Analisi e Calcolo delle Tracce in un Modello di Fratture Discrete (DFN)

Abstract

Il presente lavoro descrive un programma per l'analisi e il calcolo delle tracce in un modello di fratture discrete (DFN), e per la determinazione dei sotto-poligoni generati dal taglio delle fratture con le tracce.

Il lavoro si articola nei seguenti passi essenziali:

- Definizione delle strutture dati
- Lettura ed importazione dei dati relativi al DFN
- Esclusione delle fratture lontane
- Calcolo dei parametri dei piani che contengono le fratture
- Calcolo delle rette passanti lungo l’intersezione tra i piani
- Determinazione delle tracce differenziandole in passanti e non passanti
- Stampa dei risultati ordinando le tracce in passanti e non passanti e rispetto alla lunghezza
- Determinazione dei sotto-poligoni generati per ogni frattura
- Costruzione di mesh poligonali per la memorizzazione dei dati relativi ai sotto-poligoni

Definizione delle strutture dati

Struct DFN

La struttura DFN è progettata per rappresentare le fratture, le tracce associate e le loro relazioni. La descrizione dei campi all'interno della struttura è la seguente:

- **NumberFractures**: memorizza il numero totale di fratture.
- **FractureId**: un vettore che contiene gli ID univoci di ciascuna frattura.
- **FractureCoordinates**: un vettore bidimensionale che memorizza le coordinate dei vertici per ciascuna frattura. Ogni frattura è rappresentata da un insieme di array tridimensionali che indicano le coordinate dei suoi vertici.
- **FracturesVertices**: un altro vettore bidimensionale che mappa gli ID delle fratture alle loro coordinate dei vertici.
- **NumberTraces**: memorizza il numero totale di tracce.
- **TracesId**: un vettore che contiene gli ID univoci di ciascuna traccia.
- **TracesVertices**: un vettore di array bidimensionali, dove ogni array contiene due array tridimensionali che rappresentano le coordinate dei due vertici di una traccia.
- **TracesFractures**: un vettore di array bidimensionali che associa ciascuna traccia alle due fratture che la generano.
- **FractureTraces**: un vettore bidimensionale che mappa gli ID delle fratture agli ID delle tracce.
- **Tips**: una mappa che utilizza array bidimensionali come chiavi (associazione traccia-frattura) e un valore booleano per indicare se il punto è un tip (passante) o meno.
- **Retta**: una mappa che associa ogni coppia traccia-frattura a un array bidimensionale contenente due array tridimensionali che rappresentano un punto e un vettore direttore della retta.
- **LatiIntersecati**: una mappa che associa ogni coppia traccia-frattura a un array bidimensionale che indica i lati delle fratture intersecati.
- **Sottopoligoni**: un vettore tridimensionale che mappa gli ID delle fratture alle coordinate dei sotto-poligoni risultanti dalla suddivisione delle fratture stesse.

Struct Piano

La struttura Piano è utilizzata per rappresentare i piani geometrici associati alle fratture. Ogni piano è definito dai coefficienti della sua equazione.

- **PlaneId**: un vettore che contiene gli ID univoci dei piani, che coincidono con quelli delle fratture.
- **Plane**: un vettore di array, dove ogni array contiene i coefficienti della forma generale dell'equazione del piano.

Struct PolygonalMesh

La struttura PolygonalMesh rappresenta una mesh poligonale composta da celle di diverse dimensioni (0D, 1D, 2D).

- **NumberCell0D**: un vettore che memorizza il numero di celle 0D (punti).
- **Cell0DId**: un vettore bidimensionale che contiene gli ID delle celle 0D.
- **Cell0DCoordinates**: un vettore bidimensionale di array tridimensionali che rappresentano le coordinate delle celle 0D.
- **NumberCell1D**: un vettore che memorizza il numero di celle 1D (spigoli).
- **Cell1DId**: un vettore bidimensionale che contiene gli ID delle celle 1D.
- **Cell1DVertices**: un vettore bidimensionale di array bidimensionali che rappresentano gli indici dei vertici di ogni cella 1D.
- **NumberCell2D**: un vettore che memorizza il numero di celle 2D (poligoni).
- **Cell2DId**: un vettore bidimensionale che contiene gli ID delle celle 2D.
- **Cell2DVertices**: un vettore tridimensionale che contiene gli indici dei vertici per ogni cella 2D.
- **Cell2DEdges**: un vettore tridimensionale che contiene gli indici degli spigoli per ogni cella 2D.

Importazione dei Dati

I dati rilevanti del DFN, ovvero il numero delle fratture, l'identificativo associato ad ogni frattura ed i vertici di ciascuna frattura, sono stati letti da file e, all'interno della funzione ImportFractures, memorizzati in opportune strutture dati.

