Triangulation on a square domain : A new Look

Md Galib Hassan
Department of Physics, University of Cologne, Cologne, Germany.

November 1, 2017

Abstract

A new approach of triangulation of a surface of the form $z = f(x, y)$ defined on a square-domain (an $\mathbb{R}^2$ region) is introduced.

1 Introduction

Before we begin the algorithm, we want to define some useful quantities. Please note that these definitions may differ from the usual literature.

1.1 Triangulation

Let $\{v_1, v_2, \ldots, v_n\}$ is a set of vertices. We choose three vertices v_i, v_{i+1}, v_{i+2}, connect them, and assign a face on the triangle created by them. We repeat this process of making triangles such that no two sides of any two triangles intersect each other (see figure 1). We define this process as *triangulation* and the set of connected faces of these triangles as a *mesh*.

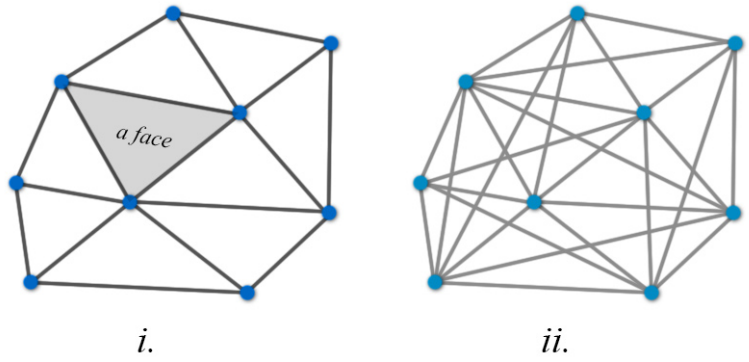


Figure 1: *i.* Triangulation, *ii.* not triangulation.

1.2 Orientation

We can connect any three vertices in either clockwise or in counterclockwise orientation. It's important to choose the clockwise orientation at this point, since we want to demonstrate the mesh generation in Unity3D [1].

2 Triangulation of square grid

Here, present the step-by-step triangulation of a square grid which starts from the origin $(0, 0)$ on a two dimensional plane and stretched up to infinity in both x and y-axis.

2.1 Marking the square grid

We assign a natural number, $i \in N$ to each vertex of a square grid as shown in figure (2) .

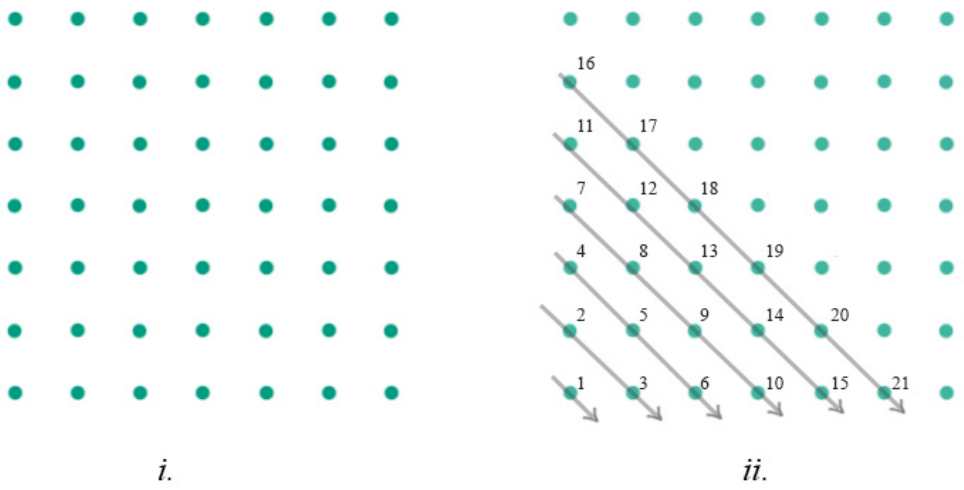


Figure 2: *i.* A square grid, *ii.* Marking process.

