CentraleSupélec

université PARIS-SACLAY

Intermediary project report

ST5 - Acoustic and electromagnetic pollution control

LUCYSZYN Edward

October 17, 2023

Contents

1 Introduction

Knowing how to solve the equations of acoustic and electromagnetic waves can help solve many of today's challenges, such as limiting noise in an aircraft engine or insulating a building from noise pollution.

Solving these equations analytically can quickly become impossible. That's why you need to know how to solve them numerically. Throughout this report, we'll look at methods for controlling, analyzing and using the mesh of a finite-difference method, for example.

First of all, during all this project, only the basic libraries (`Numpy`, `Scipy` or `Matplotlib`) have been used. After, there are three arrays that will be used during all the simulation:

- `node_coords`: represents all the coordinates of the points that are in the mesh in the form of a 2-dimensional matrix. There must be no point in the mesh that do not belong to any element.
- `elem2nodes`: indicates the index of the points present in each element. For example, if the element of index 0 has 4 nodes, then `elem2nodes` will have this format `[id`$_1$`, id`$_2$`, id`$_3$`, id`$_4$`, ...]`.
- `p_elem2nodes`: is the cursor's array of `elem2nodes`. This array indicates from which to which position to which position we have all the indexes of the nodes in one element. This basically means that if we need the indexes of the nodes of element i, we have to check the array `elem2nodes[p_elem2nodes[i]]: p_elem2nodes[i + 1]]`.

These three arrays represent the mesh and will be read of used at each question.

Also, I coded four functions to plot elements and nodes. Theses functions are `plot_elem`, `plot_node`, `plot_all_elem` and `plot_all_node`. They work with `Matplotplib.pyplot`. For example, in Figure 2, there are plots that correspond to the theoretical example of Figure 1 with inputs:
`node_coords = np.array([[0, 0, 0], [1, 0, 0], [1, 1, 0], [0, 1, 0], [1.5, 0, 0], [1.5, 1, 0], [1.5, 2, 0]])`,
`elem2nodes = np.array([0, 1, 2, 3, 1, 2, 5, 4, 2, 5, 6])`,
`p_elem2nodes = np.array([0, 4, 8, 11])`.

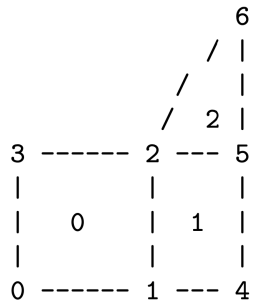

Figure 1: Classic example of mesh with 2 quadrangles and 1 triangle

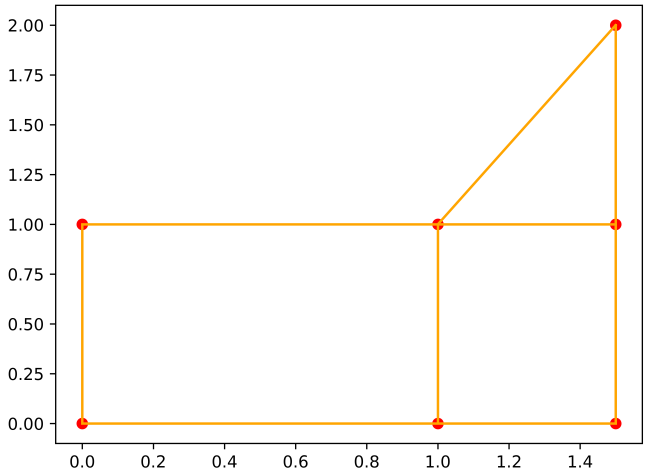

Figure 2: Plot of the classic example in Figure 1 using the plotting functions

Last but not least, during the courses one function to compute the arrays `node2elems` and `p_node2elems` was given. This function is `build_node2elems` and works with the compressed sparse row (CSR) date storage system. Knowing that, it is normally possible to understand all this report.

2 Modifying the mesh

2.1 Question 7: Adding and removing nodes and elements to the mesh

The first thing to do when you have a mesh is to be able to manipulate it. This begins by adding and removing nodes and elements.

