POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Matematica
Dipartimento di Matematica "F. Brioschi"

Flusso in un mezzo poroso fratturato: approssimazione numerica tramite Differenze Finite Mimetiche

Relatore: Dott. Anna SCOTTI
Correlatori: Prof. Luca FORMAGGIA
Dott. Paola F. ANTONIETTI

Tesi di Laurea Magistrale di:
Nicola VERZOTTI, Matr. 779867

Anno Accademico 2013-2014

Indice

Elenco delle figure

Sommario

Nella modellizzazione matematica e numerica del flusso di un fluido in un mezzo poroso è fondamentale il trattamento delle fratture, forti discontinuità presenti nel mezzo che influenzano il moto del fluido. Per le fratture si propone un modello ridotto, in cui esse vengono viste come oggetti (n-1)dimensionali immersi nel dominio. E' preferibile risolvere il problema su una griglia di calcolo che si adatti a tali fratture, con la conseguente perdita di qualità della griglia. Il metodo alle Differenze Finite Mimetiche costituisce un approccio numerico che non risente della presenza di elementi di griglia anisotropi o distorti e, in tale senso, si adatta bene al problema in esame. In questo lavoro viene proposta una discretizzazione tramite Differenze Finite Mimetiche del flusso in un mezzo poroso, descritto dalla legge di Darcy, in presenza di fratture. Lo studio ha riguardato in primo luogo l'analisi dell'errore e la stima dell'ordine di convergenza per un mezzo privo di fratture. In secondo luogo si è considerato un mezzo poroso fratturato: viene proposto un modello matematico per il flusso lungo le fratture, quindi viene discussa l'implementazione dell'accoppiamento fra mezzo poroso e fratture, si effettua l'analisi dell'errore e infine vengono risolti alcuni casi applicativi per verificare la validità del metodo.

Introduzione

Fin dagli anni settanta, le compagnie petrolifere hanno intuito l'importanza della modellistica, dell'analisi numerica e della simulazione per il raggiungimento dei propri obiettivi operativi, non solo nella fase esplorativa ma, in tempi più recenti, anche in quella produttiva. Inoltre, il notevole progresso tecnologico dall'inizio del ventesimo secolo ha trasformato e migliorato le strumentazioni che permettono di acquisire dati sul sottosuolo, rendendo possibile una comprensione più dettagliata della struttura e della storia dei giacimenti petroliferi. Lo studio del sottosuolo finalizzato alla ricerca di giacimenti richiede tuttora un enorme sforzo di ricerca e sviluppo.

In questa tesi è stato studiato il flusso di un fluido nel sottosuolo, modellato come un flusso in un mezzo poroso descritto dalla legge di Darcy [11]. Il sottosuolo è un mezzo poroso in cui sono spesso presenti delle fratture, dovute allo stato di sforzo a cui è sottoposto il materiale o, in alcuni casi, ad attività produttive. Le fratture vengono viste come zone di discontinuità del materiale, all'interno delle quali alcune caratteristiche fisiche cambiano drasticamente, modificando il moto del fluido. Uno degli aspetti più delicati di questo ambito di ricerca è proprio la modellizzazione matematica e numerica delle fratture. Le principali difficoltà derivano dal fatto che lo spessore tipico delle fratture è di qualche ordine di grandezza inferiore rispetto al mezzo in cui sono immerse, per questo motivo una griglia computazionale che ne colga le caratteristiche porterebbe a costi computazionali insostenibili. Per far fronte in questo problema in [6], [31], [7] è stato proposto un modello ridotto, che consiste nel considerare la frattura come un oggetto (n-1)dimensionale immerso in un mezzo n-dimensionale, in modo da evitare dimensioni troppo elevate.

Un secondo problema che si pone in fase di discretizzazione è rappresentato dalla qualità della griglia di calcolo. Nei casi applicativi di interesse è necessario che la griglia si adatti alle fratture [33], [20], con la conseguente generazione di elementi fortemente anisotropi e allungati. Nella maggior parte degli approcci numerici, questo rappresenta un ostacolo alla convergenza del metodo, ma questo limite non vale per le Differenze Finite Mimetiche.

Il metodo delle Differenze Finite Mimetiche (DFM) racchiude tutte le buone

proprietà dei metodi di discretizzazione più avanzati: come i Volumi Finiti si adatta a griglie poligonali e poliedriche generali, come gli Elementi Finiti ha ordini di convergenza alti [24]. Il fatto che le DFM si adattino in modo robusto a griglie distorte ed anisotrope ha fatto sì che negli ultimi anni gli studi applicativi di tale metodo siano aumentati notevolmente, alcuni ambiti di studio recente sono i problemi di diffusione, elettromagnetismo, meccanica dei continui, dinamica dei gas, fluidodinamica, si veda, per esempio [21].

Un altro importante vantaggio risiede nel fatto che gli operatori discreti mimetici preservano (o "mimano") alcune importanti proprietà degli operatori continui, tra cui il principio del massimo. Inoltre, nella costruzione delle norme degli spazi funzionali, si impongono implicitamente due condizioni: consistenza e stabilità. Tutto questo si traduce in schemi con buone proprietà di convergenza.

Il lavoro di questa tesi è strutturato come segue:

Capitolo 1. Il capitolo introduttivo riporta i principi e i fondamenti del metodo alle Differenze Finite Mimetiche. Dopo aver descritto le idee che ne stanno alla base ed aver fornito alcune ipotesi di regolarità sulla griglia computazionale, vengono definiti gli spazi funzionali e si procede alla costruzione dei due elementi principali del metodo mimetico: gli operatori discreti e i prodotti interni degli spazi. Questi ultimi saranno definiti in modo tale che soddisfino due condizioni: consistenza e stabilità, fondamentali per la continuità e la coercività della forma bilineare del formulazione debole mimetica.

Capitolo 2. Nel secondo capitolo viene applicato il metodo alle Differenze Finite Mimetiche allo studio del flusso di un fluido in un mezzo poroso; a tal scopo, si introduce il problema di Darcy. In seguito si procede alla discretizzazione mimetica di tale problema: si definiscono gli spazi funzionali in cui si lavora e i corrispettivi gradi di libertà, gli operatori discreti e il prodotto interno degli spazi. Viene poi fornita una validazione di tale discretizzazione: dopo aver introdotto alcuni casi test, si riportano i risultati numerici riguardanti il calcolo dell'errore di approssimazione e l'ordine di convergenza del metodo.

Capitolo 3. In questo capitolo si considera un mezzo poroso in cui sono presenti delle fratture. In primo luogo si presenta la derivazione del modello matematico per mezzi porosi con fratture, in particolare vengono descritti il modello per il mezzo poroso, il modello per le fratture e le condizioni che accoppiano tale modello con la matrice porosa circostante. Una volta fornito il modello matematico si propongono due tecniche di discretizzazione: ai Volumi Finiti e alle Differenze Finite Mimetiche. Per il secondo metodo viene fornita una breve analisi dell'errore di approssimazione, ricavando gli ordini di convergenza per un caso test di cui è nota la soluzione analitica al variare dei parametri fisici.

Infine vengono presentati altri casi test di complessità crescente che mostrano la validità del modello proposto.

Capitolo 4. In questa parte viene descritto il codice `C++` che è stato sviluppato per l'assemblaggio e la risoluzione dei sistemi lineari descritti nei capitoli 2 e 3. Viene fatto riferimento alle classi che portano alla generazione di una griglia che risulti utile ed efficiente e vengono descritti gli algoritmi usati per l'assemblaggio delle matrici.

Infine, l'ultimo capitolo è dedicato alle conclusioni e agli sviluppi futuri.

Capitolo 1

Differenze Finite Mimetiche

1.1 Principi e fondamenti dei metodi mimetici

Per comprendere quali sono i principi base della discretizzazione mimetica, si consideri un caso semplice, l'equazione di Poisson unidimensionale con condizioni al bordo di Dirichlet omogenee:

$$-\frac{d^2p}{dx^2} = b(x), \quad x \in (0,1), \tag{1.1}$$

$$p(0) = p(1) = 0,$$

dove $b(x)$ è un termine di sorgente sufficientemente regolare, per esempio si prende $b \in L^2(\Omega)$. Si può scrivere l'equazione del secondo ordine come un sistema di due equazioni del primo ordine:

$$u = -\frac{dp}{dx}, \quad \frac{du}{dx} = b, \tag{1.2}$$

introducendo la variabile u.

Si consideri ora una griglia di calcolo uniforme con $(n+1)$ nodi $x_i = (i-1)\Delta x$, dove $\Delta x = 1/n$ (Figura 1.1). Il passo successivo è la scelta dei gradi di libertà: alla funzione continua u si associa la funzione discreta $u_h \in \mathbb{R}^{n+1}$, che la approssima nei nodi della griglia, cioè $u_h = (u_i)_{i=1}^{n+1}$ e $u_i \approx u(x_i)$; per quanto riguarda la funzione p, essa è approssimata nel centro degli intervalli della griglia da $p_h \in \mathbb{R}^n$, cioè $p_h = (p_{i+1/2})_{i=1}^{n}$ e $p_{i+1/2} \approx p(x_{i+1/2})$. In questo modo, u è una variabile di faccia, essendo definita negli estremi degli intervalli, mentre p è una variabile di cella, essendo definita nel centro degli intervalli.

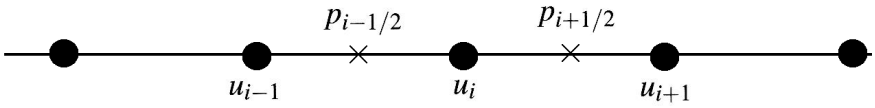


Figura 1.1: Gradi di libertà della discretizzazione mimetica in 1D

La discretizzazione classica di (1.2) tramite Differenze Finite è:

$$\begin{aligned} u_i &= -\frac{p_{i+1/2} - p_{i-1/2}}{\Delta x}, \qquad i = 1, ..., n+1, \\ \frac{u_{i+1} - u_i}{\Delta x} &= b(x_{i+1/2}), \qquad i = 1, ..., n, \end{aligned} \tag{1.3}$$

dove si impone $p_{1/2} = p_{n+3/2} = 0$ come condizioni al bordo. I metodi mimetici sono stati sviluppati originariamente usando operatori a differenze finite e da qui deriva la prima parte del nome del metodo. Partendo dallo schema (1.3), si possono riscrivere formalmente le equazioni introducendo i simboli $\widetilde{\nabla}_h$ e div_h per gli operatori gradiente e divergenza discreti, ottenendo:

$$u_h = -\widetilde{\nabla}_h p_h, \qquad \mathrm{div}_h u_h = b_h,$$

dove $b_h = (b_{i+1/2})_{i=1}^n$, mentre gli operatori sono definiti come:

$$(\widetilde{\nabla}_h p_h)_i = \frac{p_{i+1/2} - p_{i-1/2}}{\Delta x}, \qquad (\mathrm{div}_h u_h)_{i+1/2} = \frac{u_{i+1} - u_i}{\Delta x}. \tag{1.4}$$

Moltiplicando la prima equazione di (1.3) per u_i e la seconda per $p_{i+1/2}$ si può facilmente verificare che:

$$\Delta x \sum_{i=1}^{n+1} \frac{p_{i+1/2} - p_{i-1/2}}{\Delta x} u_i = -\Delta x \sum_{i=1}^{n} \frac{u_{i+1} - u_i}{\Delta x} p_{i+1/2}. \tag{1.5}$$

Usando le definizioni introdotte in (1.4), si ha la seguente espressione:

$$\Delta x \sum_{i=1}^{n+1} (\widetilde{\nabla}_h p_h)_i \; u_i = -\Delta x \sum_{i=1}^{n} (\mathrm{div}_h u_h)_{i+1/2} \; p_{i+1/2}. \tag{1.6}$$

Le relazioni (1.5) e (1.6) sono formule di integrazione per parti discrete, ossia costituiscono l'analogo discreto della formula di Green:

$$\int_0^1 \frac{dp}{dx} u dx = -\int_0^1 p \frac{du}{dx} dx \quad \forall u \in H^1(0,1), \; p \in H_0^1(0,1).$$

La dualità tra il gradiente discreto e la divergenza discreta rappresenta la proprietà naturale dello schema a differenze finite (1.3). Ne derivano conseguenze

molto utili: per esempio, l'eliminazione dell'incognita u_i porta ad un sistema di equazioni con una matrice simmetrica e definita positiva, che a sua volta implica l'esistenza e unicità della soluzione p_h.

Il primo principio che deve avere uno schema mimetico è la conservazione della proprietà di dualità discreta, anche in due e tre dimensioni e su griglie formate da poligoni e poliedri arbitrari. Questo non si riesce ad ottenere se si discretizzano gli operatori divergenza e gradiente in modo indipendente tra loro. Si consideri infatti l'espressione formale della dualità discreta:

$$[\widetilde{\nabla}_h p_h, u_h]_{\mathscr{F}_h} = -[p_h, \mathrm{div}_h u_h]_{\mathscr{P}_h} \quad \forall p_h \in \mathscr{P}_h, \forall u_h \in \mathscr{F}_h. \tag{1.7}$$

Nel semplice esempio 1D introdotto $\mathscr{P}_h = \mathbb{R}^n$ e $\mathscr{F}_h = \mathbb{R}^{n+1}$ sono gli spazi in cui sono definite, rispettivamente, p_h e u_h e $[\cdot,\cdot]$ indica il prodotto interno di questi spazi:

$$[v_h, u_h]_{\mathscr{F}_h} = \sum_{i=1}^{n+1} \Delta x \; v_i u_i \quad \forall v_h, u_h \in \mathscr{F}_h$$

e

$$[q_h, p_h]_{\mathscr{P}_h} = \sum_{i=1}^{n} \Delta x \; p_{i+1/2} q_{i+1/2} \quad \forall q_h, p_h \in \mathscr{P}_h.$$

Un altro modo per rappresentare i prodotti interni è usare le matrici di massa:

$$[v_h, u_h]_{\mathscr{F}_h} = v_h^T \mathsf{M}_{\mathscr{F}_h} u_h, \quad [p_h, q_h]_{\mathscr{P}_h} = q_h^T \mathsf{M}_{\mathscr{P}_h} p_h.$$

Nell'esempio unidimensionale trattato, le matrici di massa $\mathsf{M}_{\mathscr{F}_h}$ e $\mathsf{M}_{\mathscr{P}_h}$ sono matrici diagonali, più precisamente l'identità moltiplicata per Δx. Sfruttando la definizione di queste matrici nell'equazione (1.7), si ottiene:

$$\widetilde{\nabla}_h = -\mathsf{M}_{\mathscr{F}_h}^{-1} \mathrm{div}_h^T \mathsf{M}_{\mathscr{P}_h}. \tag{1.8}$$

Si può notare, quindi, che l'operatore divergenza discreto e le matrici dei prodotti interni definiscono univocamente l'operatore gradiente discreto. Per ogni diversa definizione di div_h^T, $\mathsf{M}_{\mathscr{P}_h}$ e $\mathsf{M}_{\mathscr{F}_h}$ si ottiene un nuovo schema, in cui gli operatori divergenza e gradiente restano aggiunti tra loro, a meno del segno. Su una griglia non strutturata, solamente definendo l'operatore gradiente discreto tramite la proprietà di dualità (1.7) si ottiene, dopo l'eliminazione di u_h, un sistema di equazioni algebriche per p_h con una matrice simmetrica e definita positiva. Riassumendo, in uno schema mimetico per un'Equazione differenziale a Derivate Parziali (EDP) generica, alcuni operatori discreti (detti *primali*) vengono definiti direttamente, come in (1.2), mentre gli altri (detti *duali* o *derivati*) vengono definiti sfruttando la dualità discreta, come in (1.8).

Sfortunatamente nelle griglie non strutturate la costruzione delle matrici di massa è non banale. Perciò, in fase di costruzione, si usano due principi

addizionali, dovuti a condizioni di consistenza e stabilità. Per semplicità, si illustrano questi principi applicati alla costruire della matrice $M_{\mathscr{F}_h}$ nell'esempio unidimensionale.

Tipicamente la matrice viene assemblata a partire dai blocchi M_i, associati ognuno ad un intervallo della griglia $[x_i, x_{i+1}]$:

$$\mathsf{M}_{\mathscr{F}_h} = \sum_{i=1}^{n} \mathscr{N}_i^T \mathsf{M}_i \mathscr{N}_i, \tag{1.9}$$

dove $\mathscr{N}_i$ sono le stesse matrici che si assemblano nel metodo degli elementi finiti e consentono di trasferire le informazioni delle matrici locali in quella globale: ad ogni riga di $\mathscr{N}_i$ (e ad ogni colonna di $\mathscr{N}_i^T$) è associato un nodo della griglia e l'unica componente non nulla della riga (colonna) occupa la posizione corrispondente alla numerazione globale del nodo in questione.

Come anticipato precedentemente, nel caso unidimensionale la matrice di massa è diagonale; quindi, si può facilmente verificare che essa soddisfa:

$$\mathsf{M}_i^{DF} = \frac{\Delta x}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Le matrici locali M_i sono 2×2 e tali che:

$$(v_i, v_{i+1})\, \mathsf{M}_i \begin{pmatrix} u_i \\ u_{i+1} \end{pmatrix} \approx \int_{x_i}^{x_{i+1}} vu dx.$$

In particolare con un calcolo diretto, si può mostrare che

$$(v_i, v_{i+1})\, \mathsf{M}_i^{DF} \begin{pmatrix} u_i \\ u_{i+1} \end{pmatrix} = \Delta x (v_i u_i + v_{i+1} u_{i+1}) = \int_{x_i}^{x_{i+1}} vu dx + \mathrm{O}((\Delta x)^3).$$

Quindi, le matrici di massa locali rappresentano una regola di quadratura e, in questo modo, si può provare che uno schema a differenze finite è del secondo ordine sia per p che per u; ciò è valido anche per griglie poligonali e poliedriche. La qualità dell'approssimazione locale degli integrali di cella influenza l'accuratezza dello schema mimetico risultante.

