Progetto: DELAUNAY

TEAM

Andrea Baccolo (282561)

Fabio Daniele Diena (282265)

Umberto Finello (283219)

DOCENTI

Prof. S. Berrone

Prof. F. Vicini

Dott.ssa G. Teora

Programmazione e Calcolo Scientifico

Anno Accademico 2022/23

Indice

0 Introduzione

0.1 Triangolazione di Delaunay

In geometria computazionale, la *triangolazione di Delaunay* per un gruppo di punti P su un piano è una triangolazione tale che nessun punto appartenente a P sia all'interno del cerchio circoscritto ad ogni triangolo della triangolazione.
Tale triangolazione massimizza il minor angolo di tutti gli angoli dei triangoli appartenenti alla triangolazione; si tende quindi ad evitare che ci siano triangoli aventi angoli relativamente molto acuti.
La triangolazione prende il nome da Boris Delaunay per il suo lavoro su questo argomento dal 1934.

Per un gruppo di punti collineari non esiste triangolazione in quanto non si possono formare triangoli non degeneri. Per un gruppo di quattro o più punti su una stessa circonferenza (ad esempio i vertici di un rettangolo) la triangolazione non è unica: ognuno dei due possibili triangoli in cui si può dividere il quadrilatero, infatti, soddisfa i requisiti di Delaunay (ovvero che le circonferenze circoscritte ai triangoli non contengano altri punti).
Considerando le sfere circoscritte, le nozioni della triangolazione di Delaunay si possono estendere a tre o più dimensioni.

0.2 Scelta del progetto

Abbiamo scelto questo progetto al posto di *Raffinamento* per il cuore del progetto stesso: la costruzione di una mesh dato un set di punti forniti a priori. Abbiamo infatti considerato in un'ottica lavorativa il caso verosimile di un'utente (come ad esempio un'azienda o un privato) analogo al progetto.
La costruzione di una mesh da zero, partendo dalle fondamenta della struttura, ci ha permesso di vedere l'edificazione stessa sotto molteplici aspetti e prospettive. L'aggiunta dei nuovi punti è stata probabilmente la parte più complessa e che ha richiesto maggior tempo.

Il controllo di Delaunay inoltre è stato un buon punto di partenza in un'ottica di simmetria e omogenizzazione (degli angoli in questo) di un insieme convesso per sfide che potremmo incontrare in futuro in un'ambito lavorativo volto in particolare alla Ricerca Operativa o ad una ricerca tramite gradiente, equazione differenziali o altro ancora.

1 Classi e Struttura

1.1 Classe Punto

E' stata utilizzata una tolleranza geometrica (geometricTol) pari a $1.0e^{-12}$ per le lunghezze al fine dei confronti. Per le aree invece è stata utilizzata una tolleranza geometrica superficiale (geometricToleranceSquare)
Gli attributi sono: identificatore (id), coordinate cartesiane (x, y), variabile booleana (inserito) per sapere se il punto è inserito nella mesh.

1.2 Classe Lato

Gli attributi sono: identificatore (id), vertici, lunghezza, lato precedente e lato successivo, identificatore dei triangoli che contengono il lato.

1.3 Classe Triangolo

Gli attributi sono: l'identificatore (id), i vertici, i lati.
L'unico metodo è "CalcolaAngolo", ovvero una funzione che restituisce l'angolo compreso tra due lati del triangolo tramite il Teorema di Carnot.

