Università della Svizzera italiana | Faculty of Informatics | Institute of Computing CI

Numerical Computing **2023**

Student: Jeferson Morales Mariciano
Discussed with: Michele Dalle Rive, Martin Lettry

Solution for Project 2 **Due date:** Wednesday, 25 October 2023, 11:59 PM

Numerical Computing 2023 — 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. The assignment

Tests environment:

- Ubuntu 22.04.1
- Linux kernel 6.2.0-35 generic
- Matlab R2023b
- Metis version provided

1.1. Implement various graph partitioning algorithms [50 points]

The results are obtained by running 'Bench_bisection.m' after implementing spectral and inertial methods in correspondent 'bisection_spectral.m' and 'bisection_inertial.m' files.

For spectral implementaion, the threshold has been choosen to be 0, since it reports better results generally for our meshesh.

Table 1: Bisection results, cutsizes for each method

Mesh	Coordinate	Metis 5.0.2	Spectral	Inertial
grid5rec(12,100)	12	12	12	12
grid5rec(100,12)	12	12	12	12
grid5recRotate(100,12,-45)	22	12	12	12
gridt(50)	72	82	70	72
grid9(40)	118	127	128	118
Smallmesh	25	12	12	30
Tapir	55	23	18	49
Eppstein	42	41	42	45

Table 2: Bisection timings for each method, matlab 'timeit()' function

Mesh	Coordinate	Metis 5.0.2	Spectral	Inertial
grid5rec(12,100)	0.0015	0.0013	0.024	3.0764e-04
grid5rec(100,12)	0.0014	0.0017	0.0261	3.1754e-04
grid5recRotate(100,12,-45)	0.0019	0.0017	0.0256	3.1954e-04
gridt(50)	0.0020	0.0019	0.0537	3.9554e-04
grid9(40)	0.0019	0.0028	0.0635	3.5784e-04
Smallmesh	3.1284e-04	3.3817e-04	0.0084	2.5849e-04
Tapir	0.0020	0.0020	0.0300	3.2250e-04
Eppstein	0.0012	0.0017	0.0200	3.3750e-04

With the provided Table 1 it's evident that all methods produce identical results for some test cases (e.g., grid5rec(12,100), grid5rec(100,12), and others), reinforcing the idea that Coordinate, Metis, and Spectral may not significantly differ in these specific scenarios. Nevertheless, it's essential to consider the computational costs and accuracy trade-offs when choosing a graph partitioning methodology, as accuracy and performance can vary substantially across different cases.

Confronting with Table 2 timings, it has been observed that Inertial is the quickest of the methods even though the one producing most edgecuts generally, hence worse quality paritioning. Spectral method is the one with fewest edgecuts and also the method taking the longest time to compute the partitioning since its computational task is to calculate eigenvectors and eigenvalues for large mesh

matrices. Coordinate and Metis in terms of timing are generally similar in timing and edgecuts, with Metis being slightly slower and producing more edgecuts.

1.2. Recursively bisecting meshes [20 points]

The results are obtained by running 'Bench_rec_bisection.m'.

Table 3: Edge-cut results for recursive bi-partitioning with $p = 8$.

Case	Spectral	Metis 5.0.2	Coordinate	Inertial
mesh3e1	51	57	63	59
bodyy4	1000	985	1065	1364
de-2010	809	491	929	1084
biplane-9	411	465	548	648
L-9	718	637	631	828

Table 4: Edge-cut results for recursive bi-partitioning with $p = 16$.

