Presentazione progetto

Bessi Ludovico, Boscolo Claudio,
Cataldo Luca, Licari Valentina
A.A. 2017/2018
Programmazione e calcolo scientifico

22/05/2018

Indice

1 Progetto 2D

Obiettivo della parte bidimensionale del progetto scritto in matlab é, data una triangolazione e un segmento definito "traccia", portare a termine le seguenti richieste:

- Individuare quali triangoli di partenza vengono tagliati dalla traccia
- Individuare tutti i triangoli vicini ai triangoli tagliati, i.e. tutti quelli che condividono un vertice o un lato con un triangolo tagliato
- Costruire sottopoligoni a partire dalle informazioni ottenute analizzando la posizione della traccia rispetto alla triangolazione di partenza.
- Costruire sottotriangoli a partire dalle informazioni ottenute analizzando la posizione della traccia rispetto alla triangolazione di partenza.
- Salvare le informazioni relative alle intersezioni tra la traccia e la triangolazione, i.e. le ascisse curvilinee dei punti di intersezione.

1.1 Lettura file e strutture dati

In questa sezione vengono brevemente presentate le scelte adottate per salvare le informazioni del file relativo alla triangolazione. Per generare le strutture dati di *node* e *info_trace* é stato utilizzato il comando *repmat* di matlab.

1.1.1 Strutture dati

Node
Si è scelto di salvare nella struttura *node* le coordinate dei punti della triangolazione del piano cartesiano 2D, un "side" del punto rispetto alla traccia (un indicatore che puó assumere valore 0,1,-1,2) che esprime dove si trova il punto rispetto alla traccia. Il campo side assume valore '0' se la posizione del punto é non nota, '1' o '-1' se il punto si trova alla sinistra o alla destra della traccia e '2' se si trova sulla retta della traccia. Il campo "toll" presenta una tolleranza relativa al problema, calcolata tenendo conto delle dimensioni dei lati della triangolazione e verrá meglio descritta in seguito. 'Edges' e 'Triangles' sono vettori che contengono rispettivamente gli indici dei segmenti e dei triangoli che hanno l'n-esimo nodo della struttura dati come estremo. 'tot_edges' e 'tot_triangles' contengono rispettivamente il numero di segmenti e il numero di triangoli condivisi da ogni nodo della struttura dati. L'ultimo campo di *node* contiene l'ascissa curvilinea s del punto nel caso in cui questo si trovi sulla traccia.

Triangle
É una matrice di n_triangles righe e 10 colonne. Le colonne 1,2,3 contengono i vertici del triangolo (indici dei nodi). Le colonne 4,5,6 contengono degli status relativi ai lati dell' n-esimo triangolo. Gli status indicano le possibili posizioni del segmento rispetto alla traccia (interno,secante,non incidente). É importante osservare che la colonna 4 della matrice contiene lo status del segmento formato dai nodi 2-3 ; la colonna 5 della matrice contiene lo status del segmento formato dai nodi 1-3 ; la colonna 6 della matrice contiene lo status del segmento formato dai nodi 1-2. Le colonne 7,8,9 contengono le eventuali intersezioni dei segmenti del triangolo con la traccia, seguendo una numerazione analoga a quella poco prima descritta. La colonna 10, ultima, contiene un flag che monitora l'operazione di messa in coda del triangolo la quale serve per verificare se é tagliato.

Edge
É una matrice contenente i segmenti indicizzati della triangolazione, le due colonne della matrice contengono gli indici degli estremi del segmento.

Neigh
É una matrice le cui colonne contengono i vicini del triangolo n-esimo.

Trace vertex
É una matrice le cui colonne contengono le *coordinate* degli estremi della traccia.

Trace
É una matrice le cui colonne contengono gli indici degli estremi della traccia riferiti alla matrice 'trace_vertex'.

Info trace
É una struttura che contiene, per ogni traccia, le informazioni relative alle richieste del problema. Il campo 'cut_tri' é una sottostruttura di 'info_trace' e contiene le informazioni relative ai triangoli tagliati. Per ogni triangolo tagliato di questa sottostruttura vengono salvati in 'points' i punti in 'poly_1' i punti del primo poligono che si forma dopo il taglio, in 'poly_2' i punti del secondo poligono che si forma dopo il taglio, in 'tri' le informazioni relative alla sottotriangolazione, 'tri' é una matrice di dimensione iniiziale 3x3 (numero plausibile di sottotriangoli) che descrive utilizzando i punti le cui coordinate sono salvate in points, i triangoli della sottotriangolazione.

