An Algorithm for Complex Surfaces Modelling Using MatLab

Manuel Pérez Vázquez [(a)], *José Vázquez Paz* [(a)]

[(a)] Universidade de Vigo. Dp. Deseño na Enxeñaría. Escola de Enxeñaría Industrial. Campus de As Lagoas, 36310 Vigo.

Article Information

Keywords:
Complex surfaces,
modelling,
interpolation,
triangulation,
MatLab.

Abstract

The aim of this paper is to provide a method to model a complex surface from some determinate control points using the Lagrange´s algorithm to interpolate them and exploiting the possibilities of MatLab to develop a useful tool.

A complex 3D surface, in a wide range of cases, can be controlled by some "main" lines (guidelines or directives). We can also assume that every line is decomposable in a finite succession of points. Conversely said, giving a number of points we can select some of them to interpolate and create a line, and so on, generating a family of guidelines each with different number of points, all of them located on the surface. The second family of lines, which determines the surface, can be created by interpolating points from the lines of the first family, thereby generating a warped quadrilateral mesh. Afterwards we proceed to the triangulation of the formed quadrilaterals. We can control both the number of lines and points defining every line.

Finally it is possible to determine the area of a complex surface by summing the areas of all the triangles.The calculation of this area may be useful in several applications such as the volume of needed material to create a certain thickness of surface, etc.

1 Introduction

This paper is in line with previous ones in which we discussed about the modeling and control of lines, approaching or interpolating points, generating surfaces from the motion of straight lines or curve ones, revolution surfaces, etc., with different examples and applications [1], [2], [3], [4]. Now we intend to provide an overview of how to exploit MatLab as a tool to create and control surfaces obtained from some determinate points using the Lagrange´s algorithm to interpolate them, even though unexpected waviness could occur in some complex cases.

In a wide range of cases, we can generate surfaces from some lines on which they are supported (guidelines), in others dragging the generatrix over the guideline. Here we use the first method. Initially, we are going to determine the "main" lines or guidelines on which we will support the surface. In order to do this we need the control points. We can assume that every line is decomposable in a finite succession of points and because of that, we can create various guidelines with some points of the surface. The number of points to create a guideline can be different from one guideline to another as is evidenced in part 2.

Initially we can select the number of points of each line. Once we have got all the lines that we want for this family, we can obtain a second family interpolating points of each guideline (for instance, the first point of each respectively guideline, the end points, etc.). After that, we can proceed to automatically create a quadrilateral mesh with points of the lines introduced earlier. Then, we can make the proper triangulation.

When we have got a primitive (or initial) interpolation we can do it more accurate by adding more guidelines and more points to each guideline, but without deforming the initial shape of the surface. Finally we triangulate all the quadrilaterals and it is possible to obtain the total area of the surface by adding all the triangles' areas.

The developed tool is useful both for teaching and working environments. We can create our own surfaces quickly with it.

2 Description of the procedure

In this part we are going to describe the variables, functions and tasks followed to develop the procedure in the MatLab language [5[, [6], [7], [8].

2.1 Main program

The main program is developed in a function called *"patch"*, which is going to be invocated from the MatLab's command window every time we want to start the program.

The first request this tool makes, is to call the function *"Data introduction"*, which is the data introduction function as it is own name indicates.

Then, once you have introduced all the data (number of lines and points per line), a menu with the next main tasks appears (fig. 1):

1) Introduce new data
2) Four types of interpolation
3) Claim for help
4) Exit the tool

Now we briefly describe each of these options we can choose.

2.1.1 Data introduction

The first time you start the tool, it asks you the number of guidelines in one direction (or family) that you want to introduce. After that, you should introduce the number of points that are going to form every line. At this moment, you must keep in mind that you can introduce a different number of points for each line and you can introduce one

point only for the last line, to make the interpolation of a non-quadrilateral grid.

To execute this task, the tool calls the function already mentioned *"Data introduction"*.

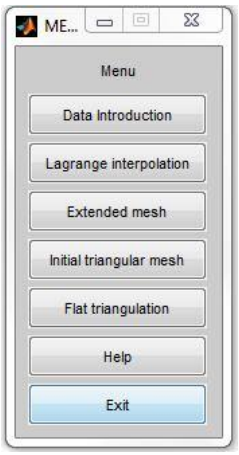


Fig. 1 Menu options

2.1.2 Lagrange interpolation

This function shows the user the primitive interpolation made by the method of Lagrange with the points previously introduced to the tool [1], [3]. This initial mesh is composed of warped quadrilaterals.