2.2 Sequences

We observe two important sequence emerging from the pattern of marking the grid. Starting from the origin, i.e. the vertex marked as 1, if we move along the x axis, we meet the vertices marked as:

$$S_h = 1, 3, 6, 10, 15, 21, \ldots \tag{1}$$

And if we move along the y axis, we meet the vertices marked as:

$$S_v = 1, 2, 4, 7, 11, 16, \ldots \tag{2}$$

The horizontal sequence S_h is particularly important, since, as we will see, S_v can be generated from S_h.

2.2.1 The horizontal sequence, S_h

Let us try to get a formula for S_h in (1). To do this, let us write the elements of S_h in terms of u_n where $n \in \mathbb{N}$.

$$\begin{array}{lll} n = 1, & u_n = n + 0 & = 1 \\ n = 2, & u_n = n + 1 & = 3 \\ n = 3, & u_n = n + 3 & = 6 \\ n = 4, & u_n = n + 6 & = 10 \\ n = 5, & u_n = n + 10 & = 15 \\ n = 6, & u_n = n + 15 & = 21 \\ \vdots & \vdots & \vdots \end{array} \tag{3}$$

Looking at the middle column of (3), we find that the residuals after n also form the same sequence S_h. So the formula for generating S_h is,

$$u_n = n + u_{n-1}, \qquad u_{(n<1)} = 0 \qquad n \in \mathbb{N} \tag{4}$$

2.2.2 The vertical sequence, S_v

Similarly we decompose S_v as previous:

$$\begin{array}{lll} n=1, & u_n = n+0 & = 1 \\ n=2, & u_n = n+0 & = 2 \\ n=3, & u_n = n+1 & = 4 \\ n=4, & u_n = n+3 & = 7 \\ n=5, & u_n = n+6 & = 11 \\ n=6, & u_n = n+10 & = 16 \\ \vdots & \vdots & \vdots \end{array} \tag{5}$$

We observe for S_v that the residuals for u_1 and u_2 are zero, and afterwards they form S_h. So, denoting the the terms of S_h as $u_n^{(S_h)}$ and terms of S_v as $u_n^{(S_v)}$, we have:

$$u_n^{(S_v)} = n + u_{n-2}^{(S_h)} \tag{6}$$

3 Triplet table

We can now generate a *triplet table* by the help of the elements of S_h and S_v. An extra condition is, in every iteration, the number of triplets will equal to n.

We define the three entries of the ith triplet of nth iteration as follows:

$$\begin{aligned} p_{n,1}^i &= u_{n-2}^{(S_h)} + n + (i-1), \\ p_{n,2}^i &= u_n^{(S_v)} + n + (i-1), \\ p_{n,3}^i &= u_n^{(S_v)} + n + (i-1) + 1, \end{aligned} \tag{7}$$

The triplet table for $n = 1$ to $n = 4$ is shown below:

iteration, n	Triplets $(p_{n,1}^i, p_{n,2}^i, p_{n,3}^i)$
1	$(1, 2, 3)$
2	$(2, 4, 5), (3, 5, 6)$
3	$(4, 7, 8), (5, 8, 9), (6, 9, 10)$
4	$(7, 11, 12), (8, 12, 13), (9, 13, 14), (10, 14, 15)$

Table 1: Triplet table for $n = 1$ to $n = 4$

4 Triangles

To make a triangle, we need three vertices. We also mentioned that we will connect the vertices only in clockwise orientation. This gives us the concept of *forward triangles* and *backward triangles*.

Consider four vertices a, b, c, d as in figure (3.i). We can choose a first, and then clockwisely complete a triangle by connecting a, b and c (shown in blue). If we take the triplet table and apply this process to its vertices, we completed triangles shown shaded in figure (3.ii). However, it is clear from that there are triangles, just like the red one in figure (3.i) which are not completed in this process. Notice also that, in case of the red triangle, we must go in the *backward* direction to maintain the clockwise orientation. For example, we can choose b first and then complete the triangle by connecting b, d, c.

