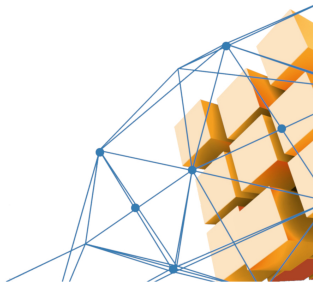
Politecnico di Torino
Dipartimento di Scienze Matematiche "G. L. Lagrange"
DISMA ECCELLENZA 2018•2022

Raffinamento Complesso

Candidati:
Matteo Racca 283581
Davide Omento 281464
Giorgio Musso 282313

Docenti:
Prof. S. Berrone
Prof. F. Vicini

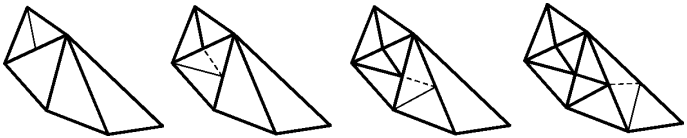
Programmazione e Calcolo Scientifico



Processo di Raffinamento

Un'iterazione del processo di raffinamento è composta da:

1. rintracciare il triangolo di area maggiore T;
2. trovare il lato e^T più lungo di T;
3. calcolare il punto medio M_{e^T} di e^T e collegarlo al vertice opposto $O^T_{e^T}$;
4. determinare il triangolo S adiacente al triangolo T attraverso e^T;
5. ripercorrere i punti (2) e (3) per il nuovo triangolo S;
6. unire i punti medi $O^T_{e^T}$ e $O^S_{e^S}$;
7. iterare il processo fino a ottenere una mesh ammissibile.



Struttura dati utilizzata

- 3 classi di oggetti:
  - **Vertices**: con attributi l'id, il marker e le coordinate x e y;
  - **Edges**: con attributi l'id, il marker, l'id del vertice iniziale e finale, la lunghezza, il parametro booleano inMesh e il metodo *MidPoint*;
  - **Triangles**: con attributi l'id, gli id dei vertici e dei lati, l'area, il parametro booleano inMesh e il metodo *FindMaxEdge*.
- 3 vettori contenenti i vertici, i lati e i triangoli della mesh.

Operatori e utilizzo tolleranze

Abbiamo creato diversi operatori utilizzati per:

- **output**: utili nella scrittura del file di output della mesh;
- **confronto**: usati per verificare uguaglianze o ordinamento degli oggetti delle classi (vengono utilizzate le tolleranze).

```
static constexpr double geometricTol = 1.0e-12;
static constexpr double geometricTol_Squared = max_tolerance(Triangles::geometricTol * Triangles::geometricTol,
                                                             numeric_limits<double>::epsilon());
```

```
inline bool operator>(const Edges& e1, const Edges& e2)
{
    return e1.length > e2.length + Edges::geometricTol * max(e1.length, e2.length);
}
```

```
inline bool operator>(const Triangles& t1, const Triangles& t2)
{
    return t1.area > t2.area + Triangles::geometricTol_Squared * max(t1.area, t2.area);
}
```

