Politecnico di Torino

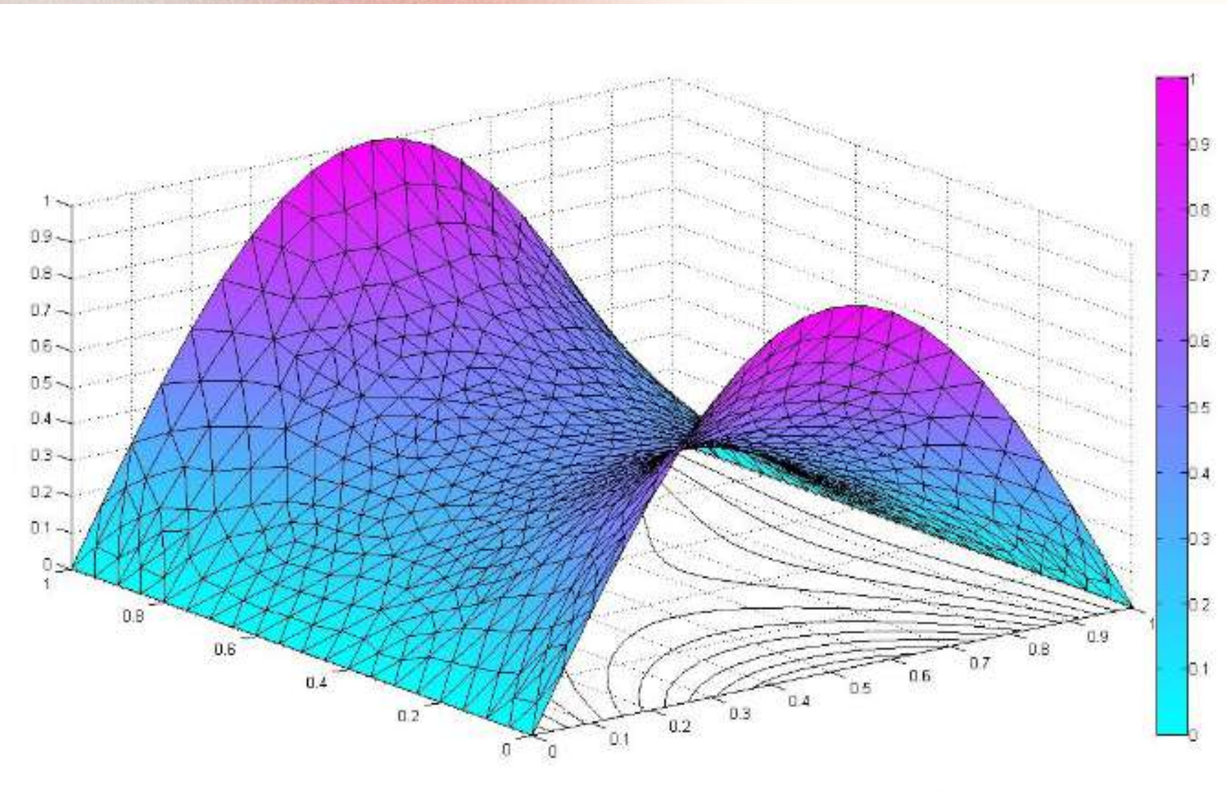
Raffinamento Complesso

Allegra Riola
Vincenzo Solimando
Gabriele Zambrano

Programmazione e Calcolo Scientifico-2022/2023
F. Vicini, S. Berrone, G. Teora

Corso di studi: Matematica Per L'Ingegneria

Applicazioni



Il raffinamento complesso é utilizzato in diversi ambiti, primo fra tutti quello della risoluzione di problemi differenziali.
Esso permette di ottenere risultati precisi e di qualità riducendo il margine d'errore.

Nota importante: il raffinamento garantisce una più o meno buona qualità della mesh finale in base al livello di qualità di quella iniziale

Qualità di una mesh: dipende da diversi criteri che dipendono dalle esigenze specifiche dell'applicazione.
Alcune caratteristiche generali sono: la coerenza topologica, l'uniformità, la bassa distorsione , la risoluzione , l'adattabilità dei suoi elementi

Descrizione

INPUT:

➢Vertici (Cell0D) :
marker , id , coordinate
➢Lati (Cell1D) :
marker , id , vertici
➢Triangoli (Cell2D) :
id , vertici , lati

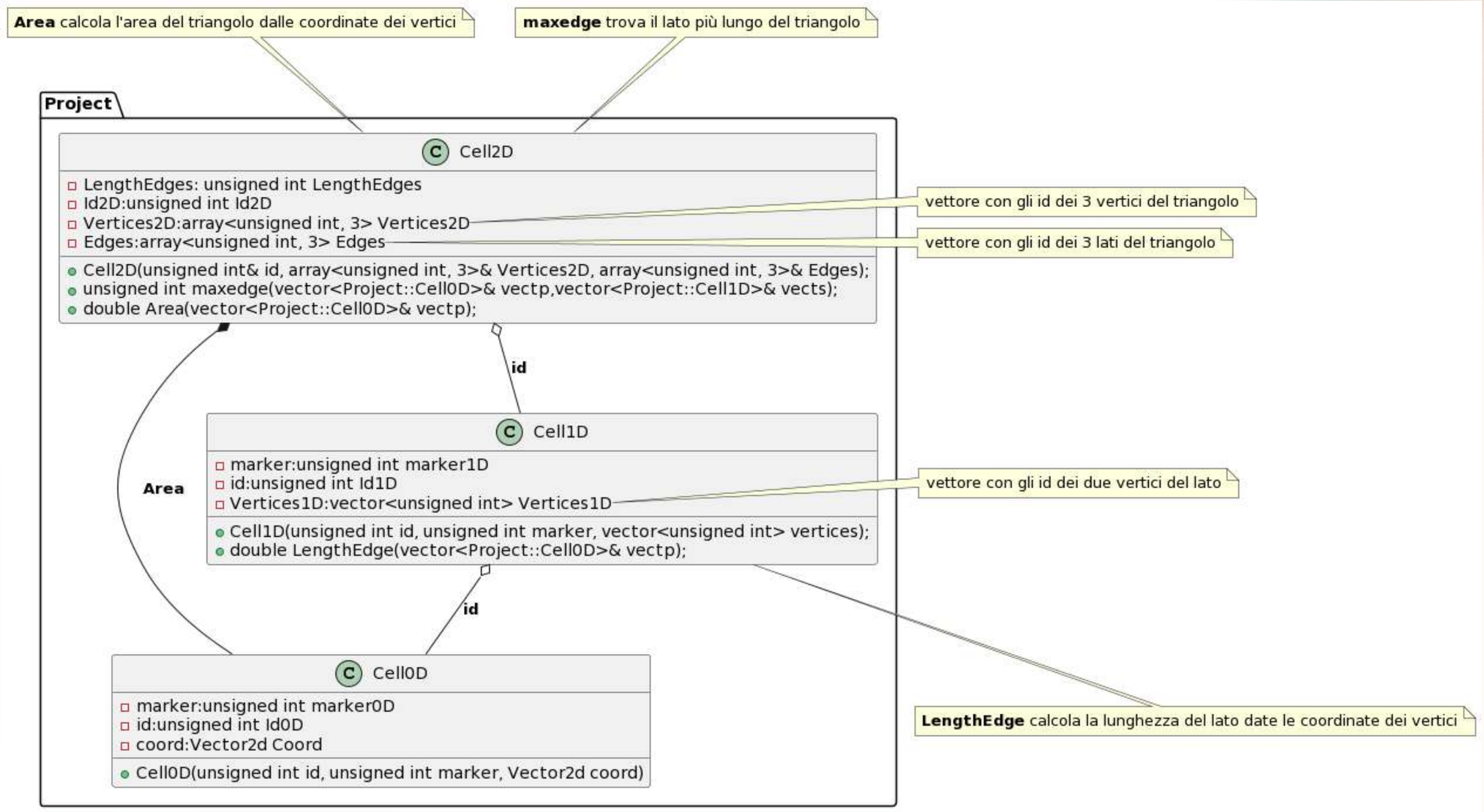
OBIETTIVO

A partire da una mesh triangolare, raffineremo un sottoinsieme di triangoli per ottenere una mesh più fine.

