Universitat Politècnica de Catalunya

Computational Mechanical Tools

Master's Degree in Numerical Methods in Engineering

Measuring Element Quality (Second Assignment)

Author:
Pau Márquez

Supervisor:
Prof. J. Sarrate

Academic Year 2019-2020

Contents

Abstract

The purpose of this text is to study the importance of the quality measures on the grid generation of simple bi-dimensional geometries with triangular and quadrilateral elements. An intermediate domain will be generated using an ideal element, different for each type of element, and so the quality of an element will be measured in terms of the deviation from this ideal element. An algebraic framework will be developed for both triangular and quadrilateral elements and shape quality measures will be extracted as well as radius-ratio measurements. To test the validity of the mentioned methodology, the code will be used to analyze a simple four-by-four squared mesh. Eventually, several meshes will be evaluated and its results plotted in an histogram. Conclusions will be extracted regarding the quality of the meshes.

1 Introduction

Here the main definitions concerning the following sections will be laid out.

The physical space is where the elements to be studied are defined. In this space the matrix A will be defined as

$$\underline{\underline{A}} = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix} \tag{1}$$

With matrix A, the Jacobian matrix S can be computed. This matrix is the one with which the shape quality of an element is calculated. To calculate S, the inverse of the matrix defining the mapping between the reference element and the ideal one is needed. The ideal element is simply the element that will give a quality of 1 when computed. For the case of triangular element, the ideal element is an equilateral triangle whose matrix W is given by

$$\underline{\underline{W}} = \begin{pmatrix} 1 & 1/2 \\ 0 & \sqrt{3}/2 \end{pmatrix} \tag{2}$$

As for the quadrilateral element, the ideal element is simply a square. However, the process is now more complex as it needs to be separated into the four triangles that may be defined with its four nodes. Consequently, the quality of the element will be derived as the mean of the qualities of the four triangles defining it. Now it is important to take into account that each of these triangles will have associated a different mapping to reach the physical space. In this way, each triangle will be given a quality of one. Fig. 1 shows the ideal quadrilateral element and the four nodes defining it. Then, in Fig. 2 the four triangles obtained from Fig. 1 are drawn in order and below them the four matrices W.

Then, the Jacobian matrix is simply computed as $\underline{\underline{S}} = \underline{\underline{AW}}^{-1}$.

2 Results

Before facing the given meshes, it is essential to validate the code. This will be done by computing the quality of a mesh which is known a priory, for instance a square composed of

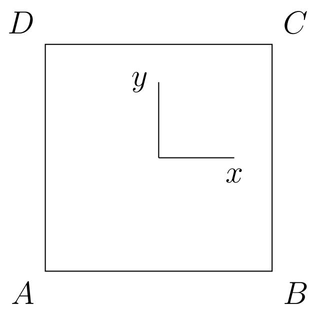
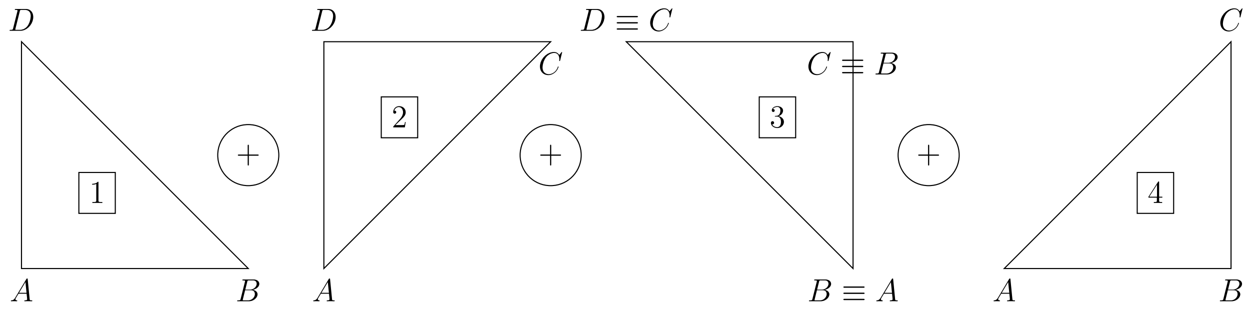


Figure 1: Arbitrary quadrilateral element defined by four nodes in the coordinates x, y.



