Università della Svizzera italiana

Institute of Computing
CI

High-Performance Computing Lab **Institute of Computing**

Student: Bella Jonatan Discussed with: FULL NAME

Solution for Project 6

HPC Lab — Submission Instructions
(Please, notice that following instructions are mandatory: submissions that don't comply with, won't be considered)

- Assignments must be submitted to iCorsi (i.e. in electronic format).
- Provide both executable package and sources (e.g. C/C++ files, Matlab). If you are using libraries, please add them in the file. Sources must be organized in directories called:
  Project_number_lastname_firstname
  and the file must be called:
  project_number_lastname_firstname.zip
  project_number_lastname_firstname.pdf
- The TAs will grade your project by reviewing your project write-up, and looking at the implementation you attempted, and benchmarking your code's performance.
- You are allowed to discuss all questions with anyone you like; however: (i) your submission must list anyone you discussed problems with and (ii) you must write up your submission independently.

1. Task: Install METIS 5.0.2, and the corresponding Matlab mex interface

2. Task: Construct adjacency matrices from connectivity data [10 points]

To accomplish this task, I first define the path to the CSV files and the output directory. Then, I make a list with the names of each country and loop over it such that, using *sprintf* to create the path to the CSV file, I read the adjacency and coordinates files using *readmatrix*, since the suggested csvread was not recommended by Matlab.

Once the edges and coords variables are set, I obtain the length and I create a matrix with 1's where the edge goes from the defined in the files as "from" to the "to". However, this will be directed, and we need a bidirection - symmetric matrix - therefore, I transpose the matrix and sum it to obtain the final required symmetric adjacency matrix.

I add the plotting through the *gplotg* function, and save the resulting matrix with the respective images:

The results are stored in : "`/Datasets/Countries_Meshes/Countries_Mat`" whereas the running file is in the Source folder as provided.

(a) Norway (b) Vietnam

Figure 1: Graph visualization of Norway and Vietnam

3. Task: Implement various graph partitioning algorithms [25 points]

For the implementation of the **spectral graph bisection** and **inertial graph bisection** I used the files `bisection_spectral.m` and `bisection_inertial.m` respectively.

In the first case, to obtain the D matrix, I sum each of the rows of the A matrix and then convert it to a diagonal matrix where each element of the diagonal is the degree of the node. Then, I obtain the Laplacian matrix $L = D - A$. Following this, I obtain the eigenvector of the second-smallest eigenvalues. Since partitioning using 0 as threshold provided better results on the Vietnam graph as test, I opted to use 0 as threshold with the following example result:

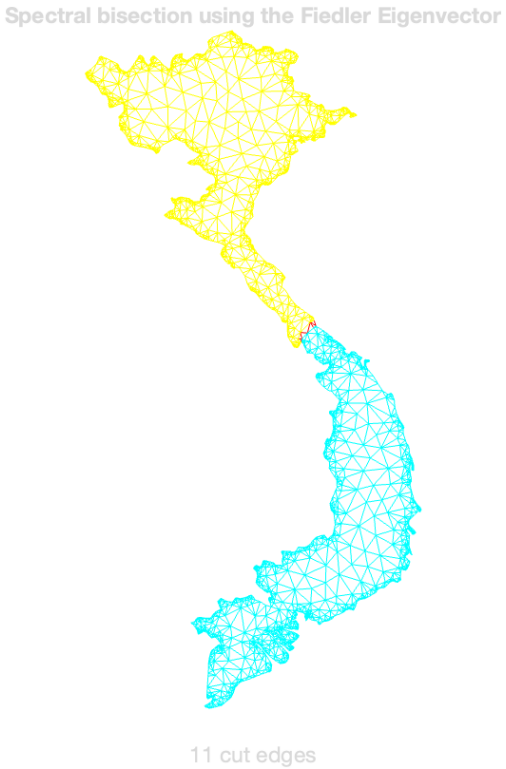


Figure 2: Spectral bisection of Vietnam

```
% Steps
% 1. Construct the Laplacian. -> L = D - A
% given A we need to construct D:
```

```
        D = diag(sum(A,2));
        L = D - A; %get the laplacian as we saw in class
        %display(L)

        % 2. Calculate its eigensdecomposition.
        [V,D] = eigs(L, 2, 'smallestreal'); % https://ch.mathworks.com/help/mat

        % 3. Label the vertices with the components of the Fiedler vector.
        fiedler = V(:,2); %as we saw in class, we take the second highest eigenv

        % 4. Partition them around their median value, or 0.
        map = zeros(length(fiedler),1); %it actually provided better results in
        map(fiedler > 0) = 1;

        %med = median(fiedler);
        %map = zeros(length(fiedler),1);
        %map(fiedler > med) = 1;

        [part1,part2] = other(map);

        if picture == 1
            gplotpart(A,coords,part1);
            title('Spectral bisection using the Fiedler Eigenvector');
        end

    end
```

