Università della Svizzera italiana

Institute of Computing
CI

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

Student: ZITIAN WANG Discussed with: NULL

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

In addition to the steps in the textbook, need to add one more step

```
1    % sudo xattr -d com.apple.quarantine
```

Listing 1: Remove Apple Quarantine Attribute

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

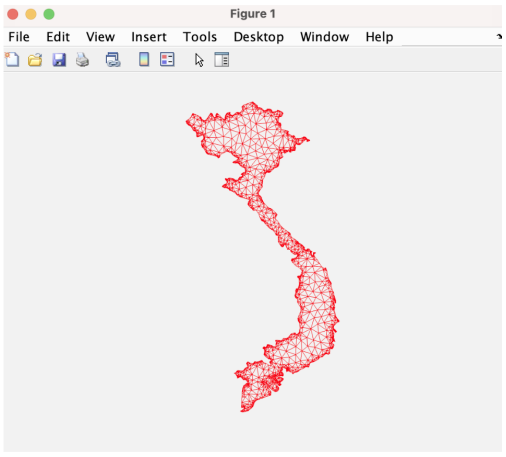
Run the Matlab script `src/read_csv_graphs.m` and complete the missing sections of the code. Your goal is to

- read the .csv files in MATLAB,
- 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, and

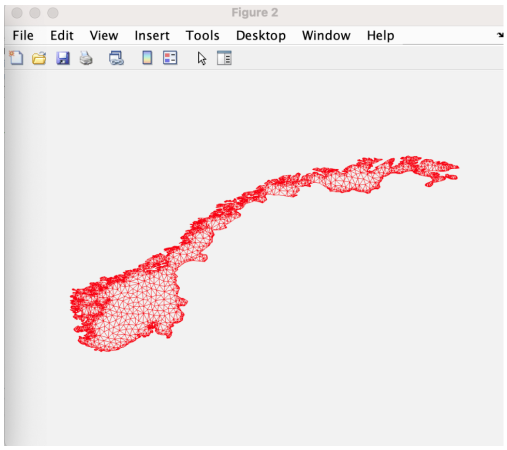
- visualize the graphs using the function `src/Visualization/gplotg.m`

```
    myDir = '../Datasets/Countries_Meshes/csv';
csvFiles_adj = dir(fullfile(myDir, '*adj.csv'));
csvFiles_pts = dir(fullfile(myDir, '*pts.csv'));

assert(length(csvFiles_adj) == length(csvFiles_pts), 'Adjacency and Points files mismatch');

numFiles = length(csvFiles_adj);
data_struct(numFiles) = struct('name', [], 'W', [], 'C', []);

for i = 1:numFiles
    adjFile = fullfile(myDir, csvFiles_adj(i).name);
    ptsFile = fullfile(myDir, csvFiles_pts(i).name);
    w_data = csvread(adjFile, 1, 0);
    c_data = csvread(ptsFile, 1, 0);

    data_struct(i).name = extractBefore(csvFiles_pts(i).name, '_pts.csv');
    data_struct(i).C = c_data;

    n = max(w_data, [], 'all');
    edges = ones(size(w_data, 1), 1);
    W = accumarray(w_data, edges, [n, n]);

    W = (W + W') / 2;
    W = sparse(W);
    data_struct(i).W = W;
end
```

Listing 2: Processing CSV Files into a Data Structure



(a) Visualization of Vietnam



(b) Visualization of Norway

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

- Run in Matlab the script `Bench_bisection.m` and familiarize yourself with the Matlab codes in the directory `Part_Toolbox`. An overview of all functions and scripts is offered in `Contents.m`.
- Implement **spectral graph bisection** based on the entries of the Fiedler eigenvector. Use the incomplete Matlab file `bisection_spectral.m` for your solution.