Esclusione delle Fratture Lontane

Mediante la funzione FrattureVicine, che implementa il metodo delle bolle, si escludono le coppie di fratture lontane, ossia quelle che sicuramente non si intersecano e che quindi non sono rilevanti nel calcolo delle tracce.
L'idea principale è di circondare ogni frattura con una sfera (o "bolla"), e di controllare prima se queste sfere si intersecano. Se le sfere non si intersecano, allora è garantito

che le fratture al loro interno non si intersecano, permettendo di evitare calcoli più complessi.
Il centro delle sfere è il baricentro dei vertici della frattura, il raggio è la distanza massima tra il centro della sfera e i vertici della frattura.
Secondo questo metodo, vengono considerate lontane le fratture le cui bolle hanno la somma dei raggi minore della distanza dei centri.
Se accade il contrario, le due fratture vengono considerate vicine e gli identificativi della coppia memorizzate in un opportuno vettore.

Calcolo dei Parametri dei Piani delle Fratture

Nella funzione `ParametriPiano`, per ogni frattura, mediante i punti che la costituiscono, vengono calcolati i coefficienti del piano che la contiene utilizzando il metodo del determinante.
I coefficienti sono associati all'identificativo della frattura tramite una mappa che prende come chiave l'identificativo della frattura e restituisce il vettore dei coefficienti.

Calcolo delle Rette di Intersezione

Nella funzione `RettaIntersezione`, a partire dai piani di ogni coppia di fratture vicine, ossia quelle che hanno passato il test del metodo delle bolle e che sono quindi restituite dalla funzione `FrattureVicine`, si è calcolato un punto di applicazione e la direttrice della retta passante lungo l'intersezione dei due piani. La direttrice della retta è stata calcolata come il prodotto vettoriale dei due piani.
Mentre come punto di applicazione si è scelto il punto di intersezione tra i due piani e un terzo piano passante per l'origine, avente come normale la direttrice della retta già calcolata. Infine, è stata creata una mappa che associa ad ogni coppia di identificativi di fratture vicine il punto e la direttrice della retta calcolati.

Determinazione delle Tracce

Nella funzione IntersezioneLati si determinano le tracce.
Per ogni frattura viene eseguito un ciclo sui suoi vertici per determinarne i lati (segmenti). L'intersezione tra la retta di intersezione di una coppia di fratture vicine e i lati di ciascuna frattura viene appurata usando il prodotto vettoriale e scalare in questo modo: data una frattura, e dati due vertici della frattura V_1 e V_2, se P è il punto di applicazione della retta e u la direttrice, allora la retta interseca il lato di estremi V_1 e V_2 quando il prodotto scalare tra i vettori $\overline{PV_1} \times \bar{u}$ e $\overline{PV_2} \times \bar{u}$ è negativo. Intuitivamente significa che i due vertici si trovano da parti opposte rispetto alla retta.
Se la retta interseca un lato, si calcola il parametro t che identifica il punto di intersezione sulla retta.
Per cui per ogni frattura si ottengono due parametri t, e quindi per ogni coppia di fratture si ottengono due coppie di parametri t.

Per determinare se esiste una traccia comune tra le due fratture, si verifica la sovrapposizione degli intervalli dei parametri t : $[t_{00}, t_{01}]$ e $[t_{10}, t_{11}]$ associati rispettivamente alle fratture della coppia:

```
if(max(Parametri_t[0][0],Parametri_t[0][1]) >=min(Parametri_t[1][1],Parametri_t[1][0])
```

Questo passaggio assicura che la retta intersechi entrambe le fratture in un segmento comune. In caso contrario, la traccia non esiste.

I valori di t per gli estremi della traccia vengono determinati come segue:

```
T1_traccia = max(Parametri_t[0][0], Parametri_t[1][0]);
T2_traccia = min(Parametri_t[0][1], Parametri_t[1][1]);
```

Il parametro T1_traccia rappresenta il massimo dei valori t iniziali delle intersezioni, mentre T2_traccia rappresenta il minimo dei valori t finali, definendo così il segmento di intersezione comune.
A questo punto si verifica se i due valori di t per gli estremi della traccia coincidono o meno con le coppie di parametri delle due fratture, ed in base a questo si determina se è passante o non passante rispettivamente per ciascuna frattura della coppia

Memorizzazione e Stampa dei Risultati

Infine, le tracce e le informazioni relative alla loro passabilità vengono memorizzate in opportune mappe che sono utili per l'export finale:

- Una mappa per associare alle tracce i loro estremi.
- Una mappa per associare ad ogni frattura le tracce generate da essa.
- Una mappa per associare alla coppia frattura-traccia l'informazione relativa al fatto che la traccia sia passante o non passante per quella frattura.

La funzione `StampaTracce` stampa i risultati ordinando le tracce in base al fatto che siano passanti o non passanti e alla loro lunghezza.
In particolare per ogni frattura accediamo alle tracce attraverso la seconda mappa dell'elenco citato prima, e attraverso la terza con un algoritmo di sorting riusciamo ad ordinarle in passanti e non passanti. A partire da questo ordinamento calcoliamo la lunghezza di ogni traccia e con un algoritmo di sorting le ordiniamo in base alla lunghezza.
Una volta ordinate, le tracce sono state esportate su file.