OUTPUT:

➢Vertici (Cell0D):
➢Lati (Cell1D) :
➢Triangoli (Cell2D) :

Diagramma di classe UML



Metodi delle classi

I metodi che abbiamo implementato sono i seguenti :

➪**LengthEdge**(vector <Project::Cell0D>& vectp) per il calcolo della lunghezza di un lato

➪**maxEdge**(vector <Project::Cell0D>& vectp,vector <Project::Cell1D>& vects) per trovare il lato più lungo in un triangolo

➪**Area**(vector <Project::Cell0D>& vectp) per il calcolo dell'area di un triangolo

Calcolo lunghezza di un lato

Per il calcolo della lunghezza abbiam usato la formula della distanza geometrica :

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} = \sqrt{(x_2 - x_1)^2 + (a - a)^2} = \sqrt{(x_2 - x_1)^2}$$

```
double Project::Cell1D::LengthEdge(vector<Project::Cell0D>& vectp){
    Vector2d coordOrigin = mesh.vectp[this->Vertices1D[0]].Coord;
    Vector2d coordEnd= mesh.vectp[this->Vertices1D[1]].Coord;
    //double len = (coordEnd-coordOrigin).norm();
    double len = sqrt(pow(coordOrigin.x() - coordEnd.x(), 2)+pow(coordOrigin.y() - coordEnd.y(), 2));
    return len;
    }
```

Ricerca del lato più lungo

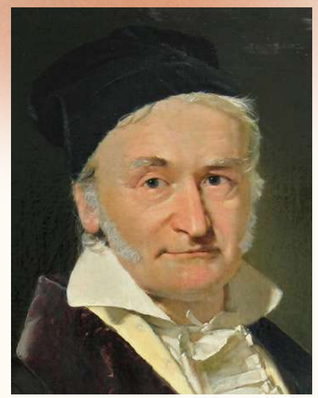
Per la ricerca del lato più lungo abbiamo effettuato dei confronti tra le lunghezze dei lati :

```
unsigned int Project::Cell2D::maxedge(vector<Project::Cell0D>& vectp,vector<Project::Cell1D>& vects){
    unsigned int indmax = 0;
    double max = mesh.vects[this->Edges[0]].LengthEdge();
    for (unsigned int i = 1; i<3; i++){
        if(mesh.vects[this->Edges[i]].LengthEdge()  > max - tol1D){
            max = mesh.vects[this->Edges[i]].LengthEdge();
            indmax = i;
        }
    }
    return this->Edges[indmax];
```

Calcolo dell'area

Per il calcolo dell'area abbiam usato la Formula di Gauss :

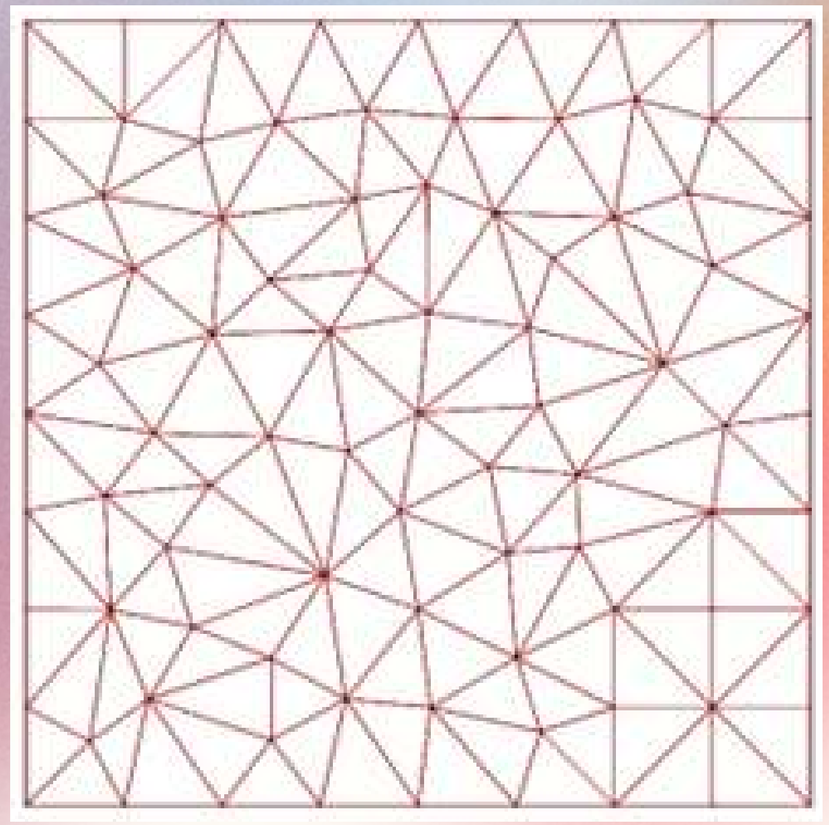
$$\frac{1}{2}\left|x_1y_2 + x_2y_3 + \ldots + x_{n-1}y_n + x_ny_1 - x_2y_1 - x_3y_2 - \ldots - x_ny_{n-1} - x_1\right|$$



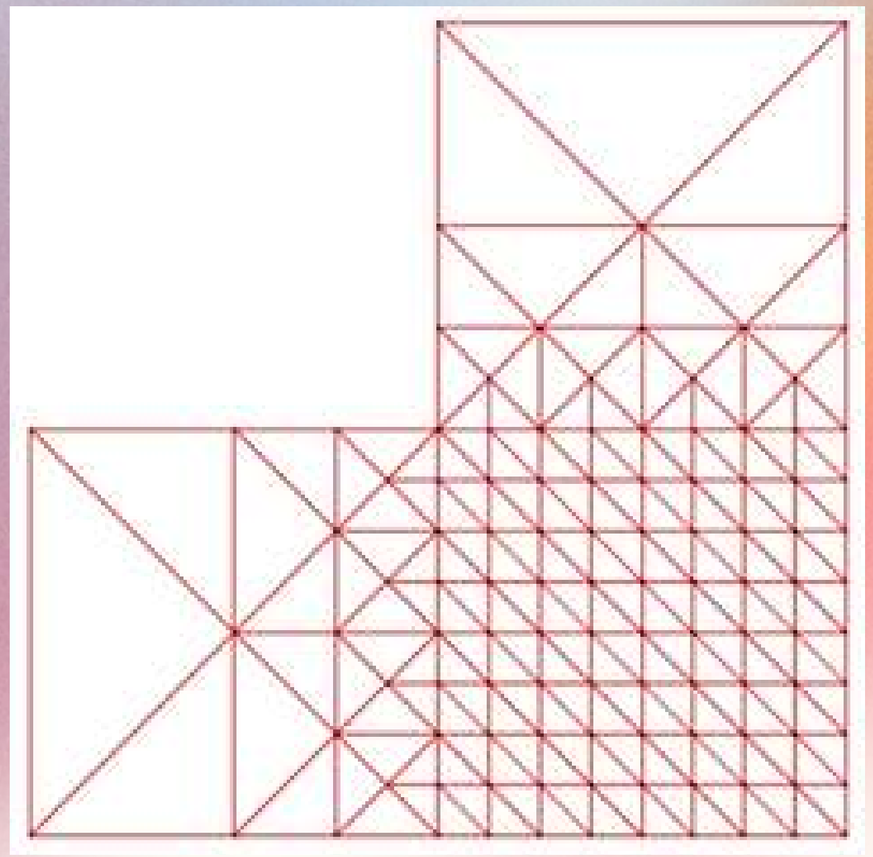
Abbiamo poi salvato gli id dei triangoli relativi alle aree calcolate in un vettore VectT

```
double Project::Cell2D::Area(vector<Project::Cell0D>& vectp){
            //Formula dell'area di Gauss
            double A_12 = (mesh.vectp[this->Vertices2D[0]].Coord[0]*mesh.vectp[this->Vertices2D[1]].Coord[1]) - (mesh.vectp[this->Vertices2D[0]].Coord[1]*mesh.vectp[this->Vertices2D[1]].Coord[0]);
            double A_23 = (mesh.vectp[this->Vertices2D[1]].Coord[0]*mesh.vectp[this->Vertices2D[2]].Coord[1]) - (mesh.vectp[this->Vertices2D[1]].Coord[1]*mesh.vectp[this->Vertices2D[2]].Coord[0]);
            double A_31 = (mesh.vectp[this->Vertices2D[2]].Coord[0]*mesh.vectp[this->Vertices2D[0]].Coord[1]) - (mesh.vectp[this->Vertices2D[2]].Coord[1]*mesh.vectp[this->Vertices2D[0]].Coord[0]);
            return abs((A_12+A_23+A_31)/2);
```