For the inertial bisection, I first obtain the center of mass by averaging over both axis. Then, I compute the difference between the entries and the corresponding center of mass. I compute the inertia matrix M and obtain the eigenvector of the smallest eigenvalue. Then, as suggested, I use the partition implementation to obtain the indices:

```
% Steps
% 1. Calculate the center of mass.
n = size(xy,1);
%center of mass as defined in equation 4 of the pdf:
x_center = (1/n) * sum(xy(:,1));
y_center = (1/n) * sum(xy(:,2));

%display(x_center)
%display(y_center)
% 2. Construct the matrix M.
%  (Consult the pdf of the assignment for the creation of M)
diff_x = xy(:,1) - x_center;
diff_y = xy(:,2) - y_center;

Sxx = sum(diff_x.^2);
Syy = sum(diff_y.^2);
Sxy = sum(diff_x .* diff_y);

M = [Syy, Sxy; Sxy, Sxx];

display(M)
% 3. Calculate the smallest eigenvector of M.
[V,D] = eigs(M, 2, 'smallestreal');
```

```
display(V)
%display(D)

u = V(:,1);
%display(u)
% 4. Find the line L on which the center of mass lies.
%projections = diff_x * u(1) + diff_y * u(2); %as we saw in class (slide 14 - gr
%since it is implemented in the partition one:
% 5. Partition the points around the line L.
%   (you may use the function partition.m)
[indices1, indices2] = partition(xy, u);
map = ones(n,1);
map(indices1) = 0;
[part1,part2] = other(map);


if picture == 1
    gplotpart(A,xy,part1);
    title('Inertial bisection using the Fiedler Eigenvector');
end
```

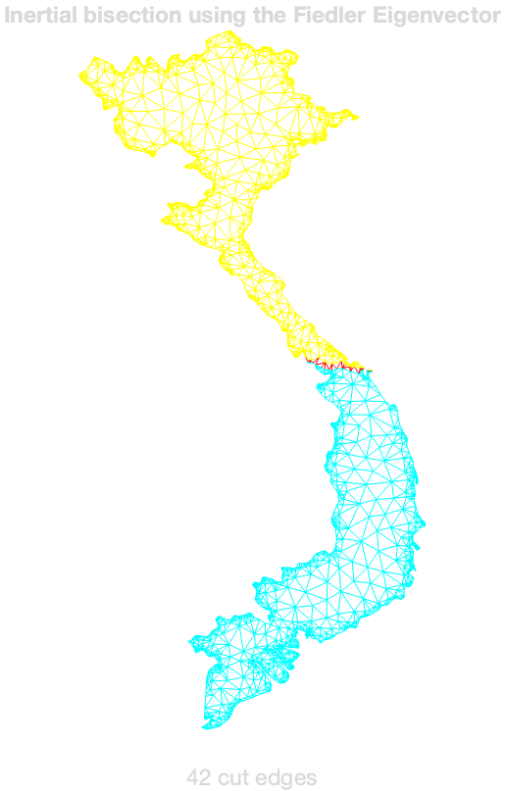


Figure 3: Inertial bisection of Vietnam

The final table with the benchmark results with respect to both implementations and the Metis and Coordinate are the following:

Mesh	Coordinate	Metis 5.0.2	Spectral	Inertial
mesh1e1	18	18	17	20
mesh2e1	37	34	37	47
mesh3e1	19	20	22	19
mesh3em5	19	20	119	19
airfoil1	94	93	59	93
netz4504_dual	25	19	24	27
stufe	16	17	16	16
3elt	172	96	94	257
barth4	206	111	96	208
ukerbe1	32	36	28	28
crack	353	220	232	384

Table 1: Mesh Partitioning Results

4. Task: Recursively bisecting meshes [15 points]

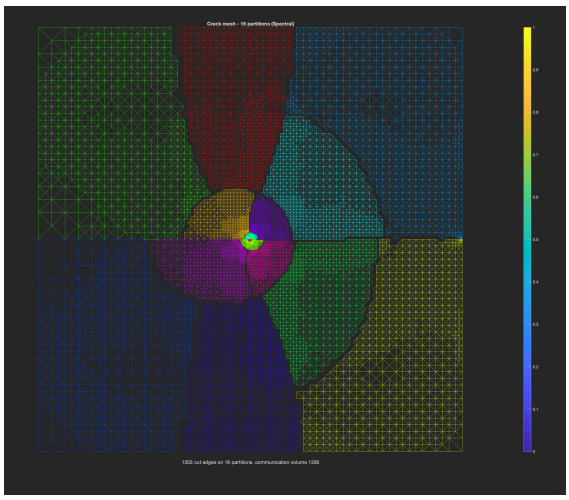
To implement the required recursive bisection, it is important to notice that the `rec_bisection` function is already provided in the toolbox and returns the assignments (indices) to each partition for a given vertex, the list of separating edges between the partitions and the adjacency matrix less the separating edges.