To execute this task the tool calls a function named *"Primitive interpolation"*, as it will be explained later (see item 2.2 Main functions)

2.1.3 Extended mesh

Once the user has got a primitive interpolation, he/she can discompose it in more points to obtain more detailed curves. At this time the user should know that he/she has not got limitation in the number of curved segments that he/she can obtain from his/her primitive surface definition.

To execute this task the tool should call a function named *"Extended mesh"*, and it is good to know that this task includes a mistake control in order to prevent user from executing it until he/she executes *"Data introduction"*.

In a first step, the lines of the same family are divided in "n" equal parts (e.g. green lines in fig. 2) and then connecting all the points we can create the other family of lines (e.g. more blue lines in fig. 2). In the same way, we can divide the lines of the second family (e.g. blue lines in fig. 2, in this example they are the initial guidelines) in another number of points introduced by the user. Connecting all of them we can obtain more lines of the other family (e.g. more green lines in fig. 2).

Besides, if we add a point to a line the other lines of that family are modified, changing the number of divisions between its first and last point.

2.1.4 Initial triangular mesh

This function of the tool has not got a real use in the world of work, but it could be interesting in the teaching environment, because what this task does is to use the points, which have been calculated in *"Extended mesh"*, to connect ones from the down guidelines to the upper ones, this connection is always made in the shortest distance to the upper point. Thus, we get an initial triangulation where each triangle has one straight side (the grey in fig. 2) and two curved sides (blue and green in fig. 2, each on a line of each family).

To execute this task the tool calls a function named *"Initial triangular mesh"*, and it should also be taken into account that this task includes a mistake control in order that the user does not execute it until he/she executes *"Extend mesh"*.

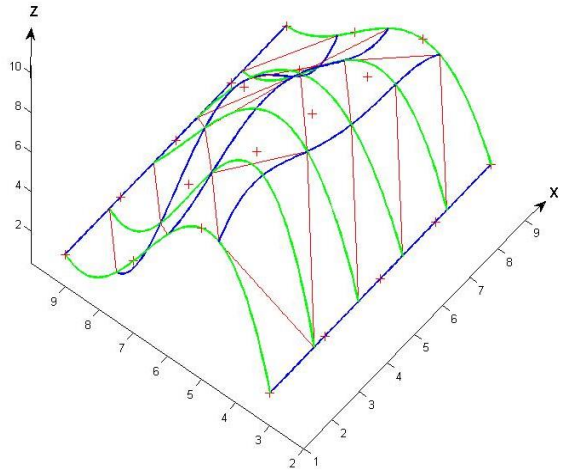


Fig. 2 Initial triangulation (warped)

2.1.5 Flat triangulation

The flat triangulation, which is the final task of the tool, outputs the triangular interpolation and the area of all triangles. This interpolation uses the points, which have been calculated in the previous *"Extended mesh"*.

To create this triangle interpolation, the tool is programed in one way that it will always build the triangle connecting two points from one guideline with another point of the next guideline. This connection to the point from the upper guideline is always made in the shortest distance to the upper point.

It is trivial to see that the precision of the triangle interpolation is related with the number of points the user gives to the tool in *"Extended mesh"*. Of course, now the triangles are flat (fig. 3)

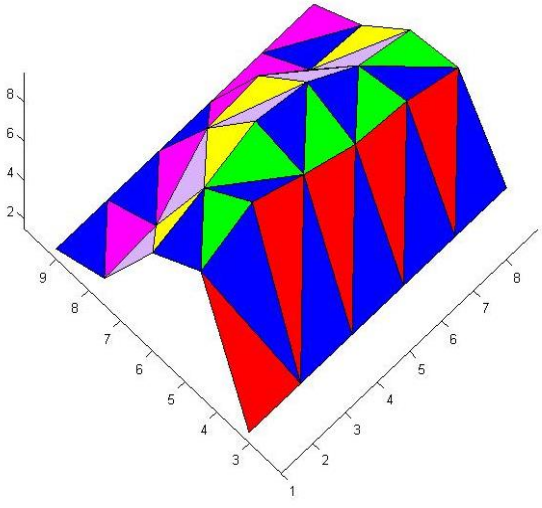


Fig. 3 Flat triangular mesh

To execute this task the tool calls a function named *"Flat triangulation"*, and it is good to know that this task includes a mistake control so that the user does not execute it until he/she execute *"Extended mesh"*.

