Università
della
Svizzera
italiana

Institute of Computing CI

High-Performance Computing **2022**

Student: Nicolai Hermann Discussed with: Michele Cattaneo

Solution for Project 6

Due date: 07.12.2022, 23:59

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

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

- **read the .csv files in MATLAB:** The csvs for Norway and Vietnam were read using the `csvread` function while skipping the header row.
- **construct the adjacency matrix $\mathbf{W} \in R^{n\times n}$ and the node coordinate list $C \in R^{n\times 2}$, where n is the number of nodes:** First of all the number of nodes was found by taking the maximum over the node numbers and subsequently the matrix was created by calling `accumarray(adj, 1, [num_nodes, num_nodes])`. After creating the adjacency matrix it was not guaranteed to be symmetric, which was checked and fixed if needed. Afterwards, a boolean mask was created by checking if each element is larger than 0. In case of duplicates or after fixing the symmetry it could have happened to get numbers other than 1 in the matrix.
- **visualize the graphs using the function `src/Visualization/gplotg.m`:** The function was simply provided with the adjacency matrix and the coordinates of the nodes and called to produce the following two plots.

(a) Norway (b) Vietnam

Figure 1: Country meshes of Norway and Vietnam

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

- First of all the Laplacian matrix was created using the formulas from the theory part of the assignment. Afterwards the two smallest eigenvectors were obtained using the `eigs` function. Finally, the Fiedler's eigenvector was thresholded with 0 and the partitions were created using the provided `other` function.
- To get the center of mass the mean was taken along the columns to get one value for each axis. Afterwards, the covariance matrix was computed. In the theory part of the assignment the first entry was depicted as xx and the last one as yy which is odd and led to odd results so they were swapped to obtain the correct covariance matrix. The smallest eigenvector was calculated and rotated by 90 deg to get the normal of the hyperplane. The last thing to do was to call the partition function with the coordinates and the normal vector.
- The results of `Bench_bisection.m` are reported in Table 1.

Table 1: Bisection results

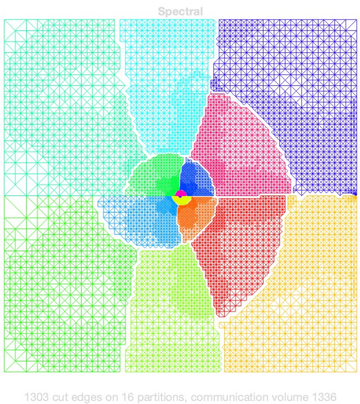
Mesh	Coordinate	Metis 5.0.2	Spectral	Inertial
mesh1e1	18	17	17	19
mesh2e1	37	37	35	47
mesh3e1	19	19	20	19
mesh3em5	19	19	17	19
airfoil1	94	77	59	94
netz4504_dual	25	23	24	30
stufe	16	16	16	16
3elt	172	124	94	209
barth4	206	97	96	194
ukerbe1	32	27	36	28
crack	353	201	232	377

4. Task: Recursively bisecting meshes [15 points]

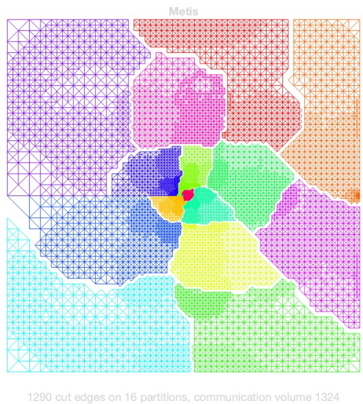
The implementation of `Bench_rec_bisection.m` was quite straight forward. The matrices and coordinates were loaded and various bisections were computed with the `rec_bisection` function with varying bisection methods. After the partitions were calculated the cut sizes were inferred with the `cutsize` function and were subsequently printed. The results are summarized in Table 2. Finally, the results for $p = 16$ for the case "crack" are visualized in Fig. 2.