Therefore, we can run the function with the corresponding bisection method, the number of partitions (ex: $nlevels = 3$ such that $2^3 = 8$ partitions), the adjacency matrix, the coordinates and the picture flag.

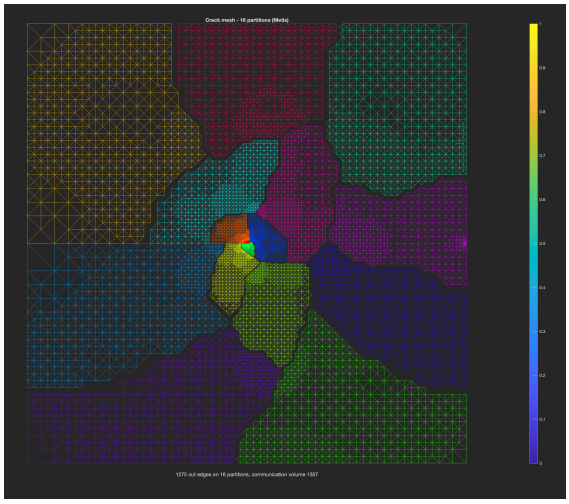
Once the indices are obtained, we can pass this, together with the adjacency matrix to the already implemented *cutsize* function to obtain the number of cut edges. Finally, we just plot each "Crack" mesh partition for p = 16 for each case and the results matrix:

Table 2: Edge-cut results for different bisection methods and partition counts

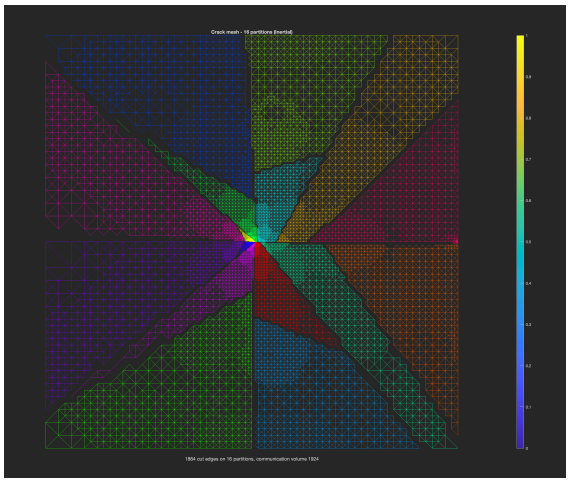
Mesh	Spectral		Metis 5.0.2		Coordinate		Inertial	
	8	16	8	16	8	16	8	16
airfoil1	327	578	318	561	516	819	670	1081
netz4504_dual	105	174	96	159	127	198	165	271
stufe	124	216	108	193	123	227	320	606
3elt	372	671	418	699	733	1168	814	1230
barth4	505	758	470	743	875	1306	977	1492
ukerbe1	116	226	147	245	225	374	340	499
crack	804	1303	808	1275	1343	1860	1351	1884



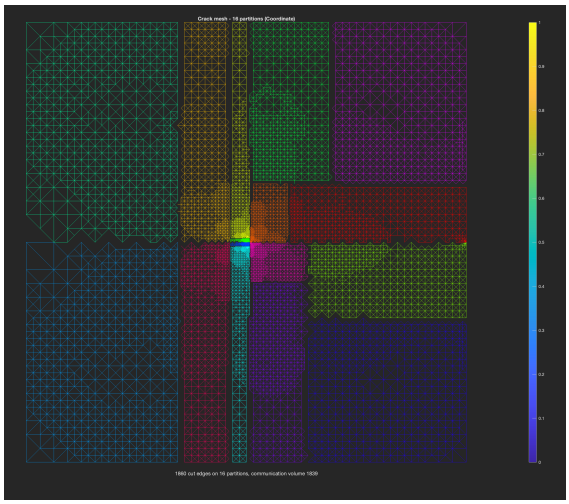
(a) Crack mesh - 16 partitions (Spectral)



(b) Crack mesh - 16 partitions (Metis)



(c) Crack mesh - 16 partitions (Inertial)



(d) Crack mesh - 16 partitions (Coordinate)

5. Task: Comparing recursive bisection to direct k-way partitioning [10 points]

First, to obtain the respective results, I manage the imports of the respective adjacency matrix and coordinates differently according to the file stored. For the USA and Luxembourg roads, I load the matrices, find by displaying the fields that $A = Problem.A$ and $xy = Problem.aux.coord$. For the csv countries, I follow the same procedure that in the first exercise. For the last case, Switzerland, I load the matrix, convert the adjacency matrix into A, guaranteeing that the symmetry is respected, and I define the coordinates as I did with the USA and Luxembourg roads.