- Implement **inertial graph bisection**. For a graph with 2D coordinates, this inertial bisection constructs a line such that half the nodes are on one side of the line, and half are on the other. Use the incomplete Matlab file `bisection_inertial.m` for your solution.
- Report the bisection edgecut for all toy meshes that are either generated or loaded in the script
"Bench_bisection.m." Use Table 1 to report these results.

Table 1: Bisection results

Mesh	Coordinate	Metis 5.0.2	Spectral	Inertial
mesh1e1	18			
mesh2e1	37			
netz4504_dual				
stufe				

```
    [n,~] = size(xy);
    D_diag = diag(sum(A));
    L = D_diag - A;

    opts.sigma = 1e-10;
    opts.tol = 1e-10;
    [Vecs, Vals] = eigs(L, 2, 'smallestabs', opts);

    u2 = Vecs(:, 2);
    split = median(u2);

    a = find(u2 < split);
    b = find(u2 > split);
    c = find(u2 == split);

    nc = length(c);
    if nc
        na = length(a);
        nca = min(max(ceil(n/2) - na, 0), nc);
        a = [a; c(1:nca)];
        b = [b; c(nca+1:end)];
    end

    part1 = a';
    part2 = b';
```

Listing 3: Spectral Clustering and Bipartition

```
    x_mean = mean(xy(:,1));
    y_mean = mean(xy(:,2));

    S = cov(xy);
    M = [S(2,2), S(1,2);   % S_xx and S_yy
         S(1,2), S(1,1)];  % S_xy

    [V, D] = eig(M);
    [~, ind] = min(diag(D));
    u = V(:,ind);
    u = [-u(1); u(2)];

    [part1, part2] = partition(xy, u);
```

Listing 4: Computing Principal Direction and Partitioning Data

The result of the bisection edgecut as following

Table 2: Bisection results

Mesh	**Coordinate**	**Metis 5.0.2**	**Spectral**	**Inertial**
mesh1e1	18	17	18	19
mesh2e1	37	37	35	47
mesh3e1	19	19	20	19
mesh3e5	19	19	24	19
airfoil1	94	77	132	94
netz4504_dual	25	23	23	30
stufe	16	16	16	16
3elt	172	124	117	209
barth4	206	97	127	194
ukerbe1	32	27	36	28
crack	353	201	233	377

4. Task: Recursively bisecting meshes [15 points]

The recursive bisecion algorithm is implemented in the file `rec_bisection.m` of the toolbox. Utilize this function within the script `Bench_rec_bisection.m` to recursively bisect the finite element meshes loaded within the script in 8 and 16 subgraphs. Use your inertial and spectral partitioning implementations, as well as the coordinate partitioning and the METIS bisection routine. Summarize your results in 3. Finally, visualize the results for $p = 16$ for the case "crack".

Table 3: Edge-cut results for recursive bi-partitioning.

Case	Spectral	Metis 5.0.2	Coordinate	Inertial
mesh3e1				
airfoil1				
3elt				
barth4				
crack				

The results of the edge cut are shown as following

Table 4: Edge-cut results for recursive 8-partitioning.

Case	**Spectral**	**Metis 5.0.2**	**Coordinate**	**Inertial**
airfoil1	398	320	516	578
netz4504_dual	111	110	127	123
stufe	128	107	123	136
3elt	469	395	733	880
barth4	550	405	875	888
ukerbe1	128	128	225	281
crack	883	784	1343	1061

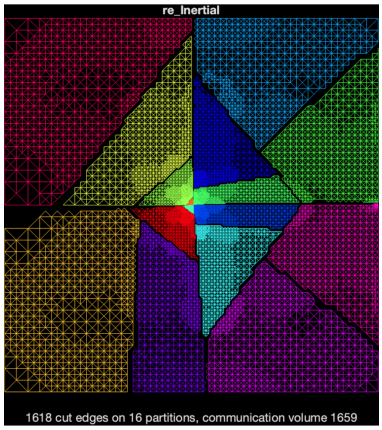
rec_bisection is a function that recursively partitions a graph. It requires specifying different partitioning methods. There are four types of bisection_spectral, bisection_metis, bisection_coordinate, and bisection_inertial. For each method, 8 and 16 partitions need to be considered. For the visualization part, following is an example

Table 5: Edge-cut results for recursive 16-partitioning.