$$\underline{\underline{W_1}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \underline{\underline{W_2}} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \qquad \underline{\underline{W_3}} = \begin{pmatrix} 0 & -1 \\ 1 & 1 \end{pmatrix} \qquad \underline{\underline{W_4}} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \tag{3}$$

Figure 2: Decomposition of a square element into four ideal rectangle triangles. Below each one, the mapping between the reference and the ideal element is given by matrix $\underline{\underline{W}}$.

four equal squares. The quality of each square should be equal to one. Fig. 3 shows the latter mesh. As a quality of one is obtained for each sub-domain, the code is validated in a first approach for both quadrilateral and triangular elements.

Then, the procedure follows by studying the simplest meshes for both quadrilateral and triangular elements, i.e. mesh files *tris-h1* and *quads-h1*. Fig. 4 shows the two meshes with all elements numbered. Then, Tables 1 and 2 show the quality for each numbered element. As is can be seen, sub-domain elements which are very similar in shape have almost equal quality values, although they have been rotated. This is another sign that the code is working properly.

Element Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Quality	0.615	0.328	0.835	0.759	0.753	0.330	0.836	0.613

Table 1: Quality of the triangular sub-domains.

To continue with, Table 3 shows the maximum, minimum and mean values for the shape quality measures of all the meshes, and Table 4 shows the radius-ratio measurements for the triangular

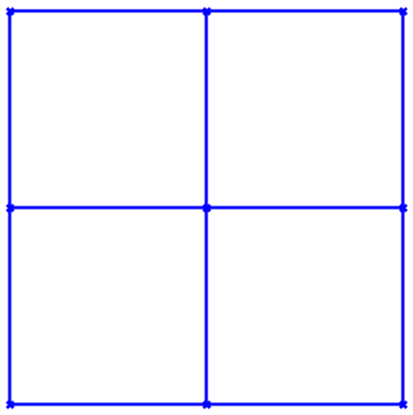


Figure 3: Square test mesh.

Element Number	**1**	**2**	**3**	**4**
Quality	0.912	0.858	0.659	0.858

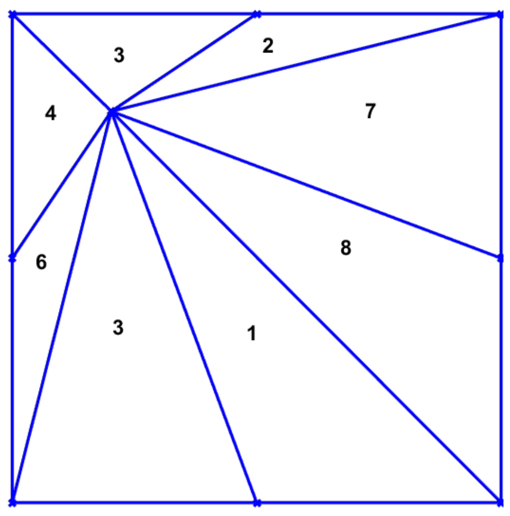
Table 2: Quality of the quadrilateral sub-domains.

meshes only. Eventually, Figures 5, 6,7,8,9 and 10 show the shape quality measurements plotted in a histogram. Then, Figures 11, 12 and 13 show the histogram plots for the radius-ratio measurements of the triangular meshes.

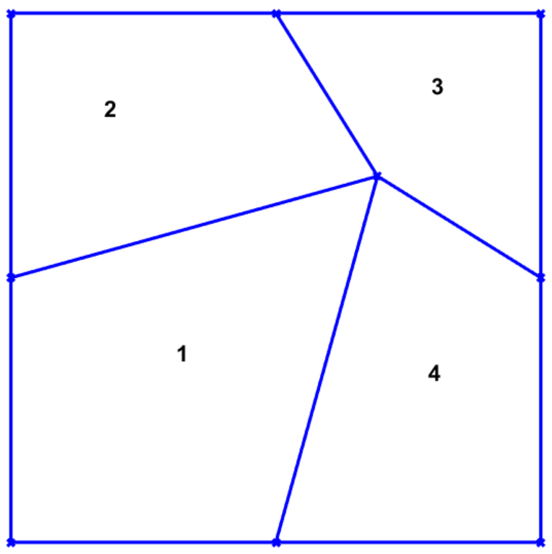
Mesh number	**Triangular**			**Quadrilateral**		
	Maximum	**Minimum**	**Mean**	**Maximum**	**Minimum**	**Mean**
1	0.836	0.328	0.634	0.911	0.659	0.822
2	0.999	0.178	0.752	0.992	0.105	0.835
3	0.998	0.258	0.790	0.990	0.095	0.791

Table 3: Shape quality measures for the given meshes, where the indications $1, 2, 3$ refer to *-h1*,*-h2* and *-h4*, respectively.