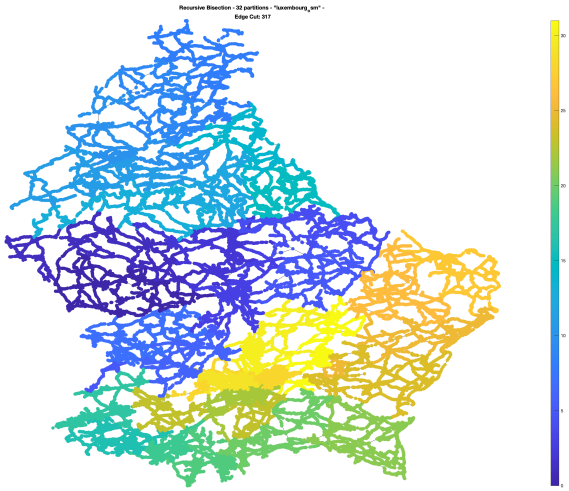
In all the cases, once I have the A and xy matrices, I can run the recursive bisection and the k-way partitioning as defined in metis where $metismex('PartGraphRecursive', A, nparts)$; and $metismex('PartGraphKway', A, nparts)$; returns the partition assignments and the number of edges that cross between partitions.

Finally, we can store each cut result and print them, with the following results:

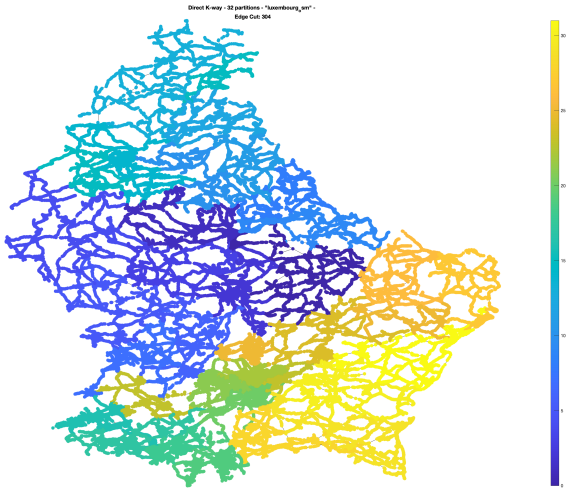
Table 3: Comparing the number of cut edges for recursive bisection and direct multiway partitioning in Metis 5.0.2.

Partitions	Luxemburg	usroads-48	Greece	Switzerland	Vietnam	Norway	Russia
16 (Recursive)	191	585	318	685	270	271	572
16 (K-way)	185	545	301	665	231	241	551
32 (Recursive)	317	983	500	1067	445	509	941
32 (K-way)	304	921	486	1013	418	436	931

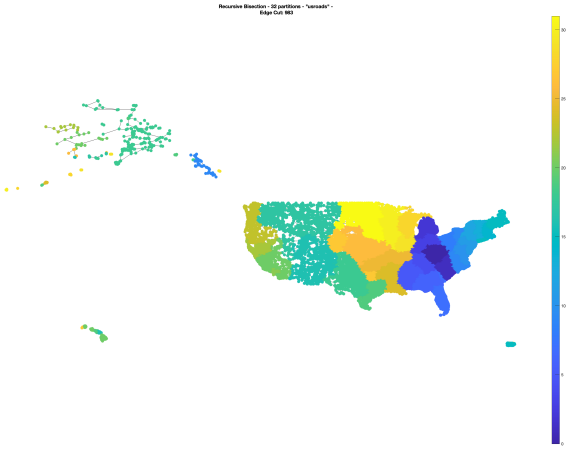
In addition, for the plots, if the number of partitions is 32 we plot the Recursive bisection and the K-way partitioning for each country. I attached the required ones here with the addition of Switzerland.



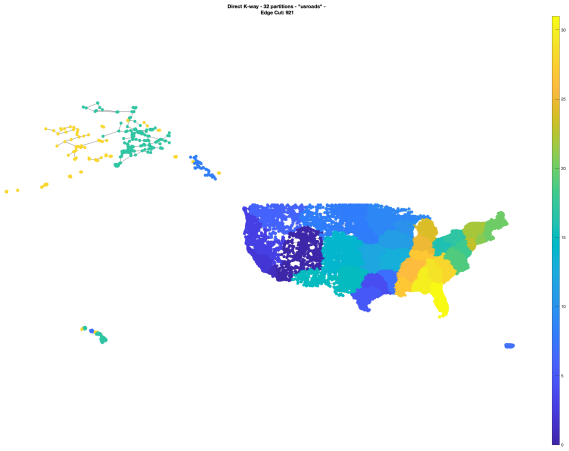
(a) Luxemburg - 32 partitions (Bisection Metis)



