times (q-p)}{A_{tot}} \\ x_3 = x_2 + \dfrac{1-q}{A_{tot}} \end{cases} \tag{3}$$

Accordingly, the desired variable x can be calculated by a uniformly distributed random variable r (which is uniformly distributed in $0 < r < 1$ range) by following criteria.

$$\begin{cases} x = \dfrac{r \times p}{x_1} & if\ r < x_1 \\ x = p + \dfrac{(r-x_1)(q-p)}{x_2 - x_1} & if\ x_1 < r < x_2 \\ x = q + \dfrac{(r-x_2)(1-q)}{1-x_2} & if\ r > x_2 \end{cases} \tag{4}$$

By applying Eq. (4) as the random distribution function, it is possible to have a region in the domain with more resolution of mesh. A sample of such mesh is depicted in Fig. 8, which is an example of mesh generation to investigate the crack formation in solid mechanics. The corresponding 2-D histogram of Fig. 8 is displayed in Fig. 9.

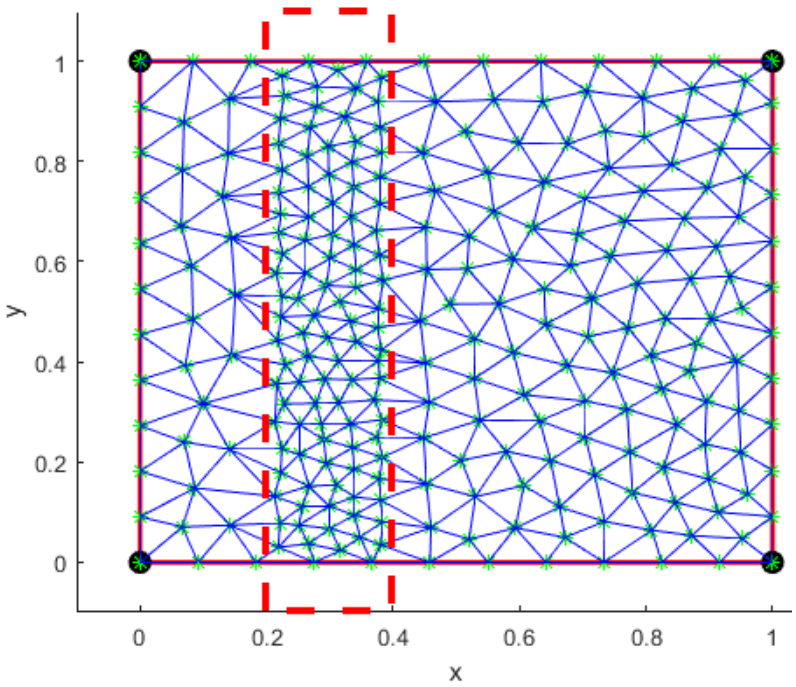


Figure 8: SOM mesh generation with high resolution region for x in range (0.2,0.4). The probability ratio of the dense region to the rest of the domain is 20.

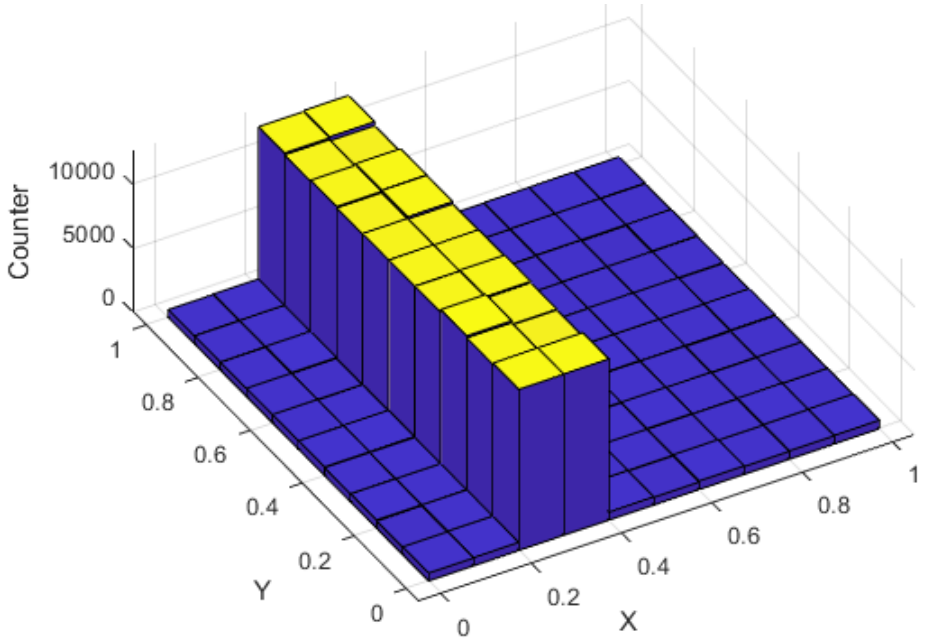


Figure 9: 2-D histogram of distribution function used for SOM in Fig. 8. The high-density region is limited to $0.2 < x < 0.4$. The probability ratio of the selected region to the rest of the domain is 20.

This method can also be used to increase the mesh density near the domain boundaries, as required in boundary layer simulations in fluid dynamics. A sample of a mesh for boundary layer simulation is depicted in Fig. 10. The boundary neurons need to be manipulated in such a way that they result in an appropriate mesh distribution.

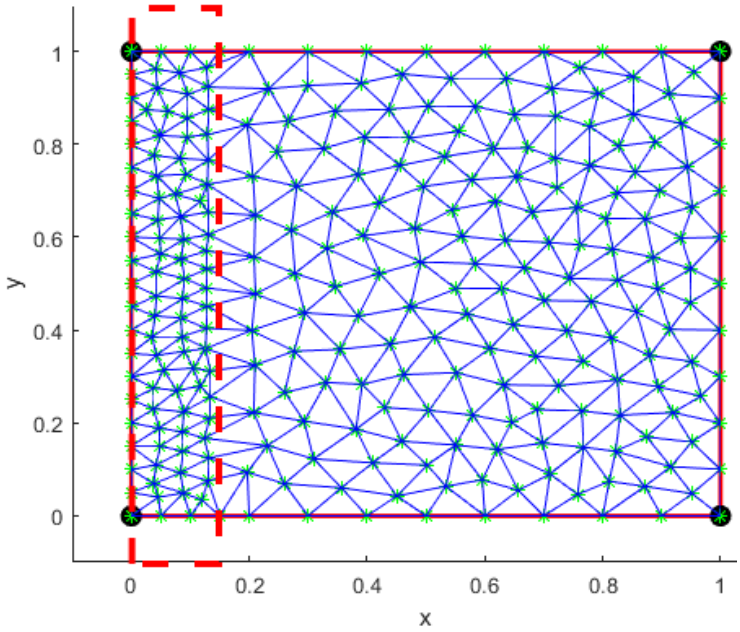


Figure 10: SOM mesh generation with high resolution region for x in range (0.0,0.15). The probability ratio of the dense region to the rest of the domain is 20. The boundary fixed points need to be manipulated to result in an appropriate mesh distribution.

The distribution function described by Eq. (4) is not however appropriate when a 2-D square is considered as the high-density mesh region. Figure 11 displays the 1-D histogram of X and Y variables using Eq. (4) independently to increase the mesh density in a square region in the domain. The result of SOM mesh generation is illustrated in Fig. 12, which contains some unwanted regions with higher mesh density. The corresponding 2-D histogram is also depicted in Fig. 13.

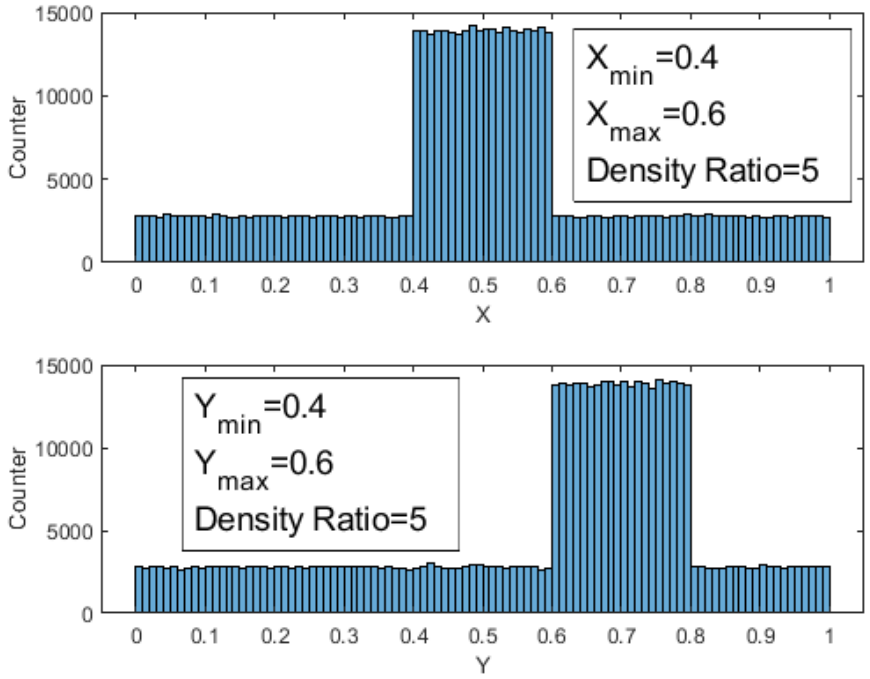


Figure 11: 1-D Histogram for X and Y variable used in SOM mesh generation to increase the mesh density in a certain region.

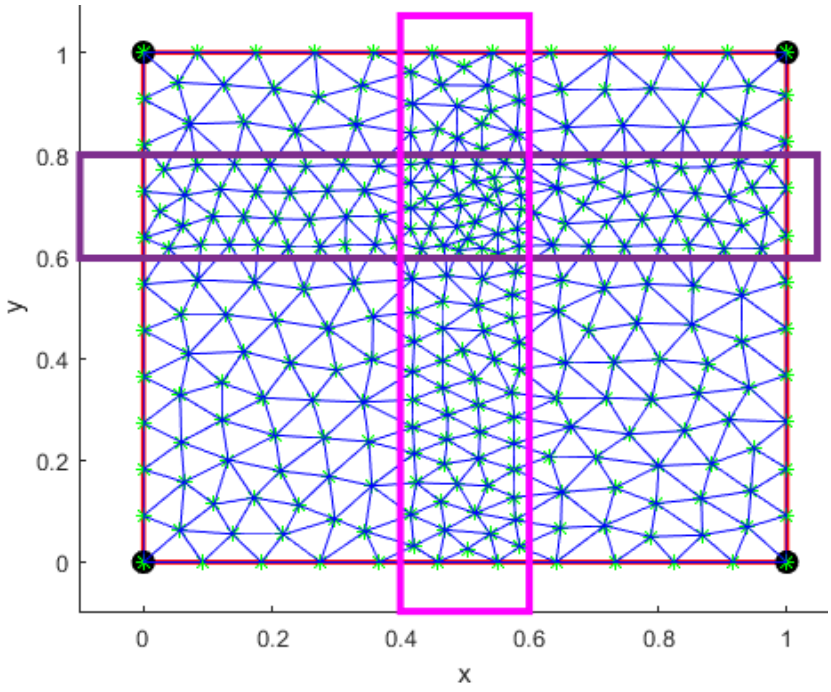


Figure 12: SOM mesh generation with 2 independent implementations of Eq. (4) for *X* and *Y* variables (Corresponding to Fig. 11).