Table 2: Edge-cut results for recursive bi-partitioning.

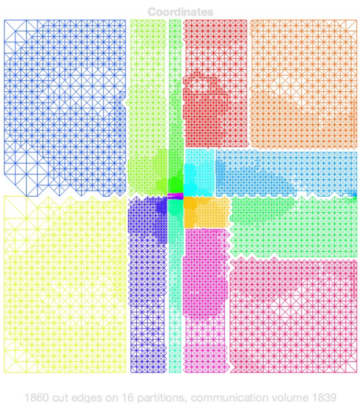
Cases	Spectral		Metis 5.0.2		Coordinate		Inertial	
Levels	8	16	8	16	8	16	8	16
airfoil1	327	578	320	563	516	819	577	897
netz4504_dual	105	174	110	161	127	198	122	200
stufe	124	216	107	194	123	227	136	269
3elt	372	671	395	651	733	1168	880	1342
barth4	505	758	405	689	875	1306	891	1350
ukerbe1	119	225	128	224	225	374	280	468
crack	804	1303	784	1290	1343	1860	1061	1618



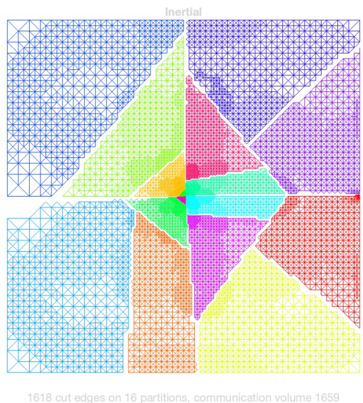
(a) Spectral bisection



(b) Metis bisection



(c) Coordinate bisection



(d) Inertial bisection

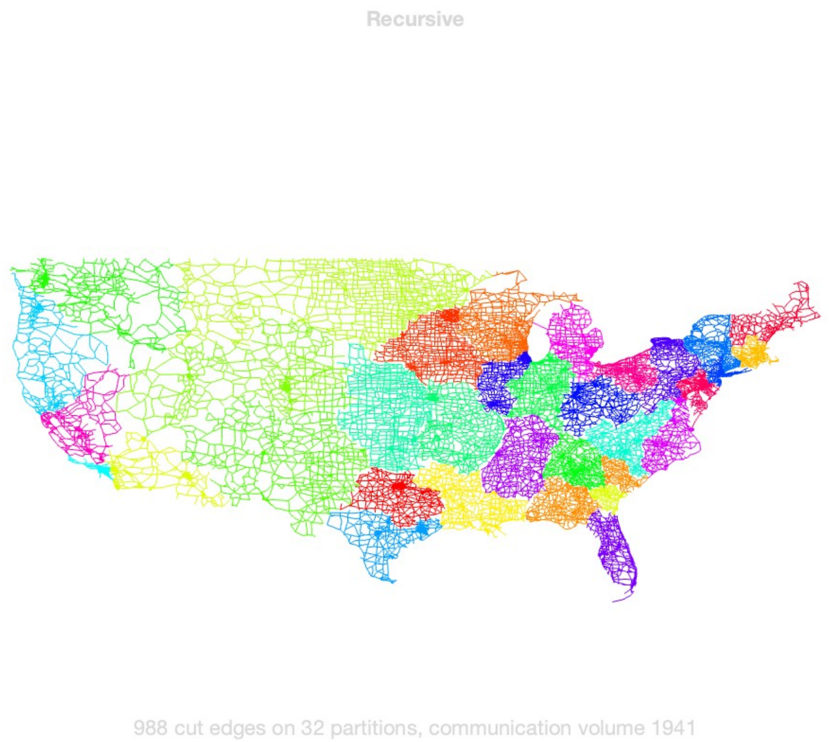
Figure 2: Partitioning plots using different bisection methods recursively using crack.