Resta da mostrare come la matrice locale M_i può essere derivata dalle due condizioni (stabilità e consistenza), che possono essere generalizzate a dimensioni arbitrarie. La matrice 2×2 M_i simmetrica e definita positiva soddisfa la condizione di consistenza dello schema mimetico se:

$$\left(v^0, v^0\right) \mathsf{M}_i \begin{pmatrix} u_i \\ u_{i+1} \end{pmatrix} = \int_{x_i}^{x_{i+1}} v^0 u^1 dx \quad \forall v^0, \forall (u_i, u_{i+1}), \tag{1.10}$$

dove v^0 è una funzione costante e u^1 è una funzione lineare. Se si prende v^0 come il valor medio di v nell'intervallo $[x_i, x_{i+1}]$ e u^1 la retta che assume i valori u_i e u_{i+1} negli estremi dell'intervallo si ottiene:

$$\int_{x_i}^{x_{i+1}} v^0 u^1 dx = \int_{x_i}^{x_{i+1}} v u dx + \mathrm{O}((\Delta x)^2).$$

Chiedendo un grado di accuratezza minore si ricava comunque uno schema del secondo ordine e, soprattutto, si possono associare gli integrali con i prodotti interni locali, ottenendo la seguente identità:

$$\int_{x_i}^{x_{i+1}} v^0 u^1 dx = \frac{\Delta x}{2} v^0 (u_i + u_{i+1}) = \left(v^0, v^0\right) \mathsf{M}_i^{DF} \begin{pmatrix} u_i \\ u_{i+1} \end{pmatrix},$$

che vale per ogni v^0, u_i e u_{i+1}.
Quindi M_i^{DF} soddisfa la condizione di consistenza. Un altro semplice esempio è il caso degli elementi finiti di Raviart-Thomas di ordine zero 1D, la cui corrispondente matrice M_i^{RT} definita come:

$$\mathsf{M}_i^{RT} = \frac{\Delta x}{6} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix},$$

soddisfa (1.10).

Apparentemente, le matrici M_i che soddisfano la condizione di consistenza possono essere sia definite positive che indefinite. Dato che si vuole ottenere una matrice globale simmetrica e definita positiva, bisogna escludere il caso di soluzione indefinita, introducendo una condizione di stabilità. Si richiede che esistano due costanti positive $\sigma_\star$ e $\sigma^\star$ indipendenti da Δx tali che:

$$\sigma_\star \Delta x (u_i^2 + u_{i+1}^2) \leq (u_i, u_{i+1}) \, \mathsf{M}_i \begin{pmatrix} u_i \\ u_{i+1} \end{pmatrix} \leq \sigma^\star \Delta x (u_i^2 + u_{i+1}^2) \quad \forall (u_i, u_{i+1}).$$

Per le differenze finite la condizione vale per $\sigma_\star = \sigma^\star = 1$, mentre per gli elementi di Raviart-Thomas vale per $\sigma_\star = 1$ e $\sigma^\star = 3$.

Le condizioni di consistenza e stabilità permettono di derivare approssimazioni accurate dell'integrale L^2 di funzioni scalari o vettoriali, rappresentate da gradi di libertà generici, tra cui valori nodali, componenti normali e tangenziali, valori di faccia o di cella, su griglie poligonali e poliedriche. Va sottolineato che le due condizioni permettono di derivare la rappresentazione discreta di forme bilineari molto più generali, anche con derivate di ordine maggiore di uno. Tra queste si possono elencare gli operatori dell'equazione di diffusione stazionaria in forma primale e in forma mista, l'equazione di diffusione-trasporto, il problema di Stokes, il problema dell'elasticità lineare, problemi magnetostatici...

Qualsiasi sia la forma bilineare considerata, la strategia per la discretizzazione è sempre analoga: si sceglie un opportuno spazio polinomiale in cui approssimare l'incognita e i suoi opportuni gradi di libertà; in questo modo il termine noto della condizione di consistenza può essere scritto in termini dei gradi di libertà. Di conseguenza, dopo aver definito due matrici N_i e R_i, calcolate usando le proprietà geometriche della cella e i coefficienti del problema, la condizione (1.10) può essere sempre riscritta come $\mathsf{M}_i\mathsf{N}_i = \mathsf{R}_i$ (i dettagli saranno riportati in seguito). Una volta noti questi termini, si deve cercare M_i che soddisfi la condizione di stabilità.

1.2 Griglia di calcolo e operatori di proiezione

Prima di affrontare la costruzione degli operatori mimetici e del prodotto interno, è necessario fare alcune considerazioni sulla griglia di calcolo e introdurre opportuni operatori di proiezione. Dato che il presente lavoro è stato svolto nel caso bidimensionale, ci si limita al caso di griglie poligonali 2D, ma tutti i concetti possono essere generalizzati al caso 3D.

Sia $\Omega \subset \mathbb{R}^2$ il dominio e $\Omega_h \subset \Omega$ la sua approssimazione discreta, esso deve essere un'approssimazione poligonale di Ω, scelto in modo tale che tutti i vertici di Ω_h collocati su $\partial\Omega_h$ appartengano anche alla frontiera di Ω. Con un lieve abuso di notazione si indica con Ω_h anche la partizione del dominio computazionale stesso in poligoni, i quali sono denotati con P. Si assume che tale partizione sia conforme, ovvero che l'intersezione tra due elementi distinti sia data da uno o più vertici della griglia, da uno o più lati, oppure sia vuota. Il diametro di un oggetto generico Q (poligono o lato) è denotato con h_Q, mentre h è definito come:

$$h = \max_{\mathsf{P}\in\Omega_h} h_{\mathsf{P}}.$$

Gli insiemi $\mathscr{P}$, $\mathscr{E}$, $\mathscr{V}$ rappresentano, rispettivamente, gli insiemi degli elementi, dei lati (che in due dimensioni coincide con l'insieme delle facce) e dei vertici, mentre le loro restrizioni sull'elemento P sono $\mathscr{P}_{\mathsf{P}}$, $\mathscr{E}_{\mathsf{P}}$ e $\mathscr{V}_{\mathsf{P}}$. Infine, sia $\mathbf{n}^{\mathsf{e}}$ il versore normale uscente dal lato e; in particolare, se e appartiene all'elemento P, la normale uscente è $\mathbf{n}^{\mathsf{e}}_{\mathsf{P}}$.

Si assume che la griglia computazionale soddisfi le seguenti proprietà: esistono due numeri reali e positivi, $\mathscr{N}^{\mathrm{S}}$ e ρ_{S}, e una sottopartizione conforme di Ω_h T_h costituita da triangoli, tali che

- (**HP1**) ciascun poligono $\mathsf{P} \in \Omega_h$ ammette una decomposizione $T_{h|\mathsf{P}}$ costituita al più da $\mathscr{N}^{\mathrm{S}}$ elementi.

- (**HP2**) ciascun triangolo $T \in \ T_h$ è *regolare* nel senso che il rapporto tra il raggio r_T della circonferenza inscritta in esso e il suo diametro h_T ha come limite inferiore la costante ρ_{S}, ossia

$$\frac{r_T}{h_T} \geq \rho_{\mathrm{S}} \geq 0.$$

Queste due ipotesi sono necessarie per evitare situazioni patologiche, quali ad esempio poligoni che si restringono ad imbuto e vanno a costituire una sorta di clessidra oppure poligoni a forma di pettine con denti che si assottigliano via via, come in Figura 1.2. Inoltre, da esse discendono utili conseguenze:

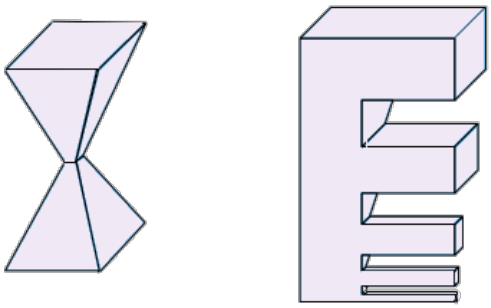


Figura 1.2: Esempi 3D di elementi che non soddisfano le condizioni [15]

- (**P1**) esistono due numeri interi e positivi $\mathrm{N_v}$ e $\mathrm{N_e}$ tali che ogni elemento P ha al più $\mathrm{N_e}$ lati ed ogni lato ha al più $\mathrm{N_v}$ vertici.
- (**P2**) le quantità geometriche relative ad ogni elemento $\mathsf{P} \in \Omega_h$ sono uniformemente limitate da sopra e da sotto. Più precisamente, esistono due costanti $a_\star$ e $l_\star$, che dipendono solo da $\mathscr{N}^{\mathrm{S}}$ e ρ_{S}, tali per cui per ogni elemento P e per ogni lato $\mathsf{e} \in \partial\mathsf{P}$ valgono le seguenti disuguaglianze:

$$a_\star h_{\mathsf{P}}^2 \leq |\mathsf{P}| \qquad l_\star h_{\mathsf{P}} \leq |\mathsf{e}|.$$

- (**P3**) esiste un numero positivo $\tau_\star$ tale per cui in ciascun elemento P è contenuta una circonferenza di centro $C_{\mathsf{P}} \in \mathsf{P}$ e raggio pari a $\tau_\star h_{\mathsf{P}}$ e tale per cui per ogni punto di P esiste un segmento interamente incluso nell'elemento che lo collega col centro della circonferenza stessa (Figura 1.3).
- (**P4**) esiste una costante C^{Agm} indipendente da h_{P} tale che:

$$\sum_{\mathsf{e}\in\partial\mathsf{P}} |\phi|^2_{L^2(\mathsf{e})} \leq C^{Agm} \left(h_{\mathsf{P}}^{-1} \|\phi\|_{L^2(\mathsf{P})} + h_{\mathsf{P}} \|\nabla\phi\|_{L^2(\mathsf{P})} \right)$$

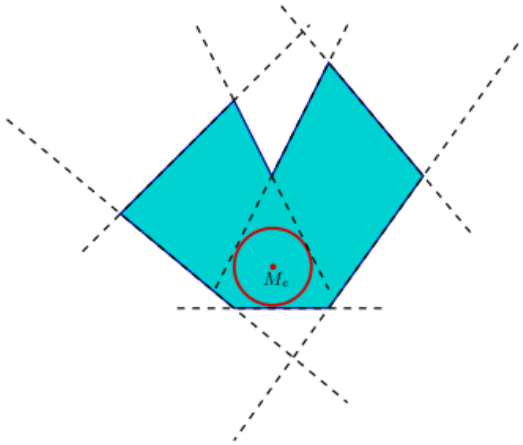


Figura 1.3: Esempio di cella 2D che soddisfa la condizione (**P3**)

per ogni funzione $\phi \in H^1(\mathsf{P})$. Questa relazione prende il nome di disuguaglianza di Agmon (si veda per esempio [10]).

Il passo successivo è definire quali sono i gradi di libertà della griglia. In generale, data una griglia in tre dimensioni, si possono definire quattro differenti tipi di gradi di libertà, che sono associati a differenti oggetti della griglia: i vertici, i lati, le facce e le celle (Figura 1.4). Essi sono:

- Un campo discreto *vertex-based* p_h. Esso si definisce associando un numero $p_{h,\mathsf{V}}$ ad ogni vertice V e può essere interpretato come l'approssimazione della funzione scalare $p(\mathbf{x})$ nei vertici di griglia, cioè $p_{h,\mathsf{V}} = p(\mathbf{x}_\mathsf{V})$. L'insieme dei gradi di libertà è lo spazio $\mathscr{V}_h$.

- Un campo discreto *edge-based* $\mathbf{u}_h$. Esso si definisce associando un numero $u_{h,\mathsf{e}}$ ad ogni lato e e può essere interpretato come l'approssimazione della componente tangenziale della funzione vettoriale $\mathbf{u}(\mathbf{x})$ nei lati di griglia. L'insieme dei gradi di libertà è lo spazio $\mathscr{E}_h$.

- Un campo discreto *face-based* $\mathbf{u}_h$. Esso si definisce associando un numero $u_{h,\mathsf{f}}$ ad ogni faccia f e può essere interpretato come l'approssimazione della componente normale della funzione vettoriale $\mathbf{u}(\mathbf{x})$ nelle facce di griglia. L'insieme dei gradi di libertà è lo spazio $\mathscr{F}_h$.

- Un campo discreto *cell-based* p_h. Esso si definisce associando un numero $p_{h,\mathsf{P}}$ ad ogni elemento P e può essere interpretato come l'approssimazione della funzione scalare $p(\mathbf{x})$ nel centro degli elementi di griglia, cioè $p_{h,\mathsf{P}} = p(\mathbf{x}_\mathsf{P})$. L'insieme dei gradi di libertà è lo spazio $\mathscr{P}_h$.

In particolare, nel caso bidimensionale, $\mathscr{E}_h$ e $\mathscr{F}_h$ coincidono, quindi è equivalente usare la prima o la seconda notazione. Per questo lavoro si farà riferimento ai gradi di libertà definiti nelle facce (o equivalentemente nei lati) e nelle celle e quindi agli spazi $\mathscr{F}_h \equiv \mathscr{E}_h$ e $\mathscr{P}_h$.

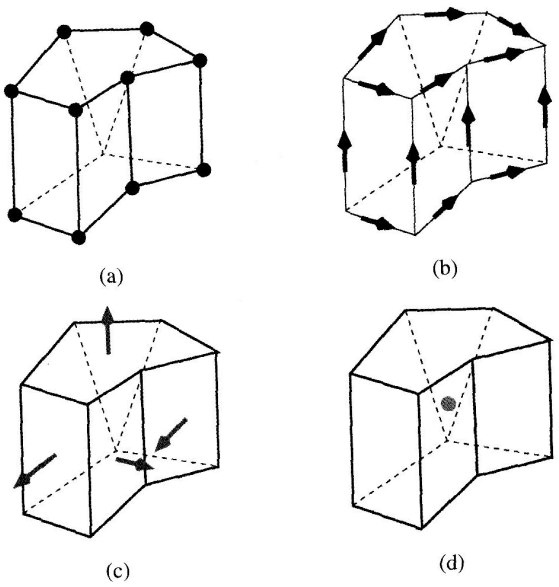


Figura 1.4: Caso 3D: Campo discreto *vertex-based* (a); Campo discreto *edge-based* (b); Campo discreto *face-based* (c); Campo discreto *cell-based* (d)

Operatori di proiezione

Gli operatori di proiezione hanno lo scopo di trasportare spazi di funzioni scalari o vettoriali sufficientemente regolari negli spazi discreti $\mathscr{V}_h$, $\mathscr{E}_h$ e $\mathscr{P}_h$; in altre parole la loro immagine è un'approssimazione discreta dei campi scalari e vettoriali continui. In generale, essi vengono indicati con il simbolo $\Pi^{\mathscr{S}}$, mentre la loro restrizione ad un generico elemento della griglia, Q, è denotata con $\Pi^{\mathscr{S}}_Q$.

Spazio dei vertici. Sia $p(\mathbf{x})$ una funzione scalare abbastanza regolare da poterne calcolare i valori puntuali, per esempio $p \in H^1(\Omega) \cap C^0(\Omega)$. La sua approssimazione $p_h \in \mathscr{V}_h$ si ottiene applicando l'operatore di proiezione $\Pi^{\mathscr{V}}$ a p:

$$p_h^I = \Pi^{\mathscr{V}}(p) = (p_{\mathsf{v}})_{\mathsf{v}\in\mathscr{V}}, \qquad p_{\mathsf{v}}^I = p(\mathbf{x}_{\mathsf{v}}). \tag{1.11}$$

La dimensione dello spazio $\mathscr{V}_h$ è pari al numero dei vertici della griglia. L'operatore di proiezione locale $\Pi^{\mathscr{V}}_Q(p)$, dove Q è un generico elemento della griglia

ha una definizione analoga:

$$\Pi_Q^{\mathscr{V}}(p) = (p_{\mathsf{v}})_{\mathsf{v}\in Q}.$$

Spazio delle facce (o dei lati). Sia $\mathbf{u}(\mathbf{x})$ una funzione vettoriale abbastanza regolare, per cui siano ben definiti gli integrali della sua componente normale su tutte le facce (lati) della griglia, per esempio $\mathbf{u} \in (L^s(\Omega))^d, s > 2$, con divergenza in $L^2(\Omega)$. La sua approssimazione $\mathbf{u}_h \in \mathscr{F}_h$ si ottiene applicando l'operatore di proiezione $\Pi^{\mathscr{F}}$ a $\mathbf{u}$:

$$\mathbf{u}_h^I = \Pi^{\mathscr{F}}(\mathbf{u}) = (u_{\mathsf{f}})_{\mathsf{f}\in\mathscr{F}}, \qquad u_{\mathsf{f}}^I = \frac{1}{|\mathsf{f}|}\int_{\mathsf{f}} \mathbf{u}\cdot\mathbf{n}_{\mathsf{f}}\, dS. \tag{1.12}$$

La dimensione dello spazio $\mathscr{F}_h$ è pari al numero delle facce della griglia. L'operatore di proiezione locale $\Pi_Q^{\mathscr{F}}(\mathbf{u})$, dove Q è un generico elemento della griglia ha una definizione analoga:

$$\Pi_Q^{\mathscr{F}}(\mathbf{u}) = (u_{\mathsf{f}})_{\mathsf{f}\in Q}.$$

Spazio delle celle. Sia $p(\mathbf{x})$ una funzione scalare abbastanza regolare, in modo tale che si possa calcolare il suo integrale su tutti i sottoinsiemi compatti di Ω, per esempio $p \in L^1(\Omega)$. La sua approssimazione $p_h \in \mathscr{P}_h$ si ottiene applicando l'operatore di proiezione $\Pi^{\mathscr{P}}$ a p:

$$p_h^I = \Pi^{\mathscr{P}}(p) = (p_{\mathsf{P}})_{\mathsf{P}\in\mathscr{V}}, \qquad p_{\mathsf{P}}^I = \frac{1}{|\mathsf{P}|}\int_{\mathsf{P}} p\, dV. \tag{1.13}$$

La dimensione dello spazio $\mathscr{P}_h$ è pari al numero delle celle della griglia.

1.3 Operatori mimetici

Per quanto detto prima, nei metodi mimetici si costruiscono in primo luogo gli *operatori primali* partendo dai due principi citati e, da questi, tramite una formula di integrazione per parti discreta, si ricavano gli *operatori derivati*. Alcuni esempi del primo tipo possono essere il gradiente, la divergenza e il rotore, ma per l'uso che si farà in questo lavoro, ci si limita a trattare la divergenza; il risultato finale sarà un operatore discreto che riproduce (o "mima") le identità fondamentali del calcolo continuo e garantisce la simmetria e definita positività [21].