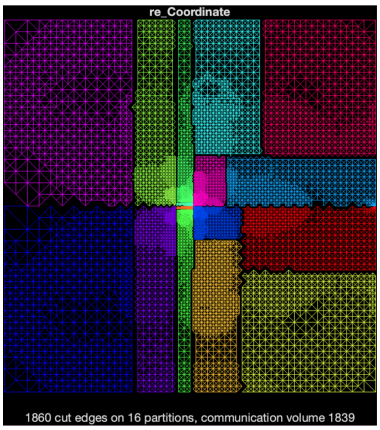
Case	Spectral	Metis 5.0.2	Coordinate	Inertial
airfoil1	633	563	819	903
netz4504_dual	184	161	198	202
stufe	243	194	227	267
3elt	752	651	1168	1342
barth4	841	689	1306	1348
ukerbe1	224	224	374	470
crack	1419	1290	1860	1618

```
[map_spe8, sepij_spe8, sepA_spe8] = rec_bisection('bisection_spectral'
    , 3, W, coords, 0);
```

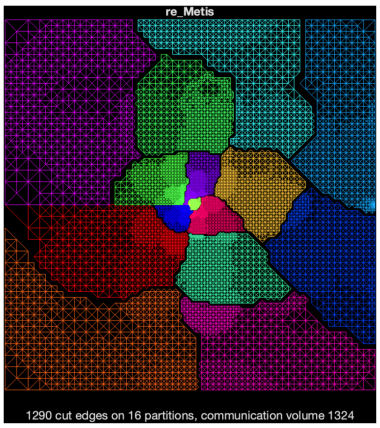
Listing 5: Recursive Bisection with Spectral Partitioning



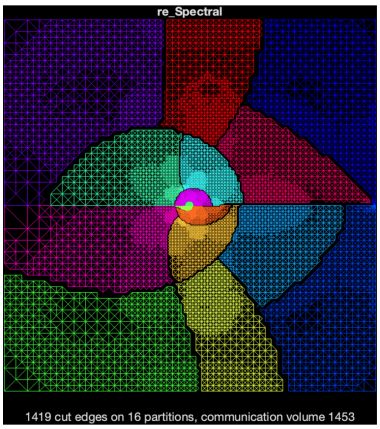
(a) Visualization of inertial method



(b) Visualization of coordinate method



(a) Visualization of metis method



(b) Visualization of spectral method

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

Use the incomplete `Bench_metis.m` for your implementation. Compare the cut obtained from Metis 5.0.2 after applying recursive bisection and direct multiway partitioning for the graphs in question.

Consult the Metis manual, and type `help metismex` in your MATLAB command line to familiarize yourself with the way the Metis recursive and direct multiway partitioning functionalities should be invoked. Summarize your results in Table 6 for 16 and 32 partitions. Comment on your results. Was this behavior anticipated? Visualize the partitioning results for both graphs for 32 partitions.

Table 6: 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							
32							

Table 7: The number of cut edges for recursive bisection

Partitions	**Luxemburg**	**usroads-48**	**Greece**	**Switzerland**	**Vietnam**	**Norway**	**Russia**
16	197	607	297	730	245	284	616
32	322	988	509	1089	445	470	1006

Table 8: The number of cut edges for direct multiway partitioning in Metis 5.0.2

Partitions	**Luxemburg**	**usroads-48**	**Greece**	**Switzerland**	**Vietnam**	**Norway**	**Russia**
16	170	579	278	673	245	255	551
32	279	961	471	1042	411	439	933

The behavior of the direct k-section method corresponds to what we anticipated, that its result is better than the recursive method.