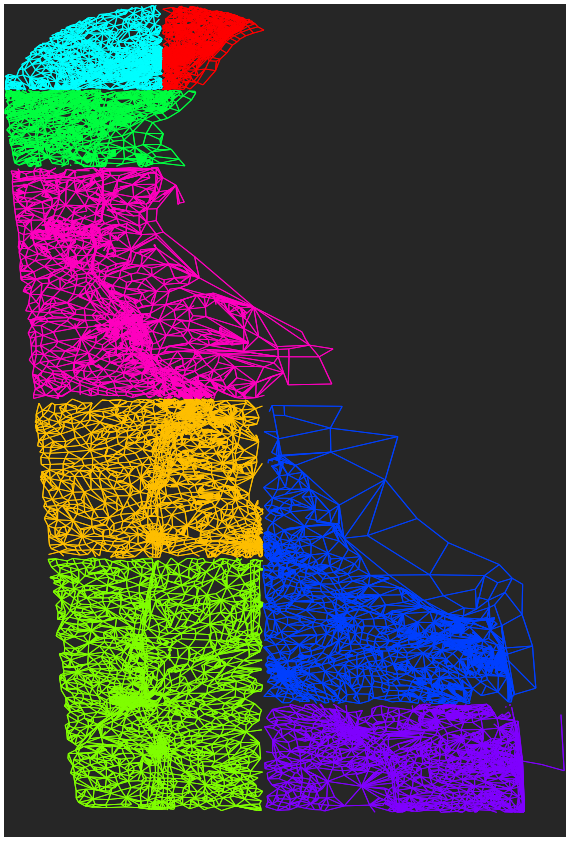
Case	Spectral	Metis 5.0.2	Coordinate	Inertial
mesh3e1	61	57	63	59
bodyy4	1675	1591	1951	2214
de-2010	1512	897	1796	2002
biplane-9	794	845	974	1093
L-9	1121	1019	1028	1377

By applying recursive partitioning algorithms, the graph is repeatedly divided into two additional subgraphs every time. Such approach has a clear advantage when aiming to parallelize and scale the computation of partitions and then the workload in a distributed system.

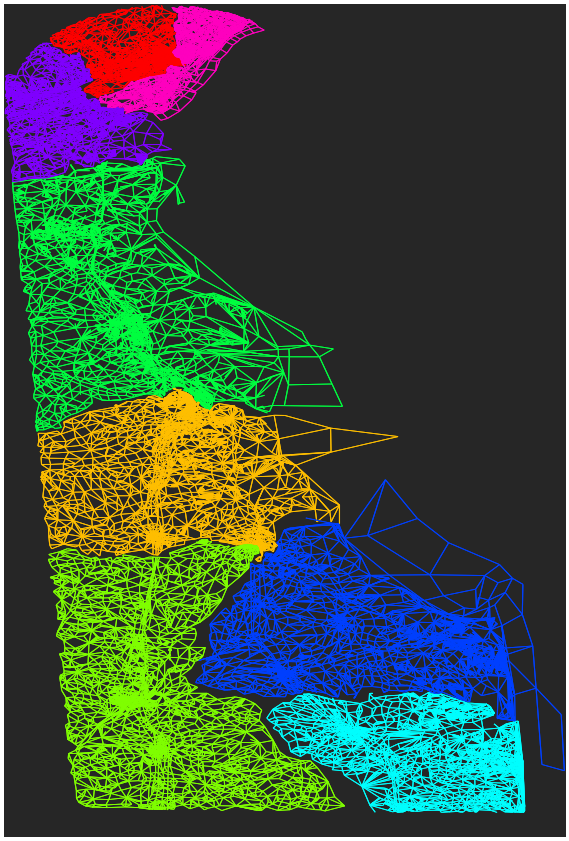
The number of subgraphs into which the graphs is splitted is 2^{levels}, where $levels$ represents the number of recursion levels. It's important to notice that the partitions generated at each recursion level have a fundamental impact on all subsequent recursions. As an example, small imbalances in partitions would get amplified at each recursion level.

During benchmarking, from tables 3, 4 it becomes evident that the spectral method suffers of performance degradation when transitioning to larger graphs and computing eigenvectors and eigenvalues. Even Metis and Coordinate suffers from such transition to larger graph, but the magnitude of the issue changes with respect to the case. Inertial method always show to have the most edgecuts, indicating lower performance, and it worse as the number of partitions increases. Coordinate show to be around the meadian of the number of edgecuts of all the other methods, since generally performing worse than spectral and metis, but not as magnitudinally worse as inertial.