Per definire la divergenza come operatore primale, si parte dal Teorema di Stokes:

$$\int_V \mathrm{div}\mathbf{u}\, d\mathrm{V} = \int_{\partial V} \mathbf{u}\cdot\mathbf{n}\, d\mathrm{S}, \tag{1.14}$$

dove $\mathbf{u}$ è un campo vettoriale ed $\mathbf{n}$ è il versore normale alla superficie ∂V. In secondo luogo, si deve formulare una relazione di dualità che permetta di ricavare l'operatore derivato. Per il caso continuo si sfrutta il Teorema di Green:

$$\int_V p \, \mathrm{div}\mathbf{u} \, d\mathrm{V} = -\int_V \nabla p \cdot \mathbf{u} \, d\mathrm{V} + \int_{\partial V} p(\mathbf{u} \cdot \mathbf{n}) \, d\mathrm{S},$$

che, per condizioni al bordo di Dirichlet omogenee, si riduce a:

$$\int_V p \, \mathrm{div}\mathbf{u} \, d\mathrm{V} = -\int_V \nabla p \cdot \mathbf{u} \, d\mathrm{V} \qquad \forall p \in H_0^1(\mathrm{V}), \forall \mathbf{u} \in [H_0^1(\mathrm{V})]^d. \tag{1.15}$$

Nell'approccio mimetico, la dualità degli operatori differenziali deve essere combinata con i coefficienti del problema: per esempio, sia K un tensore definito positivo (nel seguito verranno fornite alcune condizioni che deve soddisfare), allora la formula (1.15) viene riscritta come:

$$\int_V p \, \mathrm{div}\mathbf{u} \, d\mathrm{V} = -\int_V \mathsf{K}^{-1}(\mathsf{K}\nabla p) \cdot \mathbf{u} \, d\mathrm{V}, \tag{1.16}$$

ottenendo una relazione di dualità pesata, tra gli operatori div e $\mathsf{K}\nabla$ che tengono conto dei parametri del problema fisico; nello stesso modo saranno messi in relazione i loro analoghi discreti. In generale, sia $\mathscr{D} : \mathsf{S} \to \mathsf{S}^\star$ l'operatore a sinistra dell'uguale in (1.15), allora l'operatore a destra dell'uguale è il suo duale, $\mathscr{D}^\star : \mathsf{S}^\star \to \mathsf{S}$ e la formula (1.15) può essere scritta in modo astratto come:

$$[\mathscr{D}u, v]_{\mathsf{S}^\star} = [u, \mathscr{D}^\star v]_{\mathsf{S}} \quad \forall u \in \mathsf{S}, \ \forall v \in \mathsf{S}^\star, \tag{1.17}$$

dove $[\cdot, \cdot]$ indica il prodotto interno dei rispettivi spazi. Si noti che i termini vengono pesati in modo tale da tenere conto anche dei coefficienti del problema. Quando si discretizza, si deve ottenere una relazione simile, cioè, detti $\mathscr{D}_h$ e $\mathscr{D}_h^\star$ gli operatori discreti che agiscono negli spazi discreti S_h e $\mathsf{S}_h^\star$, a loro volta dotati dei prodotti interni $[\cdot, \cdot]_{\mathsf{S}_h}$ e $[\cdot, \cdot]_{\mathsf{S}_h^\star}$, si ha la seguente relazione di dualità:

$$[\mathscr{D}_h u_h, v_h]_{\mathsf{S}_h^\star} = [u_h, \mathscr{D}_h^\star v_h]_{\mathsf{S}_h} \quad \forall u_h \in \mathsf{S}_h, \ \forall v_h \in \mathsf{S}_h^\star, \tag{1.18}$$

che definisce univocamente l'operatore duale $\mathscr{D}_h^\star$. Un'importante proprietà dell'operatore duale discreto è la seguente: si prenda $v_h = \mathscr{D}_h u_h$, allora si ottiene la seguente disuguaglianza:

$$[\mathscr{D}_h^\star(\mathscr{D}_h u_h), u_h]_{\mathsf{S}_h} = [\mathscr{D}_h u_h, \mathscr{D}_h u_h]_{\mathsf{S}_h^\star} \geq 0 \quad \forall u_h \in \mathsf{S}_h.$$

Questo si traduce nel fatto che, dalla formulazione della dualità discreta (1.18), qualsiasi sia il prodotto interno dello spazio, l'operatore duale discreto che ne deriva è sempre consistente e, per questa ragione, il sistema lineare associato preserva la simmetria e la definita positività del problema continuo.

Si può procedere, ora, alla definizione operativa della divergenza discreta e del suo duale, il gradiente discreto. Se si applica il Teorema di Stokes (1.14) ad una singola cella P della griglia si ottiene:

$$\int_{\mathsf{P}} \mathrm{div}\mathbf{u}\ d\mathrm{V} = \int_{\partial\mathsf{P}} \mathbf{u}\cdot\mathbf{n}_{\mathsf{P}}\ d\mathrm{S}.$$

Per rispettare l'analogia al caso continuo, l'operatore discreto (applicato a $\mathbf{u}_h$ definito sulle facce di P) avrà una definizione simile:

$$(\mathrm{div}_h\mathbf{u}_h)_{\mathsf{P}} = \frac{1}{|\mathsf{P}|}\sum_{\mathsf{f}\in\mathscr{F}_{\mathsf{P}}} \alpha_{\mathsf{P},\mathsf{f}}|\mathsf{f}|u_{\mathsf{f}}. \tag{1.19}$$

Il fattore $\alpha_{\mathsf{P},\mathsf{f}} = \mathbf{n}_{\mathsf{f}}\cdot\mathbf{n}_{\mathsf{P},\mathsf{f}}$ può assumere valori ± 1 a seconda della mutua orientazione dei due vettori normali $\mathbf{n}_{\mathsf{f}}$ e $\mathbf{n}_{\mathsf{P},\mathsf{f}}$. Si ha segno positivo se l'orientazione locale della faccia è concorde con la sua orientazione globale, in caso contrario si ha segno negativo. In accordo con (1.18), l'operatore divergenza discreto agisce tra gli spazi $\mathscr{F}_h$ (gradi di libertà definiti sulle facce) e $\mathscr{F}_h^\star$, dove, nel caso specifico $\mathscr{F}_h^\star$ è relativo alle celle della griglia e coincide con $\mathscr{P}_h$ (gradi di libertà definiti sulle celle).

Si noti che, per come è stato definito, l'operatore divergenza discreto è esatto per funzioni vettoriali lineari, vale a dire che, se $\mathbf{u}\in(C^2(\mathsf{P}))^d$ e $\mathbf{u}_h$ è la sua approssimazione in $\mathscr{F}_h$, allora div_h è un'approssimazione del primo ordine di div, nel senso che:

$$(\mathrm{div}_h\mathbf{u}_h)_{\mathsf{P}} = \frac{1}{|\mathsf{P}|}\int_{\mathsf{P}} \mathrm{div}\mathbf{u}\ dV = (\mathrm{div}\mathbf{u})(\mathbf{x}_{\mathsf{P}}) + \mathrm{O}(h_{\mathsf{P}}).$$

Ricordando che un campo discreto può essere rappresentato dal vettore algebrico dei gradi di libertà corrispondenti, anche gli operatori primali possono essere interpretati come matrici. E' bene notare che c'è una forte connessione tra queste matrici e la struttura topologica della griglia; infatti gli operatori primali in forma matriciale corrispondono, dopo un opportuno riscalamento, alle matrici di adiacenza della griglia. In particolare, l'operatore div_h, a meno della misura di facce e celle, corrisponde alla matrice di connessione tra una cella e le sue facce. Infine, la rappresentazione matriciale è molto utile, perché permette di reinterpretare l'azione di un operatore lineare discreto come un prodotto matrice-vettore.

Come ultimo passo, si devono definire formalmente gli *operatori mimetici derivati*. Come precedentemente spiegato, la relazione di dualità (1.18) definisce in modo univoco un operatore aggiunto $\mathscr{D}_h^\star$ per ogni operatore primale $\mathscr{D}_h$; per quanto riguarda la divergenza discreta, possiamo identificare il suo operatore duale come $\widetilde{\nabla}_h \equiv -\mathrm{div}_h^\star$. La definizione formale degli operatori derivati

lascia libertà di scelta per il prodotto interno dello spazio che, a sua volta, influenzerà l'accuratezza della discretizzazione mimetica.

Si considerino, allora, i due spazi di interesse introdotti precedentemente: $\mathscr{F}_h$ e $\mathscr{P}_h$, e si supponga che il prodotto interno di cui sono dotati si possa rappresentare con una matrice simmetrica e definita positiva. Questo significa che, per il primo spazio, esiste $\mathsf{M}_{\mathscr{F}}$ tale che:

$$[\mathbf{u}_h, \mathbf{v}_h]_{\mathscr{F}_h} = \mathbf{u}_h^T \mathsf{M}_{\mathscr{F}} \mathbf{v}_h \tag{1.20}$$

e una costruzione analoga si può fare per il secondo spazio.

E' possibile, ora, applicare la formula della dualità generale, enunciata in (1.18), al caso in esame: l'operatore div_h. Riprendendo le definizioni di operatori primali e derivati si ha:

$$[\mathrm{div}_h \mathbf{v}_h, p_h]_{\mathscr{P}_h} = [\mathbf{v}_h, \mathrm{div}_h^\star p_h]_{\mathscr{F}_h} = -[\mathbf{v}_h, \widetilde{\nabla}_h p_h]_{\mathscr{F}_h},$$

che vale $\forall \mathbf{v}_h \in \mathscr{F}_h$ e $\forall p_h \in \mathscr{P}_h$. Applicando (1.20) ai prodotti interni degli spazi $\mathscr{F}_h$ e $\mathscr{P}_h$, si ottiene la seguente formulazione algebrica:

$$\mathbf{v}_h^T \mathrm{div}_h^T \mathsf{M}_{\mathscr{P}} p_h = -\mathbf{v}_h^T \mathsf{M}_{\mathscr{F}} \widetilde{\nabla}_h p_h,$$

dove div_h^T è la matrice trasposta di div_h. Dato che i vettori $\mathbf{v}_h$ e p_h sono arbitrari, si può scrivere che:

$$\mathsf{M}_{\mathscr{F}} \widetilde{\nabla}_h = -\mathrm{div}_h^T \mathsf{M}_{\mathscr{P}},$$

da cui segue che:

$$\widetilde{\nabla}_h = -\mathrm{div}_h^\star = -\mathsf{M}_{\mathscr{F}}^{-1} \mathrm{div}_h^T \mathsf{M}_{\mathscr{P}}. \tag{1.21}$$

In altre parole, l'operatore derivato gradiente è, a meno del segno, l'aggiunto dell'operatore primale divergenza. Da qui è chiaro come le proprietà del gradiente discreto dipendano dalle proprietà dei due prodotti interni.

In conclusione, si ricorda come gli operatori derivati debbano contenere anche informazioni sui coefficienti del problema differenziale. Per esempio, date q e $\mathbf{v}$ nulle al bordo di Ω e data la relazione

$$\int_\Omega \mathsf{K} \nabla q \cdot \mathbf{v} dV = -\int_\Omega q \cdot \mathrm{div}\ \mathsf{K}\mathbf{v} dV,$$

se si definisce l'operatore primale $\mathrm{div}_h \approx \mathrm{div}$, allora segue che l'operatore derivato corrisponde a $\widetilde{\nabla}_h \approx \mathsf{K}\nabla$.

1.4 Prodotti interni mimetici

Fatte le opportune premesse e definizioni, si può procedere alla parte decisiva della discretizzazione mimetica: la costruzione del prodotto interno. In particolare, come precedentemente anticipato, il vero obiettivo è quello di derivare una formula esplicita per la matrice M.

E' bene notare che tutti gli spazi che sono stati introdotti, ognuno associato ad una diversa scelta dei gradi di libertà, possono essere dotati di prodotto interno. Per questo motivo, sia $\mathscr{S}_h$ uno spazio generico (che può essere $\mathscr{V}_h$, $\mathscr{E}_h$, $\mathscr{F}_h$, $\mathscr{P}_h$ introdotti precedentemente) e sia $\mathscr{S}_{h,\mathsf{P}}$ la sua restrizione su un singolo elemento P. Si consideri il relativo operatore di proiezione:

$$\Pi_{\mathsf{P}}^{\mathscr{S}} \ : \ X_{|\mathsf{P}} \longrightarrow \mathscr{S}_{h,\mathsf{P}}.$$

Sia, inoltre, $\mathscr{T}_{\mathsf{P}}$ lo spazio delle funzioni test. Volendo costruire schemi mimetici dell'ordine più basso esso è lo spazio delle funzioni scalari o vettoriali costanti a tratti. Occorre definire un ulteriore sottospazio di $X_{|\mathsf{P}}$, detto $S_{h,\mathsf{P}}$, che abbia le seguenti proprietà:

- (**B1**) L'operatore di proiezione $\Pi_{\mathsf{P}}^{\mathscr{S}}$ è suriettivo da $S_{h,\mathsf{P}}$ a $\mathscr{S}_{h,\mathsf{P}}$.
- (**B2**) Lo spazio delle funzioni test $\mathscr{T}_{\mathsf{P}}$ è contenuto in $S_{h,\mathsf{P}}$.
- (**B3**) Date $q \in \mathscr{T}_{\mathsf{P}}$ e $v \in S_{h,\mathsf{P}}$, allora l'integrale $\int_{\mathsf{P}} qv \, dV$ può essere calcolato in modo esatto tramite i gradi di libertà, cioè usando le componenti del vettore $\Pi_{\mathsf{P}}^{\mathscr{S}}(v)$.

L'ipotesi (**B1**) afferma che lo spazio $S_{h,\mathsf{P}}$ deve essere sufficientemente ricco. L'ipotesi (**B2**) è legata all'accuratezza dello schema mimetico che si vuole costruire, in questo caso, volendo lo schema di ordine più basso, basta che $\mathscr{T}_{\mathsf{P}}$ contenga le funzioni costanti. Infine, la (**B3**) dipende dal problema in esame e permetterà di scrivere in modo algebrico la condizione di consistenza, necessaria per la costruzione effettiva dello schema.

Il prodotto interno mimetico dello spazio globale è costruito a partire dai prodotti interni locali:

$$[u_h, v_h]_{\mathscr{S}_h} = \sum_{\mathsf{P} \in \Omega_h} [u_{h,\mathsf{P}}, v_{h,\mathsf{P}}]_{\mathscr{S}_{h,\mathsf{P}}} \qquad \forall u_h, v_h \in \mathscr{S}_h,$$

dove $u_{h,\mathsf{P}}$ e $v_{h,\mathsf{P}}$ sono le restrizioni di u_h e v_h su P.

Definizione 1.1 (Condizione di consistenza)**.** Il prodotto interno mimetico soddisfa la condizione di consistenza se:

$$\left[\Pi_{\mathsf{P}}^{\mathscr{S}}(q), \Pi_{\mathsf{P}}^{\mathscr{S}}(v)\right]_{\mathscr{S}_h,\mathsf{P}} = \int_{\mathsf{P}} qv \, dV \quad \forall q \in \mathscr{T}_{\mathsf{P}}, \ \forall v \in S_{h,\mathsf{P}}. \tag{1.22}$$

Per una cella poligonale o poliedrica, lo spazio dei vettori $\Pi^{\mathscr{I}}_{\mathsf{P}}(q)$ può essere più piccolo dello spazio $\mathscr{S}_{h,\mathsf{P}}$. In tal caso, non esiste un unico prodotto interno che soddisfa la condizione di consistenza e diventa necessario introdurre la seconda condizione, detta di stabilità.

Definizione 1.2 (Condizione di stabilità)**.** Il prodotto interno mimetico soddisfa la condizione di stabilità se esistono due costanti positive $C_\star$ e $C^\star$ indipendenti da P e $v_{h,\mathsf{P}}$ tali che:

$$C_\star|\mathsf{P}|\,\|v_{h,\mathsf{P}}\|^2 \leq [v_{h,\mathsf{P}}, v_{h,\mathsf{P}}]_{\mathscr{S}_h,\mathsf{P}} \leq C^\star|\mathsf{P}|\,\|v_{h,\mathsf{P}}\|^2 \qquad \forall v_{h,\mathsf{P}} \in \mathscr{S}_{h,\mathsf{P}}. \tag{1.23}$$

Si introduce, ora, un operatore di ricostruzione, che è l'inverso dell'operatore di proiezione:

$$R^{\mathscr{I}}_{\mathsf{P}} \;:\; \mathscr{S}_{h,\mathsf{P}} \longrightarrow S_{h,\mathsf{P}}.$$

Se $R^{\mathscr{I}}_{\mathsf{P}}$ si potesse costruire facilmente, allora il prodotto interno locale si potrebbe definire esplicitamente come:

$$[u_{h,\mathsf{P}}, v_{h,\mathsf{P}}]_{\mathscr{S}_h,\mathsf{P}} = \int_{\mathsf{P}} R^{\mathscr{I}}_{\mathsf{P}}(u_{h,\mathsf{P}}) R^{\mathscr{I}}_{\mathsf{P}}(v_{h,\mathsf{P}})\; dV.$$

Sfortunatamente, questo è possibile solo per celle con una geometria semplice, mentre, per celle generiche, calcolare l'operatore di ricostruzione non è banale. Esistono tecniche che permettono la definizione di tali operatori, ma esulano dall'obiettivo di questo lavoro. In ogni caso, va notato che il calcolo del termine di destra in (1.22) non dipende dal comportamento degli operatori di ricostruzione, come si può vedere nell'esempio seguente.