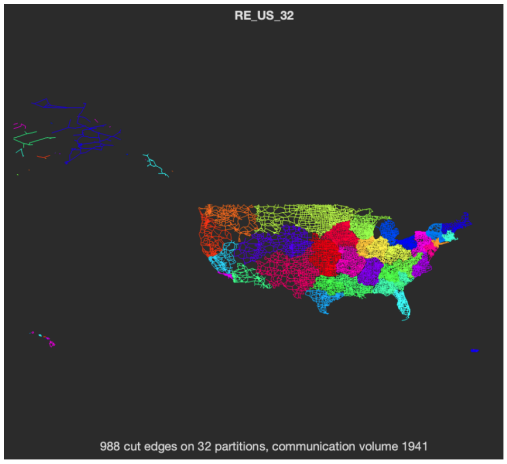
6. Task: Utilizing graph eigenvectors [25 points]

Provide the following illustrative results. Use the incomplete script `Bench_eigen_plot.m` for your implementation.

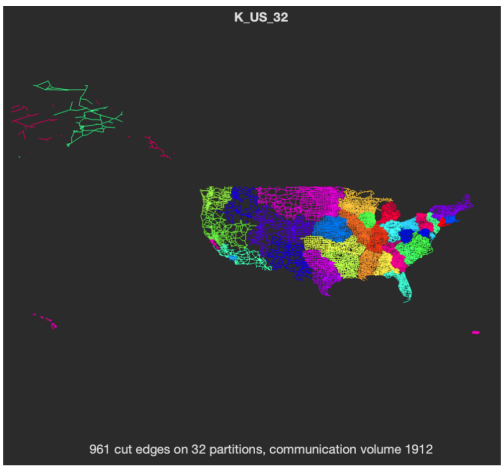
1. Plot the entries of the eigenvectors associated with the first (λ_1) and second (λ_2) smallest eigenvalues of the graph Laplacian matrix $\mathbf{L}$ for the graph "airfoil1." Comment on the visual result. Is this behavior expected?

2. Plot the entries of the eigenvector associated with the second smallest eigenvalue λ_2 of the Graph Laplacian matrix $\mathbf{L}$. Project each solution on the coordinate system space of the following graphs: mesh3e1, barth4, 3elt, crack. An example is shown in Figure 7, for the graph "airfoil1".
   Hint: You might have to modify the functions `gplotg.m` and `gplotpart.m` to get the desired result.

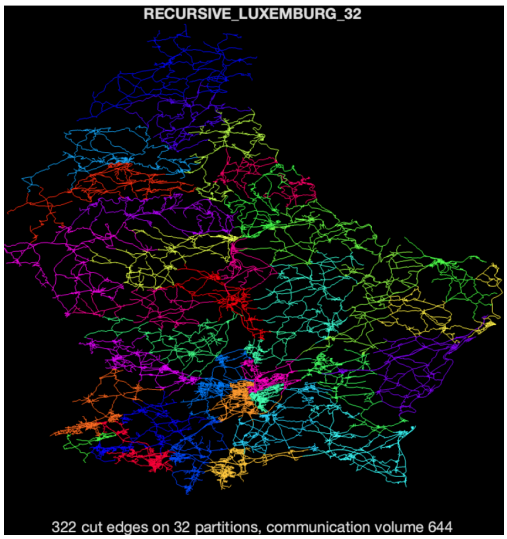
3. In this assignment we dealt exclusively with graphs $\mathcal{G}(V, E)$ that have coordinates associated with their nodes. This is, however, most commonly not the case when dealing with graphs, as they are in fact abstract structures, used for describing the relation E over a collection of entities V. These entities very often cannot be described in a Euclidean coordinate space. Therefore graph drawing is a tool to visualize relational information between nodes. The optimality of graph drawing is measured in terms of computation speed the ultimate usefulness of the resulting layout [1]. A successful layout should transmit the clearly the desired message, e.g the subsets of a partitioned graph. We will now see a spectral graph drawing method,