Mesh iniziale del test 1



Mesh iniziale del test 2



Come abbiamo ragionato

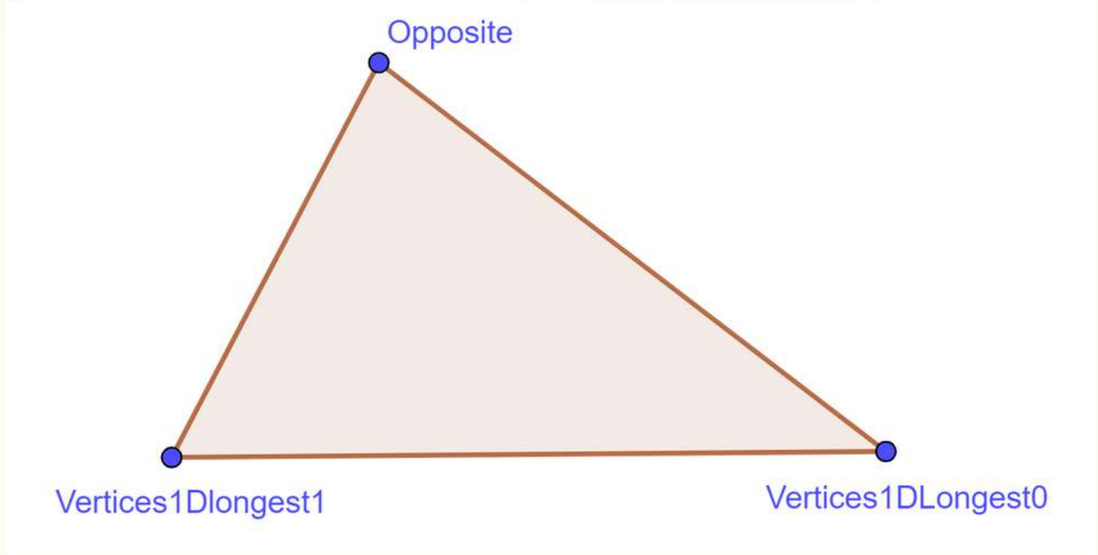
...

...

Ordiniamo i triangoli per area decrescente in un vettore vectSupp

Selezioniamo il triangolo di area massima, che verrà bisezionato (Bisect)

Andiamo a prendere il lato più lungo (latoMax) del triangolo

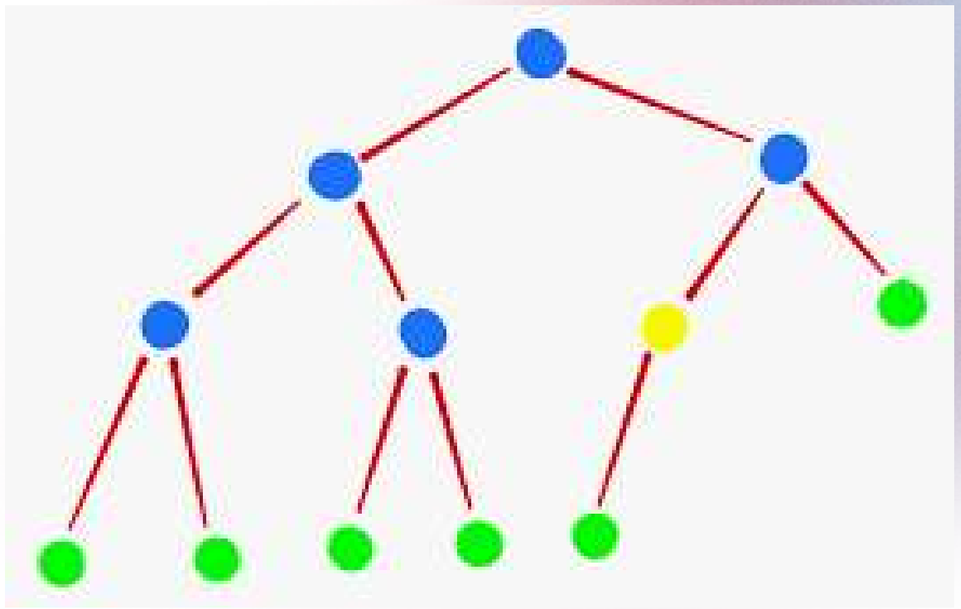


Sorting

Per ordinare le aree in ordine decrescente abbiamo usato l'algoritmo delll'**Heapsort.**

La scelta di questo algoritmo consente di ottenere un ordinamento efficiente operando in loco : non fa uso di memoria aggiuntiva, al di là di un numero costante di variabili ausiliarie.

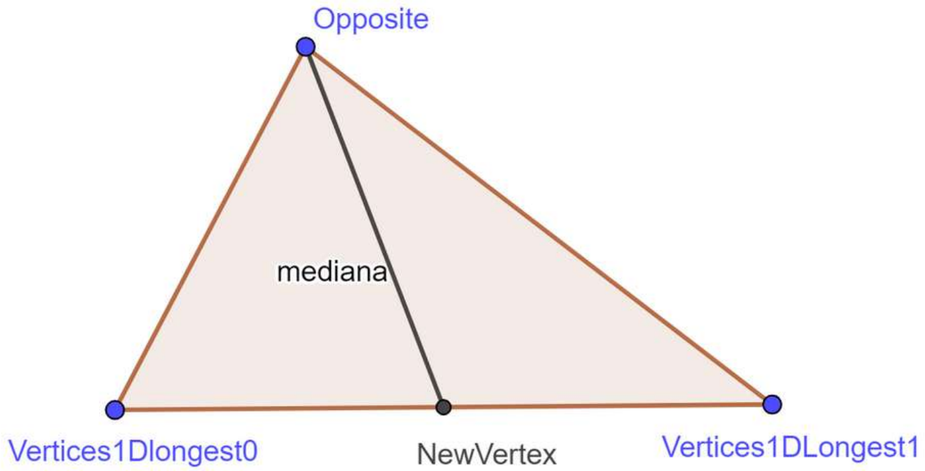
Il suo costo computazionale è quello ottimale e vale : **O (n log(n))**