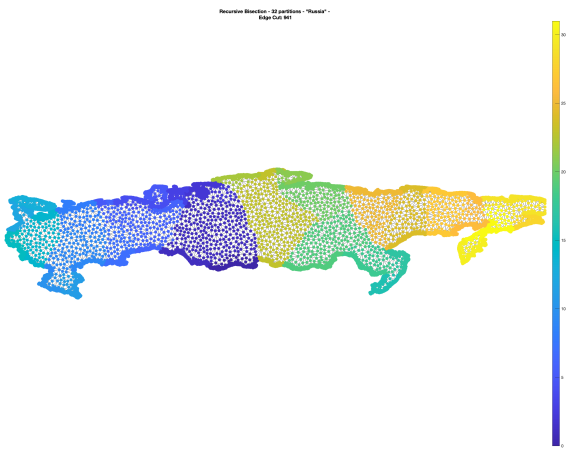
(b) Luxemburg - 32 partitions (K-way Metis)



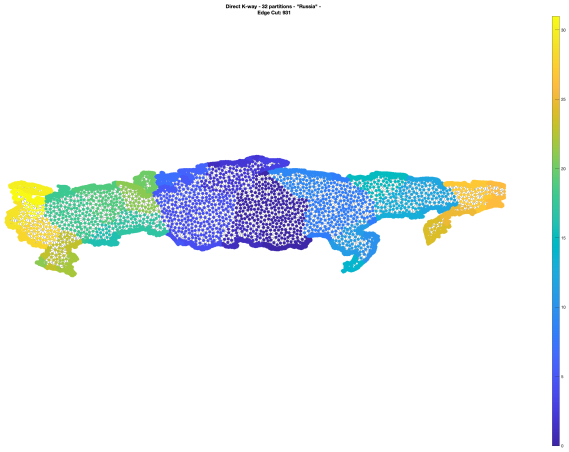
(c) USA Roads - 32 partitions (Bisection Metis)



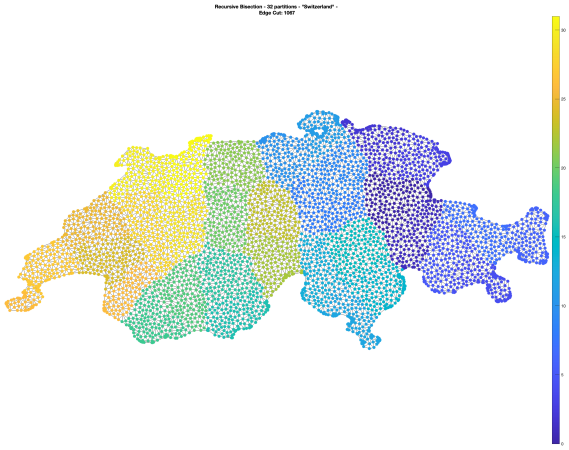
(d) USA Roads - 32 partitions (K-way Metis)



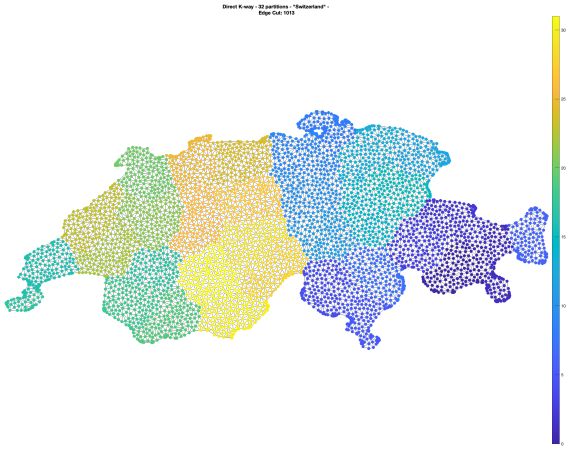
(a) Russia - 32 partitions (Bisection Metis)



(b) Russia - 32 partitions (K-way Metis)



(c) Switzerland - 32 partitions (Bisection Metis)



(d) Switzerland - 32 partitions (K-way Metis)

What I can observe from the results is that k-way partitioning consistently over performs (producing fewer cut edges) than the recursive bisection across all countries, for any size of partitions. The difference is more pronounced as the number of partitions increases. As for example, for Switzerland, it produces reduces the cut by 20 edges in 16 partitions and 54 in 32 partitions. The difference is smaller for Luxembourg and the highest improvement is actually in the USA with 40 fewer cuts in 16 partitions and 62 fewer cuts in 32 partitions. This makes sense since recursive bisection acts greedily locally at each split which may not be the global optimum since it continually divides the graph into two parts and optimizes the binary split. This can lead for a good initial 2 partitions, but then these partitions can be hard to split further.

On the other hand, K-way partitioning considers the k partitions simultaneously from the start, moving vertices between any of the k partitions during optimization. This can lead to a better global optimum, but it may be harder to find the optimal k partitions.

The k-way method for example, seems to produce more compact and balance regions than the recursive bisection. To see this in the example, we can visualize Texas regions with the center west of USA difference between the two methods or Russia's regions in general.