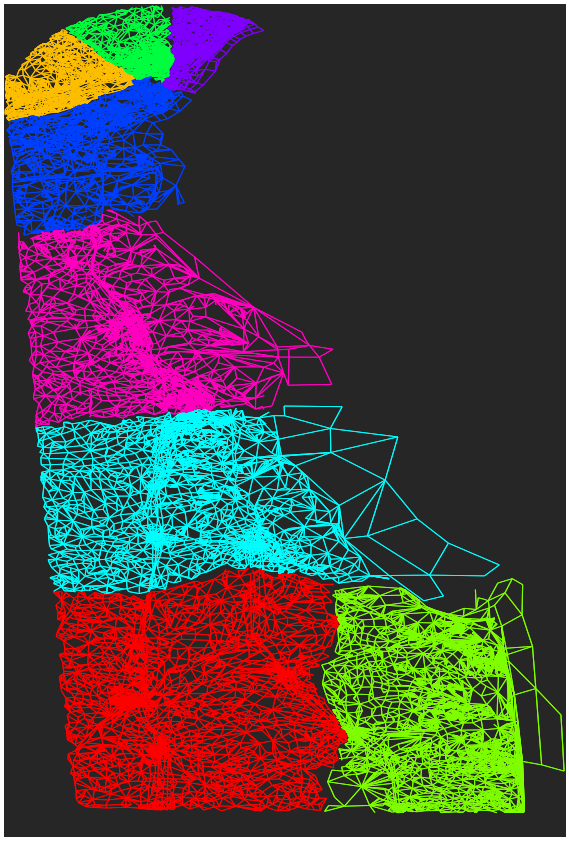
Visualization for case *"de-2010"* is provided for $p = 8$ and $p = 16$ in Figure 1 and 2 respectively.