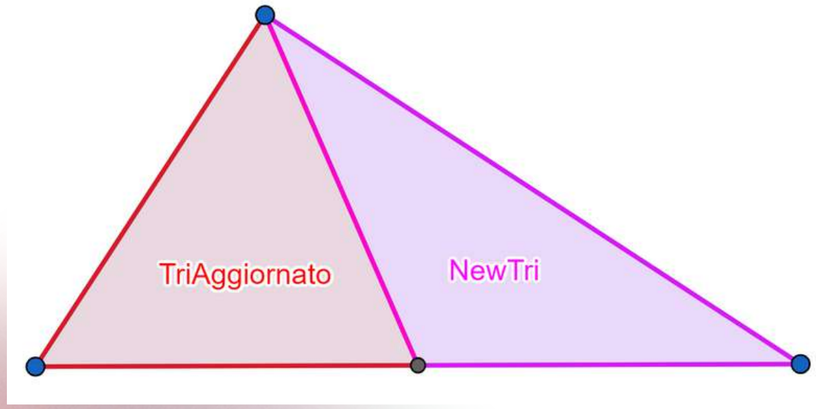


Heap : é una struttura dati logica che rispetta i vincoli
_è un HeapTree
_è un albero binario
_è completo a sinistra

Bisezione

- Cerco il lato più lungo del triangolo (funz MaxEdge)
- Trovo il punto medio
- Biseziono il triangolo unendo il punto medio con il vertice opposto
- Otteniamo così due nuovi triangoli: uno avrà l'id del triangolo originale (TriAggiornato) l'altro un id nuovo (NewTri)





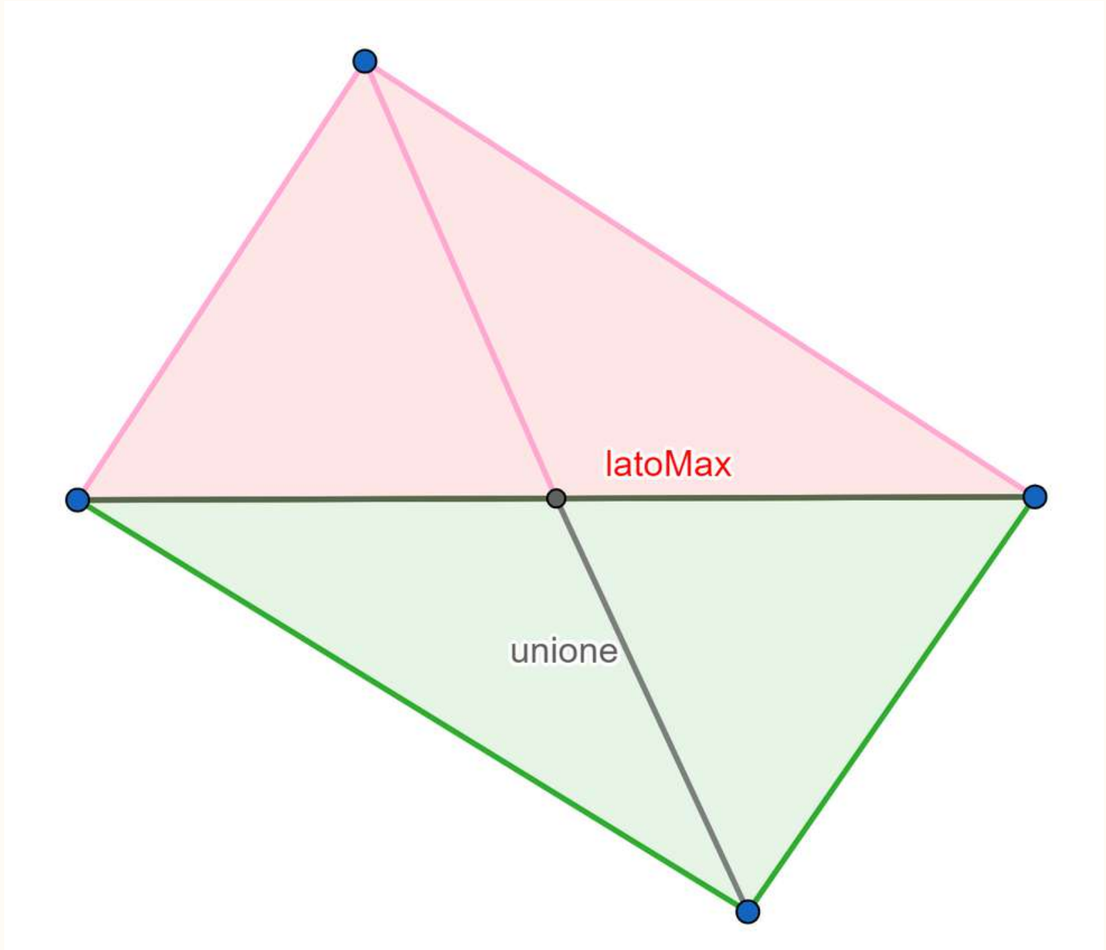
Propagazione

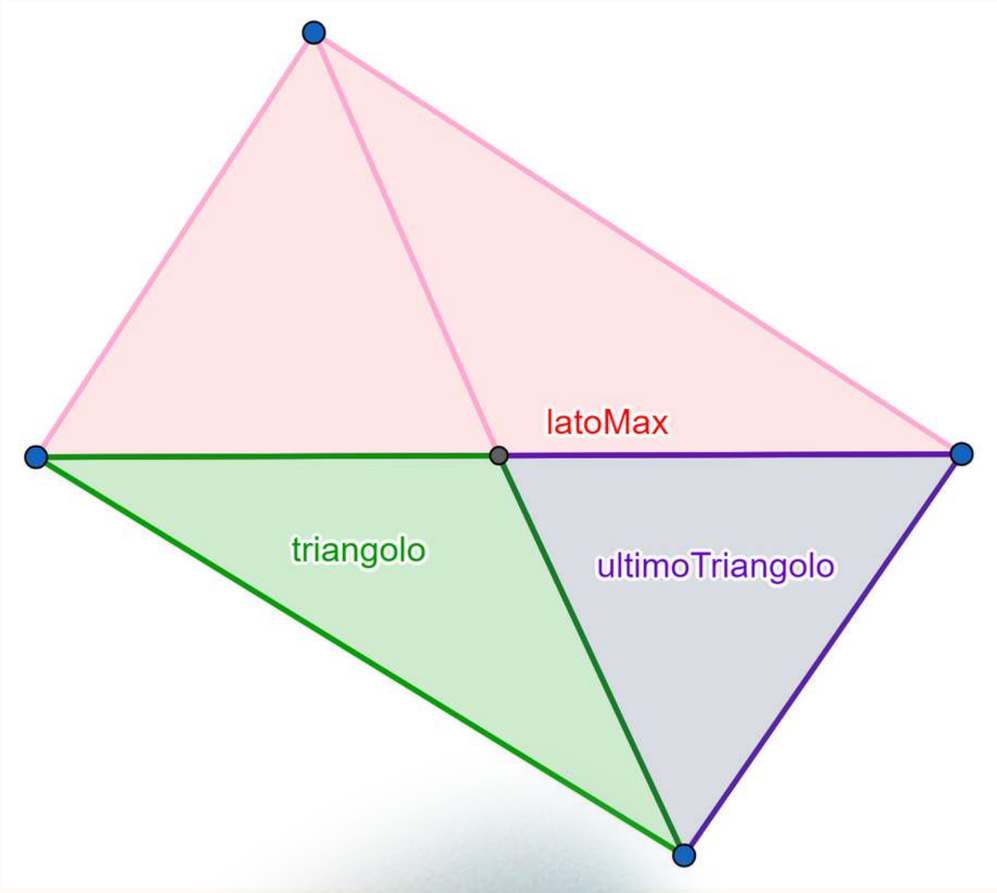
Per mantenere la mesh ammissibile, se il triangolo non si trova al bordo, dobbiamo raffinare anche il triangolo adiacente rispetto al lato bisezionato del triangolo appena raffinato:

Ci sono due casi:

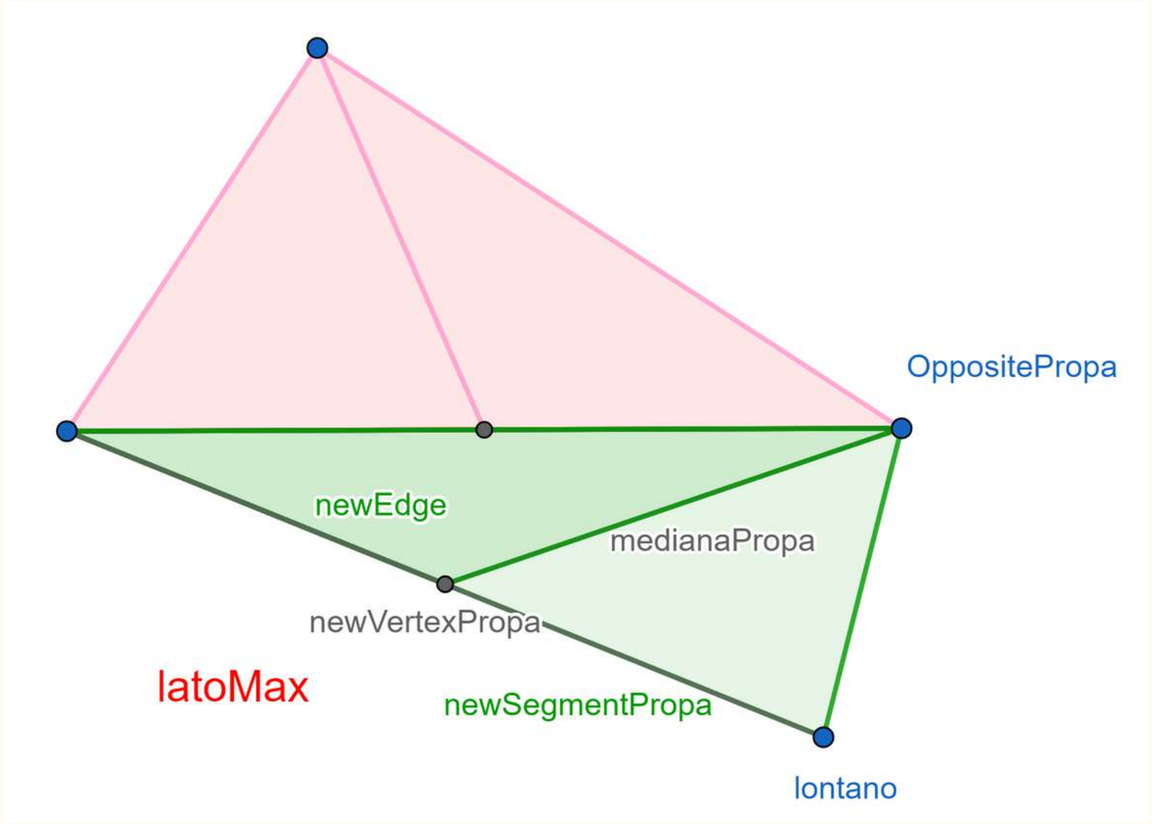
- Il triangolo adiacente ha come lato più lungo il lato precedentemente bisezionato
- Il triangolo adiacente ha come lato più lungo un lato diverso da quello precedentemente bisezionato

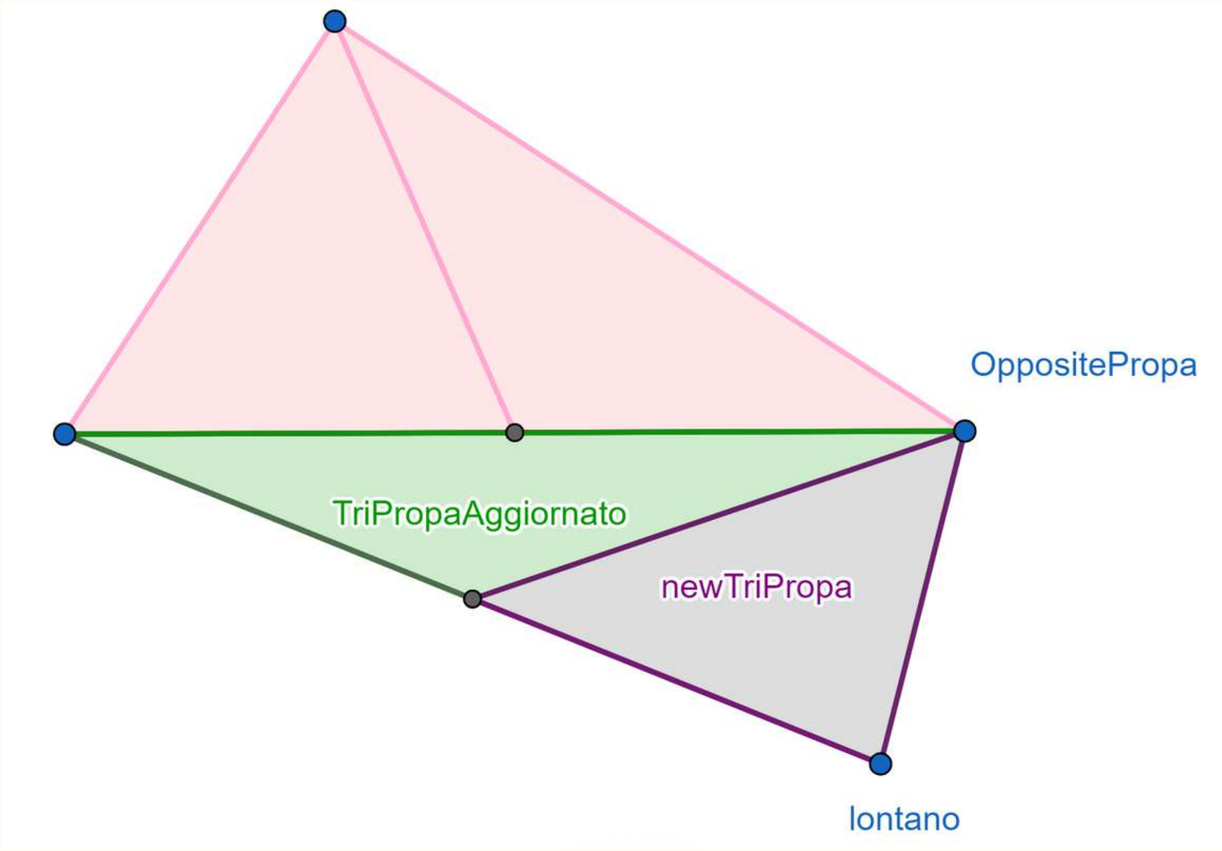
Il procedimento (I)