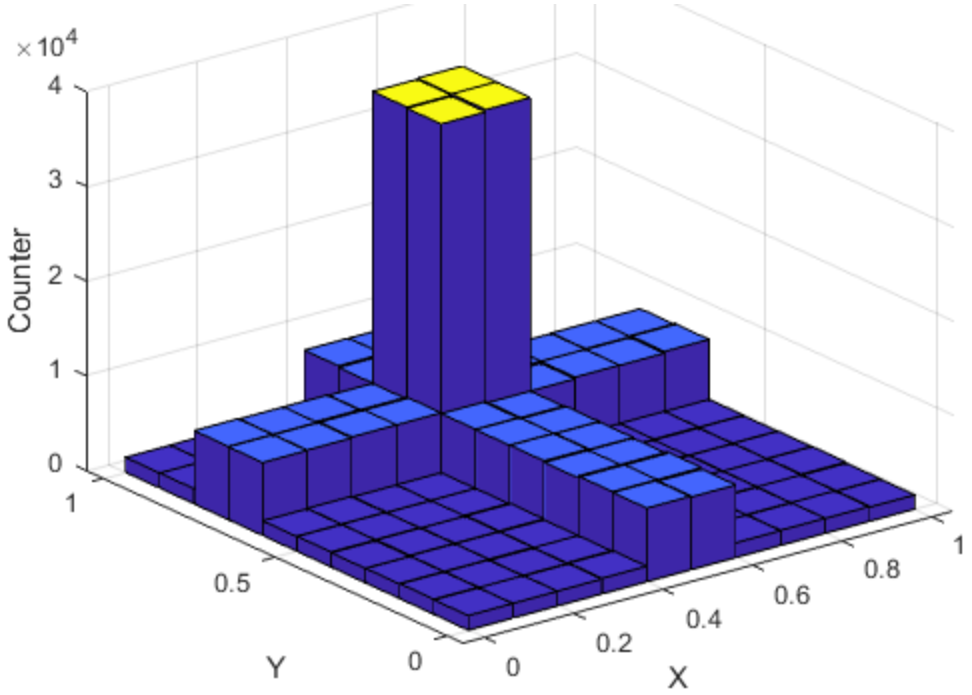


Figure 13: 2-D histogram of distribution function used for SOM in Fig. 12.

One can see in Fig. 13 that the distribution function is not what we expected. Indeed, the desired 2-D histogram for such problem is given by Fig. 14 (top), which only increase the probability of picking the point in the square of interest. The result of SOM mesh generation is illustrated in Fig. 14(bottom) which is desired as the mesh density is enhanced in the region of interest.

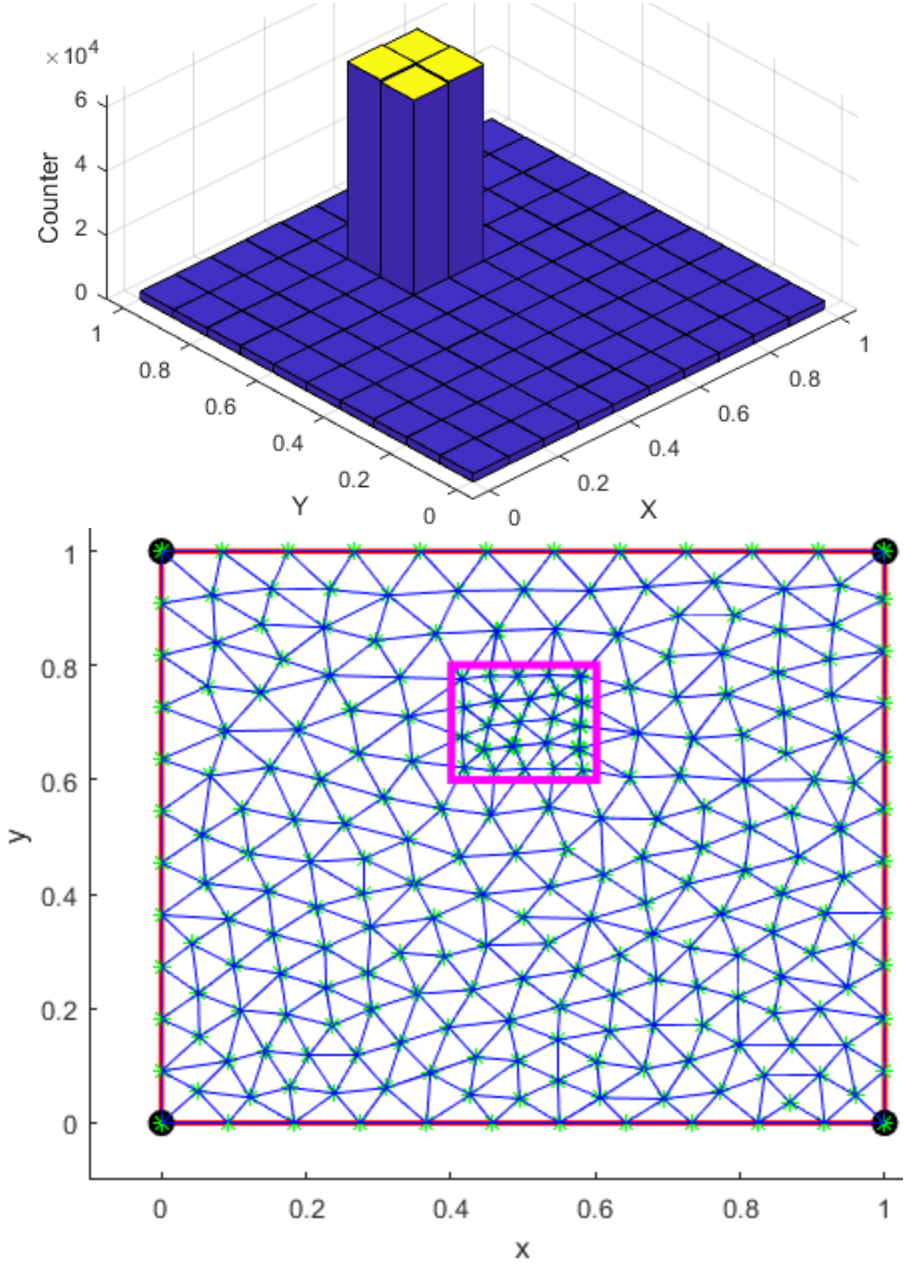


Figure 14: (top) 2-D histogram of distribution function used for increasing the mesh density in the given square. The probability ratio of the dense region to the rest of the domain is 25; (bottom) SOM mesh generation with higher resolution for a given square in the computational domain.

The mathematical procedure of obtaining such probability density function is illustrated here by taking Fig. (15) into account where the top figure depicts the required probability distribution to have a high-density mesh region in the domain and the bottom one shows the 9 possible regions in XY plane.

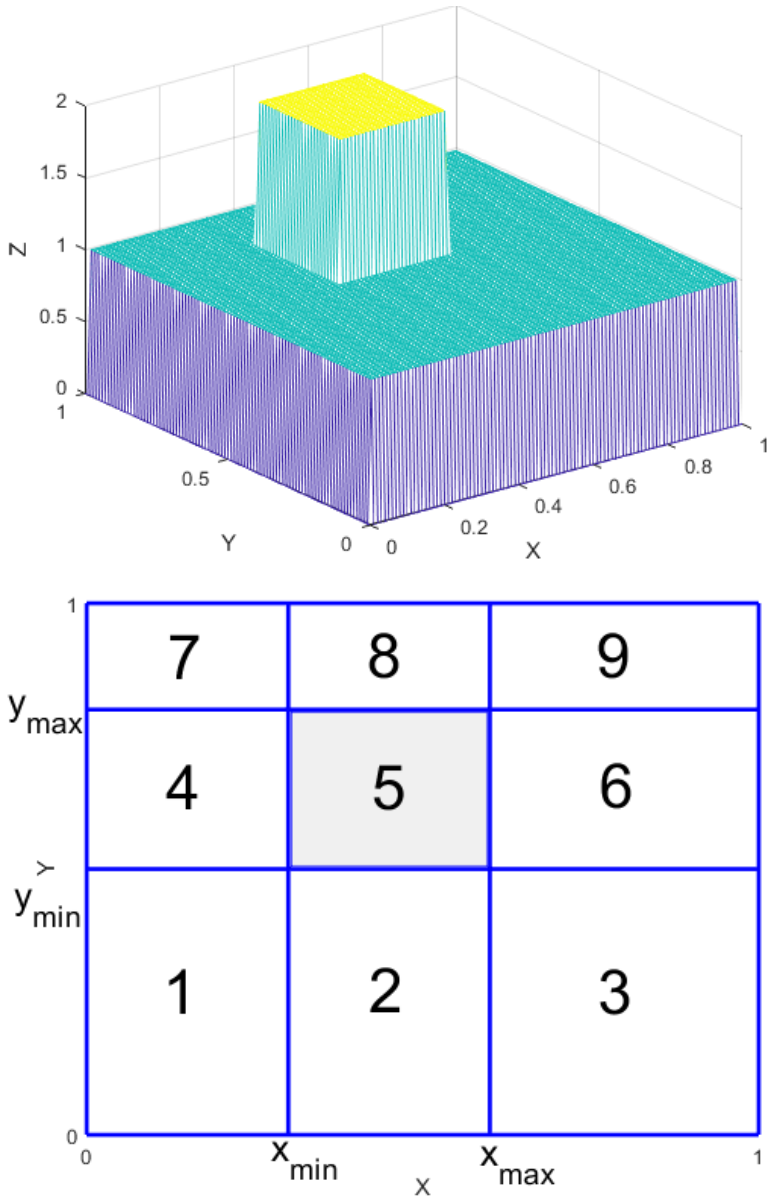


Figure 15: (top) Probability distribution required for high-density grid region inside the 1×1 square. (bottom) Regions in XY plane; zone #5 is aimed to have high-density mesh.