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

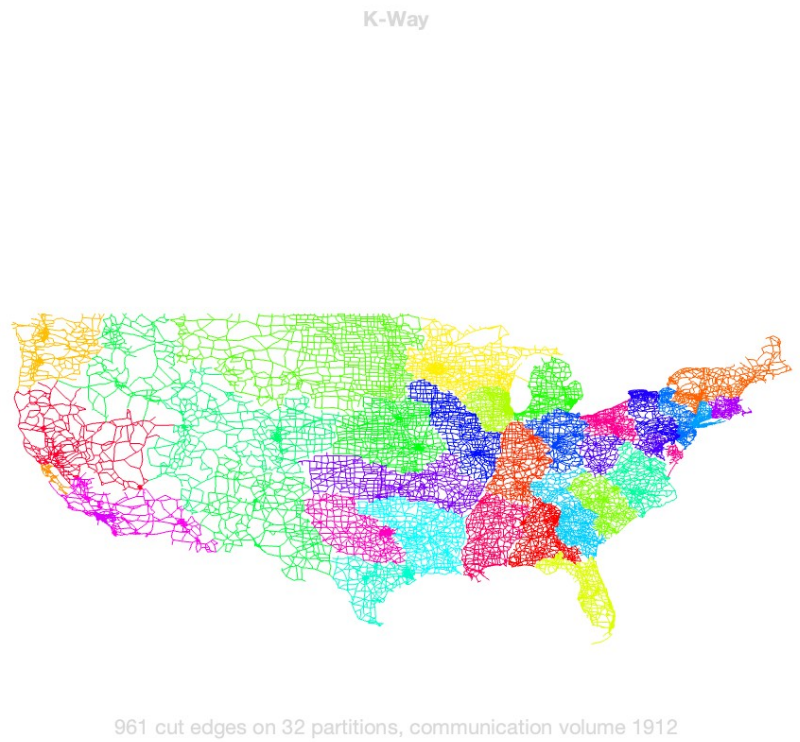
The results are summarized in Table 3 for 16 and 32 partitions. It can be observed that the direct multiway partitioning found smaller cuts. When looking at the resulting partitions it is hard to pinpoint if there was a major improvement due to the huge graph. However on a high level it looks like the partitions are placed a bit smoothly, meaning that the number of neighbours was reduced which definitely benefits the computation speed if you have to wait for fewer processes. The result was somewhat anticipated since the algorithm was carefully designed to improve the recursive pitfalls. The final partitions can be found in Fig. 3, 4 and 5.

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	
Method	Rec	K-way	Rec	K-way	Rec	K-way	Rec	K-way	Rec	K-way	Rec	K-way	Rec	K-way
16	197	170	607	579	297	278	730	673	245	245	284	255	616	551
32	322	279	988	961	509	471	1089	1042	445	411	470	439	1006	933