6. Task: Utilizing graph eigenvectors [25 points]

6.1. Eigenvectors of the first and second smallest eigenvalues of the Laplacian matrix of the airfoil graph

For the airfoil implementation of the two eigenvectors plot, the construction is identical to previous exercises with the following result:

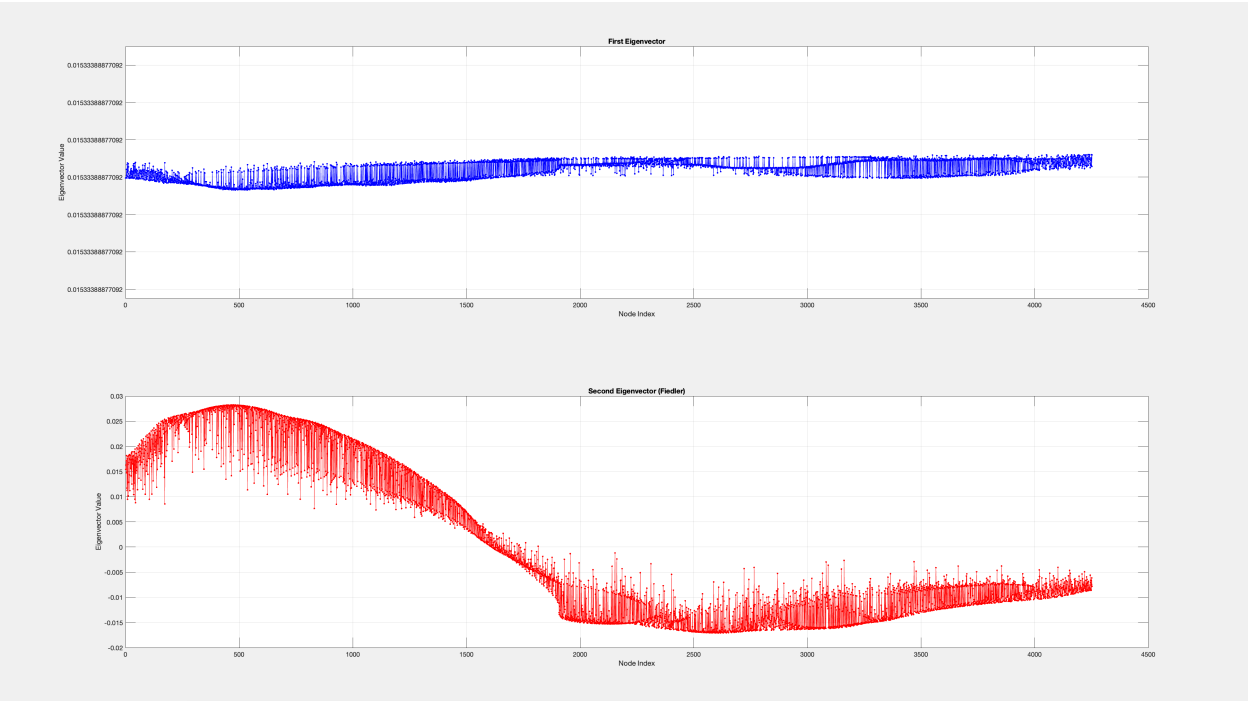


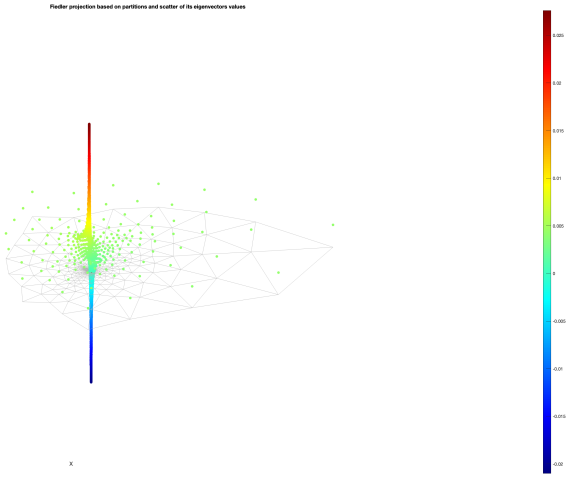
Figure 7: Eigenvectors of the first and second-smallest eigenvalues of the Laplacian matrix of the airfoil graph - 4532 nodes

To understand what is happening, it is good to remember what the Laplacian matrix captures. Which is basically how each node differs from its neighbors, encoding its topology. [1]. Now, the eigen-decomposition of the Laplacian matrix will return λ_1 and λ_2 as the smallest eigenvalues - or "lowest frequencies" [2]. $\lambda_1 = 0$ if the graph is connected, which is the case of the airfoil graph. In addition, we know by construction of the Laplacian that each row sums to zero since it is the degree of each node minus the sum of the edges which are equal. This implies that there is a stable, uniform state where the graph has no direction of variation (following Fielder's string example). By construction this constant vector exist in the null space. Then, this constant eigenvector show that the whole graph forms a connected component (no other eigenvalue = 0). Now, the second smallest eigenvalue λ_2 is the Fielder's vector, and we can see that there is a point where the eigenvector values goes under 0. Therefore, following Fielder's string example and 0 threshold, we can see that it is possible to partition the graph into two balanced parts. Therefore, the results are as expected.