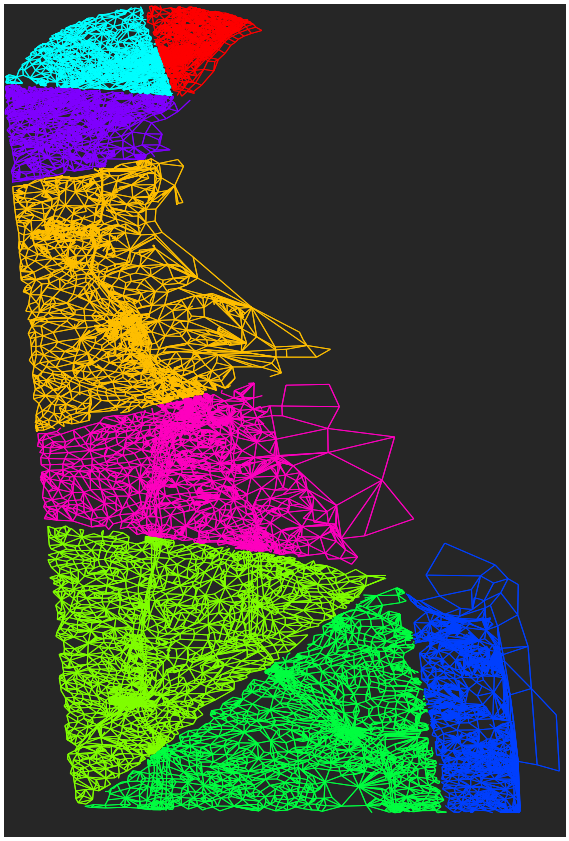
(a) Coordinate, 929 edgecuts



(b) Metis, 491 edgecuts

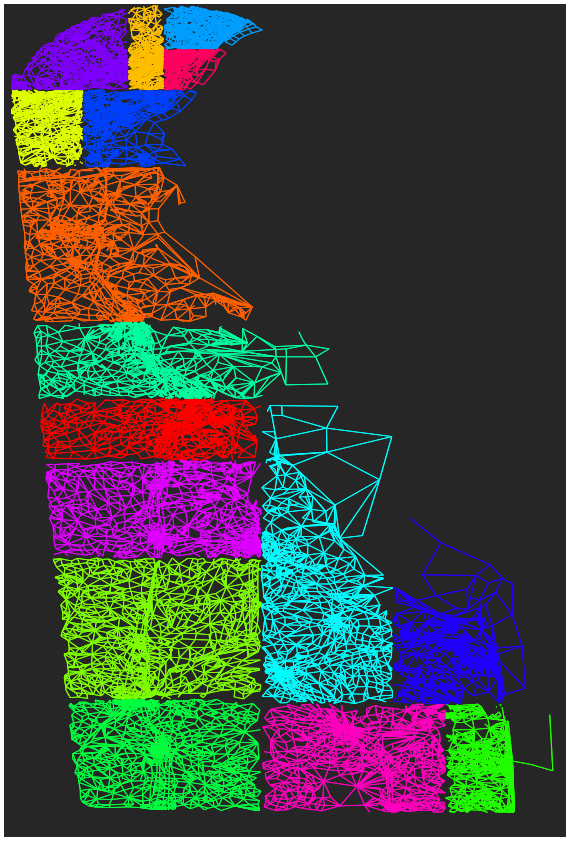


(c) Recursive bisection, 809 edgecuts

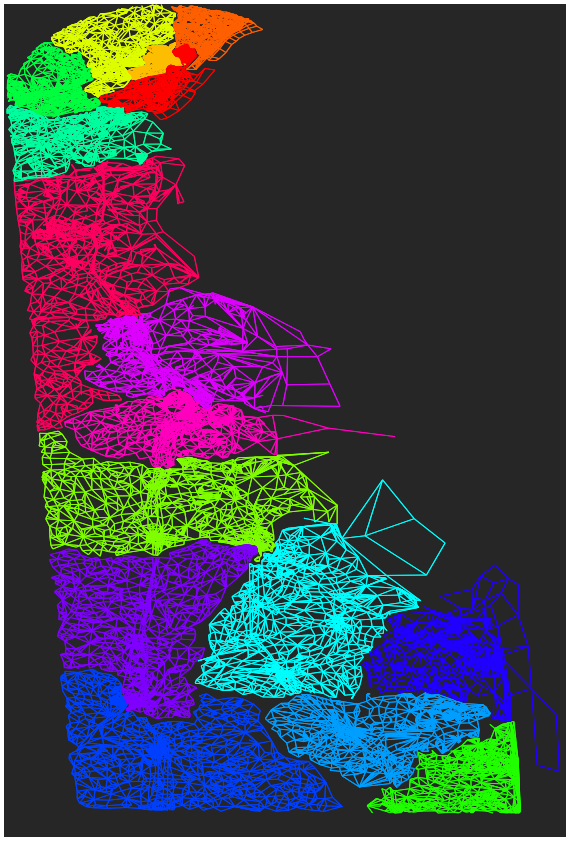


(d) K-way direct partitioning, 1084 edgecuts

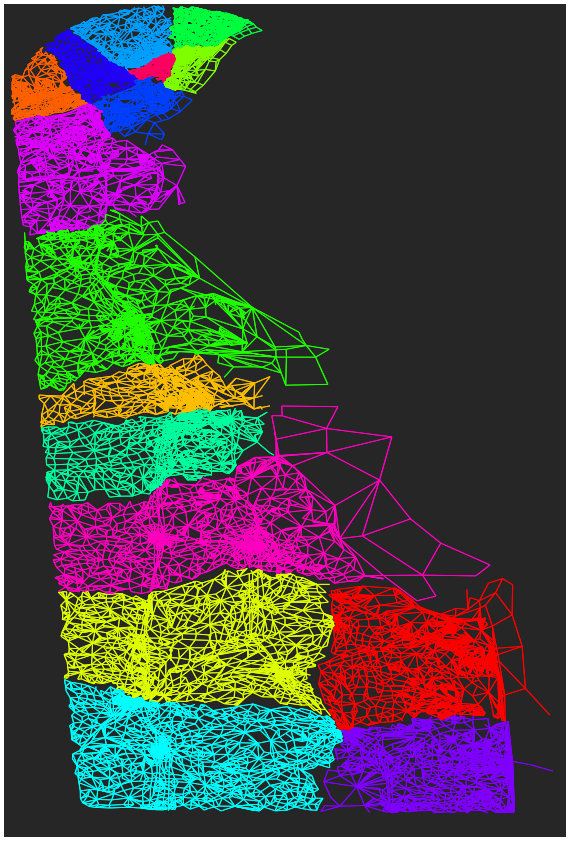
Figure 1: *de-2010* mesh - method confront with 8 partitions



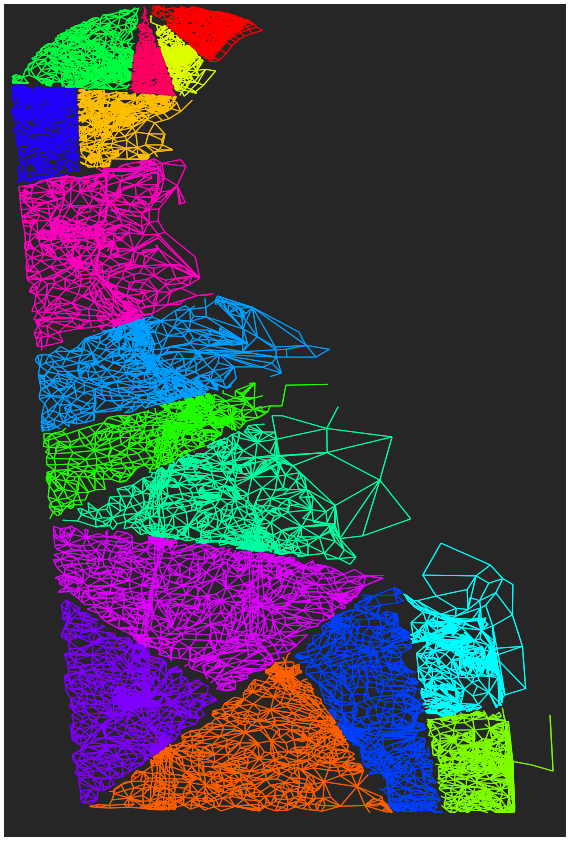
(a) Coordinate, 1796 edgecuts



(b) Metis, 897 edgecuts



(c) Recursive bisection, 1512 edgecuts