Esempio. Si consideri lo spazio mimetico $\mathscr{F}_h$, cioè quello associato alle facce di una griglia poligonale, in due dimensioni ($d = 2$). L'obiettivo è costruire un prodotto interno locale

$$[\cdot,\cdot]_{\mathscr{F}_h,\mathsf{P}} : \mathscr{F}_{h,\mathsf{P}} \times \mathscr{F}_{h,\mathsf{P}} \longrightarrow \mathbb{R},$$

che "mimi" il prodotto scalare in $L^2(\mathsf{P})$ e soddisfi la (1.22). Come spazio delle funzioni test $\mathscr{T}_{\mathsf{P}}$ è sufficiente prendere lo spazio dei vettori costanti: $\mathscr{T}_{\mathsf{P}} = [\mathbb{P}_0(\mathsf{P})]^2$. Allora deve valere:

$$\left[\Pi^{\mathscr{I}}_{\mathsf{P}}(\mathbf{c}), \mathbf{v}_{h,\mathsf{P}}\right]_{\mathscr{S}_h,\mathsf{P}} = \int_{\mathsf{P}} \mathbf{c}\cdot\mathbf{v}\; dV \qquad \forall \mathbf{c} \in \mathscr{T}_{\mathsf{P}},\ \forall \mathbf{v} \in S_{h,\mathsf{P}} \tag{1.24}$$

dove $\mathbf{v}_{h,\mathsf{P}} = \Pi^{\mathscr{I}}_{\mathsf{P}}(\mathbf{v})$. Il passo successivo è definire $S_{h,\mathsf{P}}$ come lo spazio delle funzioni definite su P che soddisfano (**B1**)-(**B3**). Si richiede, inoltre, che valgano le seguenti inclusioni:

$$[\mathbb{P}_0(\mathsf{P})]^2 \subseteq S_{h,\mathsf{P}} \subseteq \{\mathbf{w} \in H_{\mathrm{div}}(\mathsf{P}) : \mathbf{w}|_{\mathsf{f}}\cdot\mathbf{n}_{\mathsf{f}} \in \mathbb{P}_0(\mathsf{f}) \;\; \forall \mathsf{f} \in \partial\mathsf{P}, \mathrm{div}\mathbf{w} \in \mathbb{P}_0(\mathsf{P})\}.$$

La prima inclusione implica che lo spazio contiene i vettori costanti, come stabilito da (**B2**); invece la seconda inclusione è fondamentale per (**B3**). Nonostante le possibili scelte di $S_{h,\mathsf{P}}$ siano infinite, le conclusioni che si ottengono sono sempre le stesse e non è necessario sviluppare questo punto. Quindi, si prenda un qualsiasi operatore di ricostruzione

$$R_{\mathsf{P}}^{\mathscr{F}} \ : \ \mathscr{F}_{h,\mathsf{P}} \longrightarrow S_{h,\mathsf{P}},$$

definito come l'inverso dell'operatore di proiezione (e quindi che conserva i gradi di libertà). E' bene sottolineare, come la scelta di $R_{\mathsf{P}}^{\mathscr{F}}$ fornisca una definizione di $S_{h,\mathsf{P}}$ e viceversa; di conseguenza, avendo scelto lo spazio, è automaticamente definito l'operatore $R_{\mathsf{P}}^{\mathscr{F}}$. Prima di procedere, è conveniente caratterizzare lo spazio delle funzioni test come lo spazio dei gradienti di funzioni lineari:

$$\mathscr{F}_{\mathsf{P}} = \left\{ \nabla q : q \in \mathbb{P}_1(\mathsf{P}), \int_{\mathsf{P}} q dV = 0 \right\}.$$

La condizione di consistenza (1.24) afferma che il prodotto interno è esatto quando uno dei due argomenti è un campo vettoriale costante e l'altro è una funzione di $S_{h,\mathsf{P}}$. Usando la definizione di $S_{h,\mathsf{P}}$, è possibile mostrare che il termine di destra è computabile e non dipende dalla particolare scelta di $R_{\mathsf{P}}^{\mathscr{F}}$. Infatti, prendendo $\mathbf{c} = \nabla q$, con $\int_{\mathsf{P}} q dV = 0$, in (1.24), integrando per parti e sfruttando il fatto che $\mathrm{div}(R_{\mathsf{P}}^{\mathscr{F}}(\mathbf{v}_{h,\mathsf{P}})) \in \mathbb{P}_0(\mathsf{P})$ si ottiene:

$$\int_{\mathsf{P}} \nabla q \cdot R_{\mathsf{P}}^{\mathscr{F}}(\mathbf{v}_{h,\mathsf{P}}) \, dV = -\int_{\mathsf{P}} q \ \mathrm{div}(R_{\mathsf{P}}^{\mathscr{F}}(\mathbf{v}_{h,\mathsf{P}})) \, dV + \sum_{\mathsf{f}\in\partial\mathsf{P}} \int_{\mathsf{f}} R_{\mathsf{P}}^{\mathscr{F}}(\mathbf{v}_{h,\mathsf{P}}) \cdot \mathbf{n}_{\mathsf{P},\mathsf{f}} \, q dS =$$
$$= \sum_{\mathsf{f}\in\partial\mathsf{P}} \int_{\mathsf{f}} R_{\mathsf{P}}^{\mathscr{F}}(\mathbf{v}_{h,\mathsf{P}}) \cdot \mathbf{n}_{\mathsf{P},\mathsf{f}} \, q dS.$$

Ora, sfruttando la definizione (1.13) e ricordando che $\Pi_{\mathsf{P}}^{\mathscr{S}} \circ R_{\mathsf{P}}^{\mathscr{F}}$ equivale all'identità, si ha:

$$R_{\mathsf{P}}^{\mathscr{F}}(\mathbf{v}_{h,\mathsf{P}})|_{\mathsf{f}} \cdot \mathbf{n}_{\mathsf{P},\mathsf{f}} = \frac{1}{|\mathsf{f}|} \int_{\mathsf{f}} R_{\mathsf{P}}^{\mathscr{F}}(\mathbf{v}_{h,\mathsf{P}}) \cdot \mathbf{n}_{\mathsf{P},\mathsf{f}} \, dS = (\Pi_{\mathsf{P}}^{\mathscr{F}} R_{\mathsf{P}}^{\mathscr{F}}(\mathbf{v}_{h,\mathsf{P}}))_{\mathsf{f}} = (\mathbf{v}_{h,\mathsf{P}})_{\mathsf{f}} = \mathrm{v}_{\mathsf{f}}.$$

Inserendo le ultime due formule in (1.24) si ottiene una nuova condizione di consistenza

$$\left[\Pi_{\mathsf{P}}^{\mathscr{S}}(\nabla q), \mathbf{v}_{h,\mathsf{P}}\right]_{\mathscr{S}_h,\mathsf{P}} = \sum_{\mathsf{f}\in\partial\mathsf{P}} \mathrm{v}_{\mathsf{f}} \int_{\mathsf{f}} q \, dS \quad \forall \mathbf{v}_{h,\mathsf{P}} \in \mathscr{F}_{h,\mathsf{P}}, \ \forall q \in \mathbb{P}_1(\mathsf{P})/\mathbb{R},$$

che non dipende dall'operatore di ricostruzione che, nella pratica, non serve.

Si procede, ora, alla costruzione della rappresentazione matriciale del prodotto interno; per maggiore generalità, si lavorerà nuovamente sullo spazio $\mathscr{S}_h$,

che può essere $\mathscr{V}_h$, $\mathscr{E}_h$, $\mathscr{F}_h$ e $\mathscr{P}_h$. La sua restrizione sull'elemento P, $\mathscr{S}_{h,\mathsf{P}}$ può rappresentare, a sua volta, $\mathscr{V}_{h,\mathsf{P}}$, $\mathscr{E}_{h,\mathsf{P}}$, $\mathscr{F}_{h,\mathsf{P}}$ o $\mathscr{P}_{h,\mathsf{P}}$. L'obiettivo è trovare una matrice simmetrica e definita positiva tale che:

$$[u_{h,\mathsf{P}}, v_{h,\mathsf{P}}]_{\mathscr{S}_{h,\mathsf{P}}} = u_{h,\mathsf{P}}^T \; \mathsf{M}_{\mathscr{S},\mathsf{P}} \; v_{h,\mathsf{P}} \qquad \forall u_{h,\mathsf{P}}, v_{h,\mathsf{P}}.$$

Per quanto riguarda le funzioni test, esse appartengono allo spazio delle funzioni costanti, cioè $\mathscr{T}_\mathsf{P} = \mathbb{P}_0(\mathsf{P})$. Sia $v_{h,\mathsf{P}} = \Pi_\mathsf{P}^\mathscr{S}(v)$, con $v \in S_{h,\mathsf{P}}$, allora si può riscrivere la condizione di consistenza nella forma equivalente che segue:

$$\left[\Pi_\mathsf{P}^\mathscr{S}(c), v_{h,\mathsf{P}}\right]_{\mathscr{S}_{h,\mathsf{P}}} = \int_\Omega c \; R_\mathsf{P}^\mathscr{S}(v_{h,\mathsf{P}}) dV, \tag{1.25}$$

che vale $\forall v_{h,\mathsf{P}} \in \mathscr{S}_{h,\mathsf{P}}$ e per ogni funzione costante (scalare o vettoriale) $c \in \mathbb{P}_0(\mathsf{P})$.

Da questa formulazione della condizione di consistenza, si può sistematicamente derivare una forma algebrica lineare del tipo:

$$\mathsf{M}_{\mathscr{S},\mathsf{P}} \mathsf{N}_{\mathscr{S},\mathsf{P}} = \mathsf{R}_{\mathscr{S},\mathsf{P}}, \tag{1.26}$$

dove le matrici $\mathsf{N}_{\mathscr{S},\mathsf{P}}$ e $\mathsf{R}_{\mathscr{S},\mathsf{P}}$ sono computabili. In questo modo, si è ottenuto un sistema lineare in cui l'incognita è la matrice $\mathsf{M}_{\mathscr{S},\mathsf{P}}$; esso avrà una forma differente a seconda dello spazio in esame, nel prossimo capitolo verrà fornita la costruzione dettagliata per lo spazio dei vettori definiti sulle facce: $\mathscr{F}_{h,\mathsf{P}}$.

Capitolo 2

Il problema di Darcy

I modelli matematici riguardanti i giacimenti petroliferi sono stati utilizzati fin dal 1800. Essi consistono in un insieme di equazioni che descrivono il moto dei fluidi in un mezzo poroso, a cui si devono aggiungere le appropriate condizioni al bordo e iniziali.

Il moto di un fluido in un mezzo poroso è governato da leggi di conservazione, tipicamente per la massa, per il momento angolare e per l'energia. Nel caso di flusso in un mezzo poroso, l'equazione del momento è data dalla legge di Darcy: derivata empiricamente, questa legge stabilisce una relazione lineare tra la velocità relativa del fluido rispetto al solido e il gradiente della pressione del fluido.

2.1 Il modello fisico-matematico

In questa sezione, verrà fornito il modello matematico che descrive il flusso in un mezzo poroso di un fluido Newtoniano, che occupa tutto lo spazio vuoto, ossia i pori (caso saturo), ci si limita a considerare il caso di flusso a singola fase, in cui il fluido è formato da un solo componente o da una miscela omogenea.

Un mezzo poroso è un corpo composto da una parte solida (matrice solida) e da uno spazio vuoto che può essere riempito da uno o più liquidi o gas (Figura 2.1). Esempi tipici si possono trovare in ambito geologico, ingegneristico ma anche biologico. Le equazioni che ne governano il flusso sono la conservazione della massa (*legge generale*) e la legge di Darcy (*equazione costitutiva*). Un mezzo poroso è caratterizzato dalla sua porosità; essa è una grandezza scalare ed è definita come il rapporto tra il volume dei vuoti (pori) ed il volume totale del materiale considerato. Si suppone che i fenomeni di dispersione nel mezzo possano essere trascurati e la superficie di contatto tra fluido e solido sia impermeabile al fluido, cioè non possa essere attraversata.

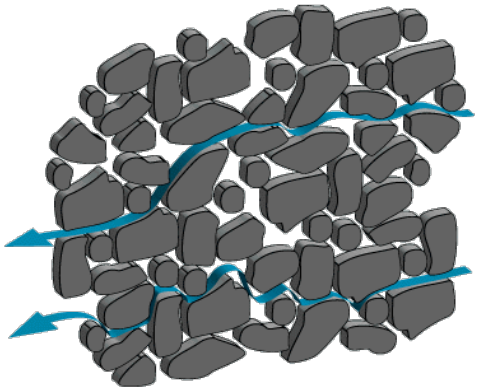


Figura 2.1: Esempio di mezzo poroso

Conservazione della massa

Sia $\Omega \subset \mathbb{R}^2$ il dominio su cui sono definite le equazioni; le variabili spaziali e temporale sono rappresentate rispettivamente da $\mathbf{x} = (x_1, x_2)$ e t. Si denota con $\phi = \phi(\mathbf{x}, t)$ la porosità del mezzo, con $\rho = \rho(\mathbf{x}, t)$ la densità volumetrica del fluido, con $\mathbf{u}(\mathbf{x}, t) = (u_1, u_2)$ la velocità del fluido e con b una forzante esterna che può essere una sorgente o un pozzo. La legge di conservazione della massa è data da:

$$\frac{\partial(\phi\rho)}{\partial t} + \text{div}(\rho\mathbf{u}) = \rho b \qquad \text{in } \Omega. \tag{2.1}$$

Nel caso in cui il fluido sia incomprimibile e quindi $\rho(\mathbf{x}, t) = \rho_0 = cost$ e ϕ sia costante, la (2.1) diventa:

$$\text{div}\mathbf{u} = b \qquad \text{in } \Omega. \tag{2.2}$$

Legge di Darcy

Le equazioni classiche che vengono usate per ricavare il campo di moto dei fluidi e il campo di pressione sono quelle di Navier-Stokes. Tuttavia nel caso di mezzi porosi sarebbe necessario risolvere tali equazioni alla microscala, ossia la scala dei pori. Essendo questa scala diversa rispetto al dominio di interesse per molti ordini di grandezza, questo richiederebbe una griglia di calcolo estremamente fitta e, di conseguenza, un costo computazionale troppo elevato per poter essere preso in considerazione. Alla macroscala si utilizza quindi la legge di Darcy:

$$\mathbf{u} = -\frac{\mathbf{K}}{\mu}(\nabla p - \rho\mathbf{g}) \qquad \text{in } \Omega, \tag{2.3}$$

che mette in relazione la pressione del fluido p e la sua velocità macroscopica attraverso il tensore di permeabilità $\mathbf{K}$ (una proprietà delle rocce che rappresenta la capacità di essere attraversati dai fluidi), la viscosità del fluido μ e l'accelerazione di gravità $\mathbf{g}$.

Tale legge è stata ricavata sperimentalmente da Henry Darcy e, più tardi, è stata legittimata rigorosamente tramite tecniche come l'omogeneizzazione delle equazioni di Navier-Stokes o la teoria delle miscele. L'idea di queste tecniche è ipotizzare che la macroscala e la microscala siano effettivamente separate e analizzare il fenomeno ad una scala intermedia (denominata mesoscala). Il segno meno in (2.3) indica che il fluido scorre da zone con pressione più alta a zone con pressione più bassa; inoltre, a parità di pressione il flusso sarà più rapido lungo le direzioni con permeabilità maggiore. Nel caso in cui le variazioni di quota siano trascurabili rispetto alle variazioni di pressione, il secondo termine di (2.3) dovuto alla gravità può essere tralasciato.

Accoppiando le equazioni (2.2) e (2.3) e completandole con le condizioni al bordo, si ottiene il modello per il flusso in mezzi porosi non fratturati:

$$\begin{cases} \mathbf{u} + \mathrm{K}\nabla p = 0 & \text{in } \Omega \\ \mathrm{div}\mathbf{u} = b & \text{in } \Omega \\ p = g^D & \text{su } \Gamma^D \\ \mathbf{u} \cdot \mathbf{n} = g^N & \text{su } \Gamma^N. \end{cases} \tag{2.4}$$

Ora, K è un campo tensoriale che tiene conto della permeabilità del mezzo, della viscosità e della densità del fluido. Si suppone che esso sia simmetrico e fortemente ellittico, ossia esistano due costanti $\mathsf{k}^\star$ e $\mathsf{k}_\star$ tali che:

$$\mathsf{k}_\star \|\mathbf{v}\|^2 \leq \mathbf{v}^T \mathrm{K}\mathbf{v} \leq \mathsf{k}^\star \|\mathbf{v}\|^2 \qquad \forall \mathbf{v} \in \mathbb{R}^2.$$

2.2 Discretizzazione mimetica

Si procede, ora, alla discretizzazione numerica, tramite Differenze Finite Mimetiche del problema (2.4). A tal proposito si veda [8], [9], [13], [16], [17], [35], [30], [22], [34], [19], [3].

2.2.1 Gradi di libertà e operatori mimetici

Il *primo* passo è la definizione dei gradi di libertà: nel caso dello schema di ordine più basso, si usa un grado di libertà per approssimare $\mathbf{u}$ in ogni faccia della griglia e un grado di libertà per approssimare p in ogni cella della

griglia; si noti che, nel caso di mesh triangolare, la discretizzazione mimetica può essere interpretata come una generalizzazione degli elementi finiti misti di Raviart-Thomas di ordine zero [14].
Più precisamente:

- Lo spazio $\mathscr{P}_h$ è definito associando un grado di libertà per ogni cella $\mathsf{P} \in \Omega_h$. Il valore associato a P è denotato con q_P e l'insieme di tutti i valori è il vettore q_h tale per cui:

$$q_h = (q_\mathsf{P})_{\mathsf{P}\in\Omega_h} .$$

- Lo spazio $\mathscr{F}_h$ è definito associando un grado di libertà per ogni faccia $\mathsf{f} \in \mathscr{F}$. Il valore associato a f è denotato con u_f e l'insieme di tutti i valori è il vettore $\mathbf{u}_h \in \mathscr{F}_h$ tale per cui:

$$\mathbf{u}_h = (u_\mathsf{f})_{\mathsf{f}\in\mathscr{F}} .$$

La pressione p è definita nel primo spazio e, per questo, sarà una funzione costante su ogni elemento. La velocità $\mathbf{u}$ è definita nel secondo spazio, per il quale è bene fare qualche considerazione in più. Infatti, come spiegato nel Capitolo 1, ad ogni faccia f di ogni cella P della mesh è possibile associare un grado di libertà $u_{\mathsf{P},\mathsf{f}}$; quindi la dimensione di $\mathscr{F}_h$ è il numero delle facce esterne più il doppio delle facce interne, poiché queste sono condivise da due elementi. Per questo motivo è necessario supporre la continuità della velocità lungo le facce, cioè, se la faccia f è condivisa dagli elementi P_1 e P_2 deve valere che:

$$u^\mathsf{f}_{\mathsf{P}_1} = -u^\mathsf{f}_{\mathsf{P}_2}. \tag{2.5}$$

In questo modo si introducono tante equazioni aggiuntive quante sono le facce interne o, viceversa, si riduce la dimensione delle incognite di $\mathbf{u}_h$ al numero delle facce.

