Report progetto PCS 2024

Gabriele Carnabuci 295510 - Filippo Demartini 294081

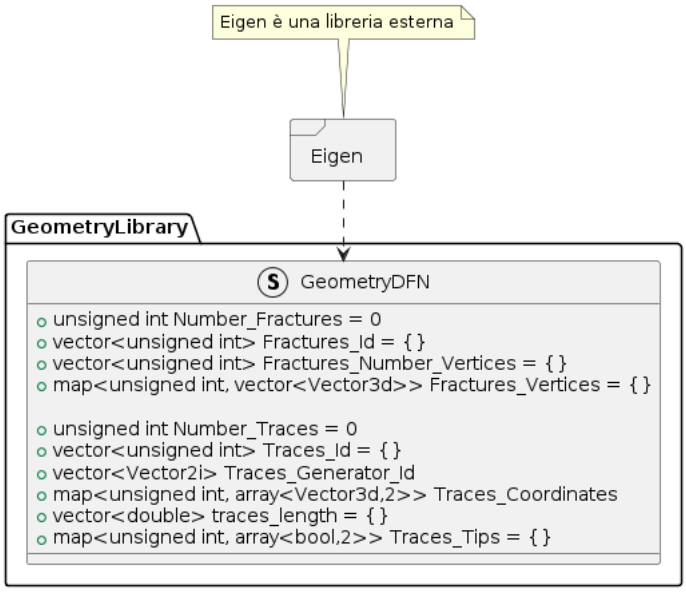
A.A. 2023/2024

Indice

1 Introduzione

Un Discrete Fracture Network (DFN) è un sistema di N fratture, rappresentate da poligoni planari che si intersecano tra di loro nello spazio tridimensionale.
Le intersezioni tra le fratture definiscono le tracce, le quali sono identificate da un segmento. Per ciascuna frattura presente nel DFN, le tracce possono essere di tipologia passante o non passante. Una traccia passante per una frattura è un segmento con entrambi gli estremi che giacciono sul bordo della frattura stessa. Una traccia non-passante per una frattura è un segmento che ha almeno un suo estremo all'interno della frattura stessa.

2 Struttura utilizzata



- **Fracture_id:** si tratta di un vettore di dimensione (1 × Number_Fractures). La dimensione del vettore è variabile ed è associata al numero di fratture, il quale può cambiare (dipende dai dati che prendiamo).
- **Fracture_Number_Vertices:** vettore di dimensione (1 x Number_Fractures), anch'esso variabile in quanto dipendente dal numero di fratture.
- **Fracture_Vertices:** è una mappa che raccoglie le coordinate della frattura, essa ha come chiave l'id della frattura che è un unsigned int e come dimensione (3 x Fracture_Number_Vertices) x Number_Fractures. Il tre ci indica che siamo in uno spazio tridimensionale.
- **Traces_id:** si tratta di un vettore di dimensione (1 × Number_Traces). La dimensione del vettore è variabile ed è associato al numero di tracce che saranno presenti, il quale può cambiare (dipende dalle fratture che prendiamo in considerazione).
- **Traces_Generator_id:** ha dimensione (2 x Number_Traces) dove memorizziamo gli Id delle due Fratture intersecanti che vanno a generare la Traccia.

- **Traces_Coordinates:** è una mappa con chiave Traces_id che è un unsigned int e ha dimensione (3 x 2) x Number_Traces. Esso è memorizzato come un array poiché il suo contenuto non varia in dimensione, in quanto l'array contiene sempre le tre coordinate dell'origine e della fine della traccia.
- **Traces_Tips:** è una mappa che ha come chiave Traces_id, che è unsigned int e associa a ciascuna chiave un array di due elementi di tipo booleano (bool) che corrispondono alle due fratture generatrici della traccia, dalla tipologia di questi booleani capiremo successivamente se la traccia per quella determinata frattura sarà passante o meno.