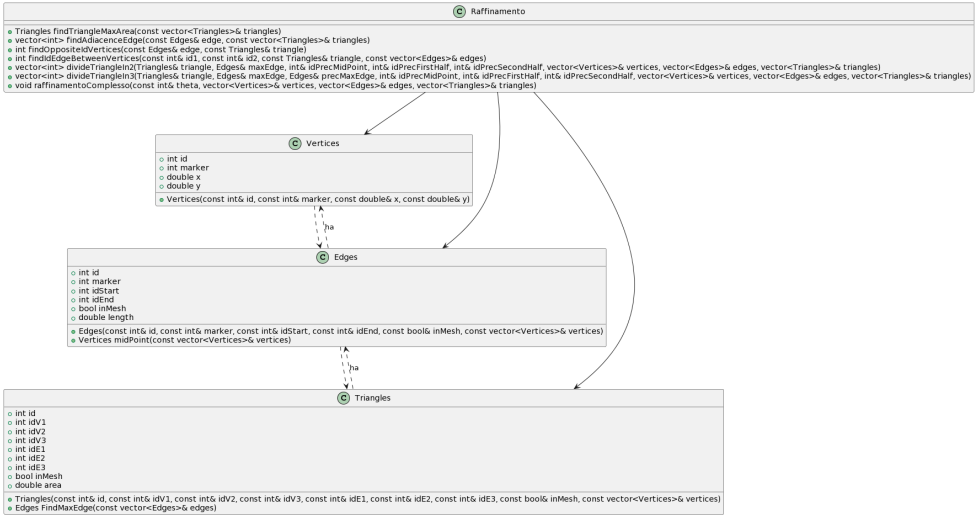
Funzioni di import

- **ImportVertices**:
  - input: const string& nameFile
  - output: vector<Vertices>
- **ImportEdges**:
  - input: const string& nameFile, const vector<Vertices>& vertices
  - output: vector<Edges>
- **ImportTriangles**:
  - input: const string& nameFile, const vector<Vertices>& vertices
  - output: vector<Triangles>

Osservazione: const vector<Vertices>& vertices

Utilizzato nel costruttore di elementi Edges e Triangles per calcolare rispettivamente lunghezza e area.

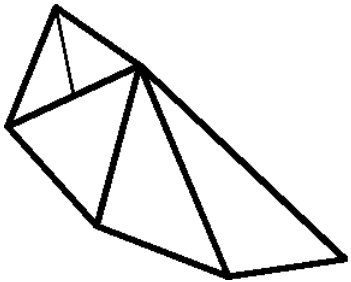
Visualizzazione UML



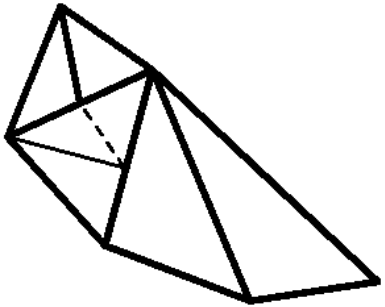
Funzioni ausiliarie

- **findAdiacenceEdge**:
  - input: const Edges& edge, const vector<Triangles>& triangles
  - output: vector<int>
- **findOppositeIdVertices**:
  - input: const Edges& edge, const Triangles& triangle
  - output: int
- **findIdEdgeBetweenVertices**:
  - input: const int& id1, const int& id2, const Triangles& triangle, const vector<Edges>& edges
  - output: int
- **findTriangleMaxArea**:
  - input: const vector<Triangles>& triangles
  - output: Triangles

- **divideTriangleIn2**:
  - input:
    Triangles& triangle,
    Edges& maxEdge,
    int& idPrecMidPoint,
    int& idPrecFirstHalf,
    int& idPrecSecondHalf,
    vector<Vertices>& vertices,
    vector<Edges>& edges,
    vector<Triangles>& triangles
  - output:
    vector<int>



- **divideTriangleIn3**:
  - input:
    Triangles& triangle,
    Edges& maxEdge,
    Edges& precMaxEdge,
    int& idPrecMidPoint,
    int& idPrecFirstHalf,
    int& idPrecSecondHalf,
    vector<Vertices>& vertices,
    vector<Edges>& edges,
    vector<Triangles>& triangles
  - output:
    vector<int>

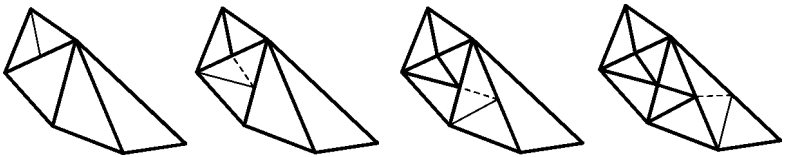


Osservazione: Test

Per verificare la corretta implementazione delle funzioni e dei metodi delle classi abbiamo creato un test per ognuna di esse.

Funzione principale

- **raffinamentoComplesso**:
  - input: const int& theta, vector<Vertices>& vertices, vector<Edges>& edges, vector<Triangles>& triangles
  - output: void



Osservazione: void

La funzione esegue diverse operazioni di aggiornamento dei vettori in input.