La seconda importante questione da considerare è il segno del grado di libertà; infatti, il valore u_f rappresenta la media sulla faccia f della componente normale di $\mathbf{u}$, ossia la velocità in direzione $\mathbf{n}_\mathsf{f}$, tuttavia il versore normale alla faccia deve essere fissato una volta per tutte. Si hanno, cioè, due possibili normali per ogni faccia: la prima è quella fissata, in modo arbitrario, a priori, viene indicata con $\mathbf{n}_\mathsf{f}$ e non dipende dall'elemento; la seconda è quella relativa alla cella in cui è contenuta la faccia, è indicata con $\mathbf{n}_{\mathsf{P},\mathsf{f}}$ e può essere concorde o discorde con quella di riferimento, nel caso fosse discorde, il grado di libertà va cambiato di segno, in accordo con (2.5). Formalmente, detto $u_{\mathsf{P},\mathsf{f}}$ il valore di velocità associato a P, sarà $u_{\mathsf{P},\mathsf{f}} = \alpha_{\mathsf{P},\mathsf{f}}\, u_\mathsf{f}$, dove $\alpha_{\mathsf{P},\mathsf{f}} = \mathbf{n}_{\mathsf{P},\mathsf{f}} \cdot \mathbf{n}_\mathsf{f}$.

Un'alternativa molto diffusa nella scelta dei gradi di libertà consiste nell'usare al posto della velocità media il flusso uscente dalle facce, definito come:

$$F_{\mathsf{e}} = \int_{\mathsf{e}} \mathbf{u} \cdot \mathbf{n}_{\mathsf{e}} \, dS.$$

Una volta definiti i gradi di libertà, si possono introdurre gli operatori di interpolazione, che agiscono tra spazi di funzioni continue regolari e gli spazi $\mathscr{F}_h$ e $\mathscr{P}_h$. Essi sono quelli definiti in (1.12) e (1.13). Queste definizioni, unite a quella dell'operatore divergenza discreta, forniscono il seguente risultato:

Lemma 2.1. *Detto* $X(\Omega) = \{\boldsymbol{v} \in (L^s(\Omega))^d, \ s \geq 2, \ \text{con } div\boldsymbol{v} \in L^2(\Omega)\}$, *per ogni* $\boldsymbol{v} \in X$, *vale*

$$(div\boldsymbol{v})^I = div_h \boldsymbol{v}^I.$$

Il *secondo* passo consiste nel dotare gli spazi $\mathscr{F}_h$ e $\mathscr{P}_h$ dei prodotti interni mimetici. Per lo spazio $\mathscr{P}_h$ si considera il seguente prodotto interno:

$$[p_h, q_h]_{\mathscr{P}_h} = \sum_{\mathsf{P} \in \Omega_h} |\mathsf{P}| \, p_{\mathsf{P}} q_{\mathsf{P}}, \tag{2.6}$$

dove con $|\mathsf{P}|$ si indica il volume della cella. Si noti che tale formula è esatta per funzioni costanti a tratti definite su Ω_h.
Per quanto riguarda lo spazio $\mathscr{F}_h$, il prodotto interno è definito come:

$$[\mathbf{u}_h, \mathbf{v}_h]_{\mathscr{F}_h} = \sum_{\mathsf{P} \in \Omega_h} [\mathbf{u}_{\mathsf{P}}, \mathbf{v}_{\mathsf{P}}]_{\mathsf{P}} \, , \tag{2.7}$$

dove $[\cdot, \cdot]_{\mathsf{P}}$ è un prodotto interno locale tale da soddisfare le due condizioni di consistenza e stabilità. Esso si può esprimere come

$$[\mathbf{u}_h, \mathbf{v}_h]_{\mathsf{P}} = \sum_{\mathsf{f}_i, \mathsf{f}_j \in \mathsf{P}} \mathsf{M}^{\mathscr{F}}_{\mathsf{P}} \ \mathbf{u}^{\mathsf{f}_i}_{h,\mathsf{P}} \ \mathbf{v}^{\mathsf{f}_j}_{h,\mathsf{P}} \qquad \forall \mathbf{u}, \mathbf{v} \in \mathscr{F}_h \ e \ \forall \mathsf{P} \in \Omega_h, \tag{2.8}$$

spostando l'obiettivo alla costruzione di una opportuna matrice $\mathsf{M}^{\mathscr{F}}$.

Il *terzo* passo è la costruzione degli operatori primali e derivati, usando la strategia descritta nel capitolo precedente. Ponendo per semplicità $g^D = 0$ e $\Gamma^N = 0$ in (2.4) e sfruttando le definizioni (1.19) e (1.21) si può scrivere la discretizzazione tramite Differenze Finite Mimetiche del problema (2.4) come:

$$\begin{aligned} \mathbf{u}_h + \widetilde{\nabla}_h p_h &= 0, \\ \mathrm{div}_h \mathbf{u}_h &= b^I, \end{aligned} \tag{2.9}$$

dove $b^I = \Pi^{\mathscr{P}}(b)$ e gli operatori discreti approssimano

$$\mathrm{div}_h \approx \mathrm{div} \qquad \widetilde{\nabla}_h \approx \mathsf{K}\nabla.$$

Per passare alla formulazione variazionale (o debole) di (2.9), si suppone che i dati al bordo siano integrabili sulle rispettive frontiere e si introduce lo spazio

$$\mathscr{F}_{h,g} = \{\mathbf{v}_h \in \mathscr{F}_h : v_\mathsf{f} = \frac{1}{|\mathsf{f}|}\int_\mathsf{f} g^N dS \quad \forall \mathsf{f} \in \mathscr{F}^N\},$$

dove $\mathscr{F}^N \subset \mathscr{F}$ è l'insieme delle facce contenute in Γ^N; analogamente si definiscono $\mathscr{F}^D \subset \mathscr{F}$ e Γ^D. Nel caso particolare $g^N = 0$ lo spazio sarà $\mathscr{F}_{h,0}$. Inoltre per ogni $g^D \in L^1(\Gamma^D)$ si definisce il funzionale lineare

$$< g^D, \mathbf{v}_h >_h = \sum_{\mathsf{f}\in\mathscr{F}^D} v_\mathsf{f} \int_\mathsf{f} g^D dS.$$

Allora la formulazione debole della discretizzazione mimetica del problema (2.9) è:
Trovare $(\boldsymbol{u}_h, p_h) \in \mathscr{F}_{h,g} \times \mathscr{P}_h$ *tali che*

$$[\mathbf{u}_h, \mathbf{v}_h]_{\mathscr{F}_h} + \left[\widetilde{\nabla}_h p_h, \mathbf{v}_h\right]_{\mathscr{F}_h} =< g^D, \mathbf{v}_h >_h \qquad \forall \mathbf{v}_h \in \mathscr{F}_{h,0},$$

$$[\mathrm{div}_h \mathbf{u}_h, q_h]_{\mathscr{P}_h} = \left[b^I, q_h\right]_{\mathscr{P}_h} \qquad \forall q_h \in \mathscr{P}_h.$$

Sfruttando la definizione di gradiente discreto si può scrivere:
Trovare $(\boldsymbol{u}_h, p_h) \in \mathscr{F}_{h,g} \times \mathscr{P}_h$ *tali che*

$$\begin{cases} [\mathbf{u}_h, \mathbf{v}_h]_{\mathscr{F}_h} - [p_h, \mathrm{div}_h \mathbf{v}_h]_{\mathscr{P}_h} =< g^D, \mathbf{v}_h >_h & \forall \mathbf{v}_h \in \mathscr{F}_{h,0}, \\ [\mathrm{div}_h \mathbf{u}_h, q_h]_{\mathscr{P}_h} = \left[b^I, q_h\right]_{\mathscr{P}_h} & \forall q_h \in \mathscr{P}_h, \end{cases} \tag{2.10}$$

Si noti che, nella formulazione debole mista, le condizioni al bordo sulla pressione (o di Dirichlet in forma primale), sono imposte in modo naturale, mentre quelle sulla velocità (o di Neumann in forma primale) devono essere imposte sullo spazio funzionale, cioè in modo essenziale.

2.2.2 Costruzione del prodotto interno

In questo paragrafo, viene applicata la strategia descritta nel Capitolo 1 per la costruzione del prodotto interno mimetico $[\cdot,\cdot]_\mathsf{P}$, applicata al caso specifico del problema di Darcy [21]. A tal proposito, si ricorda che è fondamentale selezionare, tra le infinite possibilità, il prodotto interno che soddisfa le condizioni di consistenza (1.22) e stabilità (1.23), con l'obiettivo di ottenere un metodo convergente. Partendo dall'enunciato formale delle condizioni, si arriverà a scrivere esplicitamente la matrice $\mathsf{M}^\mathscr{F}$ (che per semplicità ora verrà indicata solo con M), in modo da poterla facilmente implementare.

Sia K_P l'approssimazione del tensore di diffusione K sull'elemento P, data da:

$$\mathsf{K}_\mathsf{P} = \frac{1}{|\mathsf{P}|}\int_\mathsf{P} \mathsf{K} dV,$$

che definisce un campo tensoriale discontinuo. Se K è sufficientemente regolare, per esempio continuo su ogni elemento, l'approssimazione tramite media sulla cella può essere sostituita dal valore calcolato nel baricentro, vale a dire $\mathsf{K}_\mathsf{P} = \mathsf{K}(\mathbf{x}_\mathsf{P})$.

Procedendo nella definizione della condizione di stabilità, si parte dalla (1.23), in cui è stata usata una generica norma dello spazio $\mathscr{F}_{h,g}$, e la si riscrive per il caso specifico in esame:

(S1) Esistono due costanti positive $C_\star$ e $C^\star$ tali che

$$C_\star|\mathsf{P}|\sum_{\mathsf{f}\in\partial\mathsf{P}}|v_\mathsf{f}|^2 \leq [\mathbf{v}_\mathsf{P},\mathbf{v}_\mathsf{P}]_\mathsf{P} \leq C^\star|\mathsf{P}|\sum_{\mathsf{f}\in\partial\mathsf{P}}|v_\mathsf{f}|^2 \quad \forall \mathbf{v}_\mathsf{P}\in\mathscr{F}_{h,\mathsf{P}}. \tag{2.11}$$

Questa condizione afferma che il prodotto interno è coercivo e cioè che $[\mathbf{v}_\mathsf{P},\mathbf{v}_\mathsf{P}]_\mathsf{P} = 0$ se e solo se $\mathbf{v}_\mathsf{P} = \mathbf{0}$.

Prima di formulare la condizione di consistenza, si deve introdurre lo spazio $S_{h,\mathsf{P}}$, definito come

$$S_{h,\mathsf{P}} = \{\mathbf{v}\in(L^s(\mathsf{P}))^d, s>2, \text{con } \mathrm{div}\mathbf{v} = cost, \mathbf{v}\cdot\mathbf{n}_\mathsf{f} = cost \ \ \forall \mathsf{f}\in\partial\mathsf{P}\}.$$

In accordo con la teoria sviluppata del Capitolo 1, lo spazio deve soddisfare le tre assunzioni (**B1**)-(**B3**) e, per come è stato definito, questo si verifica. Si può allora definire la condizione di consistenza:

(S2) Per ogni funzione vettoriale $\mathbf{v}\in S_{h,\mathsf{P}}$, per ogni polinomio lineare q e per ogni elemento P vale:

$$\left[(\mathsf{K}_\mathsf{P}\nabla q)^I_\mathsf{P}, \mathbf{v}^I_\mathsf{P}\right]_\mathsf{P} = \int_\mathsf{P}\mathsf{K}_\mathsf{P}^{-1}(\mathsf{K}_\mathsf{P}\nabla q)\cdot\mathbf{v}dV. \tag{2.12}$$

Per rendere il termine di destra di (2.12) facilmente implementabile, lo si può riscrivere integrando per parti e sfruttando le proprietà di $S_{h,\mathsf{P}}$:

$$\int_\mathsf{P}\mathsf{K}_\mathsf{P}^{-1}(\mathsf{K}_\mathsf{P}\nabla q)\cdot\mathbf{v}dV = \int_\mathsf{P}\nabla q\cdot\mathbf{v}dV = -\int_\mathsf{P} q\ \mathrm{div}\mathbf{v}dV + \sum_{\mathsf{f}\in\partial\mathsf{P}}\int_\mathsf{f}\mathbf{v}\cdot\mathbf{n}_{\mathsf{P},\mathsf{f}}\ qdS =$$

$$= -\mathrm{div}_\mathsf{P}\mathbf{v}^I_\mathsf{P}\int_\mathsf{P} qdV + \sum_{\mathsf{f}\in\partial\mathsf{P}}\alpha_{\mathsf{P},\mathsf{f}}v^I_\mathsf{f}\int_\mathsf{f} qdS, \tag{2.13}$$

dove $\mathrm{div}_\mathsf{P}\mathbf{v}^I_\mathsf{P} = (\mathrm{div}\mathbf{v})_{|\mathsf{P}} = cost$ e $v^I_\mathsf{f} = \mathbf{v}\cdot\mathbf{n}_{\mathsf{P},\mathsf{f}} = cost$.

Dopo aver riscritto le due condizioni, di consistenza e stabilità adattate al problema di Darcy, si può passare alla costruzione della matrice che rappresenta il prodotto interno:

$$[\mathbf{u}_{\mathsf{P}}, \mathbf{v}_{\mathsf{P}}]_{\mathsf{P}} = \mathbf{u}_{\mathsf{P}}^T \mathsf{M}_{\mathsf{P}} \mathbf{v}_{\mathsf{P}}.$$

A tal scopo, si restringe lo spazio di q allo spazio dei polinomi di grado uno, con media nulla, definiti su P: $\mathbb{P}_1(\mathsf{P})/\mathbb{R}$, in modo tale che l'integrale di volume nel termine di destra di (2.13) si annulli e **(S2)** si riduca a:

$$\left[(\mathsf{K}_{\mathsf{P}} \nabla q)^I_{\mathsf{P}}, \mathbf{v}^I_{\mathsf{P}}\right]_{\mathsf{P}} = \sum_{\mathsf{f} \in \partial \mathsf{P}} \alpha_{\mathsf{P},\mathsf{f}} v^I_{\mathsf{f}} \int_{\mathsf{f}} q dS. \tag{2.14}$$

Si osservi che aver scelto q a media nulla non è restrittivo, perché uno schema ottenuto prendendo $q = cost$ non porterebbe informazioni aggiuntive. Infatti ponendo $q = 1$ in (2.13) si riottiene la definizione dell'operatore mimetico divergenza (1.19):

$$0 = -|\mathsf{P}| \mathrm{div}_{\mathsf{P}} \mathbf{v}^I_{\mathsf{P}} + \sum_{\mathsf{f} \in \partial \mathsf{P}} \alpha_{\mathsf{P},\mathsf{f}} v^I_{\mathsf{f}} |\mathsf{f}|.$$

Si considerano, allora, le seguenti funzioni polinomiali:

$$q^1(x, y) = x - x_{\mathsf{P}}, \qquad q^2(x, y) = y - y_{\mathsf{P}},$$

dove $\mathbf{x}_{\mathsf{P}} = (x_{\mathsf{P}}, y_{\mathsf{P}})^T$ è il baricentro della cella P, e si definisce il vettore:

$$\mathsf{N}_j = (\mathsf{K}_{\mathsf{P}} \nabla q^j)^I_{\mathsf{P}} = \Pi^{\mathscr{F}}_{\mathsf{P}} (\mathsf{K}_{\mathsf{P}} \nabla q^j), \qquad j = 1, 2.$$

Numerando le facce di P da 1 a $N^{\mathscr{F}}_{\mathsf{P}}$ (il numero delle facce di P), si può scrivere esplicitamente la formula della i-esima componente di N_j come

$$(\mathsf{N}_j)_i = \frac{1}{|\mathsf{f}_i|} \int_{\mathsf{f}_i} \mathbf{n}_{\mathsf{f}_i} \cdot \mathsf{K}_{\mathsf{P}} \nabla q^j dS = \mathbf{n}^T_{\mathsf{f}_i} \mathsf{K}_{\mathsf{P}} \cdot \nabla q^j$$

e si può definire la matrice di dimensioni $N^{\mathscr{F}}_{\mathsf{P}} \times 2$: $\mathsf{N}_{\mathsf{P}} = [\mathsf{N}_1, \mathsf{N}_2]$, mentre nel caso 3D la matrice avrà dimensioni $N^{\mathscr{F}}_{\mathsf{P}} \times 3$. Per come sono state definite le funzioni q^j, il loro gradiente è un vettore di dimensione due con l'elemento j-esimo pari a uno (zero altrove) e, conseguentemente, la i-esima riga della matrice è $\mathbf{n}^T_{\mathsf{f}_i} \mathsf{K}_{\mathsf{P}}$. Quindi,

$$\mathsf{N}_{\mathsf{P}} = \begin{pmatrix} \mathbf{n}^T_{\mathsf{f}_1} \\ \mathbf{n}^T_{\mathsf{f}_2} \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{n}^T_{\mathsf{f}_{N^{\mathscr{F}}_{\mathsf{P}}}} \end{pmatrix} \mathsf{K}_{\mathsf{P}}, \tag{2.15}$$

e si può riformulare la (2.14) come segue:

$$\left[(\mathsf{K}_\mathsf{P}\nabla q^j)^I_\mathsf{P}, \mathbf{v}^I_\mathsf{P}\right]_\mathsf{P} = (\mathbf{v}^I_\mathsf{P})^T \mathsf{M}_\mathsf{P}\mathsf{N}_j = (\mathbf{v}^I_\mathsf{P})^T \mathsf{R}_j, \tag{2.16}$$

avendo posto

$$(\mathsf{R}_j)_i = \alpha_{\mathsf{P},\mathsf{f}_i} \int_\mathsf{f} q^j dS.$$

La (2.16) deve valere per qualsiasi campo discreto vettoriale $\mathbf{v}^I_\mathsf{P}$. Il vettore R_j dipende da q^j e dalla geometria della cella P e analogamente a N_P, si può definire $\mathsf{R}_\mathsf{P} = [\mathsf{R}_1, \mathsf{R}_2]$. Essendo $\mathbf{v}^I_\mathsf{P}$ arbitrario, si ottengono le seguenti condizioni matriciali:

$$\mathsf{M}_\mathsf{P}\mathsf{N}_j = \mathsf{R}_j \quad \text{per } j = 1, 2,$$

che possono essere scritte nella forma compatta

$$\mathsf{M}_\mathsf{P}\mathsf{N}_\mathsf{P} = \mathsf{R}_\mathsf{P}. \tag{2.17}$$

Prima di proseguire, è necessario fornire una formula esplicita e quindi implementabile per il calcolo di R_P: l'integrale su una faccia di una funzione lineare equivale al suo valore nel baricentro $\mathbf{x}_\mathsf{P}$ moltiplicato per la misura della faccia. Perciò:

$$\mathsf{R}_\mathsf{P} = \begin{pmatrix} \alpha_{\mathsf{P},\mathsf{f}_1}|\mathsf{f}_1|(\mathbf{x}_{\mathsf{f}_1} - \mathbf{x}_\mathsf{P})^T \\ \alpha_{\mathsf{P},\mathsf{f}_2}|\mathsf{f}_2|(\mathbf{x}_{\mathsf{f}_2} - \mathbf{x}_\mathsf{P})^T \\ \cdot \\ \cdot \\ \cdot \\ \alpha_{\mathsf{P},\mathsf{f}_{N^\mathscr{F}_\mathsf{P}}}|\mathsf{f}_{N^\mathscr{F}_\mathsf{P}}|(\mathbf{x}_{\mathsf{f}_{N^\mathscr{F}_\mathsf{P}}} - \mathbf{x}_\mathsf{P})^T \end{pmatrix}. \tag{2.18}$$

Si può dimostrare che vale il seguente risultato.