4.1 Forward triangles

The triangles created by the vertices listed in the triplet table presented in section (3) are what we want to call as forward triangles.

4.2 Backward triangles

All the *left-over* triangles that are not listed in the triplet table are the backward triangles. Notice that, every triplet gives rise to a *backward triplet*.

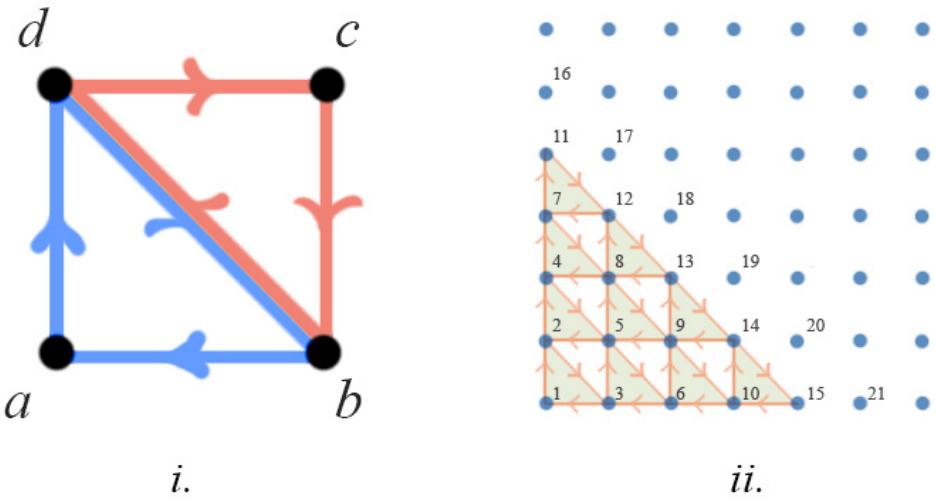


Figure 3: *i.* Forward and backward triangles, *ii.* Forward triangles (shaded).

5 Formulating backward triangles

From figure (3.ii), we list the backward triangles for each forward triplets:

iteration, n	Triplets $(p^i_{n,1}, p^i_{n,2}, p^i_{n,3})$	backward triplets
1	$(1,2,3)$	(3,2,5)
2	$(2,4,5), (3,5,6)$	(5,4,8), (6,5,9)
3	$(4,7,8), (5,8,9), (6,9,10)$	(8,7,12), (9,8,13), (10,9,14)
4	$(7,11,12), (8,12,13), (9,13,14), (10,14,15)$	(12,11,17), (13,12,18), (14,13,19), (15,14,20)

Table 2: Triplet table for $n = 1$ to $n = 4$

It is clear that the last vertex of a forward triplet is the first vertex of the corresponding backward triplet. And the middle vertex is unchanged. Notice also that the second backward triplet can be obtained by adding 1 to every vertex of the first backward triplet. The third can also be obtained applying the same process on the second, exactly as we have seen before in section: Triplet Table. So, the ith backward triplet can be found by adding $(i-1)$ to the vertices of the $(i-1)$th backward triplet (as expected). So it is sufficient to focus only on the first set of triplets $(i = 1)$ for every row n, and we have:

$$\begin{aligned} q^i_{n,1} &= u^{(S_v)}_n + n + (i-1) + 1, \\ q^i_{n,2} &= u^{(S_v)}_n + n + (i-1), \end{aligned} \tag{8}$$

Observe that the last vertex of a backward triplet in one row (n) is also the last vertex of the forward triplet of the next row $(n+1)$. So,

$$\begin{aligned} q^i_{n,3} &= p^i_{(n+1),3} \\ &= u^{(S_v)}_{(n+1)} + n + 1 + (i-1) + 1, \\ \therefore q^i_{n,3} &= u^{(S_v)}_{(n+1)} + n + i + 1 \end{aligned} \tag{9}$$

In the following section, we will express all our obtained results in terms $u^{S_h}_n$ and subsequently omit the (somewhat annoying) superscript S_h.