Costi computazionali

- **ImportVertices, ImportEdges, ImportTriangles**: $\mathcal{O}(n)$
- **findAdiacenceEdge**: $\mathcal{O}(t)$
- **findOppositeIdVertices**: $\mathcal{O}(1)$
- **findIdEdgeBetweenVertices**: $\mathcal{O}(1)$
- **findTriangleMaxArea**: $\mathcal{O}(t)$
- **divideTriangleIn2**: $\mathcal{O}(t)$
- **divideTriangleIn3**: $\mathcal{O}(t)$
- **RaffinamentoComplesso**: $\mathcal{O}(t)$

dove con n si intende il numero di linee dei file .csv da cui estrarre i dati e con t il numero di triangoli nel vettore.

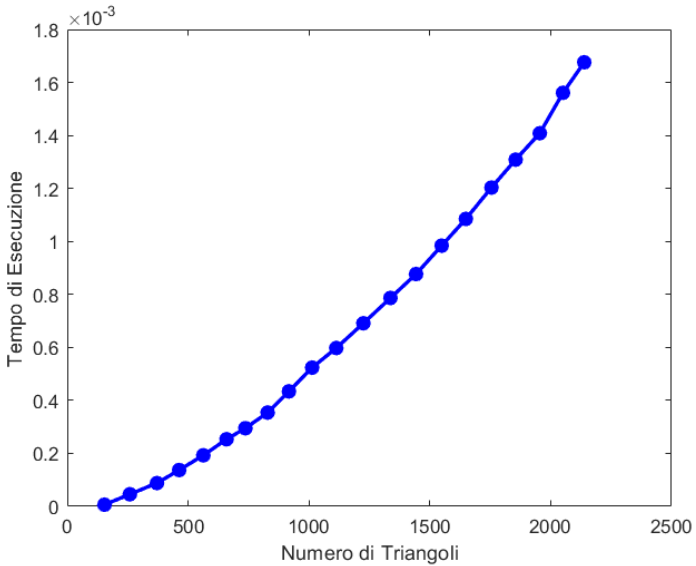