Lemma 2.2. *Data una qualsiasi cella P, vale*

$$N_P^T R_P = K_P|P|. \tag{2.19}$$

Dimostrazione. Senza perdere generalità, si può supporre che l'origine degli assi sia posta nel baricentro della cella P: $\mathbf{x}_\mathsf{P} = (0,0)^T$. Si denota la i-esima coordinata spaziale con $x^{(i)}$ e quindi $\mathbf{x} = (x^{(1)}, x^{(2)})^T$. Inoltre, $\mathbf{e}_j$ è il vettore bidimensionale la cui componente j-esima è pari a 1 e le altre sono 0, in modo da poter scrivere le colonne di $|\mathsf{P}|\mathsf{K}_\mathsf{P}$ come $|\mathsf{P}|\mathsf{K}_\mathsf{P}\mathbf{e}_j$ e svilupparle nel modo seguente:

$$|\mathsf{P}|\mathsf{K}_\mathsf{P}\mathbf{e}_j = \mathsf{K}_\mathsf{P}\int_\mathsf{P}\mathbf{e}_j dV = \mathsf{K}_\mathsf{P}\int_\mathsf{P}\nabla x^{(j)} dV = \mathsf{K}_\mathsf{P}\int_{\partial\mathsf{P}}\mathbf{n}_\mathsf{P} x^{(j)} dS =$$

$$= \mathsf{K_P} \sum_{\mathsf{f}\in\partial\mathsf{P}} \alpha_{\mathsf{P},\mathsf{f}_j} \mathbf{n}_\mathsf{f} \int_\mathsf{f} x^{(j)} dS = \mathsf{K_P} \sum_{\mathsf{f}\in\partial\mathsf{P}} \alpha_{\mathsf{P},\mathsf{f}_j} \mathbf{n}_\mathsf{f} |\mathsf{f}| x_\mathsf{f}^{(j)}.$$

che confrontata con le definizioni (2.15) e (2.18) dà

$$|\mathsf{P}|\mathsf{K_P}\mathbf{e}_j = \mathsf{N_P^T R_P}\mathbf{e}_j \qquad j = 1, 2.$$

e prova la tesi. □

Si vuole, ora, scrivere la matrice $\mathsf{M_P}$ come la somma di due matrici semi-definite positive: $\mathsf{M_P} = \mathsf{M_P^{(0)}} + \mathsf{M_P^{(1)}}$. La prima è della forma:

$$\mathsf{M_P^{(0)}} = \frac{1}{|\mathsf{P}|}\mathsf{R_P K_P^{-1} R_P^T} = \mathsf{R_P}\left(\mathsf{R_P^T N_P}\right)^{-1}\mathsf{R_P^T} \tag{2.20}$$

e si vede immediatamente che soddisfa la condizione (2.17). Inoltre essa è simmetrica per definizione, ma non sempre è definita positiva, non soddisfacendo la (2.11). Per questo motivo si rende necessaria l'aggiunta della seconda matrice, detta, appunto, matrice di stabilizzazione, che deve garantire la condizione di consistenza del metodo.

Il seguente lemma fornisce una formula generale per la matrice $\mathsf{M_P^{(1)}}$, per maggiori dettagli si veda [21]:

Lemma 2.3. *Sia $M_P^{(0)}$ data dalla formula (2.20) e siano inoltre D e U due matrici con le seguenti proprietà:*

- *D ha dimensioni $N_P^{\mathscr{F}} \times (N_P^{\mathscr{F}} - 2)$ ed è tale che $Im(D) = ker(N^T)$, ovvero $N^T D = 0$ e anche $D^T N = 0$. Questo significa che le sue colonne formano una base per lo spazio $ker(N^T)$.*
- *U ha dimensioni $(N_P^{\mathscr{F}} - 2) \times (N_P^{\mathscr{F}} - 2)$ ed è simmetrica e definita positiva.*

Allora, detta $M_P^{(1)} = DUD^T$, la matrice $M_P = M_P^{(0)} + M_P^{(1)}$ è simmetrica e definita positiva e soddisfa la (2.17).

Dimostrazione. La seconda parte del teorema è verificata per definizione delle matrici, infatti $\mathsf{M_P^{(1)} N} = \mathsf{DUD^T N} = 0$ e $\mathsf{M_P^{(0)} N} = \mathsf{R}$.

Per quanto riguarda la simmetria di $\mathsf{M_P}$, essa è legata alla simmetria di $\mathsf{K_P}$ infatti, dato che l'inversa di una matrice simmetrica si mantiene tale, si ha:

$$(\mathsf{M_P^{(0)}})^T = (\frac{1}{|\mathsf{P}|}\mathsf{R_P K_P^{-1} R_P^T})^T = \frac{1}{|\mathsf{P}|}\mathsf{R_P}(\mathsf{K_P^{-1}})^T\mathsf{R_P^T} = \frac{1}{|\mathsf{P}|}\mathsf{R_P K_P^{-1} R_P^T} = \mathsf{M_P^{(0)}}.$$

Inoltre, poiché U è simmetrica:

$$(\mathsf{DUD^T})^T = \mathsf{DU^T D^T} = \mathsf{DUD^T}.$$

Resta da provare la definita positività: ricordando la definizione, bisogna mostrare che

$$\mathbf{v}^T\mathsf{M}_\mathsf{P}\mathbf{v} \geq 0 \ \ \forall \mathbf{v} \in \mathbb{R}^{N_\mathsf{P}^\mathscr{F}} \text{ e inoltre } \mathbf{v}^T\mathsf{M}_\mathsf{P}\mathbf{v} = 0 \Longleftrightarrow \mathbf{v} = \mathbf{0}.$$

Allora, sfruttando la definizione di norma euclidea si ha:

$$\mathbf{v}^T\mathsf{M}_\mathsf{P}\mathbf{v} = \mathbf{v}^T(\frac{1}{|\mathsf{P}|}\mathsf{R}_\mathsf{P}\mathsf{K}_\mathsf{P}^{-1}\mathsf{R}_\mathsf{P}^T + \mathsf{D}\mathsf{U}\mathsf{D}^T)\mathbf{v} = \mathbf{v}^T(\frac{1}{|\mathsf{P}|}\mathsf{R}_\mathsf{P}\mathsf{K}_\mathsf{P}^{-1}\mathsf{R}_\mathsf{P}^T)\mathbf{v} + \mathbf{v}^T(\mathsf{D}\mathsf{U}\mathsf{D}^T)\mathbf{v} =$$

$$= \mathbf{v}^T(\frac{1}{|\mathsf{P}|}\mathsf{R}_\mathsf{P}\mathsf{K}_\mathsf{P}^{-1/2}\mathsf{K}_\mathsf{P}^{-1/2}\mathsf{R}_\mathsf{P}^T)\mathbf{v} + \mathbf{v}^T(\mathsf{D}\mathsf{U}^{1/2}\mathsf{U}^{1/2}\mathsf{D}^T)\mathbf{v} =$$

$$= \frac{1}{|\mathsf{P}|}(\mathsf{K}_\mathsf{P}^{-1/2}\mathsf{R}_\mathsf{P}^T\mathbf{v})^T(\mathsf{K}_\mathsf{P}^{-1/2}\mathsf{R}_\mathsf{P}^T\mathbf{v}) + (\mathsf{U}^{1/2}\mathsf{D}^T\mathbf{v})^T(\mathsf{U}^{1/2}\mathsf{D}^T\mathbf{v}) =$$

$$\frac{1}{|\mathsf{P}|}\|\mathsf{K}_\mathsf{P}^{-1/2}\mathsf{R}_\mathsf{P}^T\mathbf{v}\| + \|\mathsf{U}^{1/2}\mathsf{D}^T\mathbf{v}\| \geq 0.$$

Sia ora $\mathbf{v} \in \mathbb{R}^{N_\mathsf{P}^\mathscr{F}}$ non nullo e tale che $\mathbf{v}^T\mathsf{M}_\mathsf{P}\mathbf{v} = 0$, allora

$$\frac{1}{|\mathsf{P}|}\|\mathsf{K}_\mathsf{P}^{-1/2}\mathsf{R}_\mathsf{P}^T\mathbf{v}\| + \|\mathsf{U}^{1/2}\mathsf{D}^T\mathbf{v}\| = 0,$$

che implica $\mathsf{R}_\mathsf{P}^T\mathbf{v} = \mathbf{0}$ e $\mathsf{D}^T\mathbf{v} = \mathbf{0}$. Per questo motivo $\mathbf{v} \in ker(\mathsf{D}^T)$ e per definizione di D, $\mathbf{v} \in Im(\mathsf{N}^T)$, che significa $\mathsf{R}_\mathsf{P}^T\mathbf{v} = \mathsf{R}_\mathsf{P}^T\mathsf{N}^T\mathbf{w}$ per qualche $\mathbf{w} \in \mathbb{R}^2$. Applicando la (2.19), si ottiene che $|\mathsf{P}|\mathsf{K}_\mathsf{P}\mathbf{w} = \mathbf{0}$ ed essendo K definito positivo, implica che $\mathbf{w} = \mathbf{0}$. Ma allora:

$$\mathbf{v}^T\mathsf{M}_\mathsf{P}\mathbf{v} = 0 \Longleftrightarrow \mathbf{v} = \mathbf{0}.$$

□

Tipicamente $\mathsf{U} = \widetilde{u}\mathsf{I}$, dove $\widetilde{u} > 0$ può essere scelto opportunamente; si fa la scelta di prendere $\widetilde{u} = \mathrm{tr}\,(\mathsf{M}_\mathsf{P}^{(0)})$. Così facendo, si ha $\mathsf{M}_\mathsf{P}^{(1)} = \widetilde{u}\mathsf{D}\mathsf{D}^T$ e si può usare il proiettore ortogonale del nucleo di N_P^T.

In conclusione la matrice che rappresenta il prodotto scalare è:

$$\mathsf{M}_\mathsf{P} = \frac{1}{|\mathsf{P}|}\mathsf{R}_\mathsf{P}\mathsf{K}_\mathsf{P}^{-1}\mathsf{R}_\mathsf{P}^T + \widetilde{u}(\mathsf{I} - \mathsf{N}_\mathsf{P}(\mathsf{N}_\mathsf{P}^T\mathsf{N}_\mathsf{P})^{-1}\mathsf{N}_\mathsf{P}^T). \tag{2.21}$$

2.2.3 Formulazione algebrica

Tornando alla (2.10), si può ora fornire una formulazione matriciale dell'approssimazione del problema di Darcy con le Differenze Finite Mimetiche. Una delle sue caratteristiche principali è rappresentata dal fatto che non esistono funzioni di base dello spazio, quindi in fase di costruzione delle matrici, non è necessario fornire delle funzioni test esplicite, che influiscano in modo diretto sugli elementi delle matrici; infatti, esse si riducono ad essere delle semplici funzioni indicatrici, che aiutano in fase assemblaggio.

Dalla formulazione debole, si passa, quindi, al seguente sistema lineare:

$$\begin{bmatrix} \mathsf{M} & \mathsf{B}^T \\ \mathsf{B} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} g \\ -f \end{bmatrix}, \tag{2.22}$$

in cui i vari termini hanno i seguenti significati:

- M è la matrice globale del prodotto interno $[\cdot,\cdot]_{\mathsf{P}}$. Essa viene assemblata a partire dalle matrici locali M_{P}, a loro volta costruite come precedentemente riportato. L'assemblaggio della matrice globale avviene come nei metodi classici ad elementi finiti: le M_{P} sono costruite tramite indici locali (da 1 a $N_{\mathsf{P}}^{\mathscr{F}}$) e poi si sfruttano i corrispettivi indici globali per arrivare ad M.

- B è la matrice che rappresenta l'operatore divergenza. Essa viene assemblata sfruttando la definizione di divergenza discreta (1.19) e di prodotto interno $[\cdot,\cdot]_{\mathscr{P}_h}$ (2.6), in particolare:

$$[\mathrm{div}_h \mathbf{u}_h, q_h]_{\mathscr{P}_h} = \sum_{\mathsf{P}\in\Omega_h} |\mathsf{P}|\, \mathrm{div}_h \mathbf{u}_{\mathsf{P}} q_{\mathsf{P}} = \sum_{\mathsf{P}\in\Omega_h} |\mathsf{P}|\, q_{\mathsf{P}} \frac{1}{|\mathsf{P}|} \sum_{\mathsf{f}\in\mathscr{F}_{\mathsf{P}}} \alpha_{\mathsf{P},\mathsf{f}} |\mathsf{f}| \mathbf{u}_{\mathsf{P}} =$$

$$= \sum_{\mathsf{P}\in\Omega_h} q_{\mathsf{P}} \sum_{\mathsf{f}\in\mathscr{F}_{\mathsf{P}}} \alpha_{\mathsf{P},\mathsf{f}} |\mathsf{f}| \mathbf{u}_{\mathsf{P}} = q_h^T \mathsf{B}\mathbf{u}.$$

- B^T rappresenta l'operatore derivato della divergenza e si traduce esattamente nella trasposta della matrice B.

- g è il termine noto che racchiude le condizione al bordo, in accordo con la formulazione debole.

- f è il termine noto che rappresenta la forzante del problema di Darcy, opportunamente interpolata nello spazio mimetico.

- $\mathbf{u}$ e p sono i vettori dei gradi di libertà di velocità e pressione, come riportato all'inizio del paragrafo.

Va sottolineato che, risolvendo il sistema (2.4), si trova una soluzione per la pressione di segno opposto a quella fisica; per ottenere quella cercata si deve cambiare segno a p o, come in (2.22), risolvere il sistema con una sorgente f pari al termine forzante vero cambiato di segno, in questo modo, il segno negativo è già inglobato nei dati e la pressione risulta corretta.

Per risolvere il sistema lineare si possono percorrere due strade: la prima, se le dimensioni delle matrici lo permettono, è utilizzare in metodo monolitico per l'intero sistema, ossia risolvere $\mathsf{A}x = b$, usando preferibilmente, viste le dimensioni contenute, un metodo diretto; la seconda alternativa è partire da:

$$\begin{cases} \mathsf{M}\mathbf{u} + \mathsf{B}^T p = g \\ \mathsf{B}\mathbf{u} = -f \end{cases}$$

e, ricavando $\mathbf{u}$ dalla prima equazione e sostituendolo nella seconda, risolvere:

$$\begin{cases} \mathbf{u} = -\mathsf{M}^{-1}\mathsf{B}^T p + \mathsf{M}^{-1} g \\ \mathsf{B}\mathsf{M}^{-1}\mathsf{B}^T p = f + \mathsf{B}\mathsf{M}^{-1} g \end{cases} \tag{2.23}$$

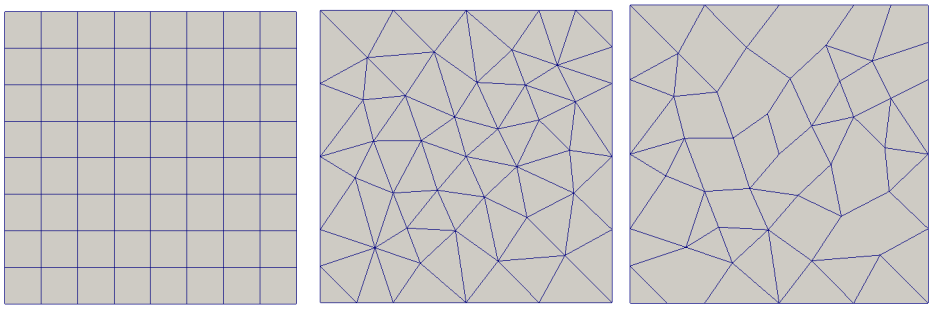
e anch'esso può essere risolto tramite un qualsiasi metodo diretto.

Nel caso in cui le dimensioni delle matrici crescano considerevolmente, i metodi diretti non sono adatti alla risoluzione del sistema lineare. E' necessario, a tal scopo, usare un solutore iterativo che, per arrivare a convergenza più velocemente, richiede l'uso di un precondizionatore efficiente.