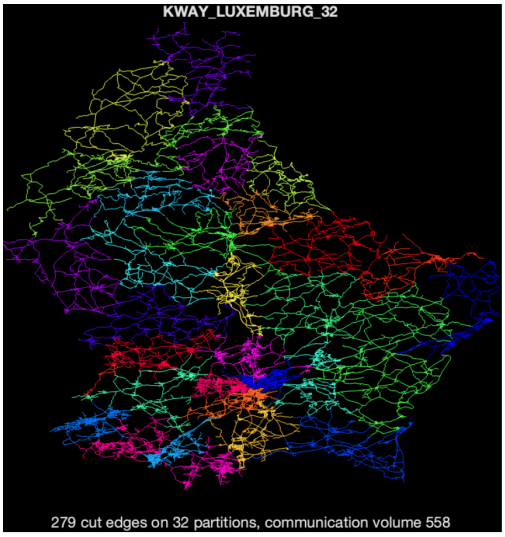
(a) Visualization of US-recursive



(b) Visualization of US-kway

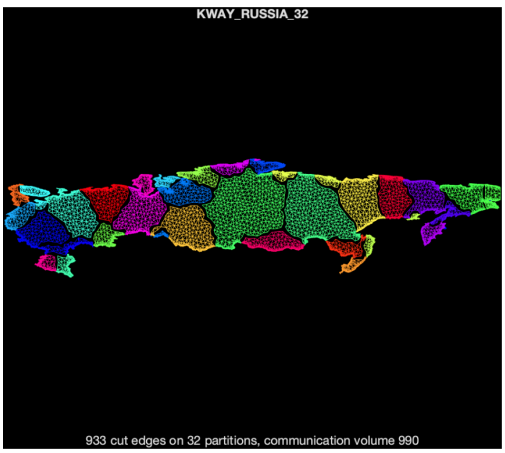


(a) Visualization of LU-recursive



(b) Visualization of LU-kway

(a) Visualization of RU-recursive



(b) Visualization of RU-kway

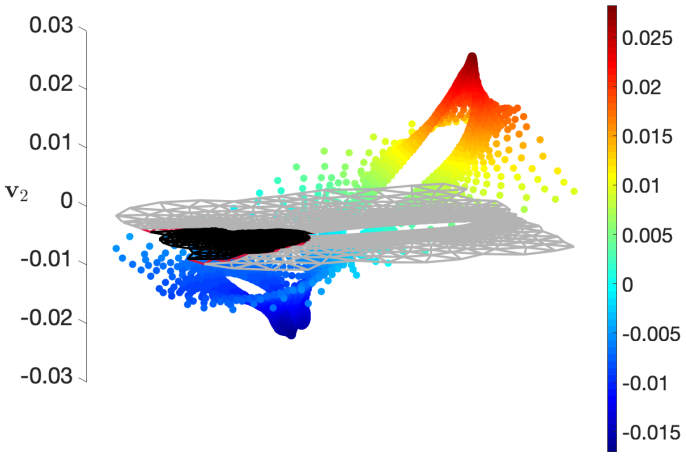


Figure 7: Partitioning the Airfoil graph based on the values of the Fiedler eigenvector. The two partitions are depicted in black and gray, while the cut edges in red respectively. The z-axis represents the value of the entries of the eigenvector.

which constructs the layout utilizing the eigenvectors of the graph Laplacian matrix $\mathbf{L}$. Draw the graphs mesh3e1, barth4, 3elt, crack, and their **spectral bi-partitioning** results using the eigenvectors to supply coordinates. Locate vertex i at position:

$$x_i = (\mathbf{v}_2(i), \mathbf{v}_3(i)),$$

where $\mathbf{v}_2, \mathbf{v}_3$ are the eigenvectors associated with the 2nd and 3rd smallest eigenvalues of $\mathbf{L}$. Figure 8 illustrates these 2 ways of visualizing the partitions of the "airfoil1" graph.