Figure: Tempo di esecuzione in funzione del numero di triangoli

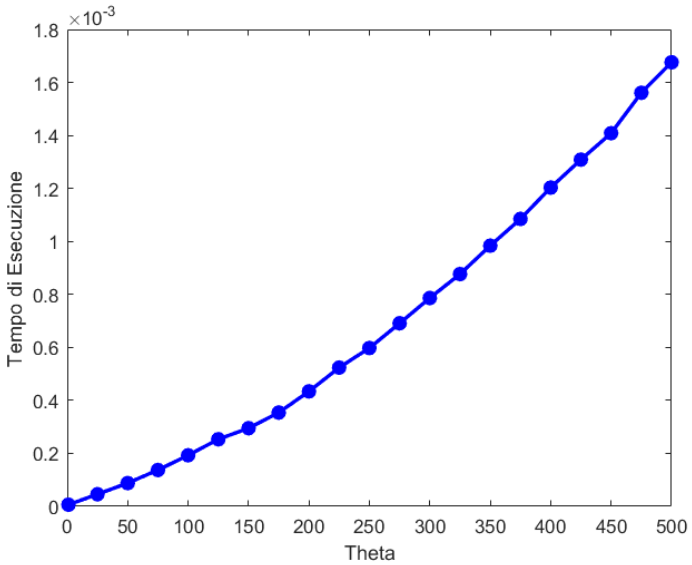


Figure: Tempo di esecuzione in funzione di θ

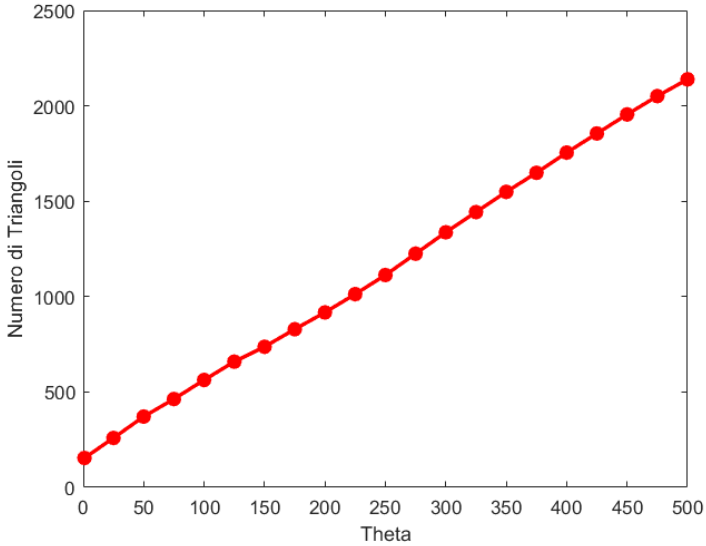


Figure: Numero di iterazioni in funzione di θ

Risultati ottenuti

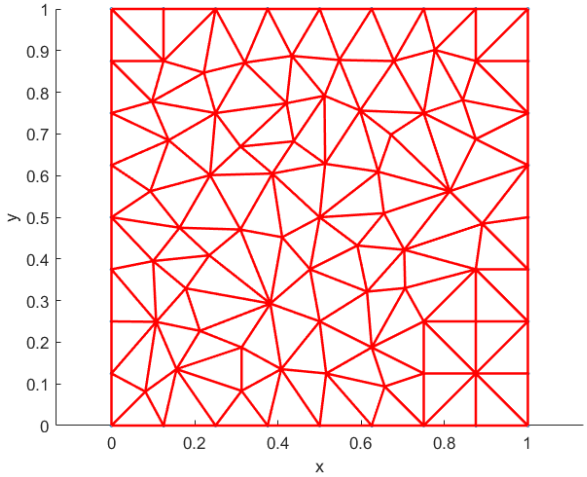


Figure: Mesh iniziale non raffinata

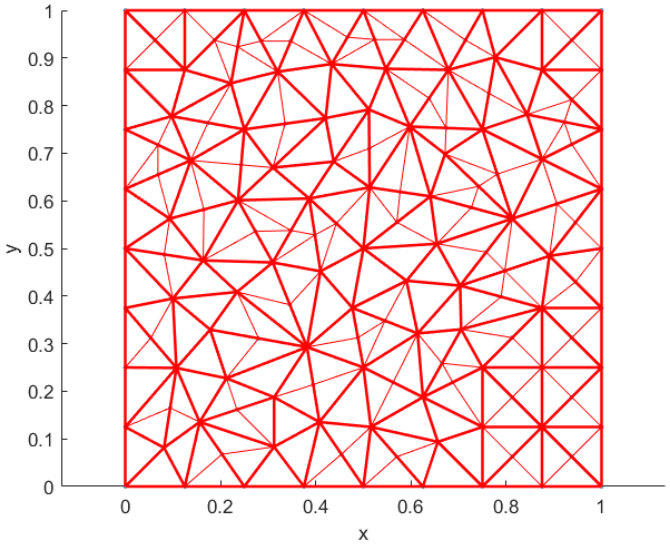


Figure: Mesh ottenuta con $\theta = 100$

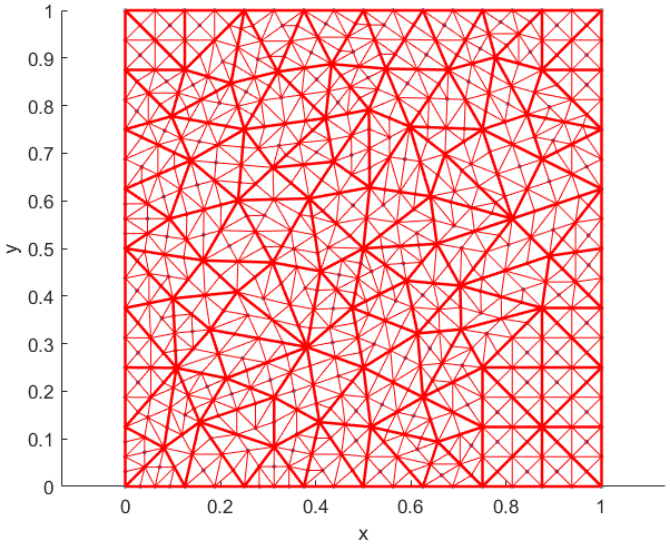


Figure: Mesh ottenuta con $\theta = 500$

Grazie per l'attenzione

Politecnico di Torino