The first two functions `add_elem_to_mesh` and `add_node_to_mesh` are

quick to do. There is no re-indexing and we just need to add points. These are simply done in one line by the functions `np.append(array, element to append)`.

Here is a test of theses functions after having used the 2 commands:
`node_coords, p_elem2nodes, elem2nodes = add_node_to_mesh(node_coords, p_elem2nodes, elem2nodes, np.array([[1, 2, 0]]))`,
`node_coords, p_elem2nodes, elem2nodes = add_elem_to_mesh(node_coords, p_elem2nodes, elem2nodes, np.array([2, 5, 6, 7]))`.

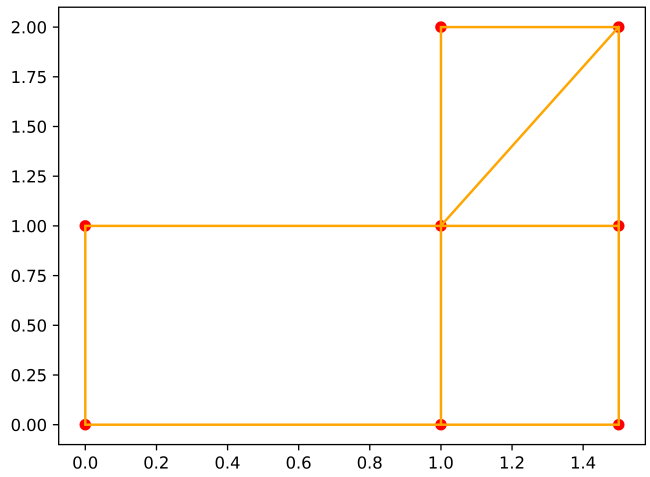

Figure 3: Plot of the classic example with one more node and one more element

Then, there is the complex part. We start by doing `remove_elem_to_mesh` since it is sure that there is just one element to remove. This constant piece of information is very useful because it shortens the algorithm. This function takes as the three classic arrays and the id of the element of to remove as arguments. Here is how the function works:

1. Collecting the indexes of the points which only belong to the point to remove;
2. Delete the points of the element to remove in `elem2nodes`;
3. Changing `p_elem2nodes` by removing `p_elem2nodes[elemid]` and shifting the values after by the number of nodes we have removed in `elem2nodes`;
4. At this point `node_coords`, `elem2nodes` and `p_elem2nodes` are "correct". But there may exist some orphan points in `node_coords` which belong to zero element. We have their index thanks to the first step, so we remove them from `node_coords` and then we change the values of the indexes in `elem2nodes` to make it correspond with the new `node_coords`.

For `remove_node_to_mesh`, I used the previous function `remove_elem_to_mesh`. Firstly, I start by collecting the indexes of the element in which the node belongs. Then, in the hypothetical case that the node belongs to 0 element, we do the same with the previous function: we remove the point from `node_coords` and then we change the values of the indexes in `elem2nodes` to make it correspond with the new `node_coords`.
In the other case, we just apply the function `remove_node_to_mesh` with every elements to whose the point belongs. Since `remove_node_to_mesh` removes the orphan points, the node to remove, its elements and the points that became orphan are removed.

We test these functions with the commands:
`node_coords, p_elem2nodes, elem2nodes = remove_node_to_mesh(node_coords, p_elem2nodes, elem2nodes, 0)` for the Figure 4, `node_coords, p_elem2nodes, elem2nodes = remove_elem_to_mesh(node_coords, p_elem2nodes, elem2nodes, 2)` for the Figure 5.