(d) K-way direct partitioning, 2002 edgecuts

Figure 2: *de-2010* mesh - method confront with 16 partitions

1.3. Comparing recursive bisection to direct k-way partitioning [15 points]

The results are obtained by running 'Bench_metis.m'.

Table 5: Comparing edge cuts number for recursive bisection and direct multiway partitioning in Metis 5.0.2.

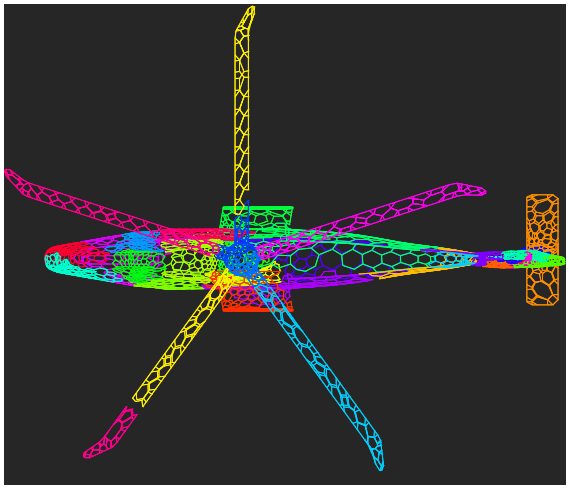
Partitions	Helicopter	Skirt
16 - recursive bisection	343	3119
16 - way direct partition	324	3393
32 - recursive bisection	537	6075
32 - way direct partition	539	6051