Considering a as the probability ratio of the center region with respect to the rest of the domain, the volume of the 2D PDF can be calculated as follows.

$$V_{tot} = 1 + (a-1)(x\_\max - x\_\min)\,(y\_\max - y\_\min) \tag{5}$$

The following equations give the volume of each region specified in Fig. (15) bottom.

$$V_1 = x_{min} \times y_{min} \tag{6}$$

$$V_2 = (x_{max} - x_{min}) \times y_{min} \tag{7}$$

$$V_3 = (1 - x_{max}) \times y_{min} \tag{8}$$

$$V_4 = x_{min} \times (y_{max} - y_{min}) \tag{9}$$

$$V_5 = a(x_{max} - x_{min}) \times (y_{max} - y_{min}) \tag{10}$$

$$V_6 = (1 - x_{max}) \times (y_{max} - y_{min}) \tag{11}$$

$$V_7 = x_{min} \times (1 - y_{max}) \tag{12}$$

$$V_8 = (x_{max} - x_{min}) \times (1 - y_{max}) \tag{13}$$

$$V_9 = (1 - x_{max}) \times (1 - y_{max}) \tag{14}$$

The probability of selecting a random point from each of the specified sections is given by the following equation.

$$p_i = v_i / A_{tot} \tag{15}$$

The cumulated probability of the regions can then be calculated by the following equation.

$$cp_i = cp_{i-1} + p_i \quad cp_0 = 0 \tag{16}$$

Accordingly, a uniform random number $0 < r < 1$ can be used to select one of the 9 regions as follows.

$$if\ cp_{i-1} < r < cp_i \Rightarrow select\ region\ i \tag{17}$$

After selecting the region of interest, the rest of the problem is similar to randomly select a point from each subdomain using a uniform distribution.

4.3. Parallelogram and triangle

For the 1×1 square (or rectangle) presented in the previous section, two independent random variables can be selected for horizontal and vertical axes to provide the possibility of freely choosing any point from the domain. Considering a domain with a parallelogram shape (see Fig. 16), the two independent random variables can no longer give the points of the domain. For this situation, a transform matrix is needed to convert the parallelogram $OACB$ into a rectangle first, which can be used for mesh generation purpose as described in the previous section. Another way to explain the process is using two independent random variables u and v each one chooses a value between 0 to 1 representing two arbitrary points on OA and OB edges, respectively. The random point of interest can then be found by drawing two parallel line from the corresponding points on the OA and OB edges that meet somewhere inside the domain (e.g., points #1 and #2). For a triangle-shape domain like OAB in Fig. 16, a similar approach can be engaged with some minor modifications. the triangle is treated like what was proposed for parallelogram to obtain the point by drawing lines parallel to OA and OB. The resulted point can be located either inside the triangle (e.g., point #1) or outside it (e.g., point #2). The former point is introduced as the output of the algorithm while the second point is selected by finding the reflected point of point #2 with respect to the midpoint of the edge AB. Thus, point #3 is the result of such algorithm as depicted in Fig. 16.

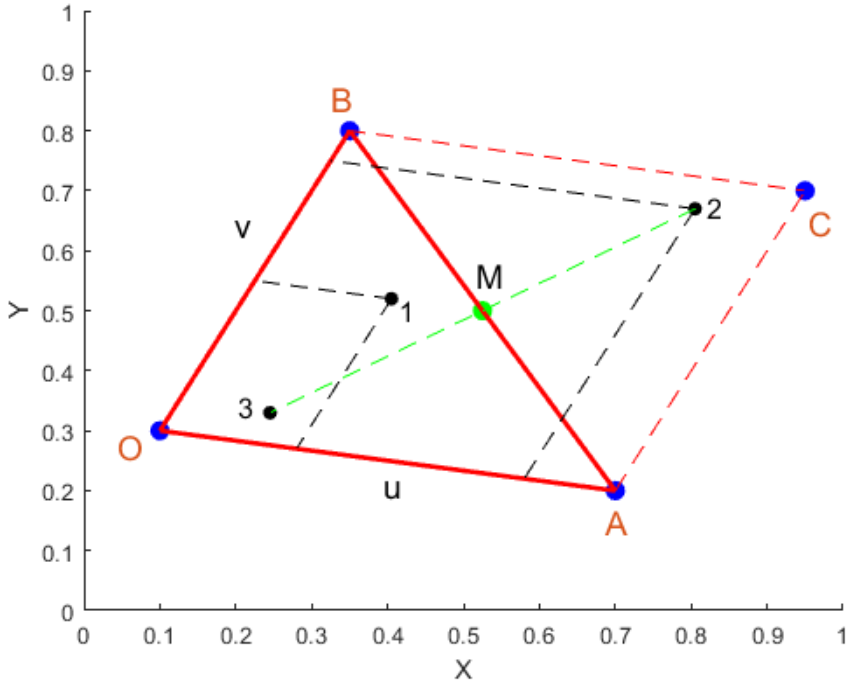


Figure 16: SOM mesh generation with higher resolution for a given square in the computational domain.

Figure 17 displays the initial locations of the SOM neurons selected by the uniformly distributed random function which was described by Fig. 16. The black points on the triangle edges are the fixed points which are defined by the number of boundary points at the beginning of the SOM calculations. As described for the square domain in the previous section, the boundary points participate in all steps of SOM calculations, including finding the winners neurons and the repelling process, but their locations are fixed and not changed over the iterations. The results of SOM approach for mesh generation of the triangular domain are depicted in Fig. 18. One can see the poor quality of the resulted mesh since some grid nodes are located outside the domain. Indeed, the repelling step enforces some SOM neurons to scape from the boundary, which is not acceptable for the final mesh.

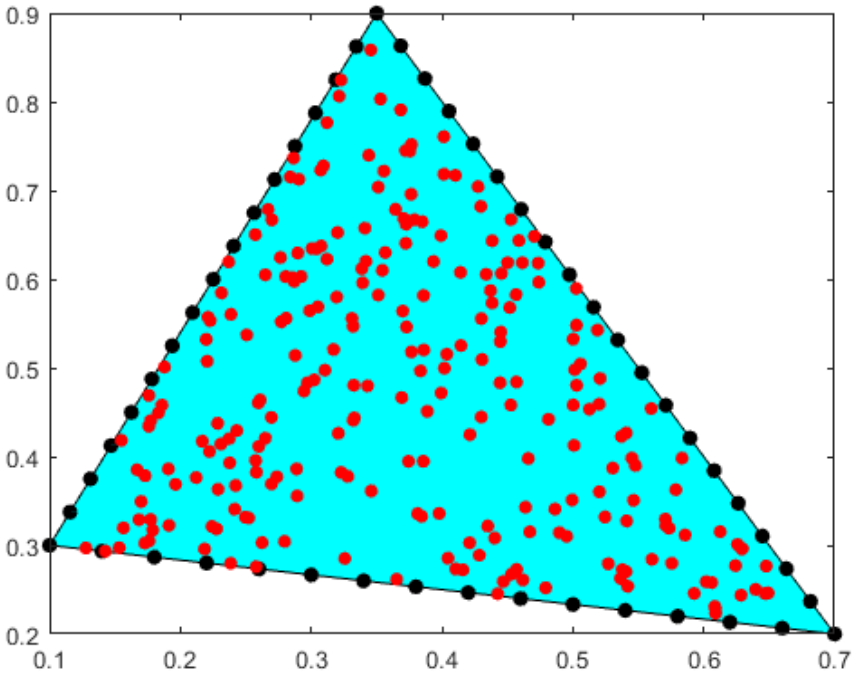


Figure 17: Initial SOM neurons on the triangle domain selected by uniformly distributed random function.

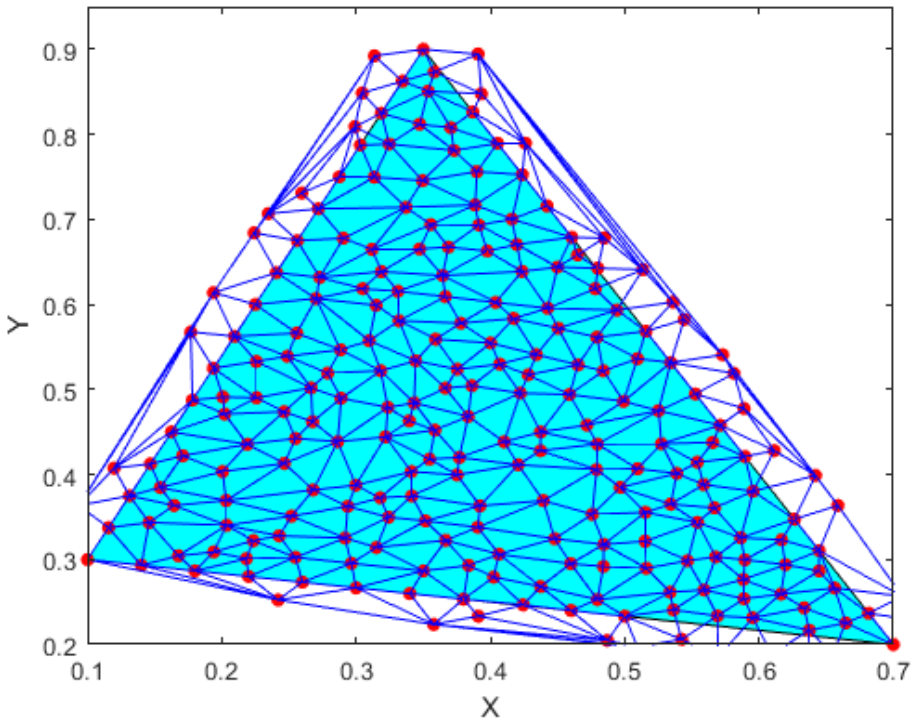


Figure 18: Results of SOM mesh generation with uniform PDF over triangular domain without limiting the internal points into the domain boundaries.

There are some remedies to prevent the internal neurons from moving out of the domain, including:

- Imposing a strong repelling function at the boundaries to push back the close neurons into the domain,
- Modifying the final locations of the internal neurons to ensure that they are located inside the domain,
- Or simply removing the outside neurons after finishing the SOM iterations.

The results of SOM mesh generation are displayed in Fig. 19 which is obtained after all required modifications.