2.2 Main functions

Now we are going to describe briefly all the functions created in the MatLab language to perform the main tasks.

2.2.1 Data introduction

Firstly this function asks the user the number of guidelines that he/she needs to define the surface and then save it in a variable called *"n"*. Then it shows a warning message that indicates you can only introduce one point in the last guideline, a fact that allows you to do interpolations to non-quadrilateral surfaces.

After that, you will be asked about the X, Y and Z coordinates of the points that are going to control the *"n"* lines, then all these points are saved in a matrix.

Then, the tool detects if the number of points of all the lines are different (without taking into account the last guideline in case the number of points is one), if the answer is yes, the tool uses a type of Lagrange's algorithm, which is implemented in a function called *"Add points"* to put more points so that all lines have got the same number of points.

The way you have to introduce all the data is shown in part 3, Examples.

2.2.2 Lagrange interpolation

This function takes the points previously introduced in *"Data introduction"* of one guideline and sends them to a function call *"Main interpolation"*, this happens with all the guidelines introduced. By doing this we obtain interpolations in the direction of the guidelines.

Then we execute the same function with all the points that are in the same position of all guidelines, in order to create additional guidelines in other direction, and as it happens with the first guidelines the process is repeated a number of times equal to the number of points. When this process is finished we obtain a poor quality quadrilateral mesh that can be improved. Moreover, when there is only a point in the last guideline all interpolations in the second direction are made to this point.

The output of this function is shown in fig. 5 and fig. 9, and as you can see, the first interpolation lines are coloured in blue and the second ones in red.

2.2.3 Extended mesh

This function asks the user the number of guidelines and the number of points for each that he/she wants in order to do a more accurate interpolation, but without losing the initial shape of the surface.

Then with this information and the points introduced earlier in *"Data introduction"* we call the function *"Add points"* twice, because in the first call we divided a family of lines (the red lines in fig.5 and fig. 9) to obtain the number of guidelines demanded by the user (blue lines in the same figures), and later in the second call we put the same number of points in all guidelines, which are the demanded by the user.

Finally we use the function *"Main interpolation"* like the previous described function, so we connected all the points in the direction of the guidelines and we also connect points of different guidelines that are in the same position to create a quadrilateral mesh. However when there is only a point in the last guideline all interpolations in the second direction are made to this point.

The outputs of this function are displayed in fig. 6 and fig. 10.

2.2.4 Initial triangular mesh

This function is going to plot its triangulation over the output of the *"Extended mesh"*, because it has got no sense to do this type of interpolation in a poor quality mesh, so it needs the points calculated in the *"Extended mesh"*.

To do the interpolation, first it chooses the points of one guideline and the next upper one, and use them in *"Diagonal drawing"*, which will generate the output, this operation is repeated the number of guidelines less one.

As you can see in fig. 2, fig. 7 and fig. 11, we mix curve lines from *"Extended mesh"* with straight lines from this *"Initial triangular mesh"*, this is why it cannot be applied to the world of work.

2.2.5 Flat triangulation

This function is going to produce a triangle interpolation and like the last one it uses the points from *"Extended mesh"*. Triangulation is usually carried in two steps: first obtaining a proper triangulation grid, and then improving it [10], [11], [12].

This function chooses the points of one guideline and the next upper one, to use them in *"Triangle"* and generate the output, this operation is repeated the number of guidelines less one.

Moreover apart from the plot generated by *"Triangle"*, this function also can calculate the area of all triangles, when the interpolation is finished.

The outputs of this function are displayed in fig. 3, fig. 8 and fig. 12.

2.2.6 Main interpolation

This function receives three coordinate vectors and uses a symbolic variable called *"t"* to do the interpolation (as it was mentioned before this tool is for both educational and work use, if it were necessary to make it only for work use, where time is crucial we have developed a faster algorithm, which does not use symbolic variables).