2.3 Risultati numerici

In questo paragrafo vengono riportati i principali numerici ottenuti su alcuni casi test, con lo scopo di verificare le proprietà del metodo. E' stata creata una libreria `C++` che implementa il metodo delle Differenze Finite Mimetiche, a partire da un codice già esistente per la risoluzione del problema 2D di Darcy tramite il metodo dei volumi finiti. Tale codice si basa sugli oggetti definiti nelle librerie CGAL [1] e EIGEN [2]. Esso si occupa, in primo luogo, della costruzione della griglia di calcolo, in seguito, tramite gli algoritmi descritti in precedenza, costruisce le matrici di (2.22) e, infine, risolve il sistema lineare che ne deriva. Per maggiori dettagli sull'implementazione si veda il Capitolo 4.

Le griglie su cui verranno effettuate le simulazioni sono di tre tipi: quadratiche, triangolari e miste: per il primo caso si ottengono quindi griglie strutturate, mentre per gli altri due no. Le griglie miste sono generate nel seguente modo: data una triangolazione, si uniscono in modo casuale coppie



(a) quadrati (b) triangoli (c) mista

Figura 2.2: Le diverse tipologie di griglia su cui sono state effettuate le simulazioni

di elementi per formare quadrilateri, in secondo luogo un quadrilatero e un triangolo vengono uniti per generare un elemento pentagonale, in questo modo nella griglia sono presenti contemporaneamente elementi di tre, quattro e cinque lati; si vedano a tal proposito le Figure 2.2-2.3. L'obiettivo delle simulazioni è calcolare gli errori, sia per la pressione che per la velocità e, nel caso di griglie strutturate, anche l'ordine di convergenza del metodo, utilizzando un passo spaziale che diminuisce di volta in volta. Inoltre, utilizzando griglie generate agglomerando in modo casuale triangoli e quadrilateri si dimostra la robustezza per griglie generiche.

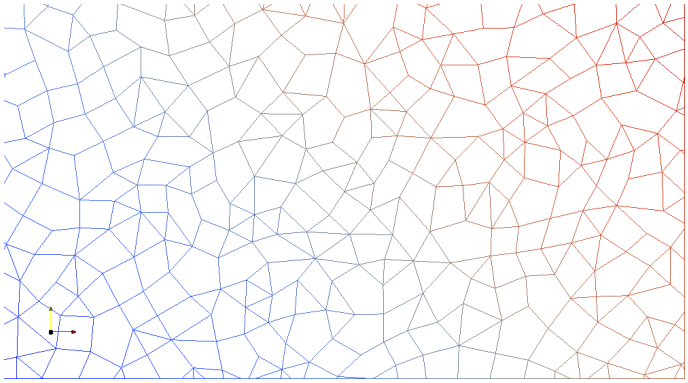


Figura 2.3: Un dettaglio della griglia mista: si possono notare elementi con 3, 4 e 5 lati

L'accuratezza della soluzione discreta $(p_h, \mathbf{u}_h)$ deve essere misurata nelle norme indotte dai prodotti scalari (2.6) e (2.7), mentre le soluzioni esatte $(p^I, \mathbf{u}^I)$ con cui vengono confrontate sono le opportune interpolazioni negli spazi considerati delle corrispondenti funzioni (1.12) e (1.13). Si ricorda che per la pressione, interpolare significa mediare sull'elemento, mentre l'interpolata della velocità è la media sulla faccia della sua componente normale. Si

definiscono allora le seguenti norme:

$$|||p^I - p_h||| = \left[p^I - p_h, p^I - p_h\right]^{1/2}_{\mathscr{P}_h} = \sqrt{\sum_{\mathsf{P}\in\Omega} \frac{1}{\mathsf{P}} \int_{\mathsf{P}} (p^I_{\mathsf{P}} - p_{h,\mathsf{P}})^T (p^I_{\mathsf{P}} - p_{h,\mathsf{P}}) dV}$$

e

$$|||\mathbf{u}^I - \mathbf{u}_h||| = \left[\mathbf{u}^I - \mathbf{u}_h, \mathbf{u}^I - \mathbf{u}_h\right]^{1/2}_{\mathscr{F}_h} = \sqrt{[\mathbf{u}^I - \mathbf{u}_h]^T \ \mathsf{M} \ [\mathbf{u}^I - \mathbf{u}_h]}.$$

Gli errori che verranno presi in considerazione sono quelli relativi, cioè:

$$err_p = \frac{|||p^I - p_h|||}{|||p^I|||} \qquad \text{e} \qquad err_u = \frac{|||\mathbf{u}^I - \mathbf{u}_h|||}{|||\mathbf{u}^I|||}.$$

A partire dagli errori è possibile stimare l'ordine di convergenza del metodo, definito come:

$$r = \frac{\log(err(h_2)) - \log(err(h_1))}{\log(h_2/h_1)}.$$

Vengono riportati ora alcuni risultati di convergenza del metodo, un studio teorico approfondito si può trovare in [21].

Teorema 2.4 (Convergenza di $(\mathbf{u}, p)$)**.** *Sia $(\boldsymbol{u}, p)$, con $p \in H^2(\Omega)$, la soluzione esatta del problema (2.4) e $(\boldsymbol{u}_h, p_h) \in \mathscr{F}_h \times \mathscr{P}_h$ la soluzione del problema discreto (2.10), allora:*

$$|||p_h - p^I|||_{\mathscr{P}_h} \leq C_1 \ h \ \|p\|_{H^2(\Omega)}$$

e

$$|||\boldsymbol{u}_h - \boldsymbol{u}^I|||_{\mathscr{P}_h} \leq C_2 \ h \ \|p\|_{H^2(\Omega)},$$

con C_1 e C_2 costanti positive indipendenti da h.

Teorema 2.5 (Superconvergenza di p)**.** *Sia $(\boldsymbol{u}, p)$, con $p \in H^2(\Omega)$, la soluzione esatta del problema (2.4), definito in un dominio Ω convesso e con condizioni di Dirichlet omogenee al bordo. Inoltre il tensore di diffusione K sia costante a tratti e il termine sorgente b in $H^1(\Omega)$. Infine sia $(\boldsymbol{u}_h, p_h) \in \mathscr{F}_h \times \mathscr{P}_h$ la soluzione del problema mimetico (2.10). Allora esiste una costante positiva C indipendente da h tale che:*

$$|||p_h - p^I|||_{\mathscr{P}_h} \leq C \ h^2(\|p\|_{H^2(\Omega)} + |b|_{H^1(\Omega)}).$$

Caso test 0. Il primo test da fare è il *check di consistenza*: in base alla proprietà (2.12), se la soluzione esatta è lineare, l'errore deve essere nullo. Le

simulazioni mostrano che se si prende una soluzione esatta del tipo $p_{ex} = x+y$, ciò è verificato per tutte e tre le tipologie di griglia.

Caso test 1. Il primo esempio che si vuole risolvere è un semplice caso sintetico: sia $\Omega = [0,1] \times [0,1]$ e $\mathsf{K} = \mathsf{I}$, cioè si assume che il mezzo poroso sia isotropo. La soluzione esatta per la pressione è:

$$p_{ex} = \sin(\pi x)\sin(\pi y).$$

Una volta nota la pressione esatta, calcolandone il gradiente e moltiplicando per K, come stabilito dalla legge di Darcy, si può ricavare anche il campo di velocità esatto:

$$\mathbf{u}_{ex} = \begin{pmatrix} \pi\cos(\pi x)\sin(\pi y) \\ \pi\sin(\pi x)\cos(\pi y) \end{pmatrix}$$

e infine, applicando la divergenza, per la conservazione della massa, si ottiene la forzante del problema:

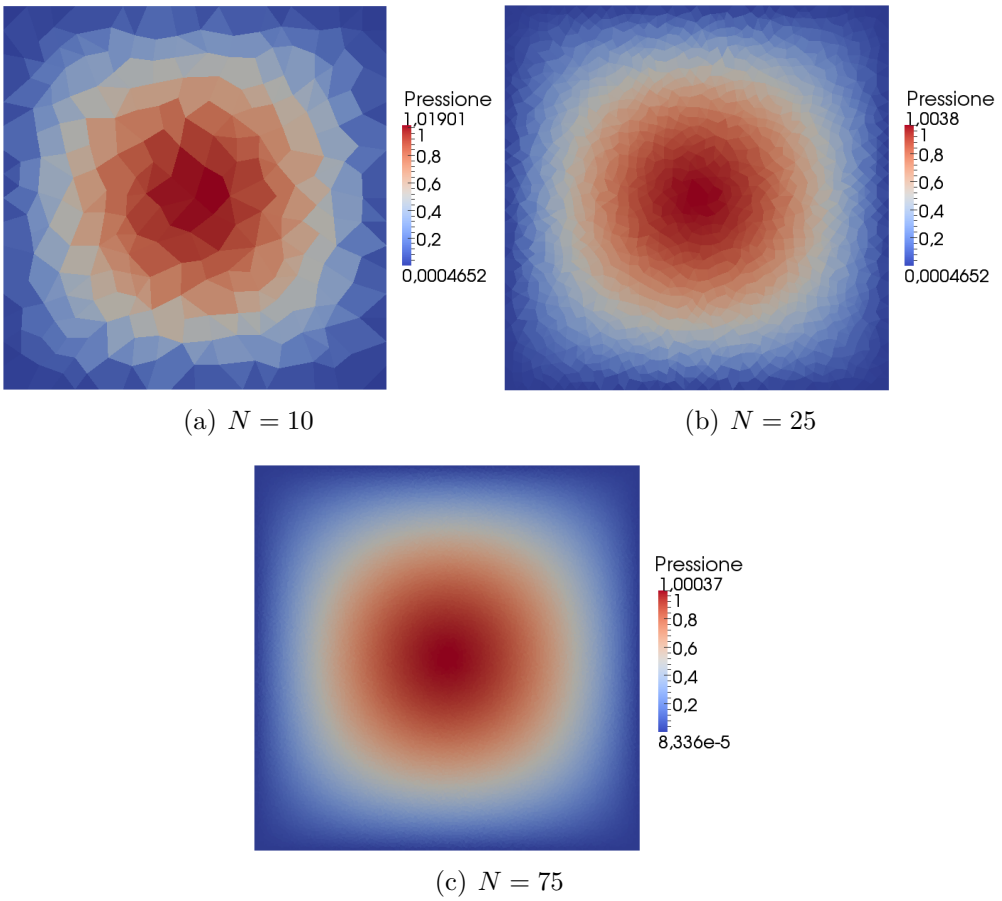
$$b = \pi^2 \sin(\pi x)\sin(\pi y).$$

Il problema che si andrà a risolvere è, pertanto:

$$\begin{cases} \mathbf{u} + \nabla p = 0 & \text{in } \Omega \\ \mathrm{div}\mathbf{u} = \pi^2 \sin(\pi x)\sin(\pi y) & \text{in } \Omega \\ p = 0 & \text{su } \partial\Omega\ . \end{cases} \tag{2.24}$$

In Tabella 2.1 sono riportati gli errori di approssimazione per le soluzioni discrete $\mathbf{u}_h$ e p_h, calcolati nelle norme sopra definite, insieme ai rispettivi ordini di convergenza. I risultati si riferiscono ad una griglia strutturata di quadrati, in cui $N = 1/h$ rappresenta il numero di elementi per lato. Per quanto riguarda la pressione, dalla tabella si può notare che l'errore decresce come $1/N^2$ e questo conferma la superconvergenza mostrata dal Teorema 2.5, inoltre, osservando la Figura 2.5 si osserva che la superconvergenza continua a valere anche nel caso di griglie non strutturate (triangolari e miste). Gli errori compiuti sulla velocità, invece, sono diversi a seconda che le griglie siano strutturate o no: nel primo caso si ha un abbattimento quadratico dell'errore, che diventa lineare per griglie non strutturate. E' bene notare che, nel caso di griglie triangolari, le simulazioni sono state fatte al decrescere del diametro massimo degli elementi, lo stesso vale per la triangolazione da cui si parte per generare le griglie miste. In Figura 2.4 sono riportate le soluzioni per la pressione, calcolate su una griglia mista a tre livelli di raffinamento.

Caso test 2. Il secondo caso ha lo stesso dominio e la stessa soluzione esatta per la pressione del primo, ma ora il mezzo poroso è eterogeneo, ossia ha un



(a) $N = 10$ (b) $N = 25$

(c) $N = 75$

Figura 2.4: Plot della soluzione discreta del caso test 1, con griglie miste

Tabella 2.1: Errori relativi caso test 1, griglia strutturata di quadrati. r_p e r_u sono gli ordini di convergenza per la pressione e la velocità

N	**Errore pressione**	**r$_p$**	**Errore velocità**	**r$_u$**
2	0.850551	-	0.570796	-
4	0.207242	2.0317	0.110721	2.3660
8	0.0515039	2.0086	0.0261722	2.0808
16	0.0128572	2.0021	0.00645454	2.0196
32	3.21315e-3	2.0005	1.60819e-3	2.0049
64	8.03215e-4	2.0001	4.01708e-4	2.0012
128	2.00799e-4	2.0000	1.00406e-4	2.0003

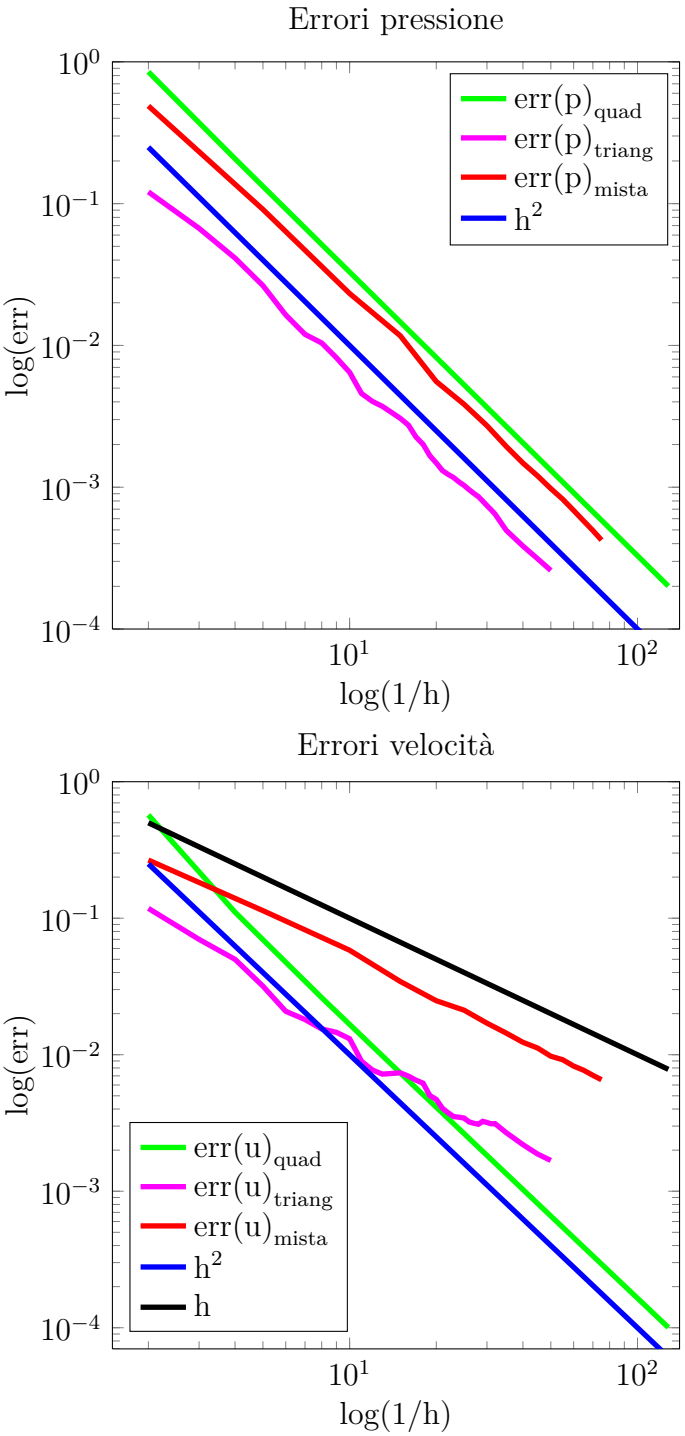


Figura 2.5: Errori di approssimazione caso test 1

tensore di permeabilità variabile, della forma:

$$\mathsf{K} = \begin{pmatrix} (x+1)^2 + y^2 & -xy \\ -xy & (x+1)^2 \end{pmatrix}.$$

Conseguenza di ciò è un termine sorgente che deve soddisfare:

$$b = \mathrm{div}[\nabla(\mathsf{K}\ p_{ex})],$$

dove, ora, K non è più costante e resta sotto il segno di derivata. Si ottiene quindi:

$$b = [1,1]d\mathsf{K}\nabla p_{ex} + \mathsf{K}_{11}\frac{\partial^2 p_{ex}}{\partial x \partial x} + (\mathsf{K}_{12} + \mathsf{K}_{21})\frac{\partial^2 p_{ex}}{\partial x \partial y} + \mathsf{K}_{22}\frac{\partial^2 p_{ex}}{\partial y \partial y}, \tag{2.25}$$

dove

$$d\mathsf{K} = \begin{pmatrix} \partial \mathsf{K}_{11}/\partial x & \partial \mathsf{K}_{12}/\partial x \\ \partial \mathsf{K}_{21}/\partial y & \partial \mathsf{K}_{22}/\partial y \end{pmatrix}.$$

Anche in questo caso si considerano condizioni al bordo di Dirichlet omogenee e si risolve:

$$\begin{cases} \mathbf{u} + \mathsf{K}\nabla p = 0 & \text{in } \Omega \\ \mathrm{div}\mathbf{u} = b & \text{in } \Omega \\ p = 0 & \text{su } \partial\Omega\ . \end{cases} \tag{2.26}$$

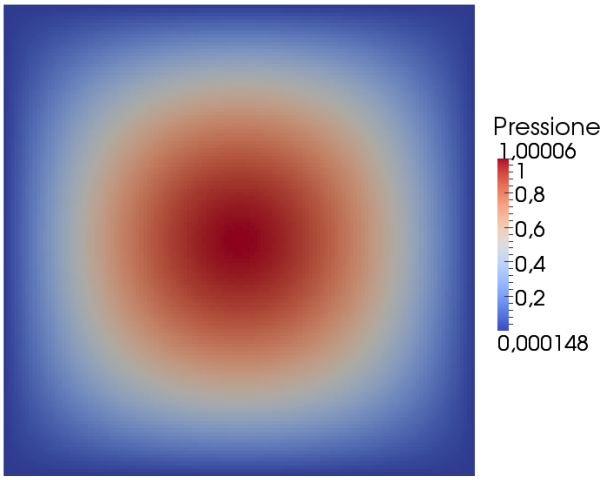


Figura 2.6: Plot della soluzione discreta del caso test 2, con griglia strutturata di quadrati ($N = 128$)

Nel secondo caso test si notano andamenti simili al primo (Figura 2.9). Nel caso di griglia strutturata, la Tabella 2.2 mostra un abbattimento quadratico

Tabella 2.2: Errori relativi caso test 2, griglia strutturata di quadrati. r_p e r_u sono gli ordini di convergenza per la pressione e la velocità

N	**Errore pressione**	$\mathbf{r}_p$	**Errore velocità**	$\mathbf{r}_u$
2	0.914246	-	0.683883	-
4	0.23297	1.9724	0.155368	2.3181
8	0.0594277	1.9709	0.0399346	1.9600
16	0.0149698	1.9891	0.0101201	1.9804
32	3.75039e-3	1.9969	2.54002e-3	1.9943
64	9.3811e-4	1.9992	6.35655e-4	1.9985
128	2.3456e-4	1.9998	1.58955e-4	1.9996

dell'errore, sia per la pressione che per la velocità. Se la griglia diventa generica, la p_h mantiene lo stesso decadimento, mentre per $\mathbf{u}_h$ si nota una pendenza pari a uno, in accordo con il Teorema 2.4. In Figura 2.6 è riportata la soluzione per la pressione calcolata su una griglia strutturata di quadrati.