All measures and plots happen to fall between the range $[0, 1]$, which is a good indication. Moreover, the results seem legit and logical, and they expand throughout all the domain, reason to think that the code is able to detect both high and low quality cells. The pattern that can be seen in all meshes with high elements is a concentration of the cell number in the medium-to-high region, meaning that the meshes are of relatively high quality, although they are all far from being meshes composed of ideal elements. The radius-ratio histograms show, however, less good results in terms of the quality, as more cells are accumulated in the region of medium-quality cells.



(a) Triangular mesh.



(b) Quadrilateral mesh.

Figure 4: Basic test meshes.

Mesh number	**Maximum**	**Minimum**	**Mean**
1	0.835	0.161	0.555
2	0.999	0.048	0.701
3	0.998	0.102	0.746

Table 4: Radius-ratio quality measures for the triangular meshes.

3 Concluding remarks

The radius-ratio and shape quality measurements have been effectively computed for six different meshes. The importance of the mapping matrix has been clearly stated and test cases have been studied to validate the code in a first extent. Then, the histogram plots for the mentioned qualities have been shown to give proper results.

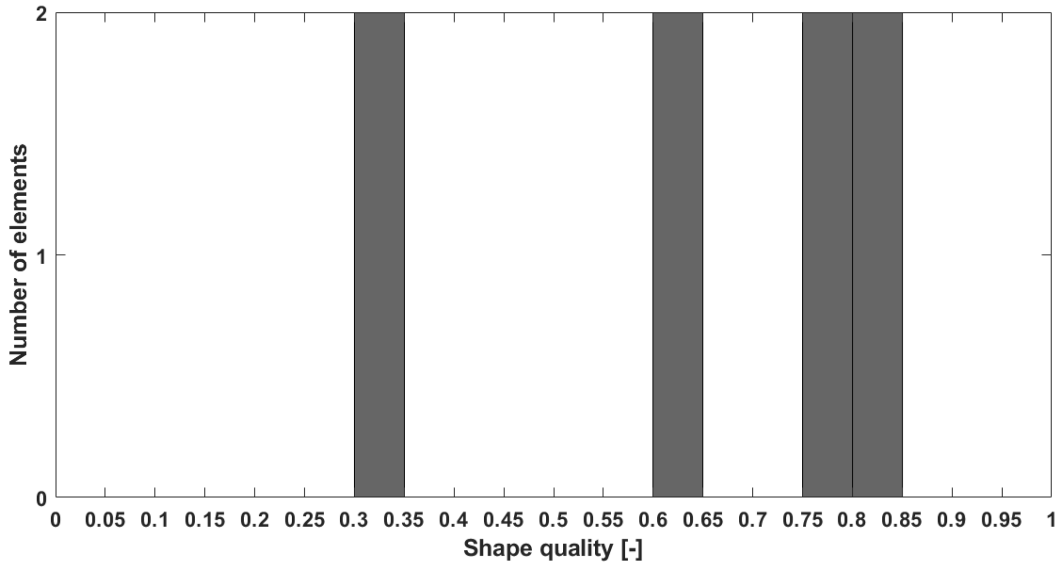


Figure 5: Histogram associated to the mesh file *tris-h1.txt*

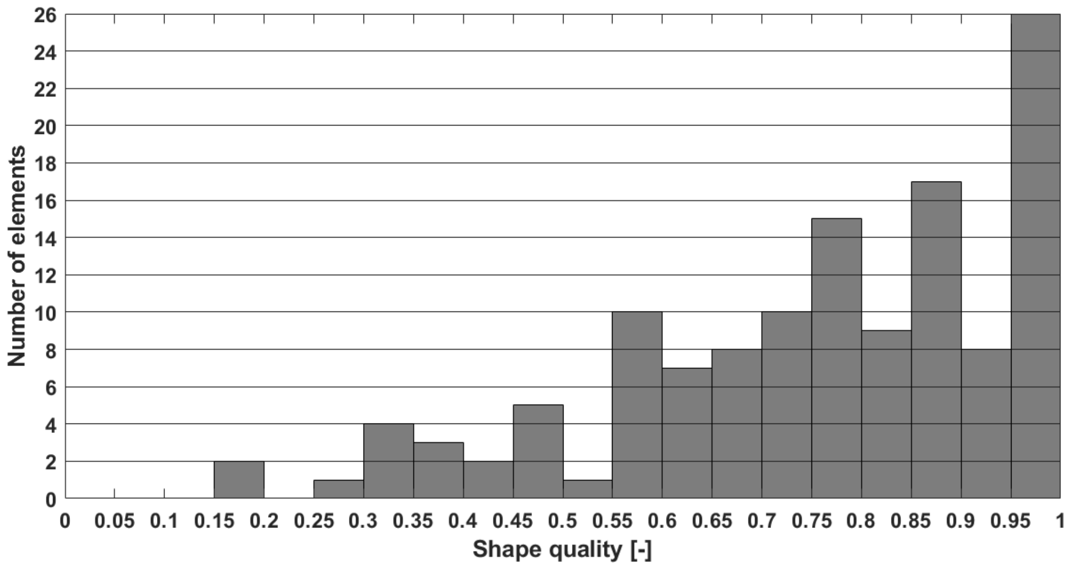


Figure 6: Histogram associated to the mesh file *tris-h3.txt*

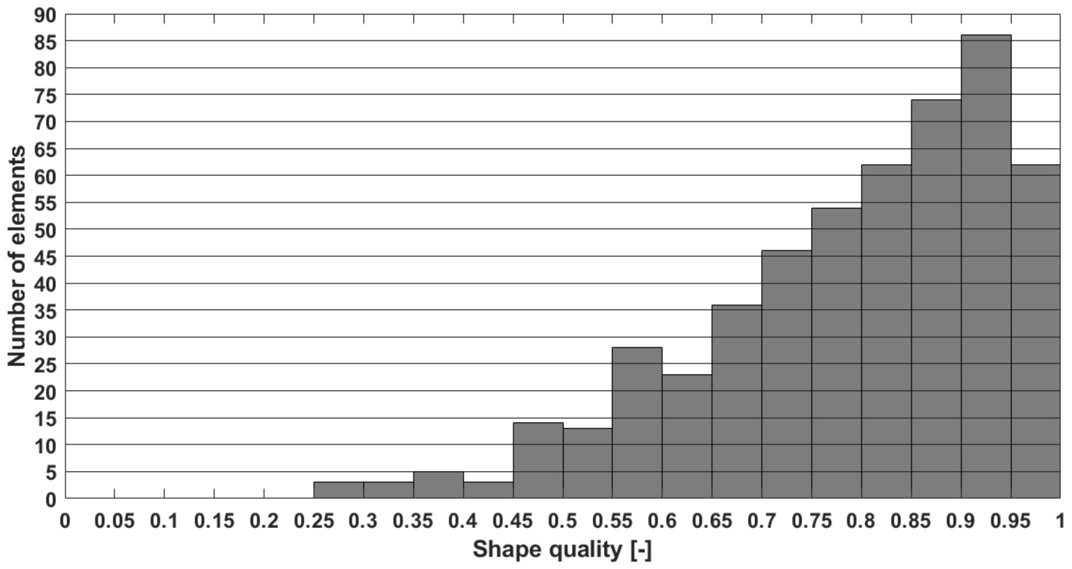


Figure 7: Histogram associated to the mesh file *tris-h4.txt*