By looking at the data, we immediately see that the switch from 16 to 32 of the same graph almost double the amount of cut edges, regardless of the graph partitioning method used.

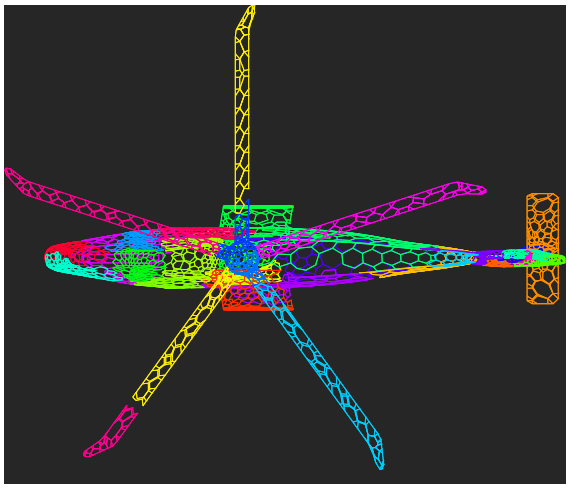
From table 5 we can see that neither recursive nor bisection method is better than the other: although there are cases where one method is better than the other, the overall total results show evenly distributed best result for both methods.

Recursive bisection is performing worst for Helicopter with 16 partitions, but notice that is performing best for skirt of the same number of partitions. Then, you would assume recursive is best for skirt and worst for helicopter, but for 32 partitions the results are inverted: recursive is best for helicopter and worst for skirt.

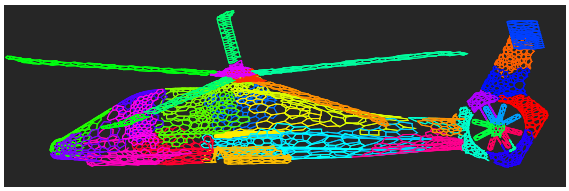
Figures of helicopter and skirt with 32 partitions are visualized for both methods using 'rotate3d' Matlab figure property.