1.4 Classe Mesh

Gli attributi sono:

- lista dei triangoli che appartengono alla mesh (listaTriangoli)
- lista dei lati che appartengono alla mesh (listaLati)
- ista dei punti che appartengono alla mesh (listaPunti)
- lato di bordo da cui partire (hullBeginLato)
- coda di Delaunauy (codaDelaunay)

I metodi sono:

- ricerca del primo triangolo di area massima (PrimoTriangolo)
- ricerca della posizione del nuovo punto rispetto alla mesh (DentroMesh)
- creazione nuovi lati collegando i nuovi punti (CollegaSenzaIntersezioni, PuntoInterno, PuntoBordoHull, PuntoBordoTriangolo)
- controllo di Delaunay sugli angoli (ControlloDelaunay)
- esportazione della mesh (ExportMesh)

1.5 Altro

Le altre funzioni sono:

- il prodotto vettoriale (CrossProduct)
- l'importazione dei punti (ImportPunti)
- la norma (Norm)
- gli ordinamenti:
  - Merge
  - MergeSort

Inoltre la struttura prevede l'enumerazione della posizione dei punti in "DentroMesh" per classificare il nuovo punto rispetto alla mesh.

2 Primo Triangolo

Come primo passo della costruzione della mesh dato il set di punti di partenza, si deve cercare un triangolo iniziale rispetto a cui si andrà successivamente a definire la posizione dei nuovi punti che si vogliono aggiungere alla mesh. L'obiettivo di tale ricerca consiste nel confrontare diversi triangoli e di salvare quello che statisticamente contiene un maggior numero di punti del set di partenza, ovvero il triagolo con area massima tra quelli considerati.

2.1 Individuazione dei punti utili

Innanzitutto si deve pensare ad un criterio di ordinamento dei punti del set di partenza e si è pensato di agire sulle coordinate cartesiane. Per poter effettuare operazioni sulle coordinate, è stata introdotta una variabile booleana (ax) per selezionare di ogni punto rispettivamente l'ascissa o l'ordinata.
Per la ricerca dei punti utili, si è voluto immaginare la mesh come un insieme abbastanza denso e largamente approssimabile da un rettangolo. Fissato virtualmente questo rettangolo, si è condotta una ricerca dei punti più vicini ai vertici ideali del poligono per avere una base di partenza abbastanza robusta e ridurre significativamente il numero di aree da calcolare e confrontare.
Questa ricerca si basa su minimizzazione e massimizzazione delle seguenti trasformazioni:

(a) x + y

(b) x - y

Le trasformazioni vengono ordinate tramite la funzione "MergeSort".
Nel caso in cui questa ricerca portasse ad ottenere due o più punti coincidenti, si procede scambiando uno dei due punti problematici con il proprio precedente o successivo nell'ordinamento effettuato precedentemente.
Trovati quindi i quattro punti desiderati, si procede con il calcolo delle $\binom{4}{3}$ combinazioni e dunque si ottengono le rispettive $\binom{4}{3}$ aree.
Infine si ricerca l'area maggiore tenendo della tolleranza geometrica superficiale come segue:

$$abs(AreaMax - Area) > geometricToleranceSquare * max(AreaMax, Area)$$

2.2 Triangolo di area massima

L'ultimo passaggio di questa ricerca consiste nella costruzione del primo triangolo della mesh. La ricerca tramite paragone delle aree restituisce i tre punti appartenenti al triangolo con area maggiore tra quelli considerati.

Il primo passo è verificare se i punti sono ordinati in senso antiorario tramite il prodotto vettoriale. In caso contrario, si inverte l'ordinamento.

Successivamente si costruiscono i lati e la frontiera della mesh. Infine si crea il triangolo desiderato passando al costruttore triangolo l'identificativo, i punti scelti per i vertici e la lista dei relativi lati.

3 Dentro Mesh

La funzione "DentroMesh" ha lo scopo di cercare e trovare la posizione di un punto non appartenente alla mesh corrente e di restituire informazioni importanti su di esso.