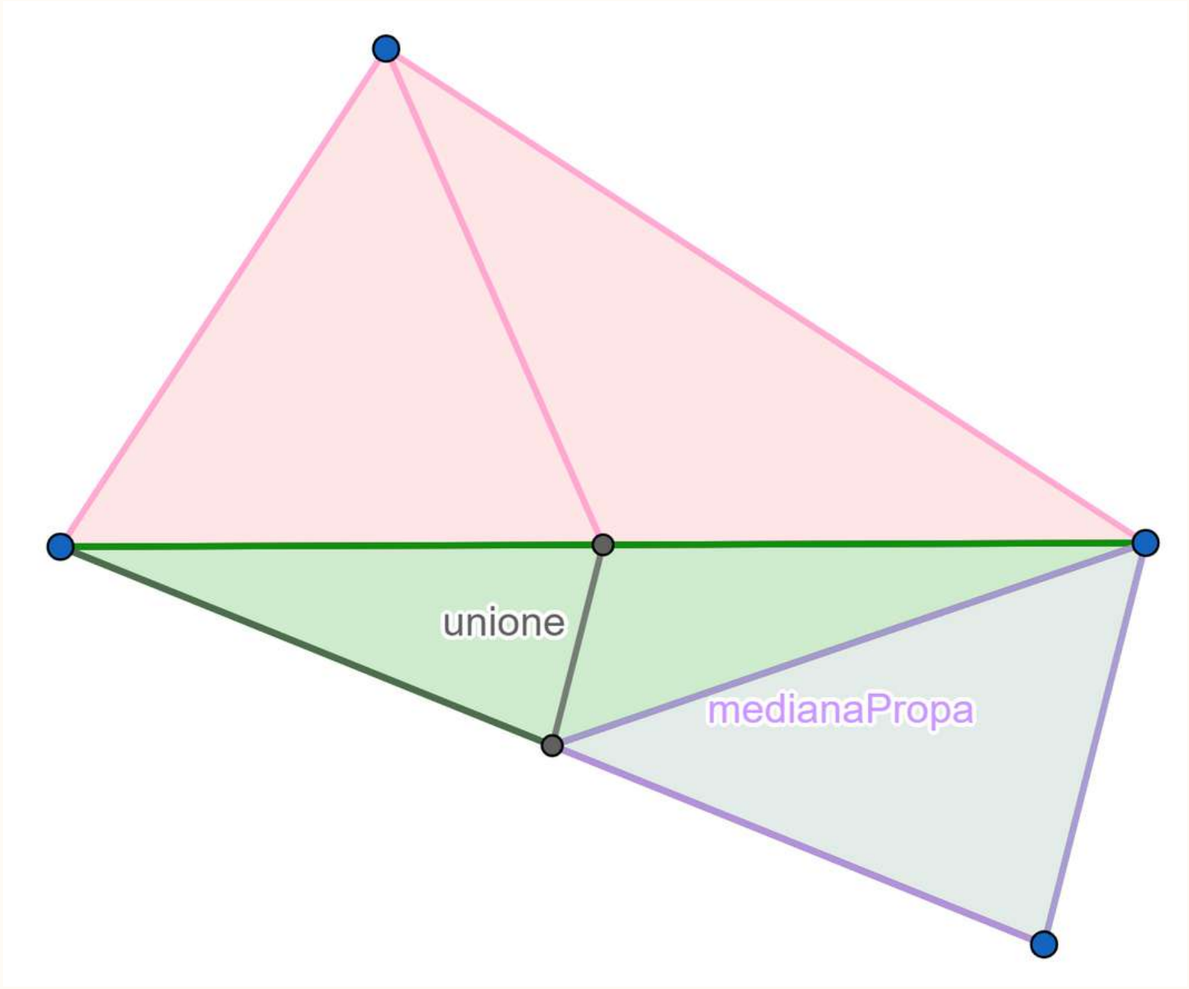


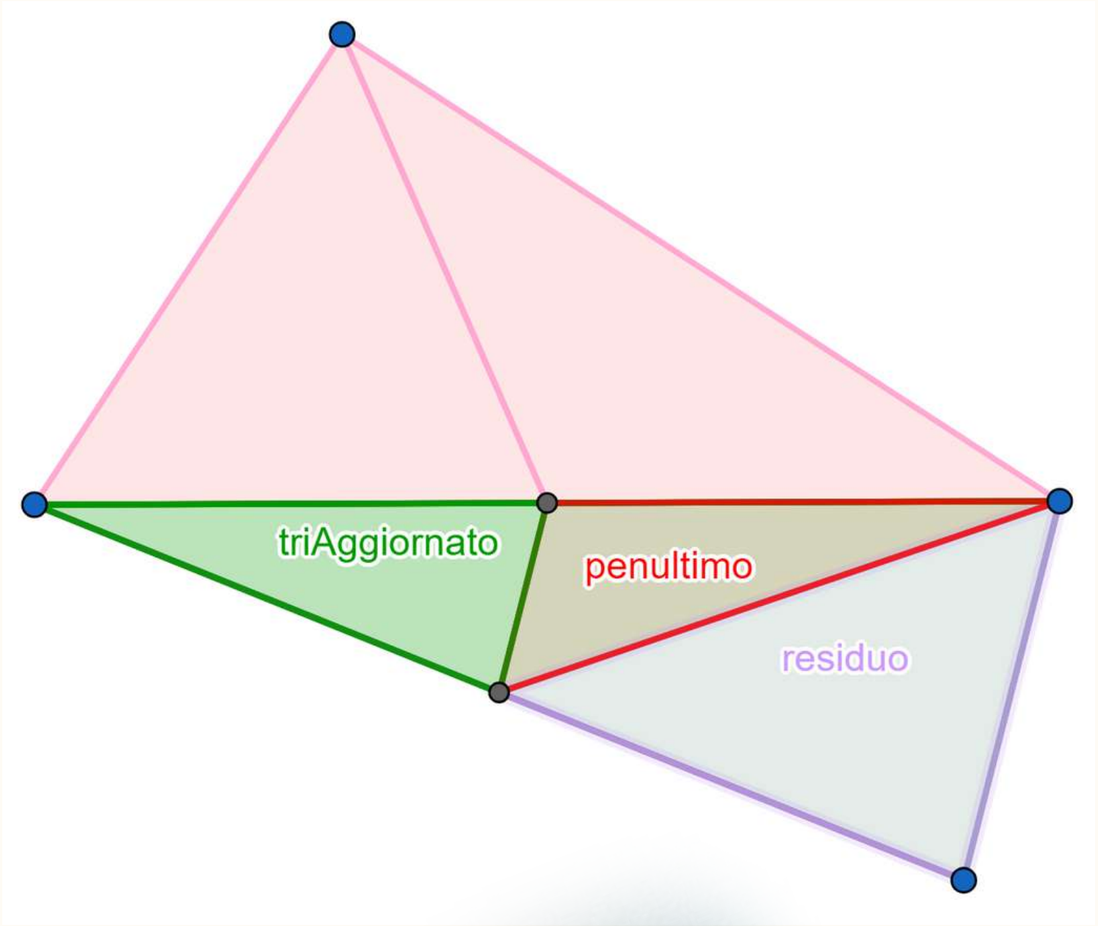
Il procedimento (II)





Il procedimento (II)