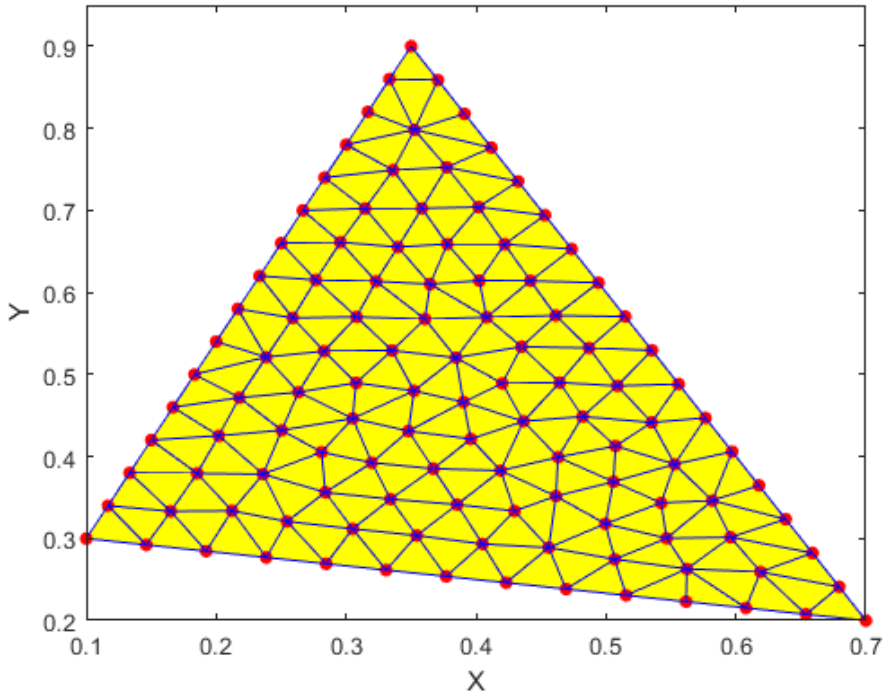


Figure 19: Final SOM mesh generation results for triangular domain with uniform distribution.

5. Complex Geometries

The SOM mesh generation has been applied on the simple geometries (i.e., 1×1 square and triangle) so far. To further delve into the topic, the implementation of the proposed technique will be studied over complicated domain shapes which brings some computational challenges. Recalling this fact that SOM is implemented through randomly selected points in the domain, the distribution function that generates the random points is crucial. The random points should be selected from the entire computational domain of interest with any customized density function defined to focus on important regions of dense grids. Hence, the mesh generation algorithm should be able to select random points from any arbitrary probability distribution function (PDF).

There are some methods to select random variables from a given PDF. Two methods are explained here including the inverse method and the rejection approach.

5.1. Selecting random variable from a PDF by 1-D inverse method

Figure 20 (top) displays an arbitrary PDF of interest which the 1-D random variable X should be selected from. the given PDF indicates the density probability of selecting each point on the horizontal axis with respect to other points of the domain. For instance, there will be no selected point for $0.5 \leq X \leq 1$ range since the corresponding PDF is equal to zero. The vertical scale of PDF diagram is selected in such a way that the area below the curve is equal to unity.

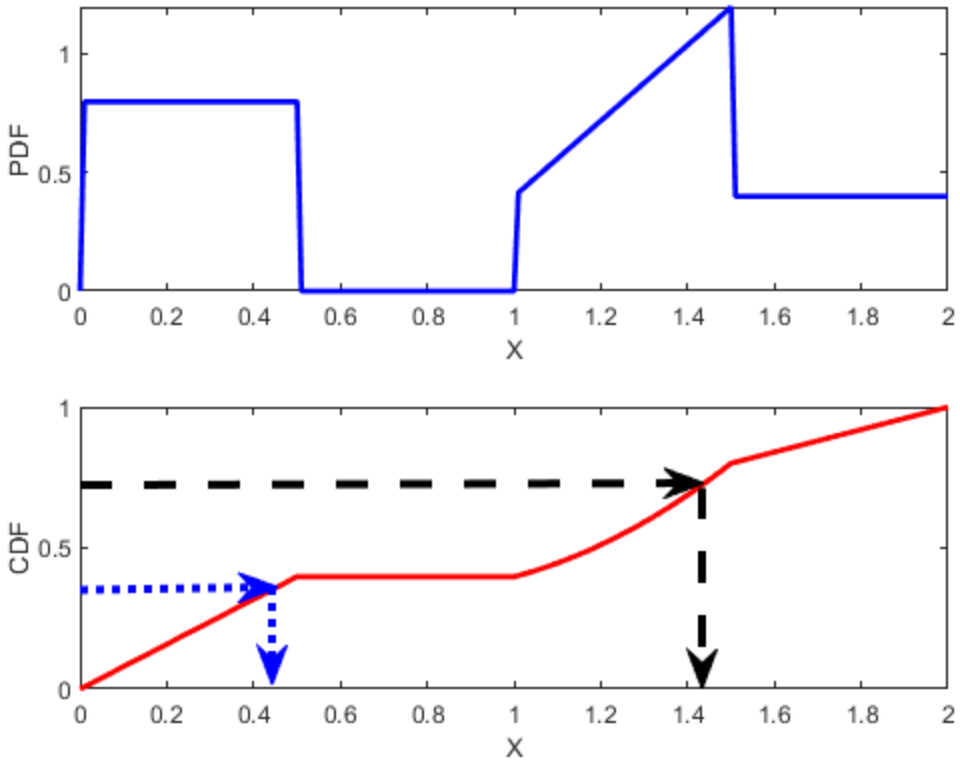


Figure 20: Description of the inverse method; top: An arbitrary PDF; bottom: The corresponding CDF (cumulative distribution function).

Figure 20 also depicts the cumulative distribution function (CDF) for the given PDF diagram, which is defined by Eq. (18). For a bounded PDF like Fig. 20 (top), the CDF value always starts from zero at the beginning (i.e., $X = 0$) and reaches to unity at the end of the range (i.e., $X = 2$).

$$CDF(X) = \int_{-\infty}^{X} PDF(s)ds \tag{18}$$

The desired random variable X can be obtained by simply selecting the standard uniform variable $0 \leq r \leq 1$ on the vertical axis and find the crossing point of the corresponding horizontal line with the CDF curve. The desired X value can then be found by projecting the resulted point on the horizontal axis, as depicted in Fig. 20 (bottom). The distribution of desired random variable (X) versus the standard uniform variable ($0 \leq r \leq 1$) is shown in Fig. 21. It can be easily seen that the plot is the inverse of the CDF curve given in Fig. 20.

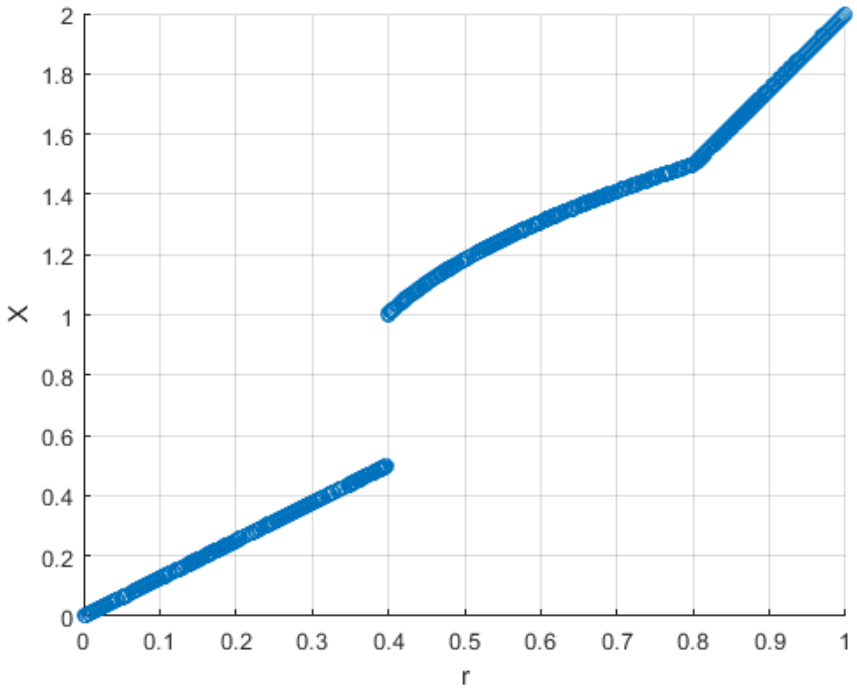


Figure 21: Distribution of desired random variable (X) versus the standard uniform variable ($0 \leq r \leq 1$). The plot is the inverse of the CDF curve given in Fig. 20 (bottom).

The inverse method approach is engaged to produce 1-D random variable from the PDF given by Fig. 20, whose results are shown as a histogram plot in Fig. 22 for $1E6$ points. One can see that the histogram of Fig. 22 is perfectly compatible with the PDF function given in Fig. 20, that indicates the accuracy of the suggested algorithm.

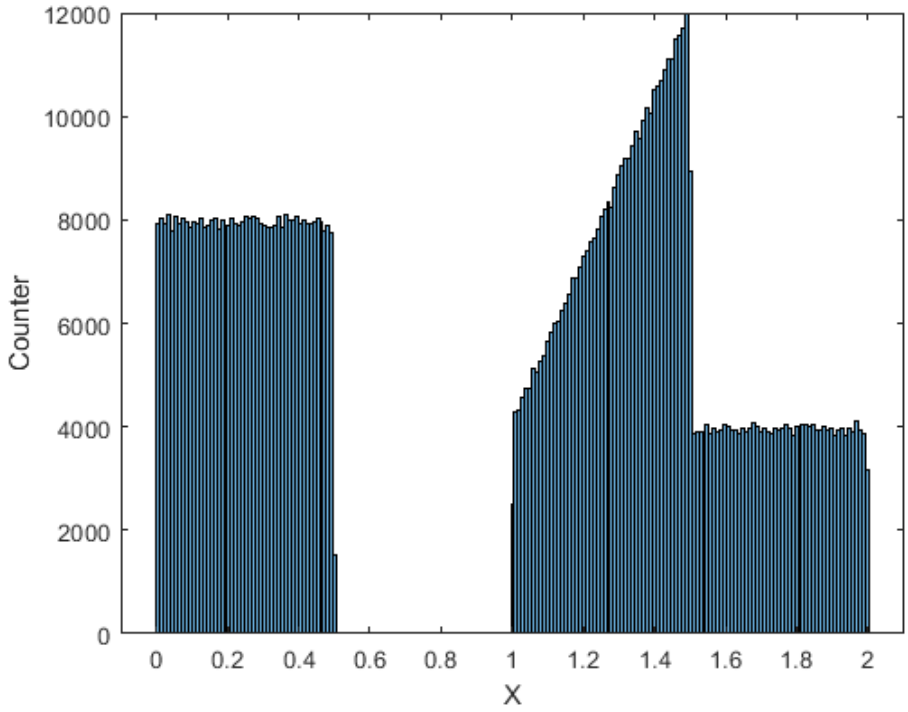


Figure 22: Histogram for random variable X obtained from the PDF in Fig. 20 using the inverse method. Number of points: 1E6.