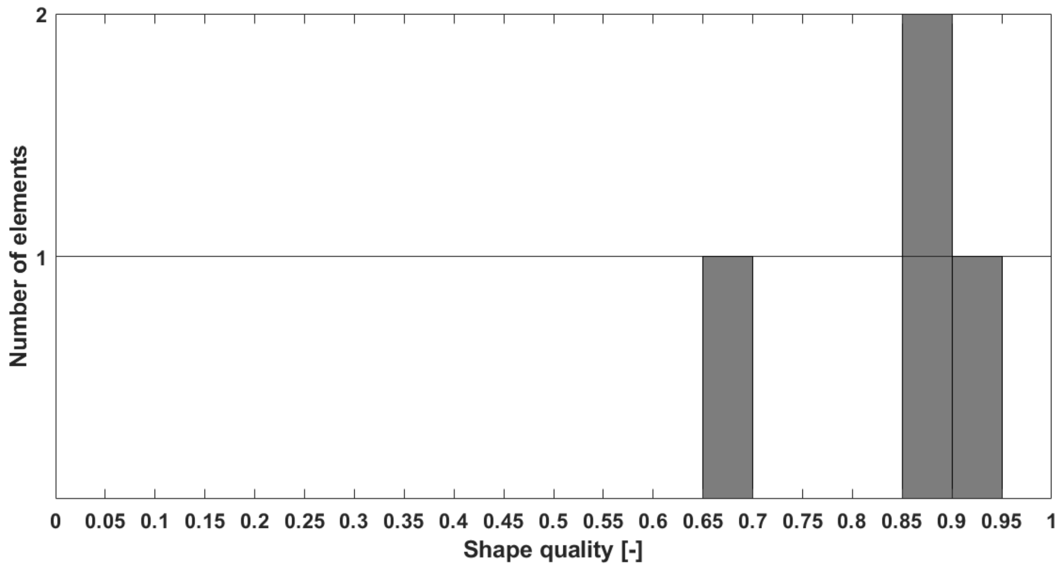


Figure 8: Histogram associated to the mesh file *quads-h1.txt*

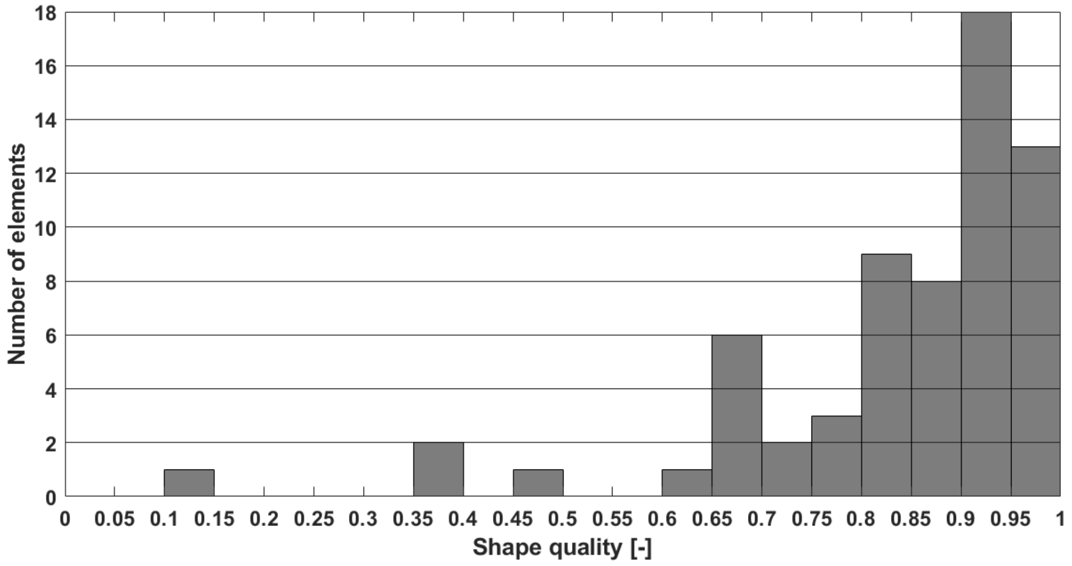


Figure 9: Histogram associated to the mesh file *quads-h3.txt*

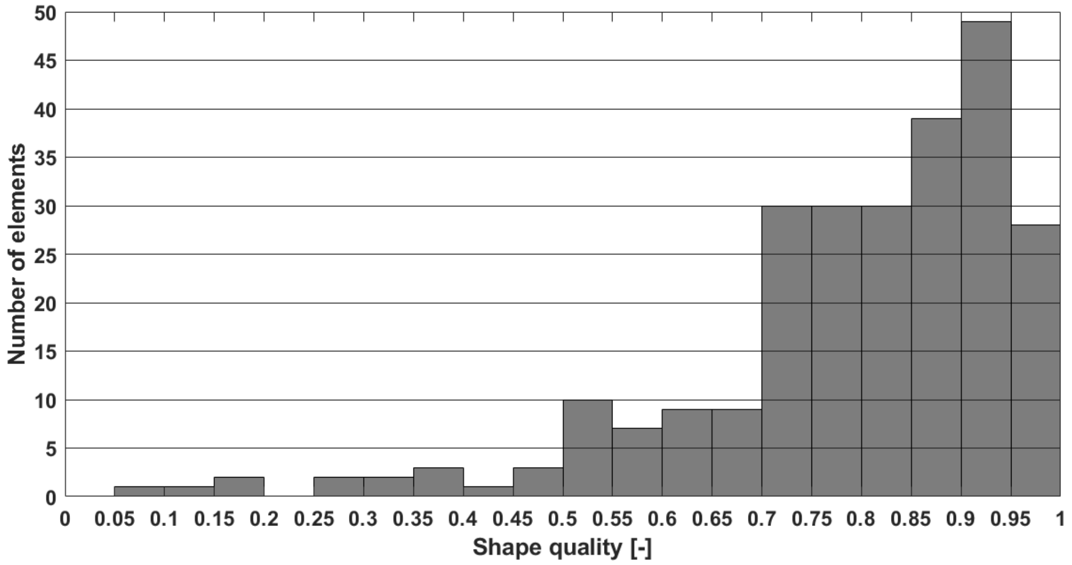


Figure 10: Histogram associated to the mesh file *quads-h4.txt*

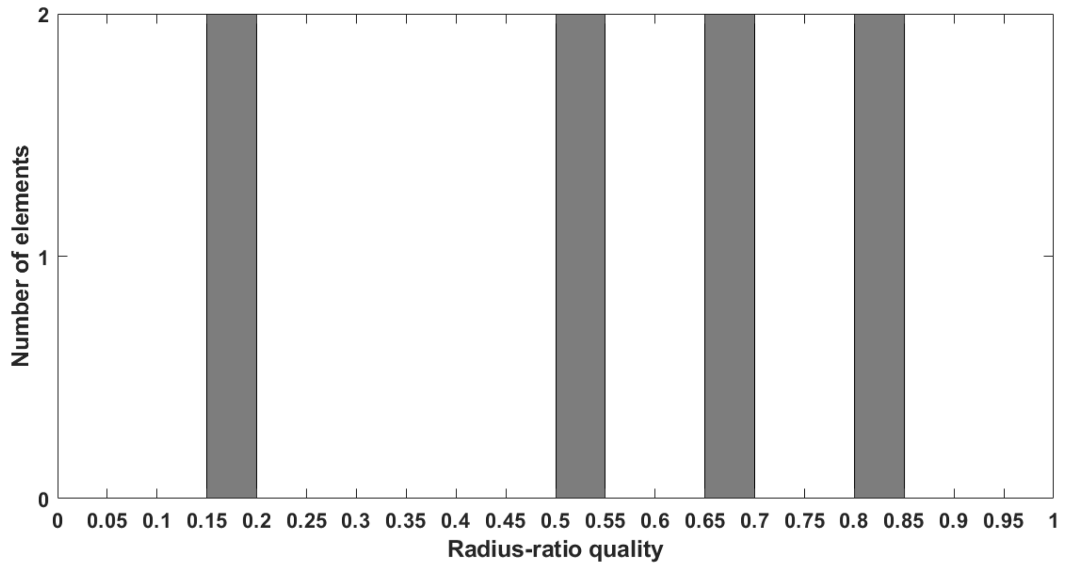


Figure 11: Histogram associated to the mesh file *tris-h1.txt*

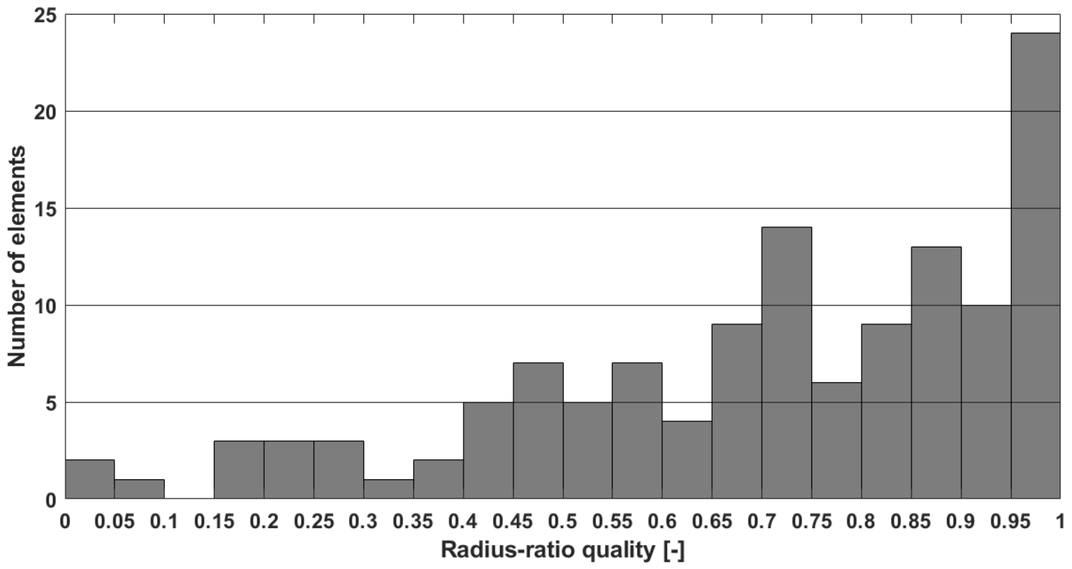