3 Funzioni implementate per il calcolo delle tracce

3.1 Eliminazione fratture molto distanti

Per eseguire una prima scrematura e quindi eliminare tutte quelle fratture che sono troppo lontane tra di loro e che sicuramente non si intersecheranno cioè che la distanza tra i baricentri di due fratture fosse maggiore della somma dei due raggi che le circoscrivono a meno di una tolleranza. Le coordinate del baricentro sono definite in questo modo:

$$B_x = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \ ; \quad B_y = \frac{1}{n} \cdot \sum_{i=1}^{n} y_i \ ; \quad B_z = \frac{1}{n} \cdot \sum_{i=1}^{n} z_i \tag{1}$$

Invece per calcolare il raggio si fa la differenza tra le coordinate del baricentro e quelle del vertice ad esso più lontano.

3.2 Calcolo tracce

Per determinare le tracce si ragiona paragonando a coppie le fratture, i calcoli che seguono sono stati fatti per tutte le fratture.
Come prima cosa abbiamo eseguito un controllo per eliminare quelle fratture che sono tra di loro parallele, ossia quelle fratture che hanno il prodotto vettoriale tra le loro norme (n) uguale a 0.
Si parte dall'equazione del piano (2) che contiene la frattura presa in considerazione, dove P_0, P_1 e P_2 sono tre punti della frattura:

$$x = P_0 + \alpha_0(P_2 - P_0) + \alpha_1(P_1 - P_0) \tag{2}$$

La normale di ogni frattura si ricava dalla formula:

$$n = \frac{\mathbf{u} \times \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|} \tag{3}$$

Per il calcolo della norma di una frattura sono stati utilizzati:

$$v = P_1 - P_0 \; ; \quad u = P_2 - P_0 \tag{4}$$

Per le fratture che hanno superato i controlli iniziali (prima scrematura e non parallelismo), si è calcolata la retta di intersezione tra i piani che contengono le fratture.
Si è partiti dal calcolare le normali alle due fratture, il cui prodotto vettoriale mi genera la direzione tangente T:

$$T = normal_i \times normal_j \tag{5}$$

Successivamente si è risolto sotto il vincolo che il determinante fosse diverso da zero (altrimenti non ci sarebbe intersezione) il seguente sistema lineare:

$$\begin{pmatrix} -normal_i- \\ -normal_j- \\ -T- \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ 0 \end{pmatrix}$$

Con rispettivamente:

$$d_1 = normal_i \cdot baricentro_i; \quad e \quad d_2 = normal_j \cdot baricentro_j \tag{6}$$

Per risolvere questo sistema lineare abbiamo utilizzato l'algoritmo di decomposizione PA=LU con pivoting totale *fullPivLu()*, il quale ha un costo computazionale pari a $O(n^3)$. In seguito abbiamo dovuto trovare la traccia generata dall'intersezione delle due fratture e in particolare bisogna calcolare i punti di intersezione tra la retta e i lati delle Fratture.
Per ottenere i punti di intersezione si risolve il seguente sistema:

$$\begin{cases} P = P_0 + \alpha(P_1 - P_0) \\ P = P_2 - \beta t \end{cases}$$

Dal quale si ottiene una matrice 3x2, dove la prima colonna rappresenta la differenza tra i vertici di due punti adiacenti mentre la seconda colonna è composta dai vertici della tangente calcolata precedentemente, la quale va a moltiplicare le incognite α e β, lo si può così tradurre:

$$\begin{pmatrix} x_1^1 - x_1^0 & T_1 \\ x_2^1 - x_2^0 & T_2 \\ x_3^1 - x_3^0 & T_3 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} x_1^2 - x_1^0 \\ x_2^2 - x_2^0 \\ x_3^2 - x_3^0 \end{pmatrix}$$