5.2. Selecting random variable from a PDF by 1-D rejection method

The rejection method is another approach to obtain the random variable from a bounded PDF which is described by Fig. 23 schematically. For a 1-D random variable X, two standard uniform variables u and v are selected over horizontal and vertical axes, respectively. the range of the former variable is equivalent to that of the desired random variable (i.e., X) whereas the latter (the vertical one) is selected uniformly between zero and the global maximum of the PDF curve. the corresponding horizontal value is accepted if the resulted point is located below the PDF curve (the green point). Otherwise, it is rejected (the magenta point).

Figure 24 displays the histogram of 1E6 tries of the PDF given by Fig. 20 obtained by the rejection method. The results show a good compatibility with the PDF function which is a sign that the method in implemented properly.

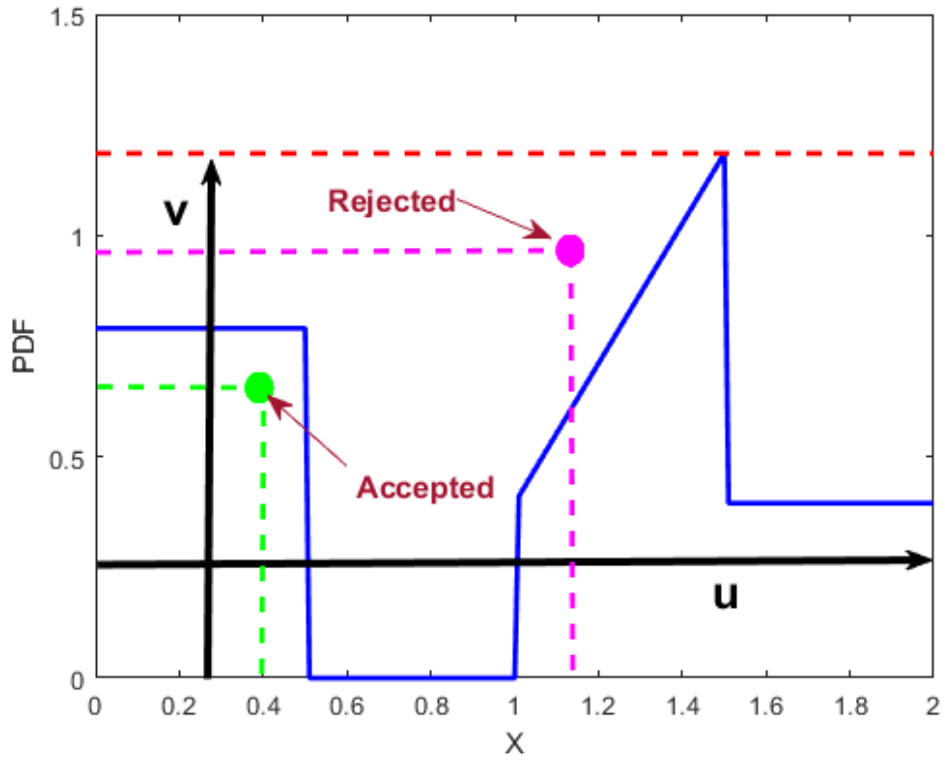


Figure 23: Description of the rejection method to select random variable from a customized PDF.

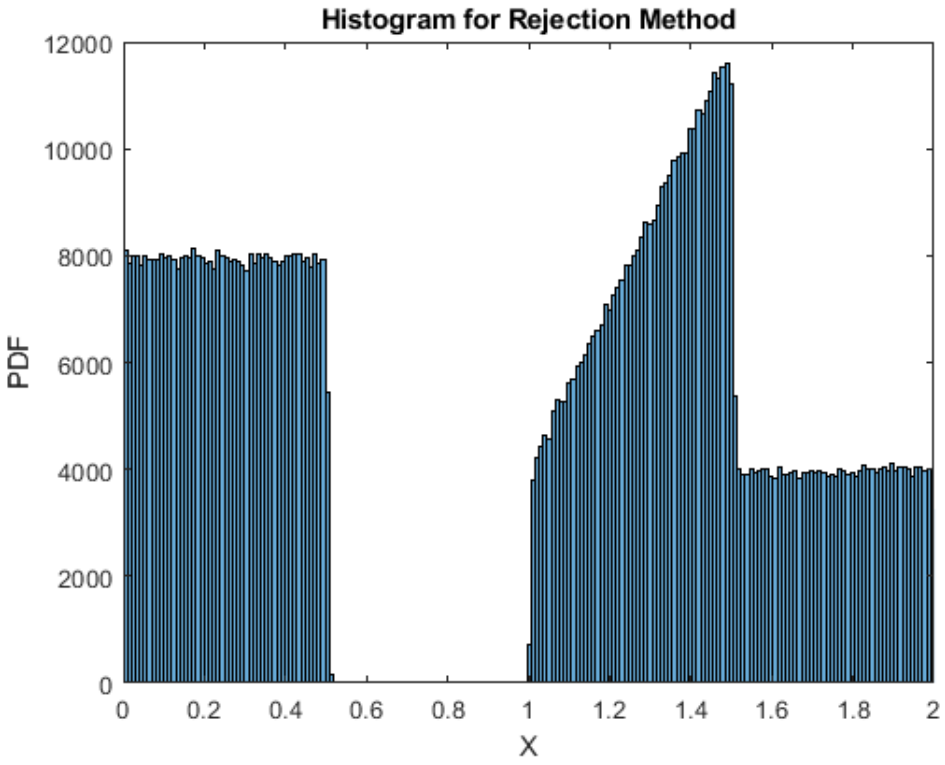


Figure 24: Histogram for random variable *X* obtained from the PDF in Fig. 20 using the rejection method. Number of points: 1E6.

5.3. Selecting random variable from a PDF by 2-D rejection method

The implementation of the rejection method for 2-D is similar to the 1-D case. The aim is to randomly choose a point in XY plane whose probability density function is given by a 2-D PDF surface like Fig. 25. Three standard uniform random variables are selected, namely u, v and w for X, Y and PDF value axes, respectively. The resulted coordinates generate a random point in 3-D space which can be located below or above the PDF surface. Those points located below the PDF surface are accepted and their projected points in XY plane are reported as the output of the algorithm. Any point located above the PDF surface is also rejected.

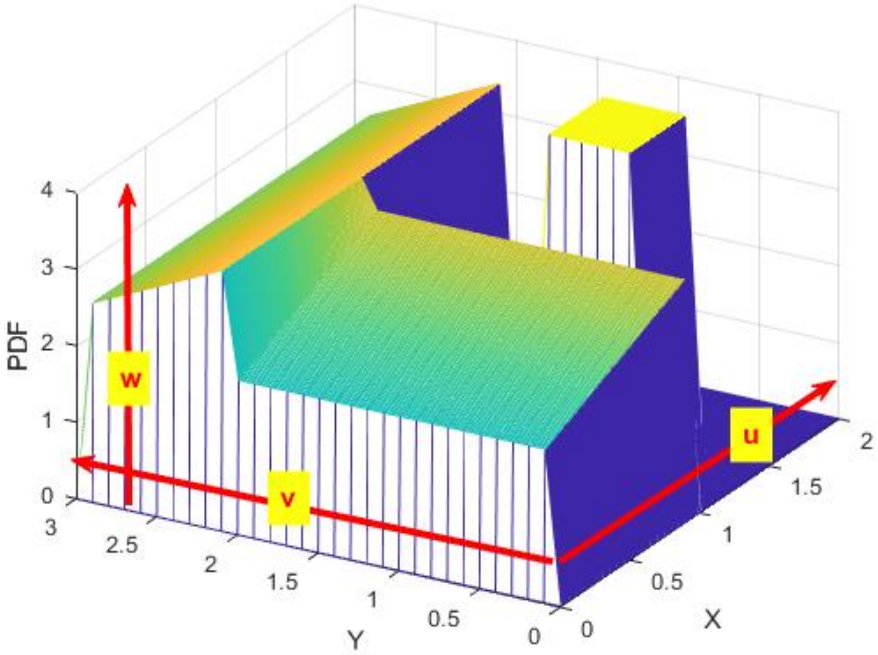


Figure 25: Description of the rejection method to select random variable from an arbitrary 2-D PDF.

Figure 26 displays the results of implementation of the 2-D rejection method for the PDF given in Fig. 25 for 1E7 points which shows a very good agreement. It should be mentioned that the rejection method for multidimensional spaces is time demanding, especially for complex PDF shapes.

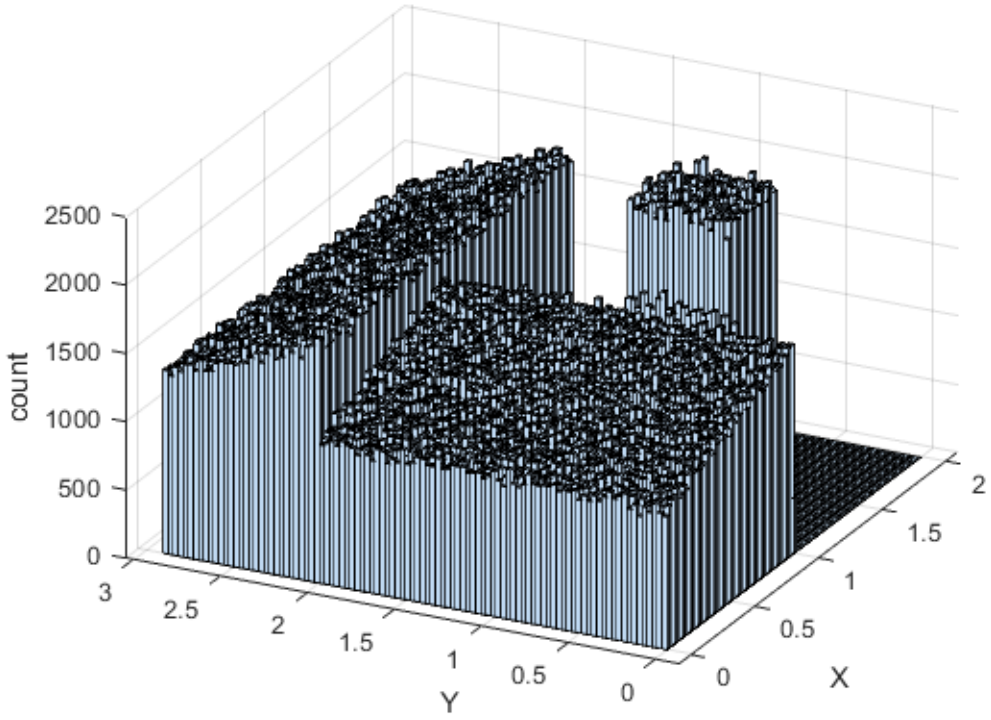


Figure 26: 2-D Histogram for random variables *X* and *Y* obtained from the 2-D PDF in Fig. 25 using the rejection method. Number of points: 1E7.