The algorithm used to make interpolations, is given by the parametric equations of the Lagrange's algorithm shown in eq.1, which is the same for the X, Y and Z coordinates (just changing the corresponding coordinate) [3], [4], [9].

$$x(t) = \sum_{i=0}^{m-1} P_X(i+1) \cdot \prod_{i=0, i\neq j}^{m-1} \frac{t - k_i}{k_j - k_i}$$
$$y(t) = \ldots \quad (1)$$
$$z(t) = \ldots$$

In eq. 1, the parameter *"t"* is a symbolic variable, *"$P_X(i+1)$"* is the value of X-coordinate to the point in the position i+1 (where $P_1, P_2, \ldots P_m$; are the points previously introduced for a determined guideline), *"m"* is the number of points of that guideline, *"m-1"* is the polynomial degree and finally *"k"* (values of the parameter in an uniform interval) is calculated according to eq. 2.

$$k = \frac{1}{m-1} \tag{2}$$

The products $(t-k_i)/(k_j-k_i)$ will give us the shape functions (or blending functions). There will be as many as the number of entered points.

According to this method every point is given by a vector whose coordinates are polynomial functions of "t". For each value of "t" there is a single point.

Finally, we can plot the equation given by the last algorithm, to obtain the outputs.

2.2.7 Add points

This function receives three coordinate vectors and a numeric variable (as stated in item 2.2.1).

Then, it uses the same algorithm described in *"Main interpolation"* (eq.1), but instead of plotting it once the algorithm is calculated, this function splits it in the number of points defined by the numeric variable. So as output we obtain a matrix of points that are going to be used by other functions.

2.2.8 Diagonal drawing

This function receives two matrixes (A and B) and a variable with the number of points of those matrixes.

The first iteration compares the diagonal lengths A(i)-B(i+1) with A(i+1)-B(i); if the first diagonal is shorter than the second one it plots it, otherwise it plots the second one (Fig. 4). Then the next iterations are the same but they control the diagonals A(i), B(i-1) and A(i-1), B(i) in order to plot the shortest one, this is repeated with all points of guidelines.

All the plots in this function are realized by the function "Plot line".

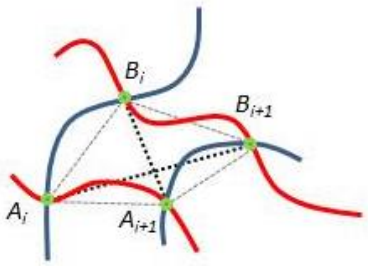


Fig 4. Diagonal selection

2.2.9 Triangle

This function is like *"Diagonal drawing"*, but instead of plotting the shortest diagonal, it uses this to mark the end points, which defined the diagonal and a point A(i+1) or A(i-1) (always from the down guideline) according to the direction of the diagonal, to plot triangles using "Plot triangle", this is repeated with all points of guidelines.

Furthermore, by means of this function we can calculate the area of each triangle to show it to the user later. Adding the areas of all triangles we get a reasonable approximation to the created surface.

2.2.10 Plot line

This function receives two vectors of points, one with the X, Y, Z coordinates of the origin point and another with the X, Y, Z coordinates of the final point, in order to plot a line using the parametric equations of a straight line.

2.2.11 Plot triangle

This function receives three vectors with all the coordinates of three points, then using MatLab's function "fill3" we obtain a triangle. The forth argument of the function "fill3" allows to assign a colour to the triangle.

3 Examples

In this part we are going to show the reader how our tool works for two different examples. In both cases we will introduce appropriate data.

3.1 Example 1

In this example we show how to model a surface with a certain number of guidelines and the different outputs provided. All lines interpolate the same number of points.

To create fig. 5 we should introduce to the program the following data:

- Number of lines: 4 (the blue lines in fig. 4)
- Number of points in each line: [5 5 5 5] (the same)
- Coordinates for points of every line.

Line 1:
X coordinates: [1 3 5 7 9]
Y coordinates: [9 9 9 9 9]
Z coordinates: [2 2 2 2 2]
Line 2:
X coordinates: [1 3 5 7 9]
Y coordinates: [7 7 7 7 7]
Z coordinates: [4 5 7 5 4]
Line 3:
X coordinates: [1 3 5 7 9]
Y coordinates: [5 5 5 5 5]
Z coordinates: [8 9 8 7 6]
Line 4:
X coordinates: [1 3 5 7 9]
Y coordinates: [3 3 3 3 3]
Z coordinates: [2 2 2 2 2]

Fig. 5 shows all points and the two line families obtained.