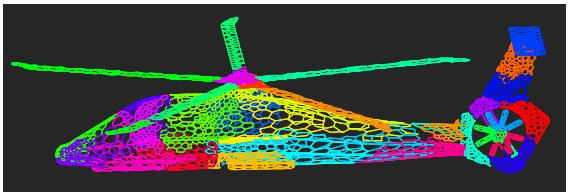
(a) Recursive bisection front view



(b) K-way direct partitioning front view

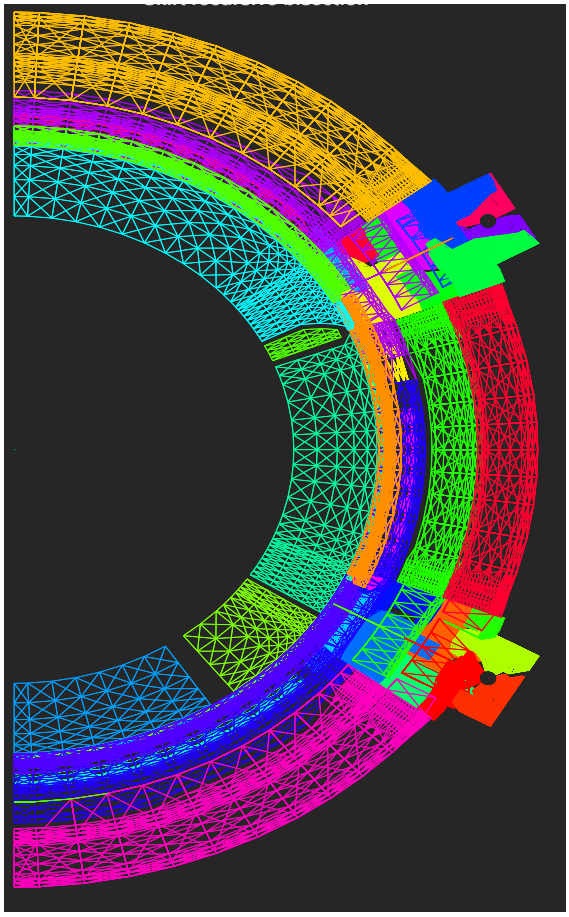


(c) Recursive bisection side view

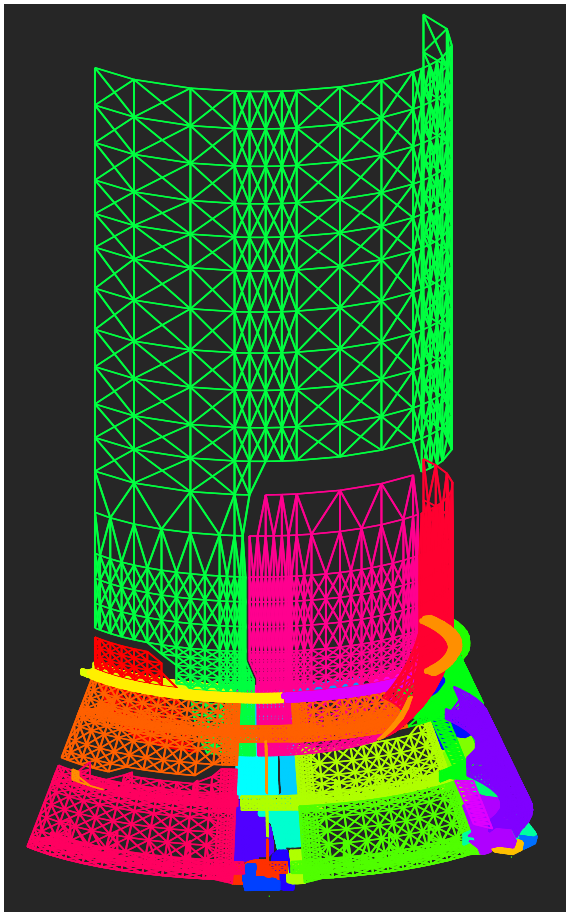


(d) K-way direct partitioning side view

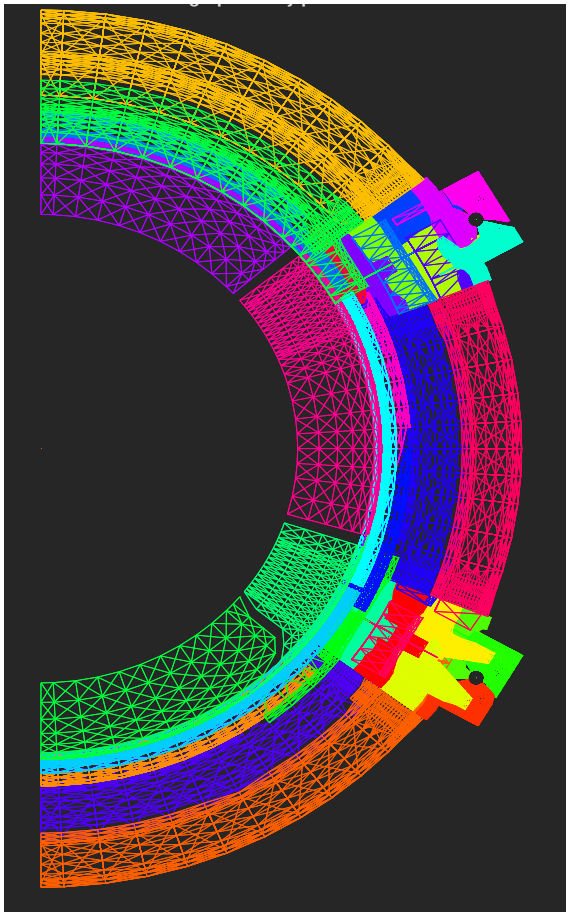
Figure 3: *commanche_dual* mesh - method confront with 32 partitions