Figure 12: Histogram associated to the mesh file *tris-h3.txt*

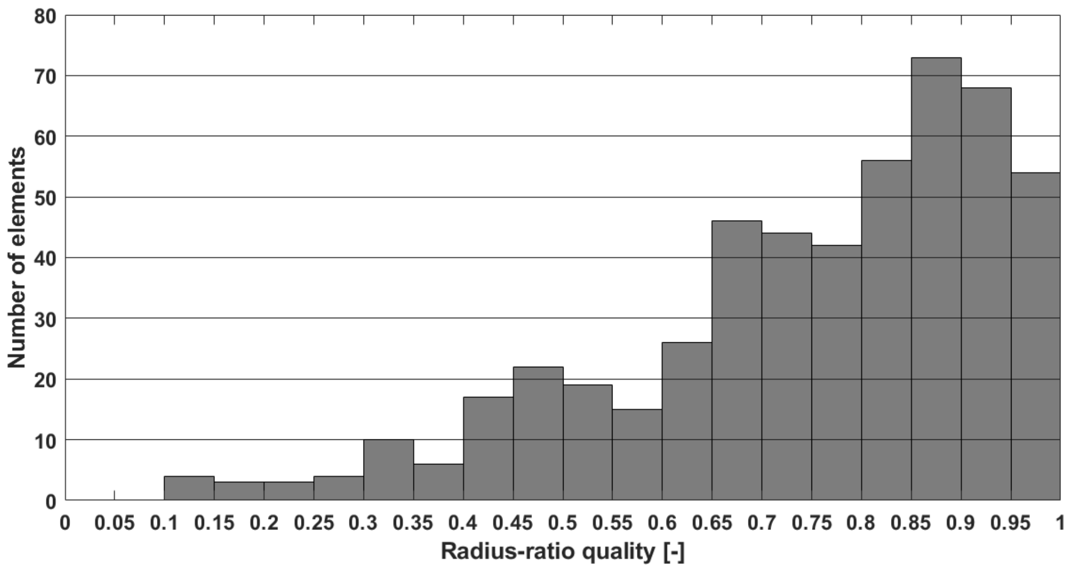


Figure 13: Histogram associated to the mesh file *tris-h4.txt*

A Appendix

A.1 Function *qualityShapeQuads*

```
function [q,minQ,maxQ,meanQ] = qualityShapeQuads(X,T)

num_elem = size(T,1);
p = 2;
n = 2;

q = zeros(1,num_elem);

    for i = 1:num_elem
        [S_l] = matrix_S_quad(X,T,i);
        C = inner_sum(S_l,n);
        B = C/4;
        A = power(B,1/p);
        q(i) = 1/A;
    end

minQ = min(q);
maxQ = max(q);
meanQ = mean(q);

end
```