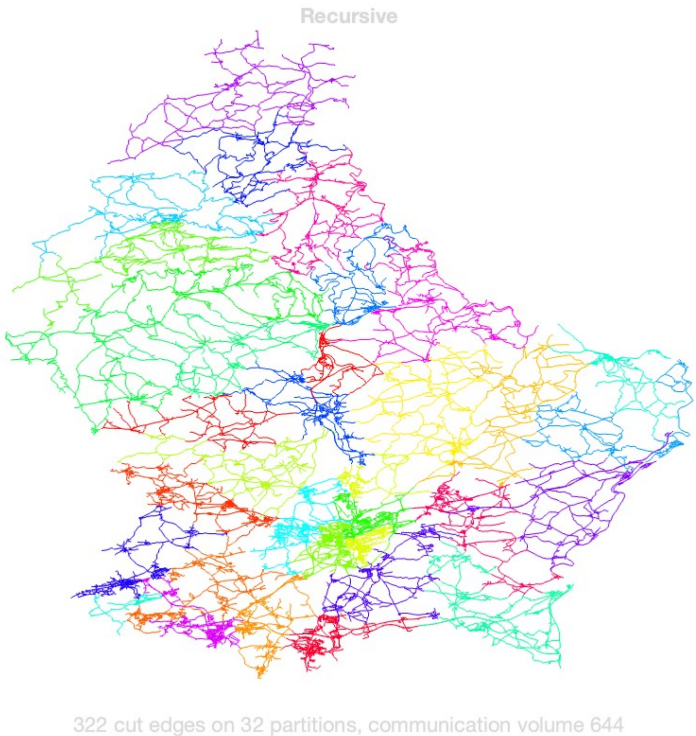


(a) Recursive partitioning

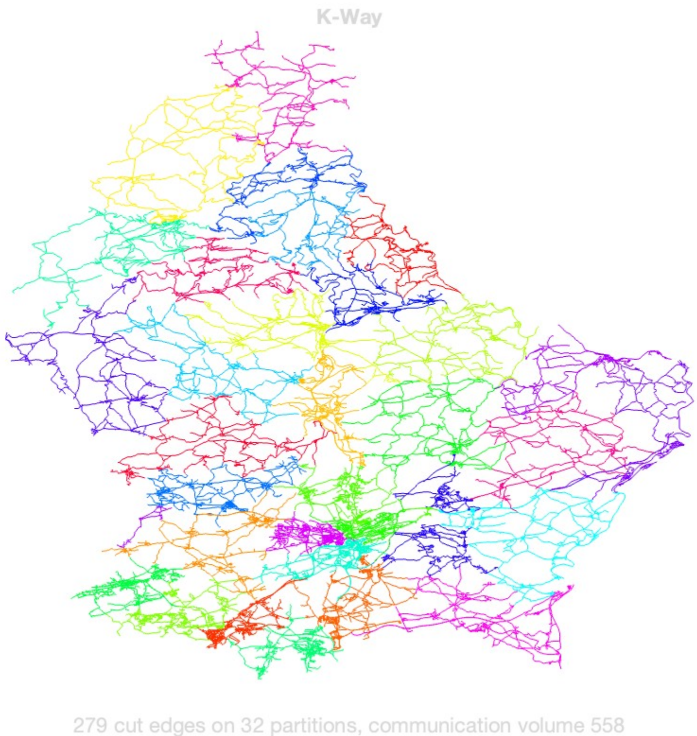


(b) Direct multiway partitioning

Figure 3: Different partitioning methods using the street network of the US.