3.1 Ricerca del punto rispetto alla Mesh

Per effettuare questa ricerca della posizione del punto rispetto alla mesh corrente, calcoliamo il prodotto vettoriale tra il vettore contetente il punto che si vuole aggiungere e un punto dell'inviluppo convesso e il vettore contenente due punti successivi dell'inviluppo.
Tale prodotto vettoriale viene definito come segue:

$$(P - A_i) \times (A_{i+1} - A_i)$$

3.2 Punto sulla frontiera o esterno

Se il prodotto vettoriale definito precedentemente è pari a zero, allora il punto che si vuole aggiungere è contenuto nella retta passante per un lato dell'inviluppo convesso. Se inoltre si verifica la condizione tale per cui il punto è contenuto nello stesso lato di bordo della mesh, allora tale punto è detto *punto di frontiera* (HULL). Infine viene restituito l'identificativo del lato contenente il punto.
Se il prodotto vettoriale è strettamente maggiore di zero con almeno una retta passante per un lato dell'inviluppo convesso, allora il punto è detto *punto esterno* (ESTERNO).

Infine viene restituito "0" come valore di default dell'identificativo della posizione in quanto non necessario a tale tipologia di punto.

3.3 Punto interno alla mesh

Se si ricade in un qualsiasi altro caso, il punto è detto *punto interno*.
Infine viene restituito l'identificativo del triangolo in cui il punto è strettamente contenuto oppure l'identificativo del lato a cui appartiene il punto. Il caso di un punto contenuto in due lati, cioè coincidente ad un vertice già esistente, non è stato considerato.

Ricerca del triangolo che contiene il punto
Questa parte è divisa in due: controllo che il punto appartenga al triangolo (modo analogo a prima) e il percorso con i triangoli.
Nel caso in cui il punto sia interno alla mesh, dentro mesh effettua una ricerca su alcuni triangoli per individuare quello in cui il mio punto è contenuto, cercando di non provare tutti i triangoli. Si parte dal triangolo relativo "hullBegin" e cercare di andare verso il lato che minimizza la distanza punto-segmento. Prima però si controlla di non avere un lato di bordo, in tal caso senza fare alcun calcolo si passa all'unico lato che rimane. Abbiamo definito la distanza punto-segmento come la distanza punto-retta nel caso in cui la proiezione del punto sulla retta generata dal segmento appartenga al segmento, altrimenti come il minimo tra la distanza degli estremi del segmento. Se esse risultano uguali, si guarda gli angoli generati dai vettori e si va verso il minore.

4 Collegare i punti

4.1 Collega Senza Intersezioni

La funzione "CollegaSenzaIntersezioni" prende in input il punto esterno e restituisce una mesh aggiornata convessa che non si autointerseca in punti non dati dall'utente.
La funzione sfrutta un'altra funzione chiamata "Accettabile", la quale dato in input un punto dell'inviluppo convesso, restituisce una variabile booleana che indica se tale punto risulta essere valido per costruire un lato che va dal nuovo punto esterno ad esso.
Verrà prima spiegata "Accettabile" e poi come essa viene utilizzata in "CollegaSenzaIntersezioni".

4.1.1 Casi di accettabilità

La funzione considera il vettore

$$v = (P - A*)$$

e dunque si cerca l'intersezione delle rette generate da un lato dell'inviluppo e il vettore v.
Considero dunque il determinante della matrice associata al sistema lineare, ovvero il prodotto vettoriale tra i due vettori nominati sopra: si generando 3 diversi casi.

1. det(M) $\neq 0$
2. det(M) $= 0$, rette parallele
3. det(M) $= 0$, rette coincidenti

caso 1: det(M) $\neq 0$
Se il determinante è diverso da zero posso cercare la soluzione del sistema lineare isolando dal sistema sopra descritto le mie variabili, ricavandone un'espressione esplicita. Ora impongo che la soluzione del sistema lineare appartenga ai segmenti: sappiamo per definizione di soluzione del sistema lineare che essa appartiene ad entrambe le rette, ma non abbiamo la garanzia che appartenga anche ai segmenti.
Nel caso in cui la soluzione non appartenga al segmento, non la si può usare per determinare se il punto sia accettabile o meno, dunque si va avanti nel ciclo passando al lato successivo dell'inviluppo.