5.4. Computational approach to find regions inside and outside a domain

The human brain can comprehensively recognize whether a point is located inside a given domain or not. However, we need to provide the computer with a metric to check if a given point is inside a domain. For simply connected 2-D domains, the domain can be given by a series of XY points indicating its boundary. The domains containing one or more holes inside can be defined by implementing the Boolean operations on the fewer complex simply connected geometries which will be discussed later in the current project. The method is described for a triangle in Fig. 27, which can be simply generalized to any arbitrary domain. For any point located inside a simply connected domain, the sum of the signed angles formed by connecting the point to two consequent nodes is equal to either $\pm 2\pi$. For the given triangle in Fig. 27-left, the sum of the internal angles is 2π when the boundary points are selected in a counterclockwise direction. The clockwise direction similarly gives the sum as -2π. As displayed in Fig. 27 right, any other point outside the domain gives the sum of the internal angles equal to zero. Figure 28 illustrates the calculation of the signed angle for given points in the 2-D plane, which is easy for code implementation. The internal signed angle can be obtained by implementation of the cross-production operation on the vectors v_1 and v_2, whose components can be easily calculated using the coordinates of the boundary points (P_1 and P_2) as well as the given point of interest (Point O). The formulation also shows that changing the direction leads to obtain the negative value for angle because of the characteristics of the cross-product operation.

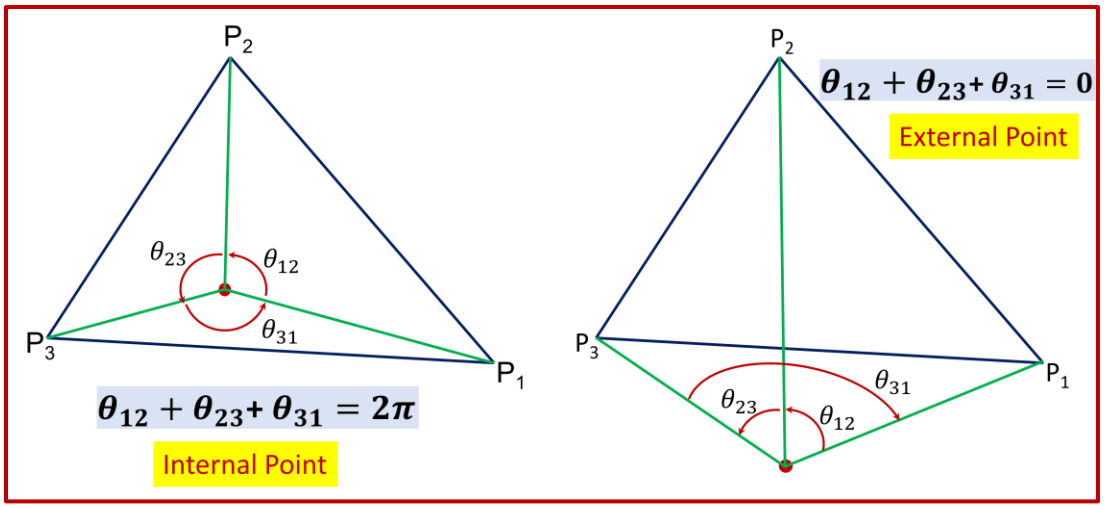


Figure 27: Mathematical metrics for inside/outside regions of a simply connected geometry: A simple triangle.

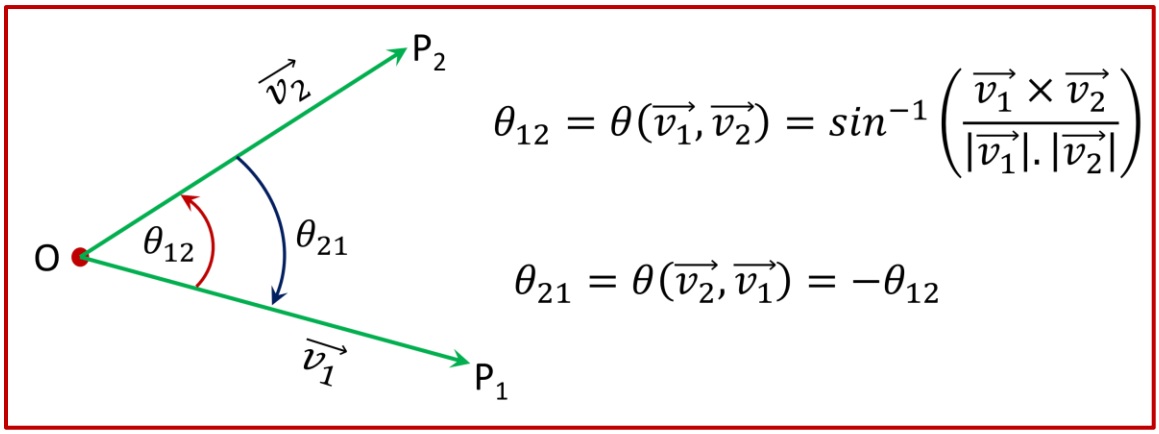


Figure 28: Calculation of the signed angle based on the cross-product operation

Figure 29 displays the evaluation of the given algorithm for distinguishing the points located inside or outside for an arbitrary simply connected domain. A Set of 100 points are picked up randomly in a 1×1 square and tested by the given algorithm. The points located inside the domain are marked blue whereas the outside points are colored red. The accuracy of the implementation of the algorithm can be evaluated by Fig. 29.

5.5. Uniform mesh generation for an arbitrary simply connected domain

The implementation of SOM for mesh generation of an arbitrary simply connected domain is shown in this section. The selected domain is the same as the one displayed by Fig. 29. Before starting SOM calculations, the set random points inside the domain are generated by a separate MATLAB program whose 2-D histogram is shown in Fig. 30 from2 different viewpoint. Total number of generated points is about 1.5E6.

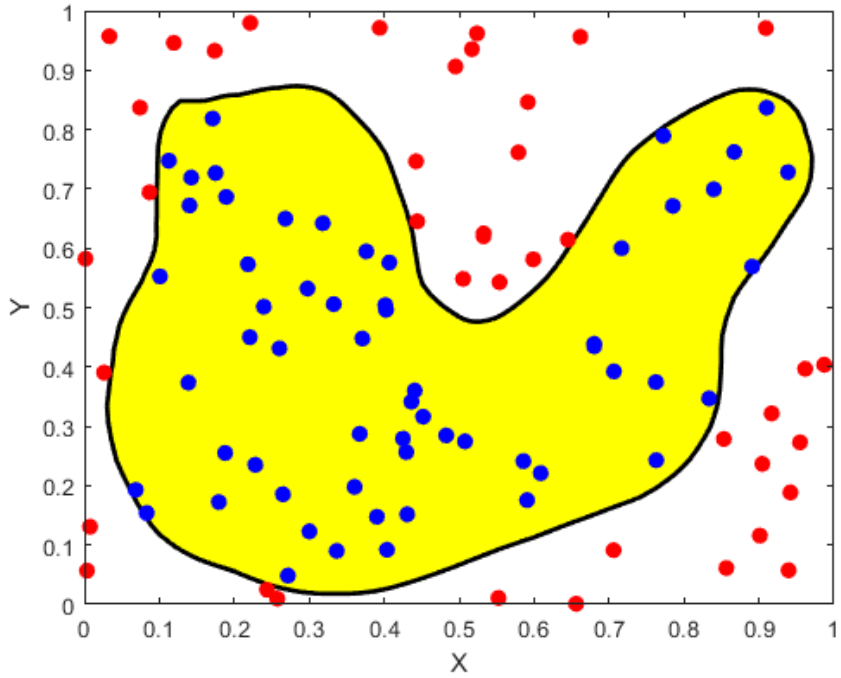


Figure 29: Evaluation of the MATLAB implementation of the algorithm to distinguish inside/outside points for an arbitrary simply connected domain. The colors given to the points by MATLAB code indicating whether they are inside (blue) or outside the domain.

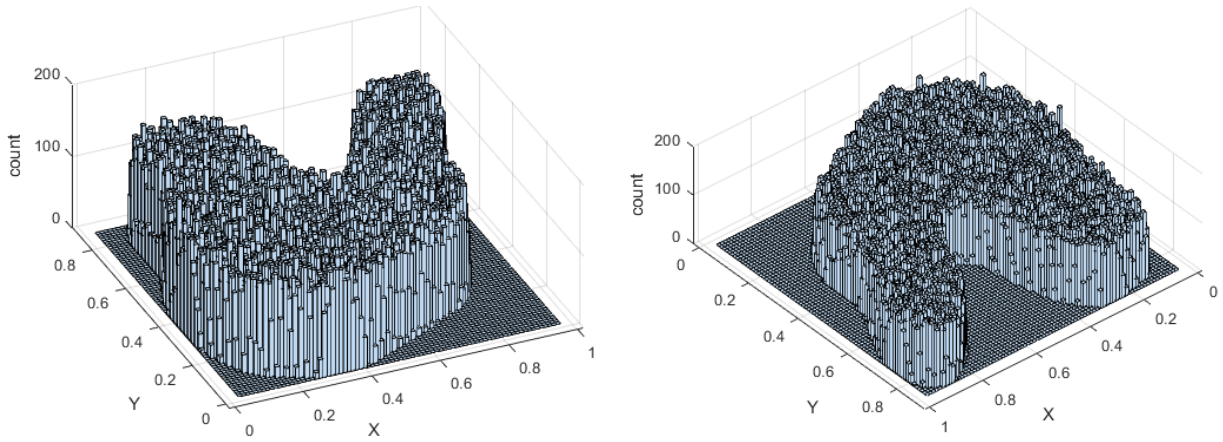


Figure 30: 2-D histogram for random points uniformly picked from the simply connected domain given by Fig. 29.

The initial SOM neurons are depicted in Fig. 31, where the black points are boundary nodes whose locations are kept fixed during the SOM iterations and the red points indicate the internal neurons whose locations are modified through introduced random points picked from the distribution shown in Fig. 30. The boundary nodes are selected uniformly over the boundary path. The result of SOM mesh generation is depicted in Fig. 32, which shows a reasonable mesh distribution for the given geometry.