Determinazione dei sotto-poligoni generati per ogni frattura

La funzione `TagliaTracce` è progettata per suddividere le fratture in sotto-poligoni basandosi sulle intersezioni con le tracce. Per ciascuna frattura viene inizializzato un vettore di sotto-poligoni. L'algoritmo, inizialmente, itera sull'insieme dei sotto-poligoni interni alla frattura, che di volta in volta verrà, eventualmente, aggiornato con i poligoni che si trovano attraverso le intersezioni con le tracce. Poi, a sua volta, itera sulle tracce che intersecano la frattura corrente e procede con due modalità diverse a seconda che la traccia corrente sia passante o non passante per il sotto-poligono considerato.

Caso passante:

Se la traccia è passante per il sotto-poligono si determinano i punti di intersezione della traccia con i lati del sotto-poligono. Questi vengono calcolati come i punti di intersezione della retta su cui giace la traccia (il punto di applicazione e la direttrice sono determinati mediante la mappa dfn.Retta utilizzata nella prima parte) e i lati del sotto-poligono con lo stesso algoritmo presente nella funzione IntersezioneLati, spiegato nella sezione Determinazione delle tracce. Una volta ottenuti, si determinano i sotto-poligoni ottenuti tagliando il sotto-poligono più "grande" con la retta su cui

giace la traccia. Si itera sui vertici del sotto-poligono più “grande” e si determinano i due sotto-poligoni attraverso il seguente algoritmo:

```
for (unsigned int i = 0; i < sottopoligono.size(); i++ ) {
 if ( estremo1 != sottopoligono[i] && estremo2 != sottopoligono[i]) {
 if (i<=J[0]) {
 Sotto1.push_back(sottopoligono[i]);
 }
 else if ( i == J[1]) {
Sotto1.push_back(estremo1);
Sotto1.push_back(estremo2);
}
else if ( i > J[1] ) {
Sotto1.push_back(sottopoligono[i]);
}
 if ( i == J[0]) {
Sotto2.push_back(estremo2);
Sotto2.push_back(estremo1);
}
else if ( i > J[0] && i <= J[1]) {
Sotto2.push_back(sottopoligono[i]);
}
}
}
```

Il ciclo itera su tutti i vertici i del sotto-poligono.
La prima condizione verifica se il vertice corrente sottopoligono[i] non è uno degli estremi della traccia: estremo1 e estremo2

Riempimento del sotto-poligono 1 (Sotto1):
Se l'indice i è minore o uguale a J[0] (primo estremo del lato intersecato), il vertice corrente viene aggiunto a Sotto1.
Se l'indice i è uguale a J[1] (secondo estremo del lato intersecato), gli estremi estremo1 e estremo2 vengono aggiunti a Sotto1, indicando che questi due vertici formano parte del lato intersecato.
Se l'indice i è maggiore di J[1], Il vertice corrente viene aggiunto a Sotto1.
Riempimento del sotto-poligono 2 (Sotto2):
Se l'indice i è uguale a J[0] (primo estremo del lato intersecato):
Gli estremi estremo2 e estremo1 vengono aggiunti a Sotto2, in ordine inverso rispetto a Sotto1, per mantenere l'orientazione corretta del sotto-poligono.
Se l'indice i è compreso tra J[0] e J[1]:
Il vertice corrente viene aggiunto a Sotto2.
Alla fine del ciclo i due sotto poligoni (Sotto1 e Sotto2) aggiornano la lista complessiva di poligoni iniziale.

Caso non passante:

Se la traccia è non passante, i sotto-poligoni, che si generano col taglio del suo prolungamento all'interno di un sotto-poligono, vengono determinati allo stesso modo del caso passante. Qui, però, prima di procedere si deve controllare se gli estremi della traccia si trovano all'interno del sotto-poligono, e ciò vien fatto sfruttando le proprietà dell'inviluppo convesso in questo modo:
si costruisce una matrice A con le coordinate dei vertici del sotto-poligono;
si costruisce un vettore b con le coordinate dell'estremo della traccia;
si risolve il sistema $A \cdot coeffs = b$ per trovare i coefficienti della combinazione convessa.
Se i coefficienti trovati soddisfano le condizioni (sono non negativi, compresi tra 0 e 1 e la loro somma è 1), allora l'estremo della traccia è considerato interno al sotto-poligono.
Una volta ottenuta la lista di sotto-poligoni la si associa alle fratture da cui si generano mediante una mappa.

Costruzione di mesh poligonali per la memorizzazione dei dati relativi ai sotto-poligoni

All'interno della funzione StampaSottopoligoni si costruiscono mesh poligonali a partire da fratture suddivise in sotto-poligoni.
Si crea un ciclo in cui si itera sulle fratture. Per ogni frattura si crea una nuova mesh (`PolygonalMesh`). Per ogni sotto-poligono della frattura corrente, si inizializzano vettori per i vertici e i lati. Si itera sui punti del sotto-poligono per mappare i vertici 3D

in indici univoci (mappa0D) e si costruiscono i lati (segmenti) del sotto-poligono, mappandoli in indici univoci (mappa1D).
Così si aggiungono i vertici e i lati 2D alla mesh e, infine, si aggiornano i contatori degli elementi della mesh (numero di celle 0D, 1D e 2D).