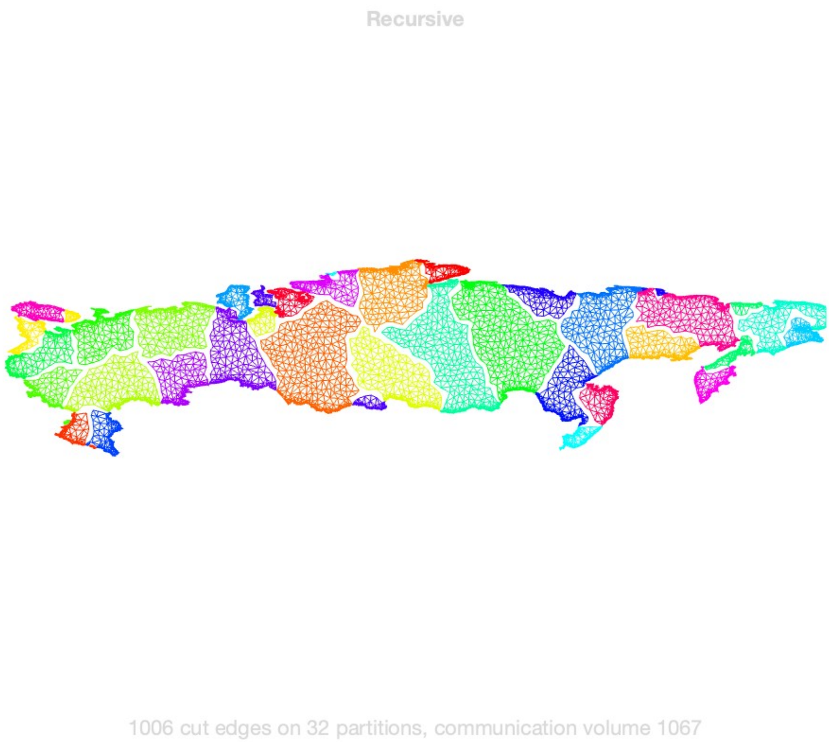


(a) Recursive partitioning

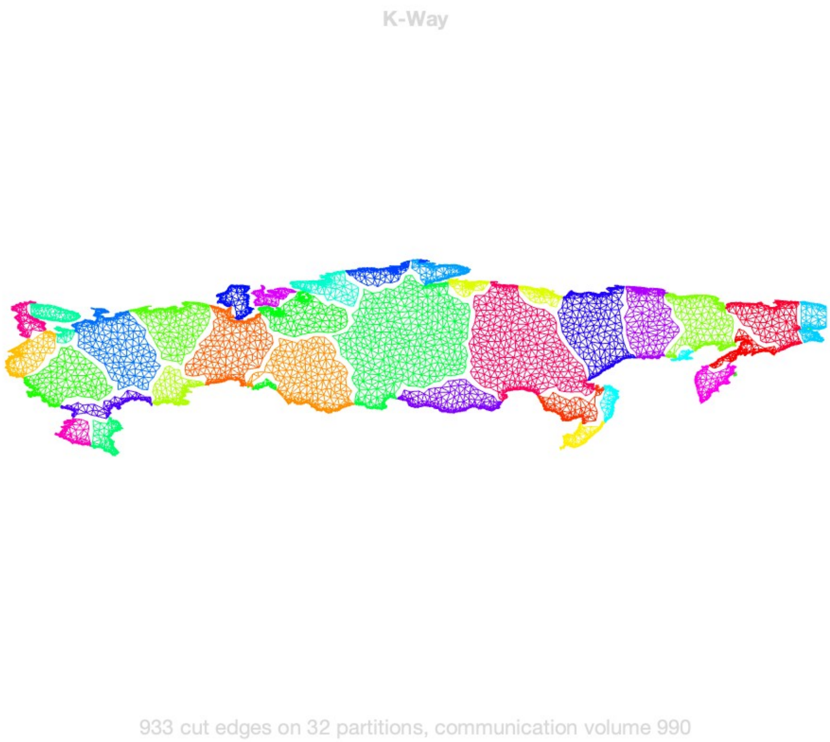


(b) Direct multiway partitioning

Figure 4: Different partitioning methods using the street network of Luxemburg.



(a) Recursive partitioning



(b) Direct multiway partitioning

Figure 5: Different partitioning methods using the street network of Russia.

6. Task: Utilizing graph eigenvectors [25 points]

1. The plot of the two smallest eigenvectors of the Laplacian matrix airfoil1 can be found in Fig. 6. Since we already know that the first eigenvector has an eigenvalue of 0 and is constant, it was not surprising to see that it is actually true. When looking at the second eigenvector depicted in orange it can be seen that the number of points can be roughly divided by two by drawing a line through y=0 which makes sense as well. The results were expected.