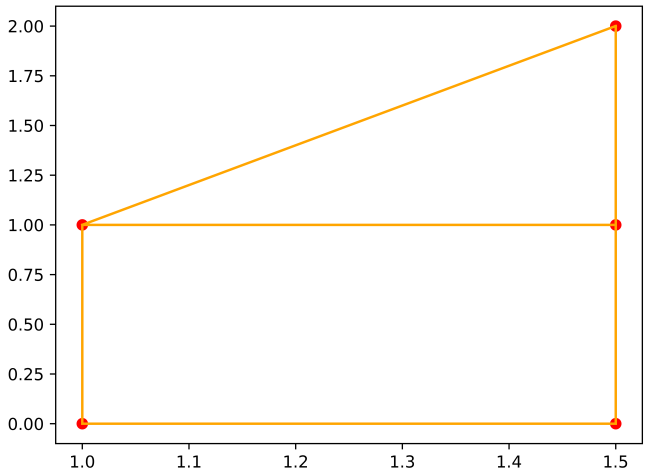

Figure 4: Plot of the classic example with the node 0 removed

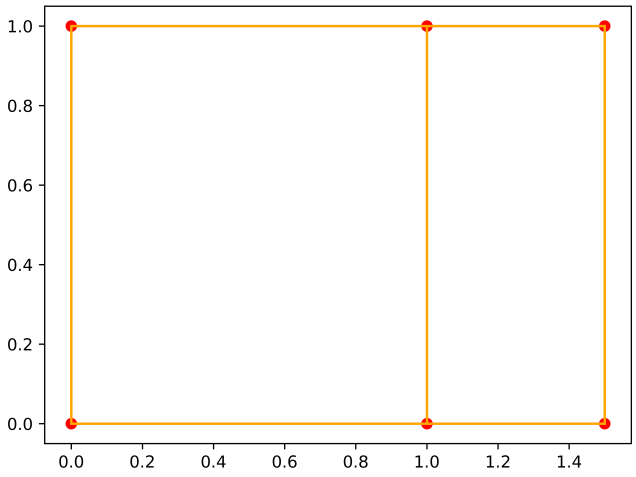

Figure 5: Plot of the classic example with the element 2 removed

3 Analyzing and calculating mesh quality

3.1 Question 1: Aspect ratio calculation

The aspect ratio is a quality metric for the quality of elements in a mesh. Its formula depends on the shape of the element. If it is a triangle, then the formula is

$$Q = \alpha \frac{h_{max}}{\rho}$$

where $\alpha = \frac{\sqrt{3}}{6}$, h_{max} is the longest edge of the triangle and ρ is the radius of the inscribed circle. To compute ρ we can use the formula:

$$\rho = \sqrt{\frac{(s-a)(s-b)(s-c)}{s}} \text{ with } s = \frac{1}{2}(a+b+c).$$

Something interesting is that the aspect ratio for triangles is greater or equal to 1, and the equality case holds for equilateral triangle. The more the triangle will be "degenerated", for example with a very large angle, the more the aspect ratio will big.

For quadrangles, the formula changes and becomes:

$$Q = 1 - \frac{\sum_{i=1}^{4} |\boldsymbol{e_i e_{i+1 \bmod 4}}|}{4}$$

where $\boldsymbol{e_i}$ is the elementary vector supporting each side of the quadrangle. Now we remark that the aspect is between 0 and 1, and is equal to 1 for a rectangle.

So the more the quadrangle is degenerated, the more the aspect ratio will tend to 0.

To compute it we simply have to copy the formulas above. The function which does it is `compute_aspect_ratio`. To test the function, we compute the aspect ratio of the elements in the classic example in introduction. I found:

$$Q_0 = 1,0; Q_1 = 1,0; Q_2 = 1,69$$

where Q_i is the aspect ratio of element i.

3.2 Question 2: Length factor calculation

The length factor is one other metric to class the quality of elements in a mesh. The formula is the same whatever is the number sides that the element has. It is:

$$E = \frac{h_{min}}{h_{medium}}.$$

The function that computes the length factor is `compute_length_factor`. We also test it on the classic example. I obtain:

$$E_0 = 1,0; E_2 = 0,66; E_2 = 0,573$$

where E_i is the edge length factor of element i.