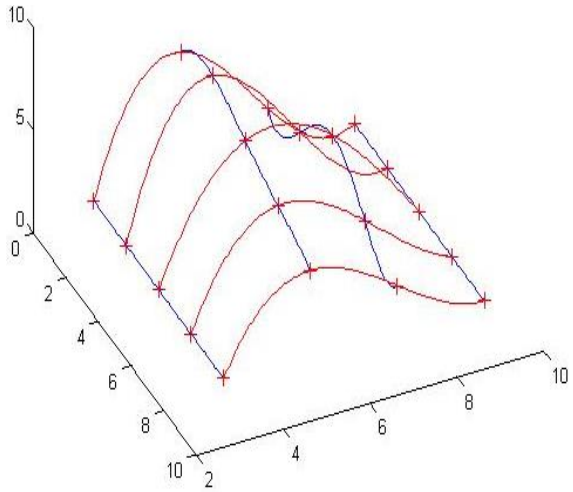


Fig. 5 Lagrange interpolation to 4 lines with 5 points.

To create more lines of every family (see fig. 6) we select *"Extended mesh"* in the menu and should introduce a greater number of lines and points in each case, like the following data:

- Number of lines: 50
- Number of points in each line: 50

The number of points and the number of lines are usually different (although in this case have introduced the same).

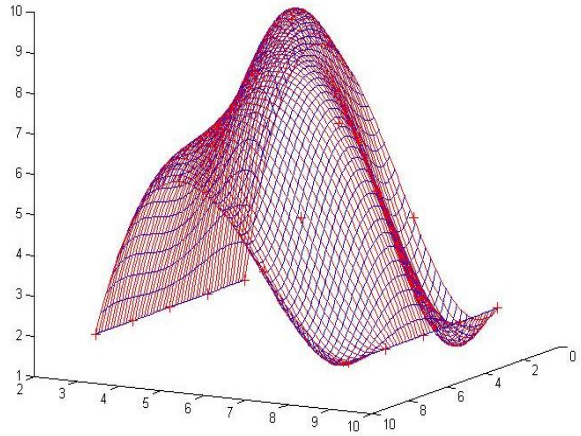


Fig. 6 Example of an extended mesh

Fig. 7 shows the initial triangulation where each triangle has two curved sides and one straight, corresponding with the selected diagonal. We can perceive this effect, because observed triangles in figure are not flat.

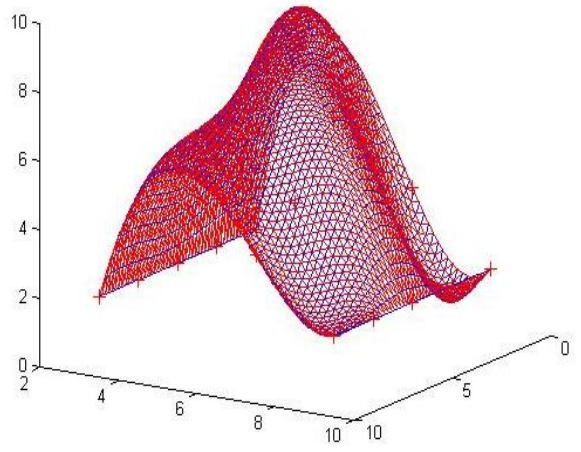


Fig. 7 Initial triangular mesh (warped)

The final result of this triangulation is showed in fig. 8, where triangles are flat and we can paint them or select some of them and assign a different colour like we have made in fig. 3.

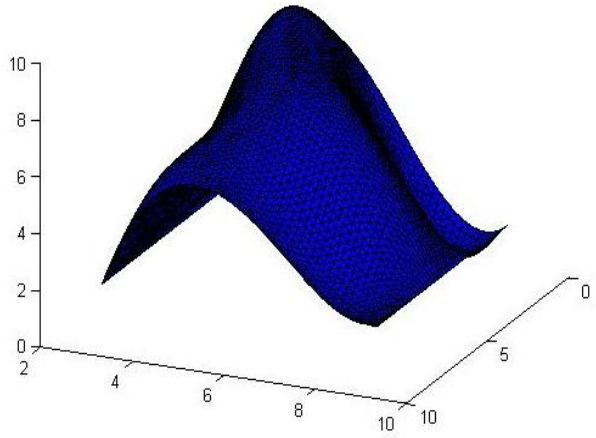


Fig. 8 Example of flat triangulation

3.2 Example 2

In this example we show how to model a surface from three lines, with only one point in the third guideline, four points in the intermediate and five in the first, as well as the different outputs it provides. Lines 1 and 2 are closed (initial and final points coincident).