Se il punto risulta appartenere ad entrambi i lati allora mi chiedo se l'intersezione trovata sia uno dei due punti estremi che mi definisce il lato considerato dell'inviluppo. Se così non è, l'intersezione non fa (forse, ancora per il momento) parte della mesh: la funzione termina subito con falso e il punto risulta non accettabile.
Se il punto considerato risulta essere uno dei due estremi del lato dell'inviluppo, devo controllare che in altri lati non ci sia intersezione, ma devo anche controllare di non essere nel caso in cui le uniche intersezioni distinte che trovo sono punti dell'inviluppo, ma esse risultano essere più di una. In tal caso è stato implementato un contatore che si abbia non più di due soluzioni non distinte.

caso 2: det(M) = 0, rette parallele
Nel caso in cui le rette risultino essere parallele, non ci sono punti di intersezione e dunque non ci sono punti problematici che influiscono nell'accettabilità del punto, dunque il programma passa al lato successivo.

caso 3: det(M) = 0, rette coincidenti
Per verificare questo caso, è stato aggiunto un else if che tiene conto del fatto che i due vettori considerati abbiano un punto in comune.
Viene considerato A*, ovvero il punto dell'inviluppo che controllo essere accettabile, e non P perchè quest'ultimo sappiamo sicuramente non appartenere ancora alla mesh.
A questo punto è importante cercare di non rendere accettabile un possibile punto A* allineato sopra e sotto da altri due punti dell'inviluppo. Per fare ciò si calcola il minimo tra la distanza del punto da mettere e gli estremi del lato, e la si confronta con la distanza di A* dal punto esterno. Se queste due distanze sono uguali, non risultano al momento punti che compromettano l'accettabilità e si passa al lato successivo. Altrimenti si può affermare che il punto non risulta accettabile, terminando il ciclo con falso.
Al termine dei cicli, se il programma non è uscito prima, il punto risulta essere accettabile, dunque la funzione termina restituendo vero.

4.1.2 Ricerca dei punti di interesse

La funzione inizia chiedendosi se il punto p1 dell'inizio dell'inviluppo memorizzato in mesh sia o meno accettabile, l'obbiettivo è quello di trovare il primo e l'ultimo punto dell'inviluppo che risultano essere accettabili, cercando di evitare di percorrerlo tutto per scoprirlo.

Se il primo punto risulta essere accettabile allora vado avanti tramite i lati successivi dell'inviluppo a cercare punti non accettabili, quando ne trovo uno allora torno al precedente e modifico una variabile apposita chiamata "fineLato", che rappresenta l'ultimo lato accettabile. Successivamente verifico che il punto precedente al punto p1 relativo all' "hullBegin" sia accettabile o meno, se lo è vado indietro finchè non trovo un punto non accettabile, una volta trovato passo al successivo e aggiorno di conseguenza la variabile "inizioLato", se il precedente è non accettabile aggiorno l'inizio.

Se il primo punto risulta non accettabile, vado avanti finchè non trovo il primo punto accettabile e successivamente vado ancora avanti finchè non trovo il primo punto non accettabile, aggiornando le variabili sopra descritte con opportuni valori. Sono stati evitati possibili casi di cicli infiniti, dove tutti i punti sono accettabili.

4.1.3 Creazione dei nuovi triangoli

Trovato inizio e fine, si procede a creare i lati accettabili e di conseguenza i triangoli generati da essi, i lati vengono creati coerentemente con l'inviluppo, in modo da mantenere l'ordinamento antiorario. Nei lati sono stati salvati inoltre gli id per l'adiacenza dei nuovi triangoli creati.

4.2 Punto interno a un triangolo

Il metodo inizia creando una copia del triangolo contenente il punto.

STEP 1
Crea i lati tra il punto nuovo e i primi due vertici del triangolo.
Il metodo crea il triangolo che sostituisce quello originario nella lista dei triangoli della mesh con lati: il secondo lato nuovo, il primo lato originario e il primo lato nuovo. Inoltre ha come punti: il nuovo punto e i

primi due punti originari.
Se il primo lato originario non è di frontiera, viene aggiunto alla Coda di Delaunay con l'idenficativo del triangolo originario.