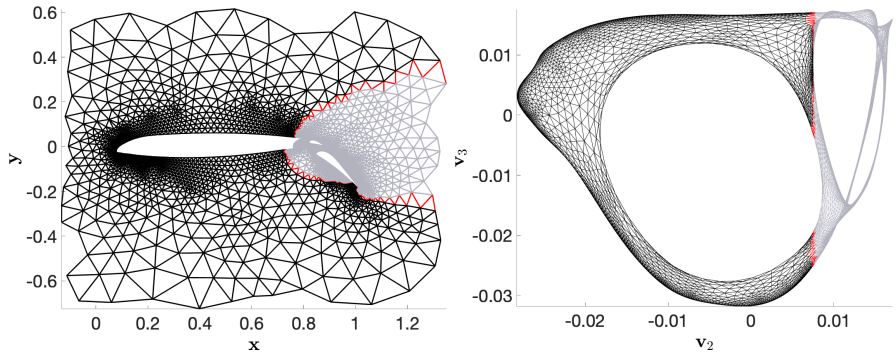
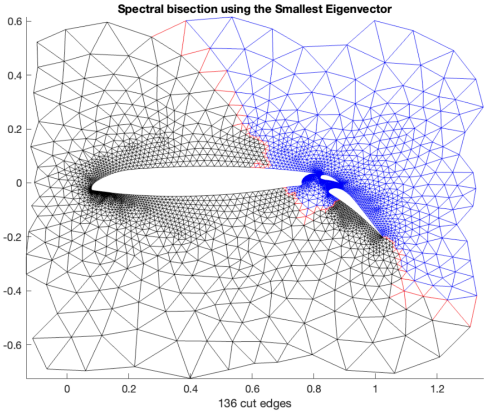


Figure 8: Visualizing the bipartitioning of the graph "airfoil1" with 4253 nodes and 12289 edges. Left: Spatial coordinates. Right: Spectral coordinates.

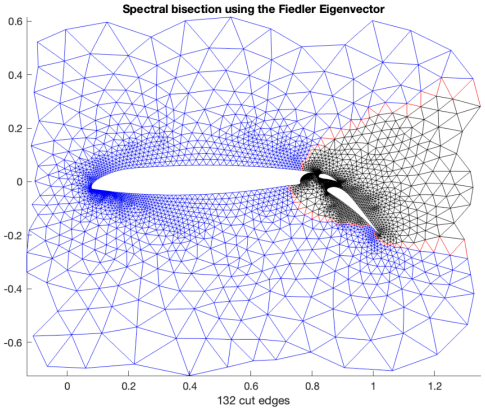
It is predicted that partitions based on the second smallest eigenvalue perform better than those based on the smallest eigenvalue.

References

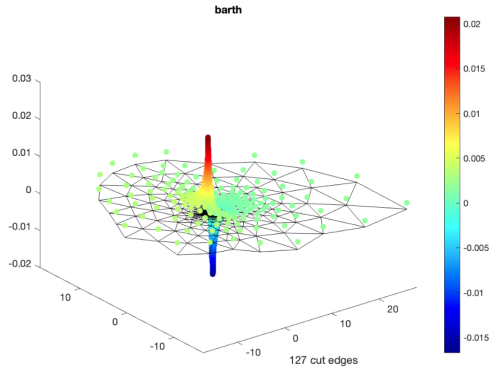
[1] Y. Koren. Drawing graphs by eigenvectors: Theory and practice. *Comput. Math. Appl.*, 49(11–12):1867–1888, June 2005.