1.2 Script di riempimento

In questa sezione vengono brevemente presentati tre script che calcolano informazioni utili per portare a termine le richieste del problema. I dati calcolati da queste funzioni vengono salvati nelle strutture prima presentate.

- **triangles_for_node**
  Calcola per ogni nodo i triangoli aventi quel nodo come estremo. La strategia é quella di scorrere la matrice triangle e per ogni vertice del triangolo inserire l'indice dell'i_esimo triangolo al campo 'triangles' della struttura 'node'. Viene salvato il numero totale dei triangoli condivisi come giá spiegato.

- **edges_for_node**
  Analoga allo script "triangles_for_node', calcola le stesse informazioni per i segmenti e inserisce gli indici degli edges in 'edges'. Il numero totale dei segmenti condivisi é salvato in 'tot_edges'.

- **toll_for_node**
  L'idea utilizzata per avere una tolleranza con cui lavorare é quella di fare una media sulle lunghezze dei segmenti della triangolazione. Per ogni nodo é quindi stata calcolata una media delle lunghezze di tutti i segmenti aventi quel nodo come estremo. Per *diminuire il costo computazionale in tempo* viene utilizzata la norma infinito invece della norma euclidea per il calcolo delle lunghezze. La differenza tra scegliere una norma o un'altra é soltanto di una costante moltiplicativa.

1.3 Le funzioni which_side e intersect

In questa sezione vengono descritte due funzioni importanti per il programma che servono per capire come i segmenti di un triangolo siano messi rispetto alla traccia e se questi la intersecano e come. Grazie a queste informazioni sará possibile capire quali triangoli siano tagliati o meno.

- **which_side**
  É una function a cui vengono passati l'id della traccia e il punto del quale si vuole conoscere la posizione rispetto alla traccia. La funzione restituisce 1 o -1 se il punto non sta sulla retta della traccia e 2 altrimenti. La funzione tiene conto della tolleranza.

- **intersection**

1.4 Script di riempimento

In questa sezione vengono brevemente presentati tre script che calcolano informazioni utili per portare a termine le richieste del problema. I

dati calcolati da queste funzioni vengono salvati nelle strutture prima presentate.

- **triangles_for_node**
  Calcola per ogni nodo i triangoli aventi quel nodo come estremo. La strategia é quella di scorrere la matrice triangle e per ogni vertice del triangolo inserire l'indice dell'i_esimo triangolo al campo 'triangles' della struttura 'node'. Viene salvato il numero totale dei triangoli condivisi come giá spiegato.
- **edges_for_node**
  Analoga allo script "triangles_for_node', calcola le stesse informazioni per i segmenti e inserisce gli indici degli edges in 'edges'. Il numero totale dei segmenti condivisi é salvato in 'tot_edges'.
- **toll_for_node**
  L'idea utilizzata per avere una tolleranza con cui lavorare é quella di fare una media sulle lunghezze dei segmenti della triangolazione. Per ogni nodo é quindi stata calcolata una media delle lunghezze di tutti i segmenti aventi quel nodo come estremo. Per *diminuire il costo computazionale in tempo* viene utilizzata la norma infinito invece della norma euclidea per il calcolo delle lunghezze. La differenza tra scegliere una norma o un'altra é soltanto di una costante moltiplicativa.

1.5 Le funzioni which_side e intersect

In questa sezione vengono descritte due funzioni importanti per il programma che servono per capire come i segmenti di un triangolo siano messi rispetto alla traccia e se questi la intersecano e come. Grazie a queste informazioni sará possibile capire quali triangoli siano tagliati o meno.

- **which_side**
  É una function a cui vengono passati l'id della traccia e il punto del quale si vuole conoscere la posizione rispetto alla traccia. La funzione restituisce 1 o -1 se il punto non sta sulla retta della traccia e 2 altrimenti. La funzione tiene conto della tolleranza.
- **intersection**

2 Fratture

... delle nostre ossa

3 C++

std::vector per tutto