A.2 Function *qualityShapeTris*

```
function [q,minQ,maxQ,meanQ] = qualityShapeTris(X,T)

    n = 2;
    num_elem = size(T,1);
    q = zeros(1,num_elem);

    for i = 1:num_elem
        [S] = matrix_S(X,T,i,n);
        q(i) = n*det(S)/(norm(S,'fro'))^2;
    end

    minQ = min(q);
    maxQ = max(q);
    meanQ = mean(q);
end
```

A.3 Function *qualityRadiusRatio*

```
function [q,minQ,maxQ,meanQ] = qualityRadiusRatio(X,T)

    alpha = 2;
    num_elem = size(T,1);
    q = zeros(1,num_elem);

    for i = 1:num_elem
        L1 = pdist([X(T(i,1),1) X(T(i,1),2); X(T(i,2),1) X(T(i,2),2)],'euclidean');
        L2 = pdist([X(T(i,1),1) X(T(i,1),2); X(T(i,3),1) X(T(i,3),2)],'euclidean');
        L3 = pdist([X(T(i,2),1) X(T(i,2),2); X(T(i,3),1) X(T(i,3),2)],'euclidean');
        B(1,1) = X(T(i,2),1) - X(T(i,1),1);
        B(1,2) = X(T(i,3),1) - X(T(i,1),1);
        B(2,1) = X(T(i,2),2) - X(T(i,1),2);
        B(2,2) = X(T(i,3),2) - X(T(i,1),2);
        A = 0.5*det(B);
        p = (L1 + L2 + L3)/2;
        r = A/p;
        R = (L1*L2*L3)/(4*A);
        q(i) = alpha*r/R;
    end

    minQ = min(q);
    maxQ = max(q);
    meanQ = mean(q);
end
```

A.4 Function to calculate matrix S for each quadrilateral element

```matlab
function [S] = matrix_S_quad(X,T,k)
%This function calculates matrix S for each quadrilateral element i
%It will have four matrices

    S = zeros(2,2,4);

    W(:,:,1) = [1 0; 0 1];
    W(:,:,2) = [1 0; 1 1];
    W(:,:,3) = [0 -1; 1 1];
    W(:,:,4) = [1 1; 0 1];


    Subdivisions = [4 1 3; 4 2 3; 1 2 3; 4 1 2];

    for i = 1:4
        A = matrix_A_quad(X,T,k,Subdivisions(i,:));
        S(:,:,i) = A/W(:,:,i);
    end




end
```

A.5 Function to calculate matrix S for each triangular element

```matlab
function [S] = matrix_S(X,T,i,n)
%This function calculates matrix S for each element i

    A = zeros(2,2);
    if n == 2
        W = [1 1/2; 0 (sqrt(3))/2];
    else
        W = [1 0; 0 1];
    end

    A(1,1) = X(T(i,2),1) - X(T(i,1),1);
    A(1,2) = X(T(i,3),1) - X(T(i,1),1);
    A(2,1) = X(T(i,2),2) - X(T(i,1),2);
    A(2,2) = X(T(i,3),2) - X(T(i,1),2);

    S = A/W;

end
```

A.6 Function to calculate matrix A for each quadrilateral element

```matlab
function [A] = matrix_A_quad(X,T,i,s)
    A(1,1) = X(T(i,s(2)),1) - X(T(i,s(1)),1);
    A(1,2) = X(T(i,s(3)),1) - X(T(i,s(1)),1);
    A(2,1) = X(T(i,s(2)),2) - X(T(i,s(1)),2);
    A(2,2) = X(T(i,s(3)),2) - X(T(i,s(1)),2);

end
```

A.7 Function to calculate the sum of qualities of the sub-domain triangles of each quadrilateral element

```
function H = innersum(S_l,n)

H = zeros(1,4);

for i = 1:4
    A = norm(S_l(:,:,i),'fro');
    B = det(S_l(:,:,i));
    H(i) = (A^2/(n*B^(2/n)))^2;
end

H = sum(H);
end
```