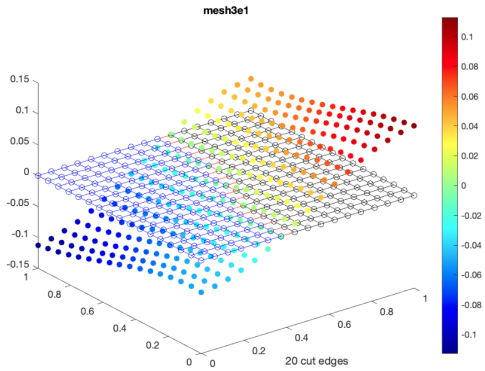
(a) Smallest eigenvalue



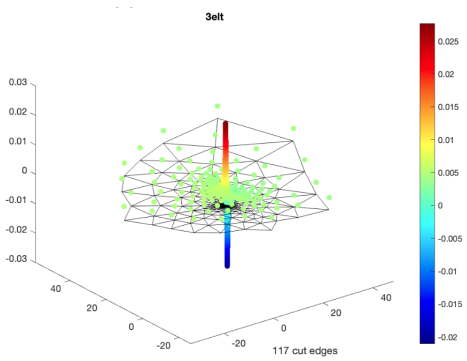
(b) Second smallest eigenvalue



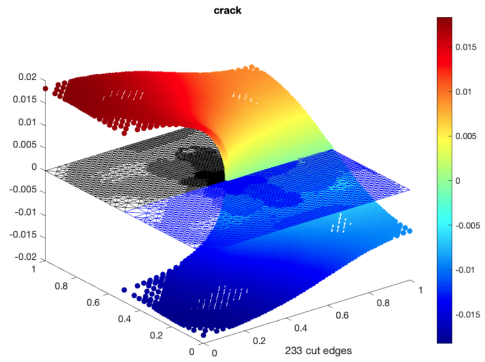
(a) barth



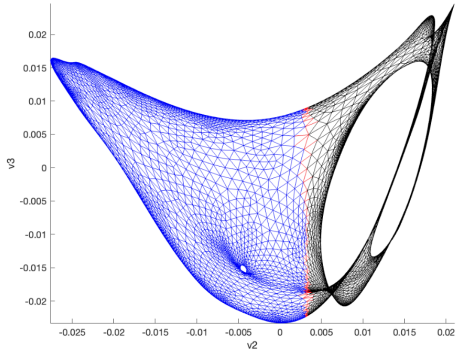
(b) mesh3e1



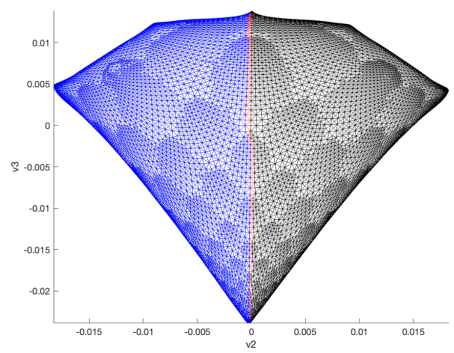
(a) 3elt



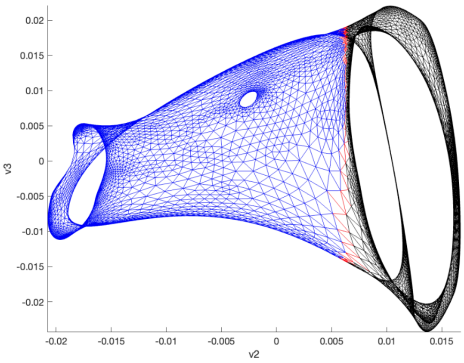
(b) crack



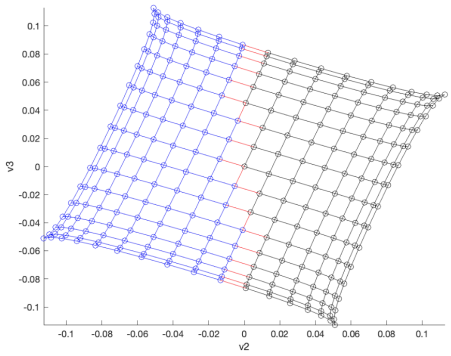
(a) bi-3elt



(b) bi-crack



(a) bi-barth



(b) bi-mesh3e1

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.