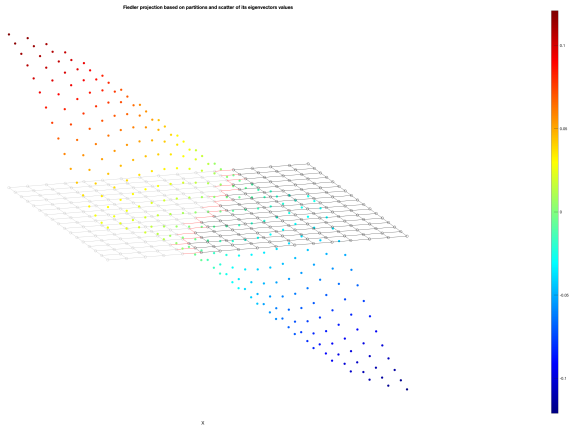
6.2. Plotting the entries of the eigenvector associated with the second smallest eigenvalue with the coloured projection

For implementing this second part of the *Bench_eigen_plot.m*, i obtain both parts of the bisection. Then, I rescale the eigenvector values since if not the plots can be hard to read since the differences can be minimal (given that the z value will be this scaled eigenvectors values). I use *gplotpart* to get the projection plane with the red color for the cut edges and the black and gray for the different partitions. Then, I overlay the scatter 3d points of the eigenvectors values.

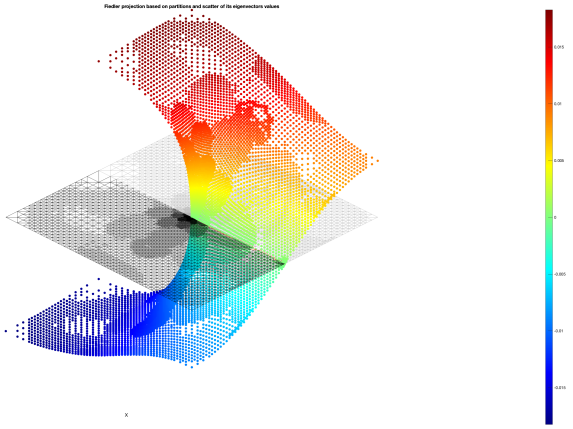
[1] `https://en.wikipedia.org/wiki/Laplacian_matrix`
[2] `http://cs-www.cs.yale.edu/homes/spielman/sagt/sagt.pdf`



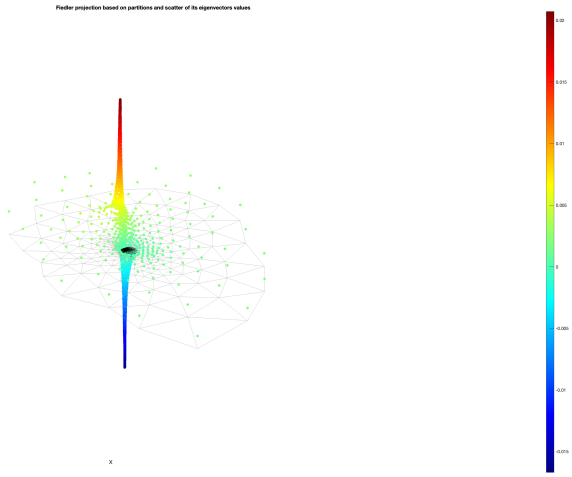
(a) "3elt" eigenvector values and its projection keeping color scaling from example



(b) "mesh3e1" eigenvector values and its projection keeping color scaling from example



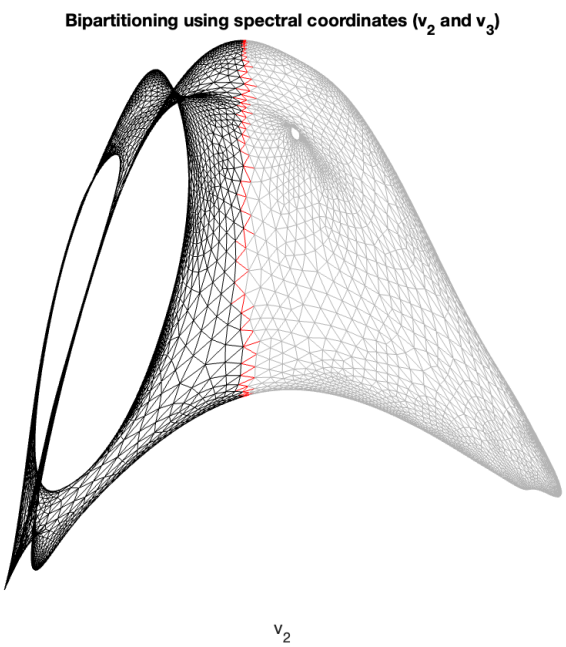
(c) "crack" eigenvector values and its projection keeping color scaling from example