6 Triangulation summary

If $p^i_{n,j}$ with $j = 1, 2, 3$ be the vertices of a forward triangle, and $q^i_{n,j}$ be the vertices of the corresponding backward triangle on a square grid, then

$p^i_{n,1} = u_{n-2} + n + i - 1$	$q^i_{n,1} = u_{n-2} + 2n + i$
$p^i_{n,2} = u_{n-2} + 2n + i - 1$	$q^i_{n,2} = u_{n-2} + 2n + i - 1$
$p^i_{n,3} = u_{n-2} + 2n + i$	$q^i_{n,3} = u_{n-1} + 2n + i + 2$

Table 3: Triangulation summary

Here $i = 1$ to n, $n = 1$ to ∞ and u_n is the nth term in the sequence S_h, i.e,

$$u_n = n + u_{n-1}, \qquad u_{(n<1)} = 0, \qquad n \in \mathbb{N} \tag{10}$$

7 Triangulation of a surface on square domain

Let us now consider a function of the form $z = f(x, y)$ defined on a square two dimensional domain,

$$D = \{(x, y) \quad : \quad x_0 < x < \infty, y_0 \leq y < \infty\} \tag{11}$$

Notice that, if we discretize the domain D and want to tag the grid-points exactly in the same manner shown in figure (2), we can then iteratively generate the grid as follows:

$$D' = \{(x, y) : x = x_0 + i\ dx, \quad y = y_0 + (n - i)\ dy\}, \qquad 0 \leq i < n, \quad 0 \leq n < \infty \tag{12}$$

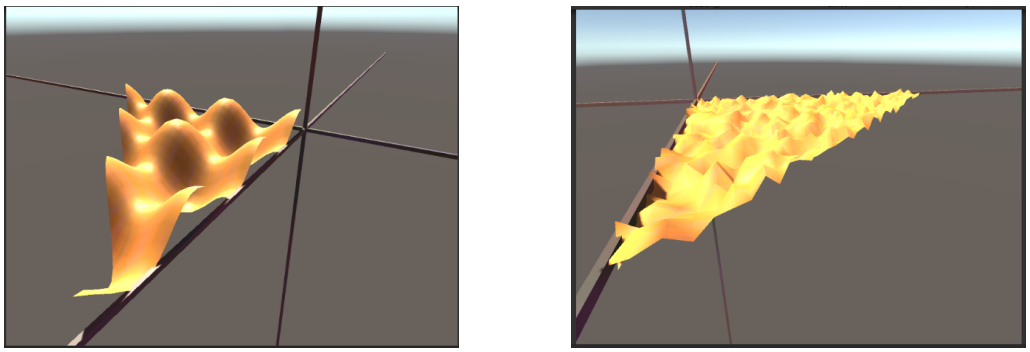
where dx and dy are sufficiently small line elements in x and y direction respectively. So the function-values of f on D' are,

$$f(x_0 + i\ dx, y_0 + (n - i)\ dy) \tag{13}$$

This values can be readily triangulated by tagging them in the order described in table: Triangulation Summary and connecting the vertices.

8 Examples

Any computer algebra system with 3D rendering facility can show the generated surface using our triangulation technique. Here we used Unity3D's `Mesh` class and `MeshFilter` component [2] to fill the triangles.



(a) Triangulated surface of $z = \sin x + \cos y$

(b) Triangulated mesh of $z = Random(-1, 1)$

Figure 4: Triangulation examples generated in Unity3D.

9 Comments

The algorithm presented here did not cover a full square grid in finite case. In equation (11), if we consider $x_0 \leq x \leq x_f$ and $y_0 \leq y \leq y_f$ for finite x_f and y_f, then only the lower triangular portion of the domain is covered by this technique. However, the upper triangular portion can be generated by applying the same algorithm starting from (x_f, y_f) and rotating the coordinate system by 180^o about z-axis.

References

[1] https://unity3d.com/

[2] https://docs.unity3d.com/ScriptReference/Mesh.html