To create example 2 we introduce to the program the following data:

- 3 lines
- Points in every line [5 4 1]
- Coordinates for:

Line 1
X coordinates: [3 5 3 2 3]
Y coordinates: [0.5 3 4 2 0.5]
Z coordinates: [1 1 1 1 1]
Line 2
X coordinates: [4 4 3 4]
Y coordinates: [2 3 2 2]
Z coordinates: [3 3 3 3]
Line 3
X coordinates: [3.5]
Y coordinates: [2.5]
Z coordinates: [5]

Fig. 9 shows these three guidelines with blue colour. Now the third guideline is a point, the second and third are closed with different number of points. The program adds a point to the second line, performing the appropriate spacing between the first and the last point. Then it plots red lines, all passing through the vertex (as the first and last coincide, we only can see four red lines).

Thus, we can prove that the algorithm draws as many red lines as the number of different points in line 1 (which has more points).

So we can represent a sort of tapered surface possessing at least one vertex and more than one guideline.

All of this gives us an initial idea of the possibilities of the created application. Modifying timely points we can get different kinds of surfaces.

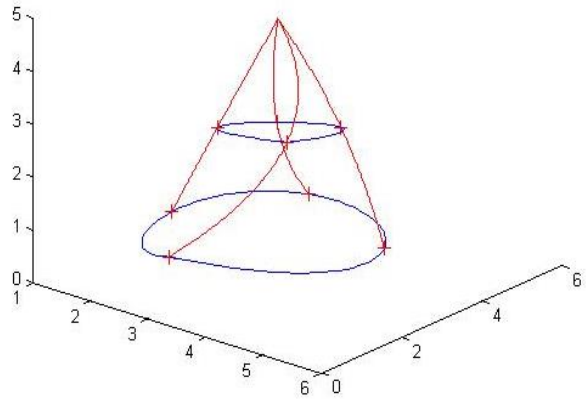


Fig. 9 Interpolation to lines with different number of points

To create fig. 10 we give to the tool the following data:

- 20 lines
- 20 points

We obtain a quadrilateral mesh enclosing a cone-like surface.

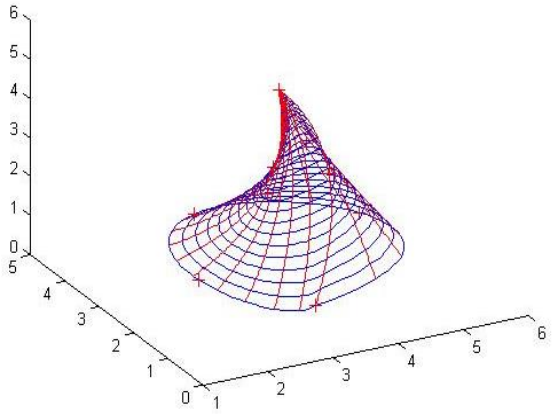
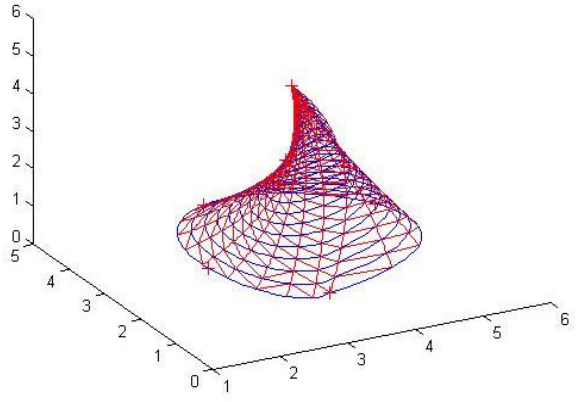


Fig. 10 Example of extended mesh

Fig. 11 shows the warped triangulation like fig. 7. Here we can see the straight diagonal (red colour) in each warped quadrilateral of the mesh.



Fi

Fig. 11 Initial triangular mesh (warped)

Finally, fig. 12 shows the flat triangulation, where triangles are flat with their vertex over the previous quadrilateral mesh.