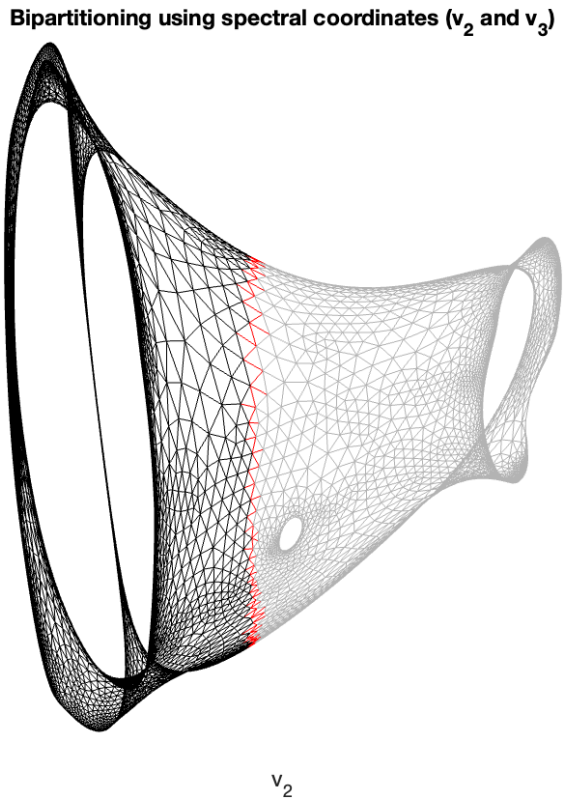
(d) "barth4" eigenvector values and its projection keeping color scaling from example

6.3. Spectral graph drawing method utilizing the eigenvectors of the graph Laplacian matrix L

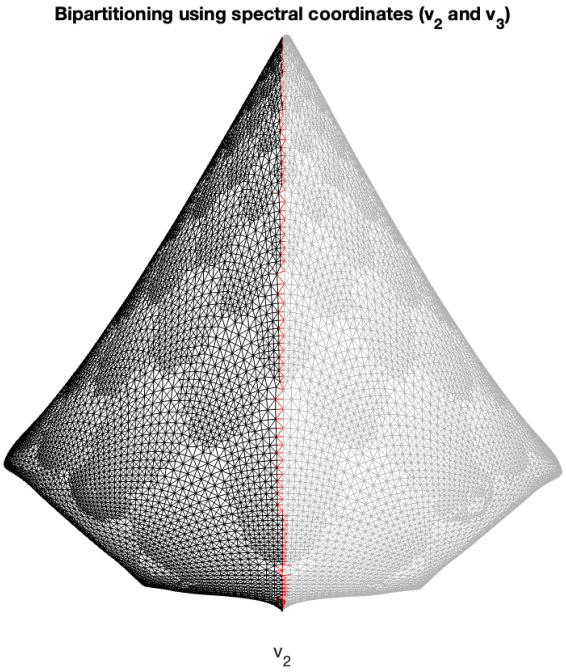
Lastly, for this last part, we create the coordinates of the pair of eigenvector values V_2 and V_3. Then, I pass those coordinates to *gplotpart* to plot the W matrix with these coordinates. The following are the results:



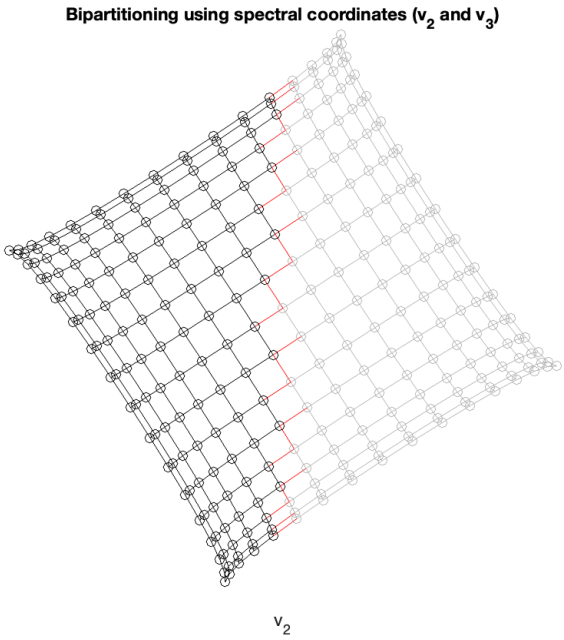
(a) "3elt" spectral coordinates



(b) "barth4" spectral coordinates



(c) "crack" spectral coordinates



(d) "mesh3e1" spectral coordinates