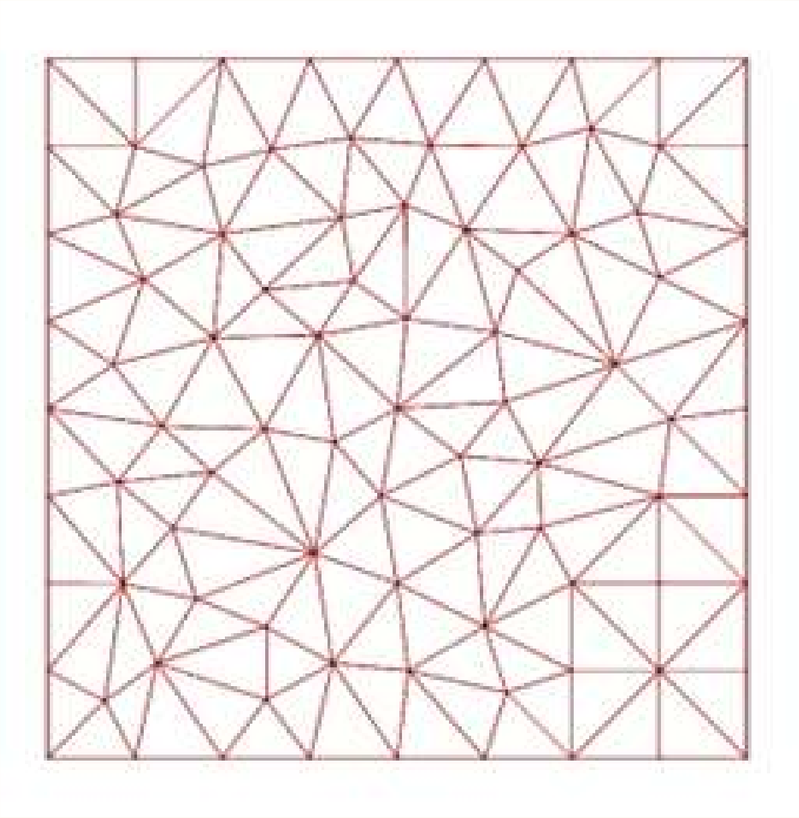


Unit Tests realizzati

- **Test sui costruttori delle classi** :
  - classe Cell0D (vertice)
  - classe Cell1D (lato)
  - classe Cell2D (triangolo)
- **Test sull' algoritmo di sorting:**
  - Heapsort
- **Test sull’ import:**
  - Cell0D (vertici)
  - Cell1D (lati)
  - Cell2D (triangoli)
- **Test sull’ algortimo di raffinamento:**
  - funzione Bisect.
  - funzione Propagazione
- **Test sui metodi**
  - area
  - maxedge
  - lenghtedge

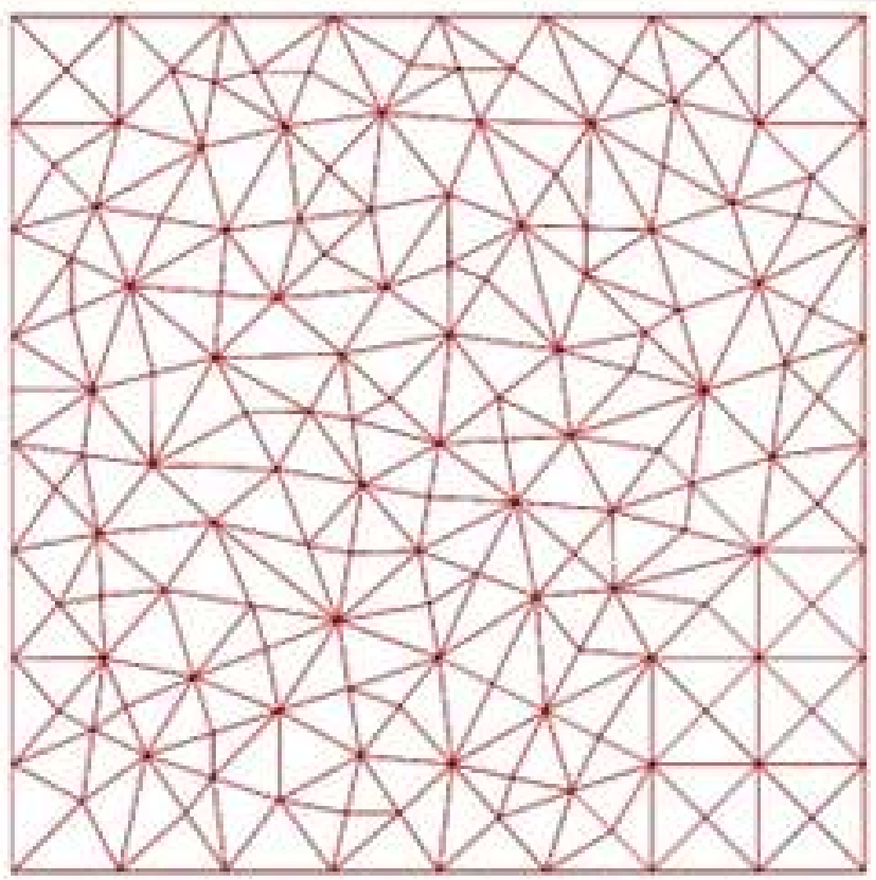
Risultati sulla mesh del test 1

Mesh iniziale :