3.3 Question 4: Analysis of the mesh and diagrams

For the analysis of the mesh we need to compute the edge length factor and the aspect ratio of every elements in the mesh. In order to have quantity always between 0 and 1, we will take the inverse of the aspect ratio that we know for triangles. So the new formula will be:

$$Q' = \frac{1}{Q} = \frac{\rho}{\alpha h_{max}}.$$

Know in every case, the more the quantity tend to 1, the more the mesh is structured and regular. Then we create a histogram and we plot it thanks to `np.hist` and `matplotlib.pyplot.stairs`. The function which does this is `analysis_of_the_mesh`. For the tests, there will be shown after question 3 because with the classic example it is not interesting since there is not enough nodes.

4 Controlling mesh data

This part is really about seeing how it is possible to use mesh data and to change it.

4.1 Question 3: Shifting internal nodes

It is asked to shift the internal nodes knowing already the indexes of the nodes on the border. The function which does this is `shift_internal_nodes`. This

function starts to determine what is the minimal length of all the sides of all the elements. This allows to do a shift that is not too large whatever is the size of the mesh.

With the test in `quadrangles_grid_example`, we obtain this type of shift.

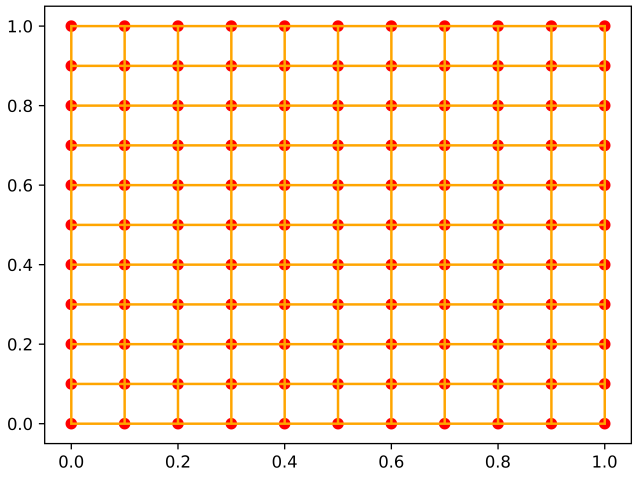

Figure 6: Original grid with quadrangles

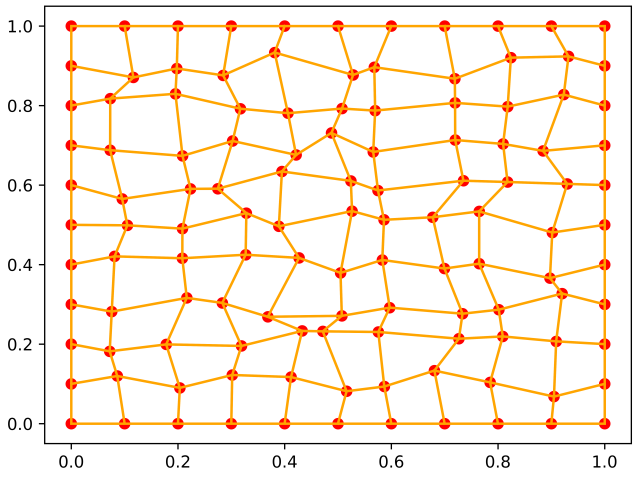

Figure 7: The same grid but with the internal nodes shifted

And for the results about the aspect ratios and the edge lengths, we obtain this.

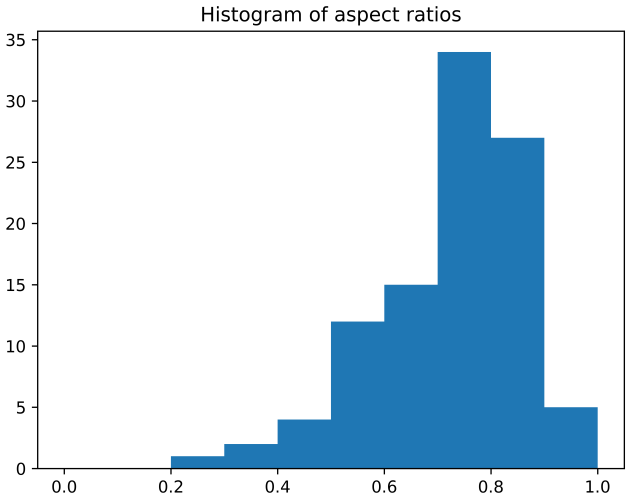

Figure 8: Histogram of aspect ratios of the elements in the shifted grid just above