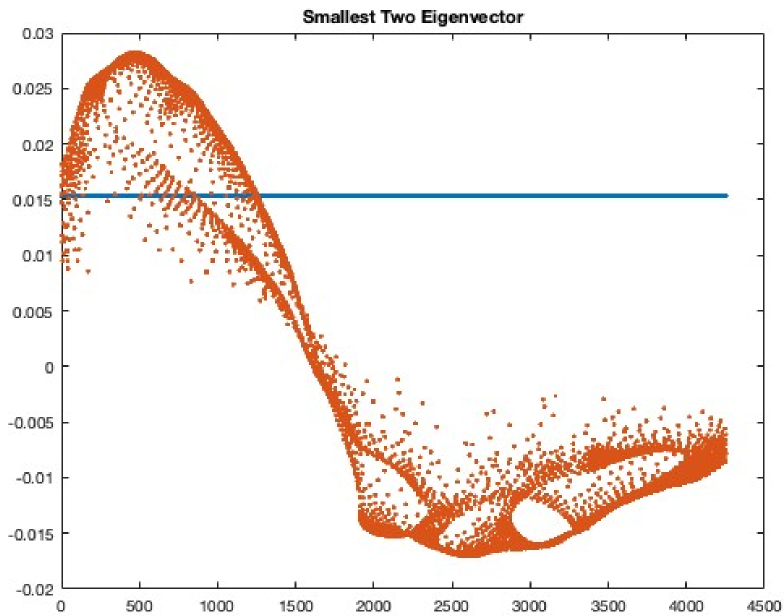
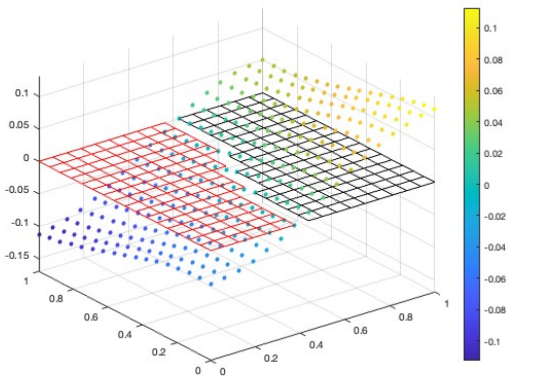
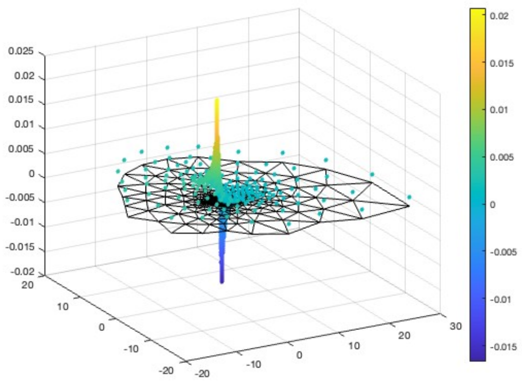


Figure 6: The two second smallest eigenvectors of the Laplacian matrix of airfoil1.mat.

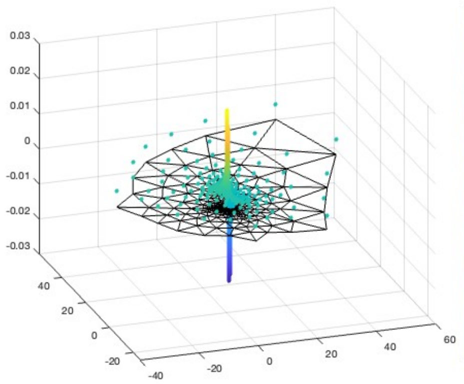
2. The resulting plots can be seen in Fig. 7.



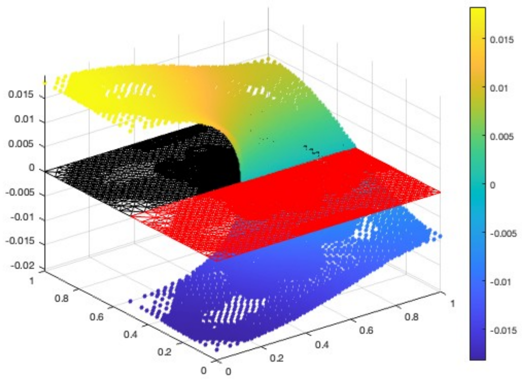
(a) mesh3e1 with projected Fiedler eigenvector.



(b) barth4 with projected Fiedler eigenvector.