STEP 2
Crea il lato tra il punto nuovo e il terzo vertice originario. Costruisce il triangolo di id nuovo con lati: il primo lato creato allo step 1, il secondo lato originario e il lato nuovo appena creato. Inoltre ha come punti: il nuovo punto, il secondo e il terzo punto originario. Questo triangolo viene aggiunto al fondo della lista dei triangoli della mesh.
Nella lista dei triangoli di adiacenza del secondo lato originario viene sostituito l'id originario con l'id nuovo; se questo lato non è di frontiera, viene aggiunto alla Coda di Delaunay con l'id nuovo.

STEP 3
Crea il triangolo con un altro id (id3) e lo aggiunge al fondo della lista dei triangoli della mesh. Tale triangolo ha come lati: il terzo lato nuovo, il terzo lato originario e il primo lato nuovo. Inoltre ha come punti: il nuovo punto, il terzo e il primo punto originario.
Nella lista dei triangoli di adiacenza del terzo lato originario, viene sostituito l'id originario con l'id3; se questo lato non è di frontiera, viene aggiunto alla Coda di Delaunay con l'id3.

4.3 Punto sul lato di un triangolo

Il metodo inizia creando una copia del lato contenente il punto.
Poi esamina il primo triangolo della lista di adiacenza di questo lato, ne memorizza l'id e ne fa una copia. Prende i primi due punti (p1, p2) di questo triangolo che con il nuovo punto generino un'area diversa da zero e crea i lati e il triangolo in base al fatto che il primo dei due appartenga o no al lato originario (condizione 1). Poi prende p2 e il successivo ed in base alla condizione 1 crea un nuovo lato e un nuovo triangolo.
Successivamente passa al secondo triangolo della lista di adiacenza del lato contenente il punto nuovo, ne memorizza l'id e ne fa una copia. Prende i primi due punti (p1', p2') di questo triangolo che con il nuovo punto generino un'area diversa da zero e crea i lati e il triangolo in base al fatto che il primo dei due appartenga o no al lato originario (condizione 2) e in base alla condizione 1. Poi prende p2' e il successivo ed in base alle condizioni 1 e 2 crea un nuovo lato e un nuovo triangolo.
Per ogni i-esimo triangolo creato, nella lista di adiacenza del lato originario contenuto viene sostituito l'id originario con il rispettivo i-esimo id nuovo; se questo lato non è di frontiera, viene aggiunto alla Coda di Delaunay con l'i-esimo id nuovo.

4.4 Punto sul bordo dell'hull

Il metodo inizia creando una copia del lato contenente il punto nuovo (*pnew*), salvando l'id di del triangolo contenente questo lato e facendo una copia del triangolo con questo id. Poi cerca la posizione nel vettore dei vertici e l'id del punto non appartenente al bordo dell'inviluppo convesso della mesh (*pnf*, punto non frontiera). Successivamente crea il lato tra il primo punto del lato originario e *pnew* (*ln1*, lato nuovo 1) e il lato tra *pnew* e *pnf* (*ln2*, lato nuovo 2). Crea quindi il triangolo avente come id quello salvato prima, come lati *ln2*, il lato originario con la posizione cercata prima e *ln1* creato e con vertici *pnew*, *pnf* e il primo punto del lato originario. Se questo lato originario non è di frontiera, lo aggiunge alla Coda di Delaunay con l'id del triangolo originario.
Crea il lato tra *pnew* e il secondo punto del lato originario. Crea quindi il triangolo con id nuovo, con lati il terzo lato nuovo, il lato originario con la posizione cercata prima meno 1 e *ln2* creato e con vertici *pnew*, il secondo punto del lato originario e *pnf*. Se questo lato originario non è di frontiera, lo aggiunge alla Coda di Delaunay con l'id nuovo. Nella lista di adiacenza del lato originario viene sostituito l'id originario con l'id nuovo.
Infine viene aggiornata la frontiera della mesh.