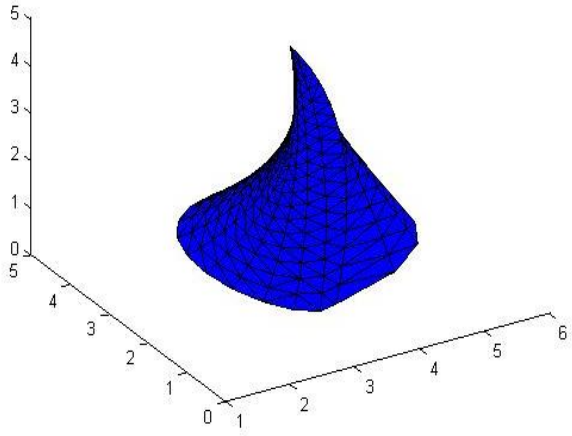


Fig. 12 Flat triangulation

4 Conclusions

Initially this tool was developed with two possible uses in mind: educational and work environment. These uses were possibly achieved because it has been developed in collaboration with students

As a result we have obtained a useful tool that allows us to represent desired surfaces quickly with high level of accuracy.

If we think of the educational use, this tool can help students to improve their space vision and they can learn how to model surfaces using families of guidelines, introducing them in the MatLab programming.

Once we have achieved the first interpolation (which is really the most difficult) they can improve the approximation level. Thus, they can create his own triangular meshes by interpolation from surfaces data (real or no, simples or complex) with pretty precision.

Managing MatLab language to solve graphics problems is a goal that should not lose sight, and its connection with Simulink.

On the other hand, if we think of the work environment, when the user obtains a surface and its area, and immediate application is to manage the surface through normal operations like: assigning a thickness, offseting, assign textures, colour, etc. For example, to calculate the amount of paint necessary to give a certain layer thickness, the fabric needed for a piece of clothing, the volume of metal, wood, etc., necessary to cover any surface in study.

In addition, it can be helpful to calculate the area of a tile or mosaic in order not to waste any material.

The calculus of the examples shown in part 3, even in the most accurate case, had not taken us more than a few seconds.

Acknowledgements

We are pleased to dedicate this paper to our colleagues of Engineering Graphics Area and to the students of the last Computer Aided Design course (2012-13), who have contributed at this work notoriously.

References

[1] M. Pérez. Aplicaciones prácticas en la docencia para la graficación de líneas generadas a partir de puntos. Actas XXI Congreso Ingegraf (Lugo-2009), p.86.

[2] J. D. Foley, A. Van Dam, S. K. Feiner, J. F. Hughes, R. L. Philips. *Introducción a la Graficación por Computadora*. Addison-Wesley Iberoamericana (1996), pp 371-404.

[3] G. Farin. *Curves and surfaces for computer aided geometric design.* Academic Press (1997).

[4] J. M. Cordero, J. Cortés. *Curvas y superficies para modelado geométrico*. Ed. Ra-Ma (2002) pp. 27-93.

[5] C. Pérez. *MatLab a través de ejemplos*. Ed. Garceta (2012)

[6] C. F. Van Loan, K.-Y. Daisy Fan. *Insight Through Computing: A MATLAB Introduction to Computational Science and Engineering.* SIAM, 2010.

[7] S. Lantarón, B. Llanas. *MatLab y matemática computacional*. Ed Bellisco, 2010.

[8] A. Gilat, V. Subraniam. Numerical Methods for Engineers and Scientists: and Introduction with Applications Using MatLab, 2e. John Wiley and sons, Inc. (2011).

[9] M. A. Kovačević. *A simple algorithm for the construction of Lagrange and Hermite interpolating polynomial*. Facta Universitatis: Ser. Math. Inform. Vol. 22, 2 (2007), pp 165-174.

[10] B.K. Choi, H.Y. Shin, Y.I. Yoon, J.W. Lee. *Triangulation of scattered data in 3D space*. Computer-Aided Design. Vol. 20, 5 (June 1988), pp 239-248.

[11] J.L. Brown. *Vertex based data dependent triangulations.* Computer Aided Geometric Design. Vol 8, 3 (August 1991), pp 239-251.

[12] M. Vigo, N. Pla, P. Brunet. *Directional adaptive surface triangulation*. Computer Aided Geometric Design. Vol. 16, 2 (February 1999), pp 107-126.