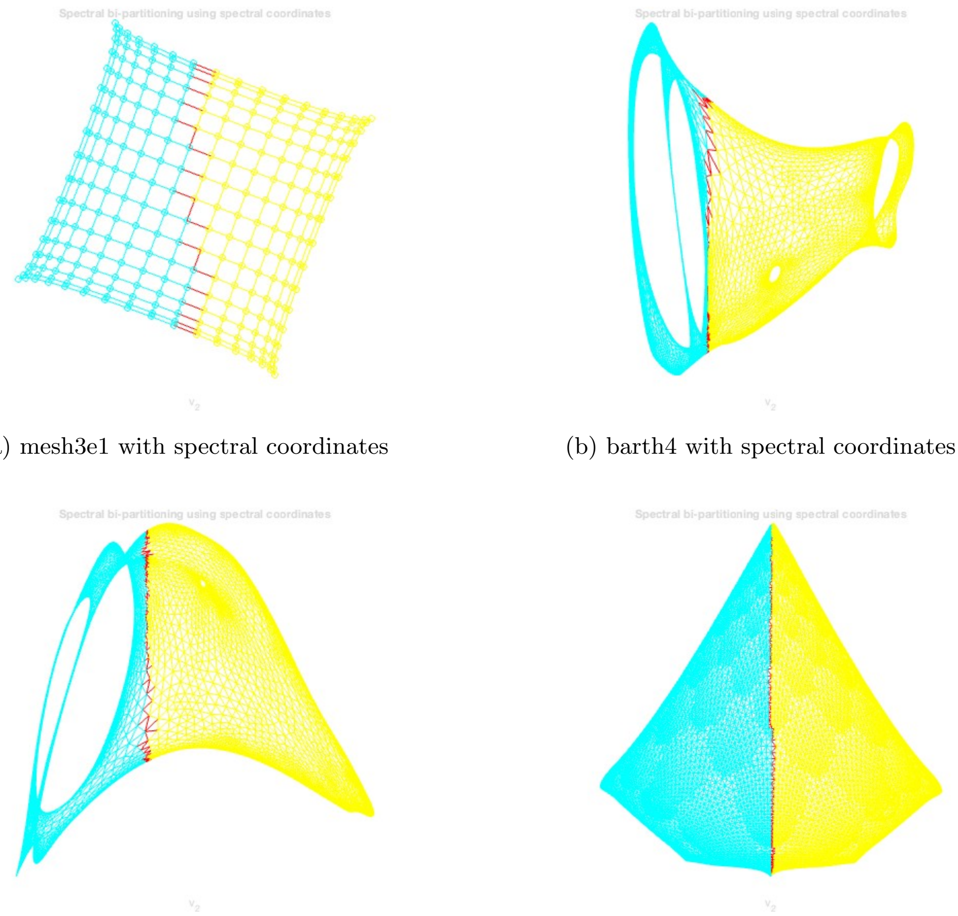
(c) 3elt with projected Fiedler eigenvector.



(d) crack with projected Fiedler eigenvector.

Figure 7: Visualization of different graphs with projected Fiedler eigenvector.

3. The resulting plots can be seen in Fig. 8.



(a) mesh3e1 with spectral coordinates

(b) barth4 with spectral coordinates

(c) 3elt with spectral coordinates

(d) crack with spectral coordinates

Figure 8: Visualization of different graphs with spectral coordinates.

References

7. Task: Quality of the Report [15 Points]

Additional notes and submission details

Submit the source code files (together with your used `Makefile`) in an archive file (tar, zip, etc.), and summarize your results and the observations for all exercises by writing an extended Latex report. Use the Latex template from the webpage and upload the Latex summary as a PDF to iCorsi.

- Your submission should be a gzipped tar archive, formatted like project_number_lastname_firstname.zip or project_number_lastname_firstname.tgz. It should contain:
  - all the source codes of your MATLAB solutions;
  - your write-up with your name project_number_lastname_firstname.pdf.
- Submit your .zip/.tgz through Icorsi.