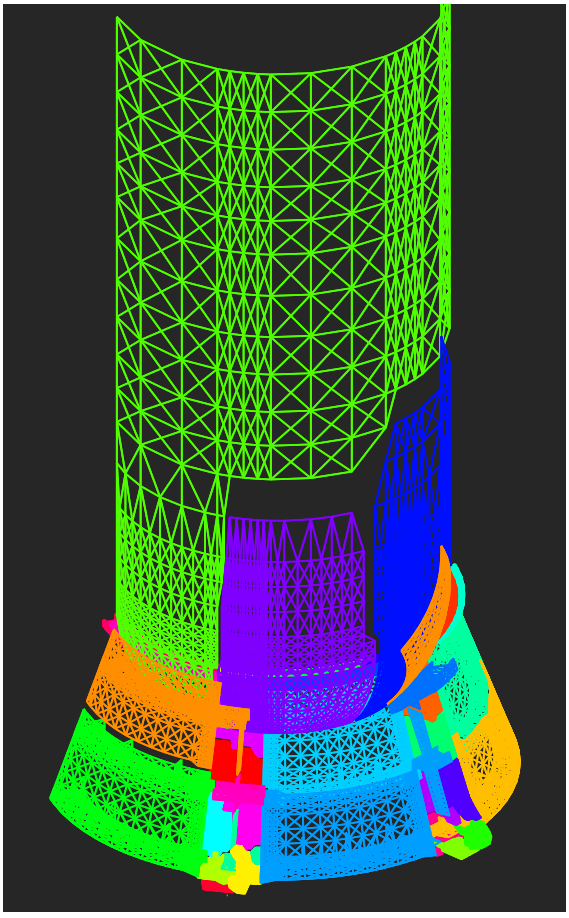
(a) Recursive bisection front view



(b) Recursive bisection side view



(c) K-way direct partitioning front view



(d) K-way direct partitioning side view

Figure 4: *Skirt* mesh - method confront with 32 partitions

Extra: Windows 11 implementation

Initially for the assignment, my test environment was:

- Windows 11 x64 Pro Version 22H2 Build 22621.2428
- Matlab R2023b
- Metis 5.0.2
- CMake 3.28.0-rc2
- Visual Studio 17 2022 c++ compiler

Compiling process specifics are explained in this issued I commented on metismex github.

The reason to why I didn't finish my report with such environment is that of getting different results from the ones of my peers and TAs I talked with, and that in order to prevent any further issues I decided to switch to a Linux environment.

Implement various graph partitioning algorithms

From table 6, we notice how the Metis 5.0.2 version result changes giving 4 better results out of 6 graphs with respect to the precompiled GNU/Linux version provided by the professor. Some doubts arise about the Metis version provided by the professor being really the 5.0.2.

Coordinate and inertial methods remains the same while spectral implementation is worse in most cases.

Table 6: Win 11 - Bisection results

Mesh	Coordinate	Metis 5.0.2	Spectral	Inertial
grid5rec(12,100)	12	12	12	12
grid5rec(100,12)	12	14	12	12
grid5recRotate(100,12,-45)	22	14	12	12
gridt(50)	72	76	82	72
grid9(40)	118	122	136	118
Smallmesh	25	12	14	30
Tapir	55	22	58	49
Eppstein	42	41	47	45