Per risolvere questo sistema non abbiamo usato la decomposizione PA=LU, bensì la decomposizione QR che è più adatta a risolvere un sistema che presenta una matrice m x n. La decomposizione QR ha un costo computazionale

pari a $O(2mn^2 - \frac{2}{3}n^3)$ che può essere arrotondato a $O(mn^2)$.
A questo punto dobbiamo fare due ulteriori controlli sui risultati ottenuti, ovvero che:

- il valore di $\alpha \in [0, 1]$
- il punto ottenuto risulti interno all'altra frattura intersecante, per fare ciò abbiamo dovuto risolvere la seguente equazione:

$$I = P_0 + \alpha(P_1 - P_0)$$

dove con I indichiamo il nostro punto di intersezione, il quale avendo superato i controlli diventa un punto appartenente alla traccia.

4 Verifica se le tracce sono passanti o non passanti

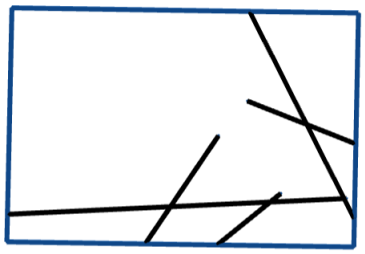
Dopo aver calcolato le tracce dobbiamo suddividerle in due gruppi, le tracce che non hanno entrambi gli estremi appartenenti ai lati della frattura sono identificate come tracce non passanti, invece quelle che hanno entrambi i punti estremi appartenenti ai lati della frattura sono passanti; in particolare quelle non passanti sono identificate nel nostro progetto come TRUE, mentre quelle passanti come FALSE.
Per determinare la tipologia è necessario controllare se gli estremi della traccia sono allineati con le coppie di vertici della Frattura, verificando se entrambi gli estremi appartengono a uno dei lati della Frattura e garantendo che le loro coordinate siano sempre comprese all'interno dei limiti del segmento corrispondente.
Il controllo che abbiamo fatto viene identificato come controllo della collinearità dei punti, ossia devo verificare questa condizione:

$$P_{\mathrm{po}} + P_{\mathrm{pe}} - L = 0$$

- con P_{pe} intendiamo la differenza tra un estremo della traccia p e un vertice della frattura e;
- con P_{po} intendiamo la differenza tra un estremo della traccia p e l'altro vertice della frattura o;
- mentre con L indichiamo la distanza tra i due vertici della frattura;

Se questa condizione è soddisfatta per entrambi gli estremi della traccia ossia se abbiamo come risposta un doppio FALSE allora la traccia è passante, se invece anche solo una delle due risposte è TRUE allora la traccia sarà non passante.

Esempio: tracce presenti nella frattura con id 2 nel dataset FR_10



5 Ordinamento tracce

Dopo aver suddiviso le tracce si deve ordinarle in base alla loro lunghezza in modo decrescente e alla loro tipologia, in particolare prima quelle passanti e successivamente quelle non passanti.
Si è deciso di utilizzare il metodo generico sort della libreria standard di C++, al quale passiamo un vettore che accoppia l'identificatore delle tracce con le lunghezze ad esse associate, ordinandole in modo da avere prima le coppie con il valore maggiore del primo elemento (cioè la lunghezza). La funzione sort della libreria standard di C++ è utilizzata per ordinare gli elementi di un intervallo, questa funzione è definita nell'header *algorithm*.
Essa ha un costo computazionale pari a $O(n\ log(n))$, poiché esegue un numero di operazioni proporzionale al numero di elementi nell'array (n) e al numero di livelli di ricorsione necessari per ordinare l'array (log(n)).

Esempio: output delle tracce sulla frattura 2 del dataset FR_10 precedentemente illustrata (0 indica una traccia passante, 1 una traccia non passante)

```
# FractureId; NumTraces
2; 5
# TraceId; Tips; Length
11; 0; 6.3578961714677895e-01
10; 1; 9.3340054310694387e-01
0; 1; 3.5618104983908305e-01
12; 1; 3.2110792003121580e-01
9; 1; 2.2298321335259683e-01
```