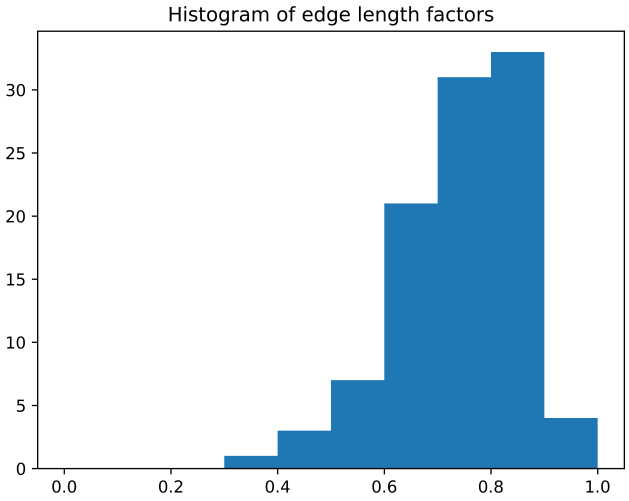

Figure 9: Histogram of edge length factors of the elements in the shifted grid just above

4.2 Question 5: Computing barycenters of element

This function was given during the lectures. It just computes the average of the coordinates of the nodes which belong to the element. The code is in the function `compute_barycenter_of_element`.

5 Creating fractals

5.1 Question 8

Fractals are very important when it comes to controlling the acoustic waves because they can prison the waves. To create a fractal we need to repeat one pattern again and again. To make this surface constant, I chose to repeat one pattern that has a zero area (i.e. the integral of the pattern is equal to 0).

In my code a pattern is an array of coordinates of points that represent the one side of the fractal between points $(0, 0)$ and $(0, 1)$. Then, for each side where I need to apply the pattern I take the extrema of the sides and I apply a rotation, a homothety and a translation to have the coordinates of the points to insert. It gives this:

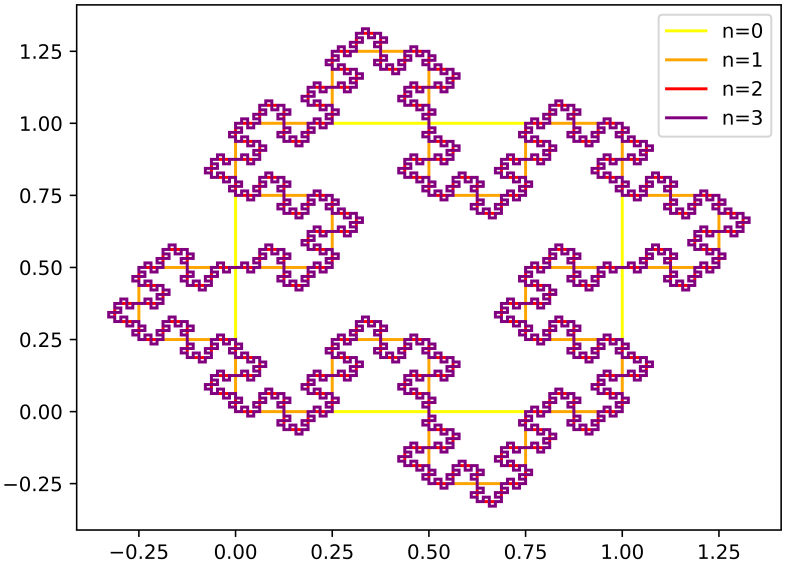

Figure 10: Fractal obtained by starting with a square and a squared pattern

For the meshing of these fractals, i am currently working on the Rupper's algorithm that creates good mesh whatever is the form. I have already done the constrained Delaunay algorithm.