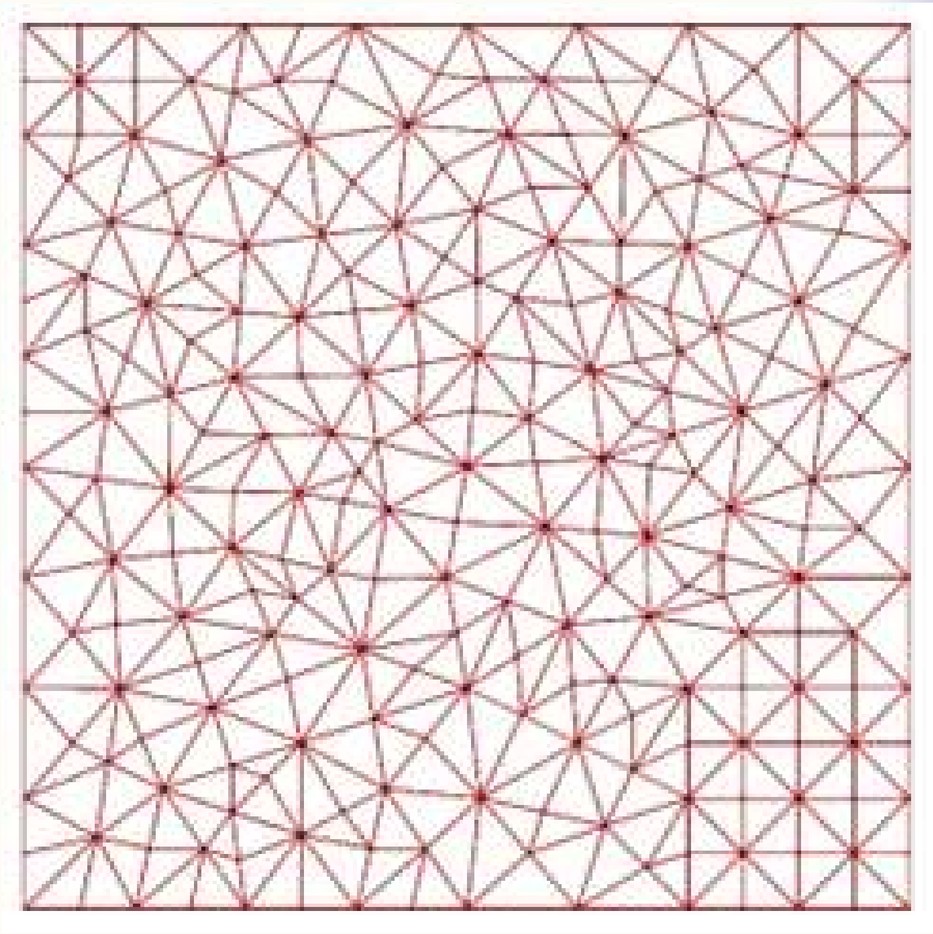
Risultati sulla mesh del test 1

Mesh raffinata con iter(Θ): 50 %



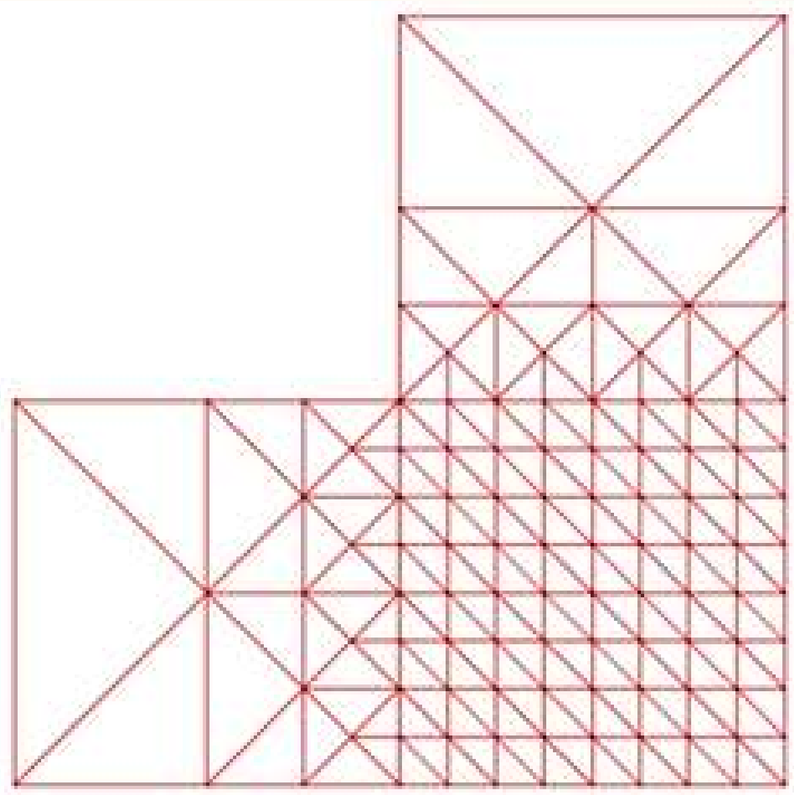
Risultati sulla mesh del test 1

Mesh raffinata con iter(Θ): 100 %:



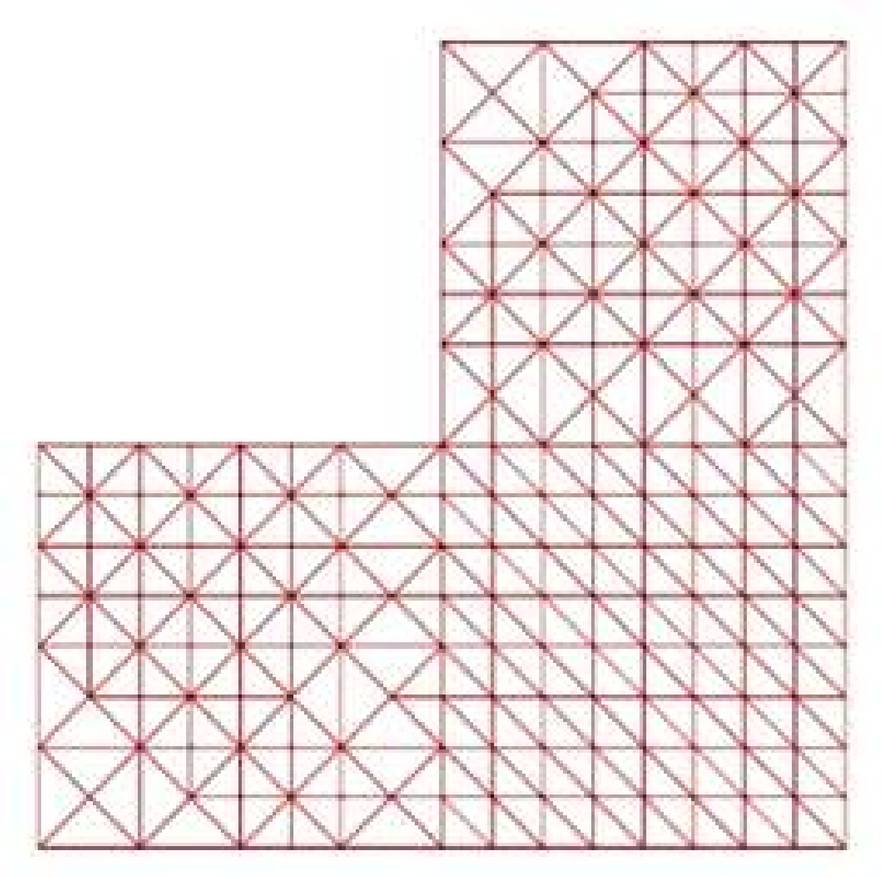
Risultati sulla mesh del test 2

Mesh iniziale :



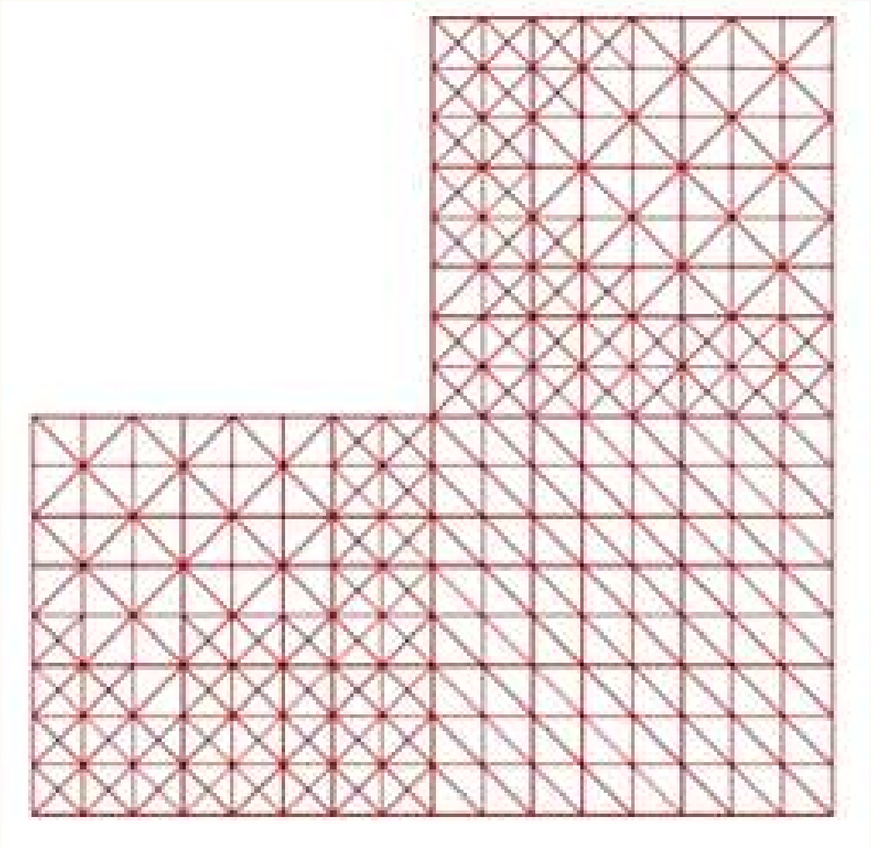
Risultati sulla mesh del test 2

Mesh raffinata con iter(Θ): 50 %



Risultati sulla mesh del test 2

Mesh raffinata con iter(Θ): 100 %:



Politecnico di Torino

Grazie per l'attenzione

Programmazione e Calcolo Scientifico-2022/2023
F. Vicini, S. Berrone, G. Teora

Allegra Riola
Vincenzo Solimando
Gabriele Zambrano

Corso di studi: Matematica Per L'Ingegneria