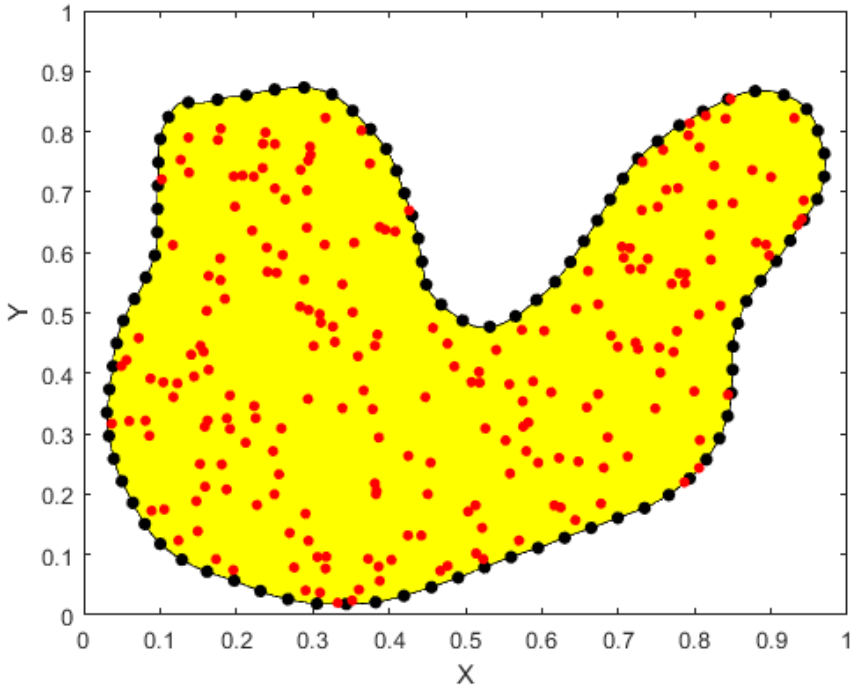


Figure 31: Initial distribution of the SOM neurons as well as the boundary nodes for the given simply connected geometry

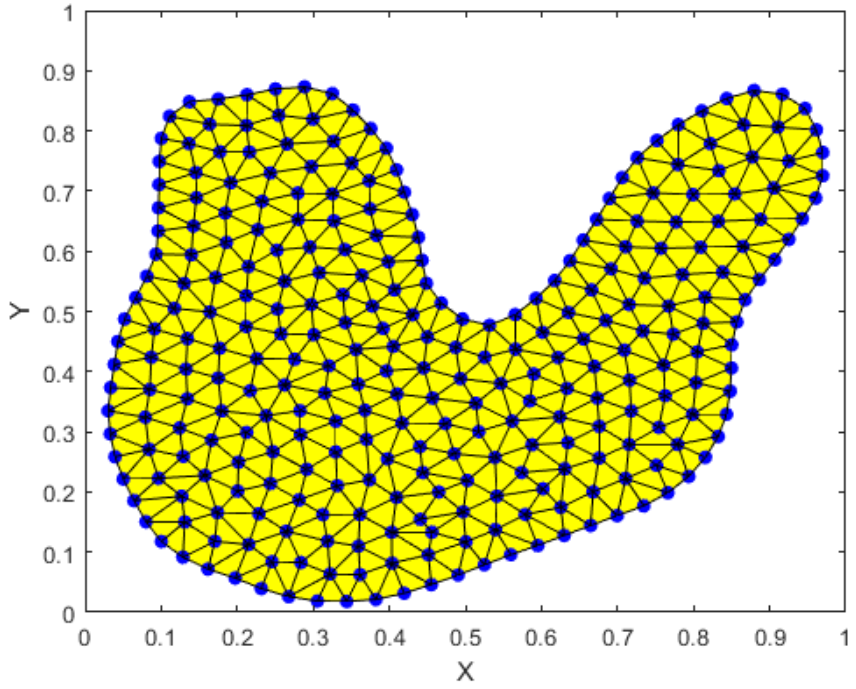


Figure 32: Results of mesh generation with SOM obtained by random points uniformly distributed across the domain (Picked from the distribution shown in Fig. 30).

5.6. Simply connected arbitrary domain with nonuniform mesh distribution

For nonuniform mesh generation of an arbitrary simply connected domain, the random points should be picked up from a given PDF. The rejection method can be implemented over the given domain to select the random points needed for the SOM calculations. To simply illustrate the approach, the geometry presented in the previous section (Fig. 29) is considered as the computational domain where a circle of diameter 0.2 whose center located at $[x_0, y_0] = [0.3,0.3]$ is selected as the sub-region with a higher mesh density. The density of the circle of interest is considered 6 times greater than the rest of the domain.

Figure 33 depicts the 2-D histogram of the points generated randomly for the given domain with a high-density circle inside the domain. Number of points is about 2E6. The distribution of the initial SOM neurons is displayed in Fig. 34, where the fixed boundary nodes are depicted in red, and the inside neurons are colored red. Since the initial neurons are also picked from the nonuniform distribution shown in Fig. 33, the density of the initial neurons is higher in the circle (depicted in blue) as well.

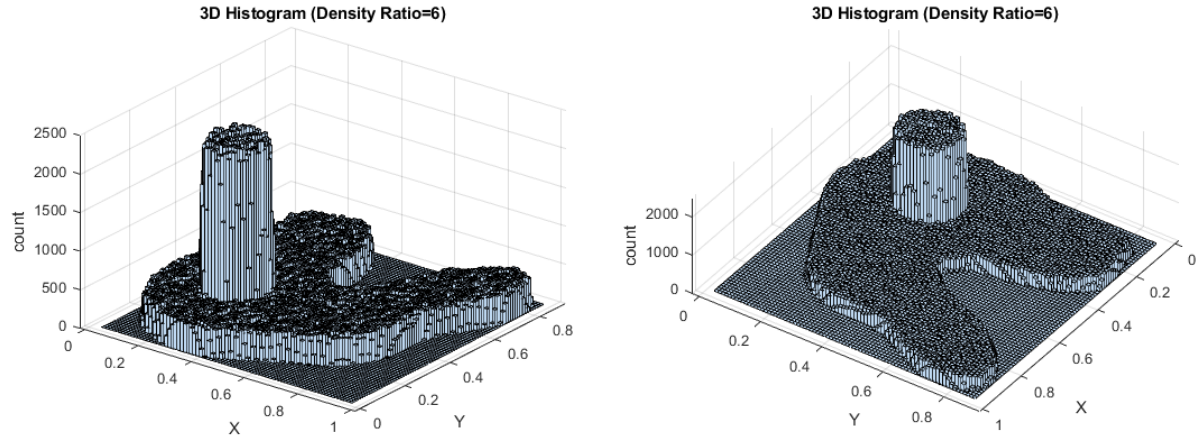


Figure 33: 2-D histogram of nonuniform PDF for the domain shown in fig. 29.

The results of nonuniform mesh generation with SOM is shown in fig. 35, where the mesh density obviously seem greater in the circle of interest. However, the quality of the produced mesh inside the circle is not as good as the other parts of the domain. More investigations are required to obtain an appropriate nonuniform mesh for arbitrary domains.

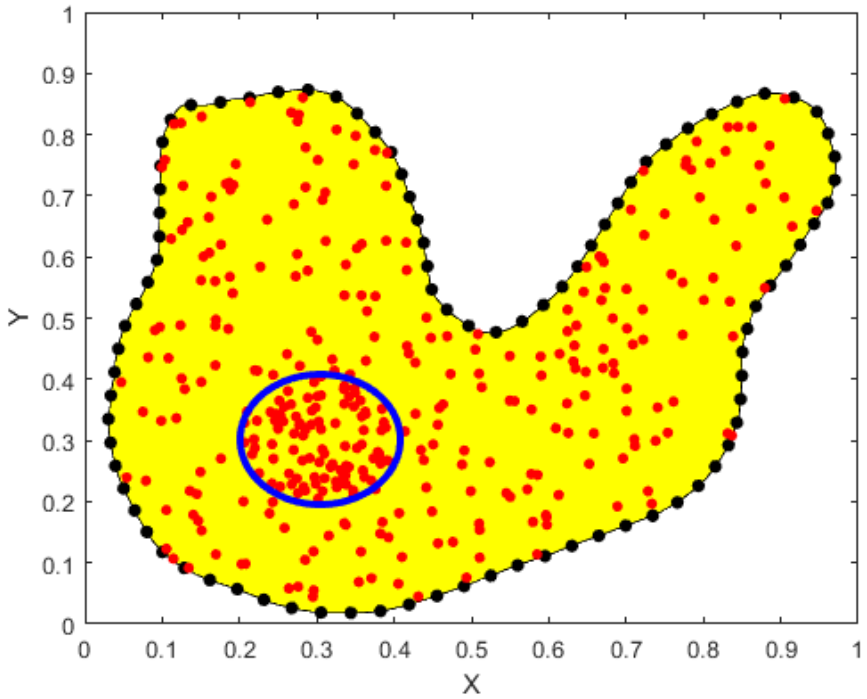


Figure 34: Initial distribution of the SOM neurons as well as the boundary nodes for the given simply connected geometry with nonuniform distribution.

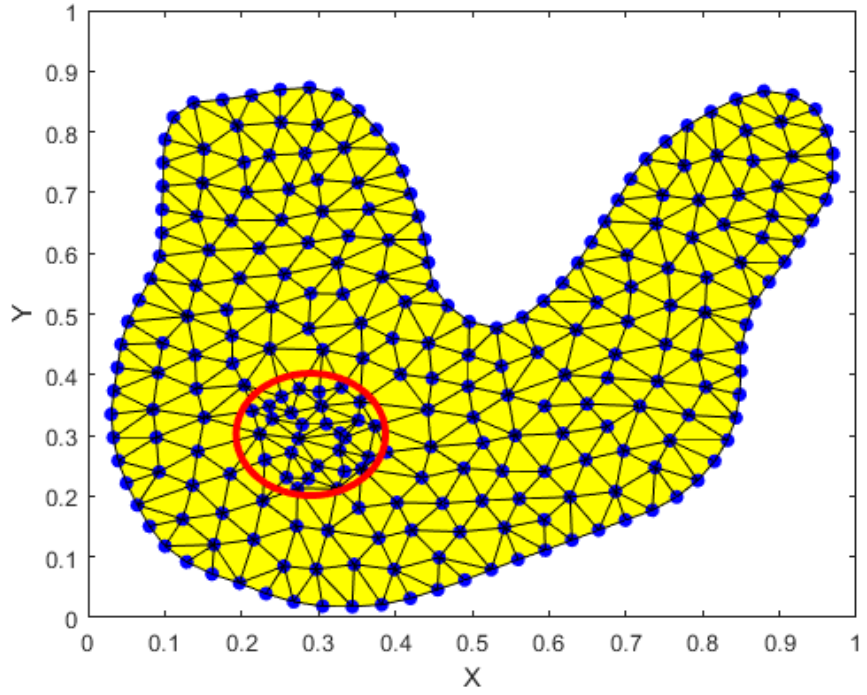


Figure 35: Results of mesh generation with SOM obtained by random points nonuniformly distributed across the domain (Picked from the distribution shown in Fig. 33).