5 Controllo di Delaunay

Finché la coda di Delaunay non è vuota il metodo esegue le seguenti operazioni:

- memorizza il lato in cima alla coda (*lic*, lato inizio coda) e l'id del triangolo da cui arrivo (*tv*, triangolo vecchio)
- calcola gli angoli opposti a *lic* nei due triangoli adiacenti, se è maggiore di pi greco:
  - Trova l'id del triangolo nuovo (*nt*) e memorizza in `lat_cd` i suoi lati non di bordo e diversi da *lic*;
  - Trova i vertici opposti a *lic* nei due tiangoli e crea il lato tra questi due vertici sostituendolo a *lic* nella lista dei lati;
  - Trova nei due triangoli le posizioni dei due vertici di *lic*;
  - Crea i due triangoli prendendo uno dei vertici di *lic* il successivo punto in uno dei due ex triangoli tale che non sia un altro vertice di *lic* ed il restante vertice opposto a *lic* e di conseguenza prende i lati. Aggiorna inoltre i valori della lista di adiacenza dei lati;
  - Aggiunge i lati che sono in `lat_cd` alla coda di Delaunay con id del triangolo corretto o *nt* o *tv*;
- elimina il primo elemento della coda.

6 Risultati e conclusioni

6.1 Costi computazionali

Il costo computazionale complessivo risulta essere O(nlog(n)), assunto che la cardinalità dei punti sulla frontiera della mesh sia molto minore di quella dei punti totali.

Per la funzione "PrimoTriangolo" la complessità computazionale è O(nlog(n)). "DentroMesh" in un caso medio ha una complessità computazionale che non esplode e risulta essere migliore di controllare tutti i triangoli. Le funzioni di aggiunta dei punti interni, sulla frontiera della mesh e sul bordo di un triangolo hanno ciascuna costo computazionale O(1). La funzione "Accettabile" ha complessità computazionale O(l), dove l è il numero di punti dell'inviluppo convesso della mesh. CollegaSenzaIntersezioni ha costo computazionale O(l^2). Il controllo dell'ipotesi di Delaunay ha complessità computazionale O(1).

6.2 Tempo di esecuzione

Il programma in modalità *release* impiega indicativamente 5,3 ms per il test 1 e circa 2,5 ms per il test 2 tramite la funzione "`steady_clock`" della libreria *Chrono*.

6.3 Mesh ottenute

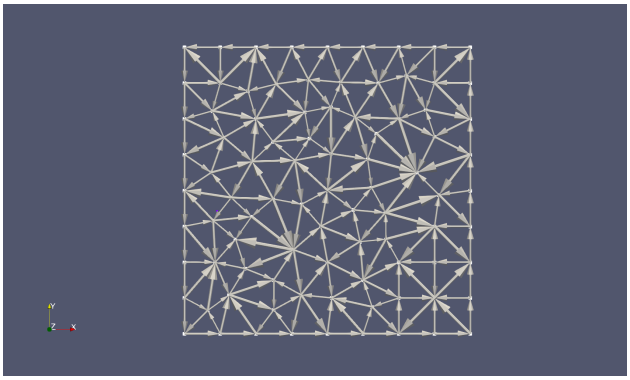

Figura 1: *Test 1*

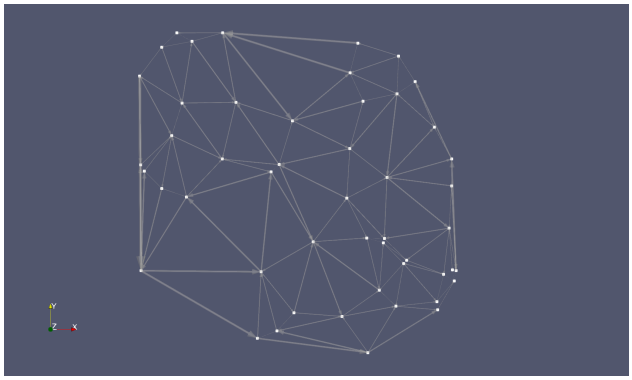

Figura 2: *Test 2*