As described earlier, the domains with one or more holes inside them can be obtained by applying Boolean operations on the simply connected geometries. Figure 36 illustrates the 2-D histogram for the domain given by Fig. 29 where the points in the internal circle $(x-0.3)^2+(y-0.3)^2=0.1^2$ are excluded to resemble a circular hole inside the domain. The corresponding distribution of the initial neurons is also depicted in Fig. 37. There is a new set of boundary nodes indicating the fixed neurons of the hole boundary, which is required to be defined before running the SOM iterations.

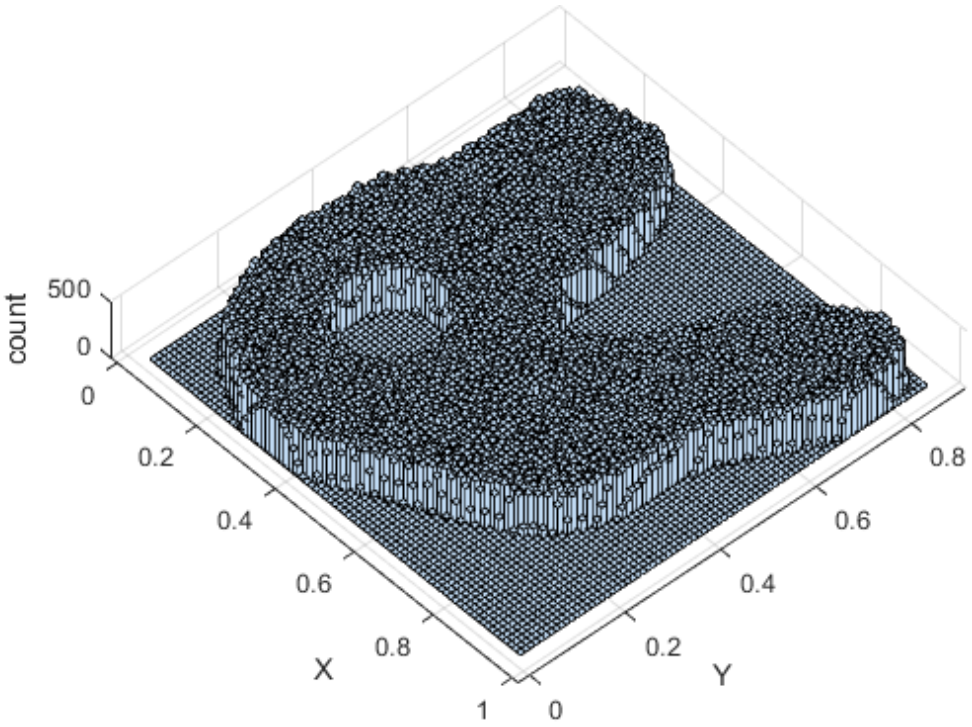


Figure 36: 2-D histogram for random points uniformly picked from a domain with a hole inside.

The results of the mesh generation for the given boundary with a hole inside is depicted in Fig. 38, which shows an appropriate grid distribution over the domain.

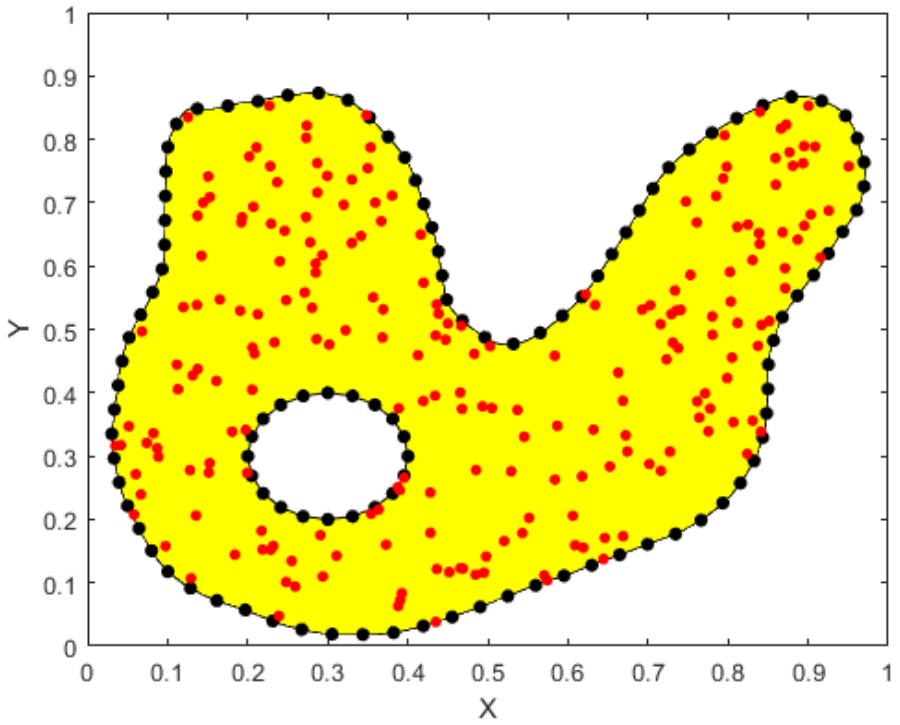


Figure 37: Initial distribution of the SOM neurons as well as the boundary nodes for the given domain with a hole inside.

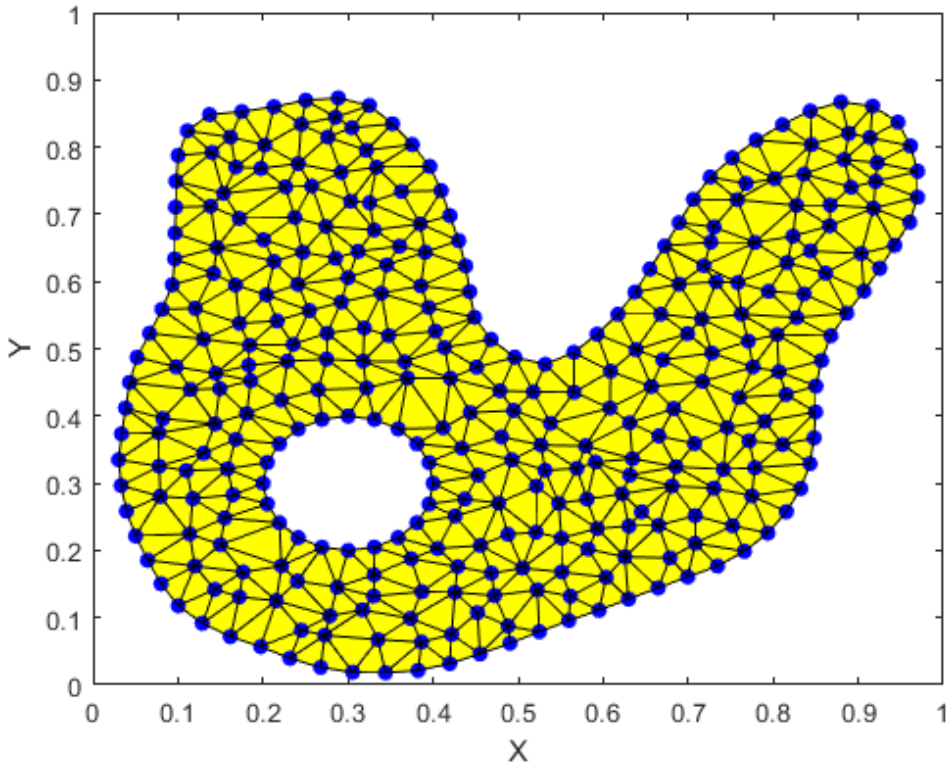


Figure 38: Results of mesh generation with SOM obtained by random points uniformly distributed across the domain with a hole inside (Picked from the distribution shown in Fig. 36).

6. Conclusions

The application of the SOMs for mesh generation was studied. As an unsupervised classification method, the SOM is used for clustering a huge amount of unlabeled datapoints into limited number of clusters which is somehow like mesh generation where the infinite number of points in a continuous domain are represented by a limited number of mesh cells/nodes.

The SOM calculations were performed by introducing random points from the domain and find the winner neurons, whose locations were consequently modified toward the given random point in each iteration. The winner neuron (i.e., 1st place) was receiving the maximum effect from the datapoint (i.e., lateral coefficient of unity) while the rest of the winner neurons were receiving less effects according to their distance from the winner one. The total effect from the newly introduced datapoints was controlled by the learning rate, which was gradually decreasing during the iterations. A repelling mechanism was defined to avoid the neurons to be located too close to each other. At the end of any iteration, the neurons whose distance from the winner neuron was less than a threshold were enforced to radially move away from it.

For a 2-D rectangular domain, two independent standard random variables were chosen to generate datapoints in the domain required for SOM calculations. Appropriate probability formulations were engaged to create subregions with higher density functions.

For 2-D arbitrary simply connected domains, the challenge was finding a solution to freely pick the datapoints from any desired probability density functions, that was solved by using two different methods namely, the inverse method and the rejection method. Some examples were given to show the effectiveness of the SOM in mesh generation of the arbitrary geometry. The method was also examined for domains with higher mesh density regions as well as the domains with one or more holes inside.

Future investigations can be performed to stabilize the iterative calculations in order to have a better mesh distribution, especially for the sub-domains with varying mesh density. 3-D implementation of SOM for mesh generation can also be a hot topic for future investigations.

7. References

[1] Bern, Marshall W., and Paul E. Plassmann. "Mesh Generation." *Handbook of computational geometry* 38 (2000).

[2] Kohonen, Teuvo. "Exploration of very large databases by self-organizing maps." Proceedings of international conference on neural networks (icnn'97). Vol. 1. IEEE, 1997.

[3] H. K. Kwan, Intelligent Computing: A Computing Approach to Artificial Intelligence, Edition 1.5, Publisher: dfisp.org, eBook, 721 pages, 4 February 2020.

[4] Ross, Sheldon M. Introduction to probability models. Academic press, 2014.