Caso test 3. Per completare la validazione del metodo e del codice, come terzo esempio si considera una pressione esatta che non si annulli al bordo:

$$p_{ex} = x^3y^2 + x\sin(2\pi xy)\sin(2\pi y).$$

Essa è definita in $\Omega = [0,1]^2$ e il mezzo ha permeabilità pari a:

$$\mathsf{K} = \begin{pmatrix} (x+1)^2 + y^2 & 0 \\ 0 & (x+1)^2 \end{pmatrix},$$

quindi, il termine sorgente si calcola ancora con la (2.25). Il problema diventa:

$$\begin{cases} \mathbf{u} + \mathsf{K}\nabla p = 0 & \text{in } \Omega \\ \text{div}\mathbf{u} = b & \text{in } \Omega \\ p = p_{ex} & \text{su } \partial\Omega\ . \end{cases} \tag{2.27}$$

Anche nel terzo caso test si osservano gli andamenti dell'errore attesi (Figura 2.9). L'errore commesso sulla pressione mostra un decadimento quadratico per tutti i tipi di griglia; quello sulla velocità mantiene tale andamento solo per la griglia strutturata, mentre, analogamente agli esempi precedenti, per griglie non strutturate l'errore viene abbattuto come $1/N$. In Figura 2.8 è riportata la soluzione per la pressione calcolata su una griglia triangolare.

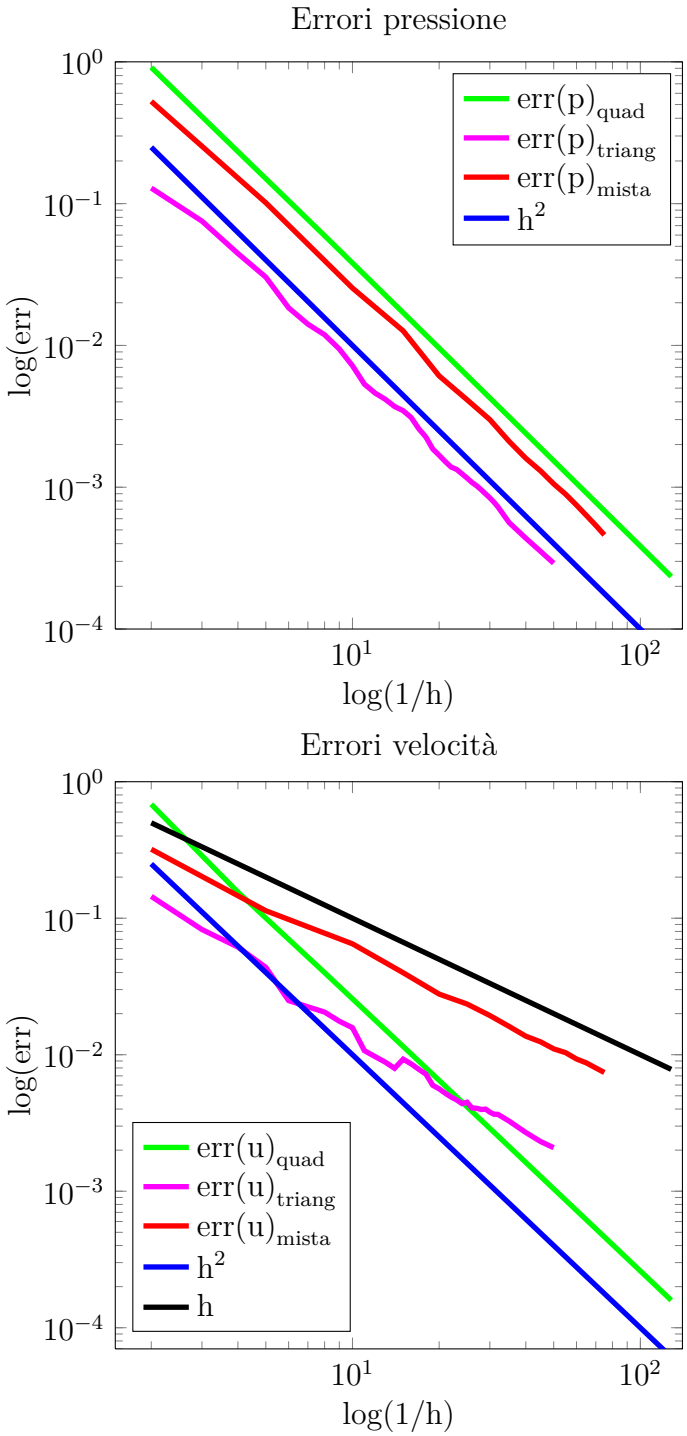


Figura 2.7: Errori di approssimazione caso test 2

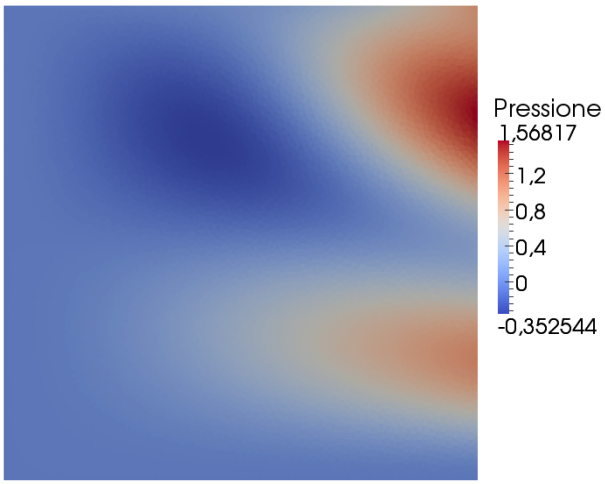


Figura 2.8: Plot della soluzione discreta del caso test 3, con griglia triangolare con lato massimo $h = 0.02$

Tabella 2.3: Errori relativi caso test 3, griglia strutturata di quadrati. r_p e r_u sono gli ordini di convergenza per la pressione e la velocità

N	**Errore pressione**	$\mathbf{r}_p$	**Errore velocità**	$\mathbf{r}_u$
2	2.93332	-	3.68848	-
4	0.806956	1.8620	0.935273	1.9796
8	0.205016	1.9768	0.285251	1.7132
16	0.0525855	1.9830	0.103374	1.4644
32	0.0133959	1.9729	0.0316403	1.7080
64	3.380e-3	1.9867	8.85358e-3	1.8374
128	8.47814e-4	1.9952	2.3909e-3	1.8887

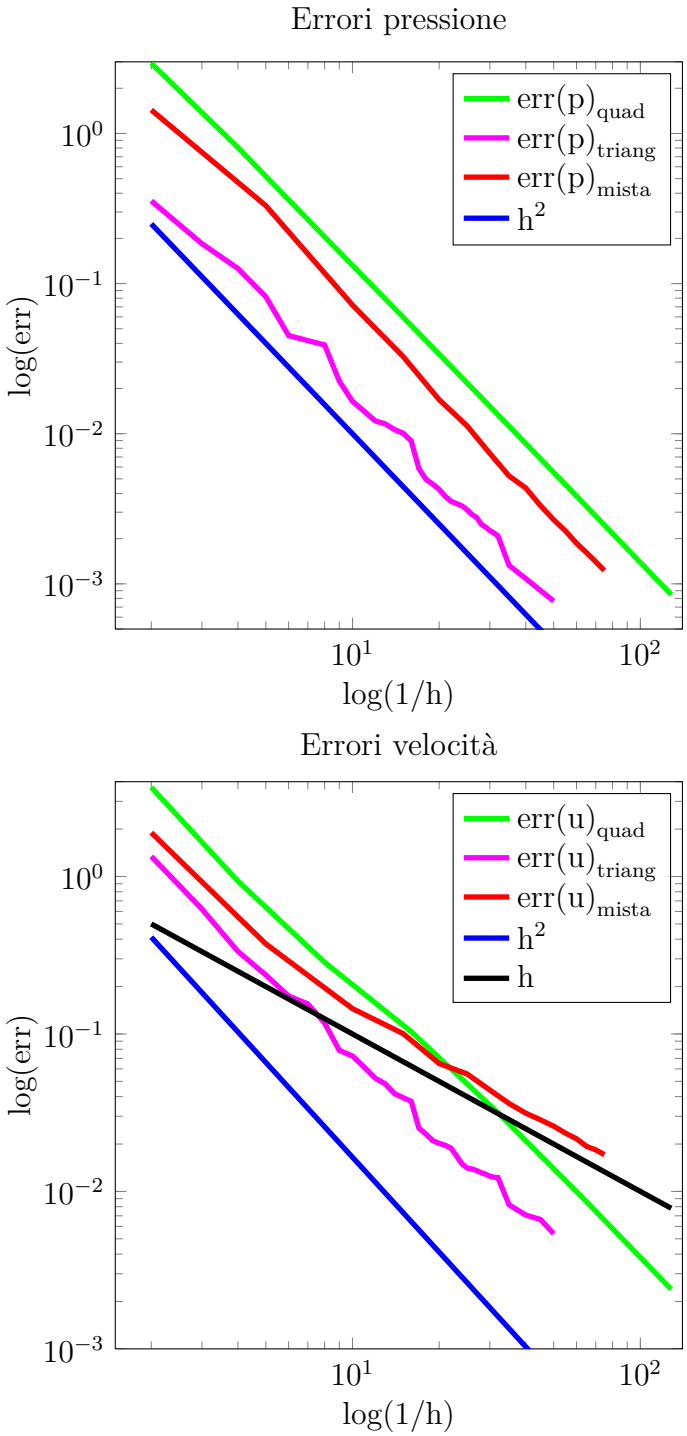


Figura 2.9: Errori di approssimazione caso test 3

Capitolo 3

Flusso in mezzi porosi fratturati

3.1 Introduzione

In ambito geologico, osservando un mezzo poroso da vicino, per esempio una roccia sedimentaria, si può notare che non è completamente omogeneo. Esso infatti presenta spesso delle discontinuità evidenti, come mostrato in Figura 3.1. Tali discontinuità vengono dette *fratture* e hanno un ruolo rilevante nel flusso dei fluidi lungo il mezzo. Esse si possono considerare come spazio vuoto tra due superfici ruvide (si veda ad esempio [5]) ed è importante sottolineare come il volume occupato dal mezzo poroso e dai suoi pori sia molto più grande rispetto a quello occupato dalle fatture stesse; inoltre lo spessore della frattura è, generalmente, di qualche ordine di grandezza più piccolo della scala del mezzo poroso circostante e di qualche ordine di grandezza più grande della scala dei pori.

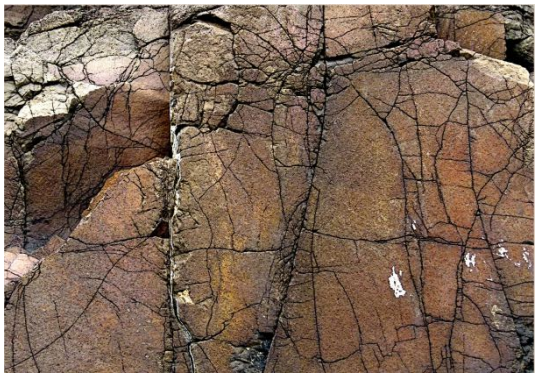


Figura 3.1: Esempio di roccia fratturata

Geologicamente con il termine frattura si possono intendere diverse tipologie di discontinuità. Un *giunto* è una frattura vuota, cioè non riempita da

altri materiali, in cui i blocchi distinti non hanno subito uno spostamento reciproco; lo spessore è dell'ordine dei centimetri. Una *vena* è una frattura rimineralizzata, ossia riempita dopo la rottura da altro materiale sedimentario: per questo motivo il fluido non può più scorrere al suo interno. Una *faglia* è una frattura in cui c'è stato un significativo spostamento dei blocchi, in direzione parallela alla faglia; solitamente, le faglie hanno uno spessore più grande delle normali fratture (nell'ordine dei metri, con un'estensione anche di alcuni kilometri), ma comunque piccolo se comparato alla dimensione del dominio in cui sono immerse. Nel caso le fratture siano molteplici, si parla di *network di fratture*.

In presenza di fratture, le proprietà macroscopiche del mezzo cambiano drasticamente: faglie e fratture infatti, a seconda dello spessore e della permeabilità, possono fungere da barriere o direzioni preferenziali per il moto del fluido. In particolare, se la permeabilità della frattura è inferiore a quella del materiale in cui è immersa, questa si comporterà come una barriera per il fluido, viceversa, il fluido scorre più facilmente in fratture molto permeabili. Per questo motivo è importante fornire una caratterizzazione accurata dei mezzi porosi fratturati, allo scopo di avere modelli matematici che ben rappresentano il fenomeno fisico. Un esempio di applicazione che richiede un'accurata modellazione delle fratture è lo stoccaggio della CO_2, sia dal punto di vista geomeccanico, che da quello idrologico: al momento dalla sua iniezione, essendo più leggera del materiale circostante, la CO_2 tende a salire verso l'alto e la presenza di fratture potrebbe portare ad una perdita verso strati superficiali e all'accumulazione inaspettata di anidride carbonica. Altre applicazioni in cui le fratture hanno un effetto rilevante sono lo studio delle falde acquifere, dei campi geotermici, dei giacimenti di petrolio e gas e delle fonti di idrocarburi non convenzionali. In questi contesti le fratture influenzano pesantemente il moto del fluido e le tecniche numeriche classiche (come l'omogeneizzazione) non sono più in grado di riprodurre al meglio il fenomeno.

Nel caso in cui lo spessore della frattura sia piccolo rispetto alla scala del mezzo, l'approccio che si segue nella modellizzazione consiste, appunto, nel trascurare tale spessore. Questo significa che viene usato un modello ridotto per tener conto del flusso nella frattura: se il mezzo poroso ha n-dimensioni, la frattura è rappresentata come un oggetto $(n-1)$dimensionale e deve essere accoppiata con il resto del mezzo tramite opportune condizioni. Una volta scelti il modello per la frattura e le condizioni di accoppiamento bisogna fornirne una discretizzazione numerica: le tecniche usate nei recenti studi comprendono i metodi delle differenze finite, dei volumi finiti e degli elementi finiti classici e misti (si vedano [36], [39]). In tutti questi casi, le fratture sono rappresentate come interfacce tra le celle della griglia e, quindi, si dice che questa è confor-

me alle fratture. L'alternativa sono le griglie non conformi, in cui le fratture non sono rappresentate dai bordi delle celle [27]; metodi che fanno uso di tali griglie sono, per esempio, gli elementi finiti estesi (XFEM), [26]. Rimuovendo il vincolo di conformità della griglia, si possono modellizzare casi molto più realistici, con un network di fratture complesso; inoltre, nel caso di fratture dinamiche, cioè che vengono generate o evolvono nel tempo, non è necessario creare una nuova griglia dopo ogni modifica, ma le simulazioni si possono fare a partire sempre dalla stessa.

Se lo spessore della frattura non è trascurabile, cade l'ipotesi alla base del modello ridotto e si richiede una rappresentazione dell'oggetto nella stessa dimensione del mezzo poroso; sono necessarie, a tal scopo, tecniche di generazione di griglia avanzate. Lo svantaggio di questo approccio è legato alla dimensione delle celle del dominio, più queste sono piccole e più il sistema lineare che ne deriva sarà grande, portando a costi computazionali enormi o, talvolta, insostenibili.

In questo capitolo vengono presentati in dettaglio due dei metodi di discretizzazione precedentemente accennati, con lo scopo di calcolare il campo di pressione e di velocità di un fluido che scorre in un mezzo poroso in cui siano presenti delle fratture. La prima tecnica si basa sui Volumi Finiti, la seconda sulle Differenze Finite Mimetiche. Quest'ultima, in particolare, rappresenta uno dei campi di ricerca più recenti, data la sua adattabilità a griglie generiche e le sue buone proprietà di convergenza; per tale motivo, dopo aver introdotto il modello e la sua discretizzazione, è stato fatto in primo luogo lo studio di convergenza per un caso test di cui si conosce la soluzione esatta, in secondo luogo questa tecnica è stata applicata a degli esempi realistici, per verificare la dipendenza dai parametri fisici e fare un confronto con il metodo dei Volumi Finiti.

3.2 Il modello matematico

Per calcolare il flusso di un fluido in un mezzo poroso fratturato sono necessari tre elementi:

- Un modello per la matrice solida
- Un modello per le fratture
- Le condizioni di accoppiamento per i due modelli

Così facendo, invece di ricavare un modello completo per il mezzo poroso fratturato, si opera un disaccoppiamento dei singoli sottosistemi, che devono essere riaccoppiati con opportune condizioni (Figura 3.2).

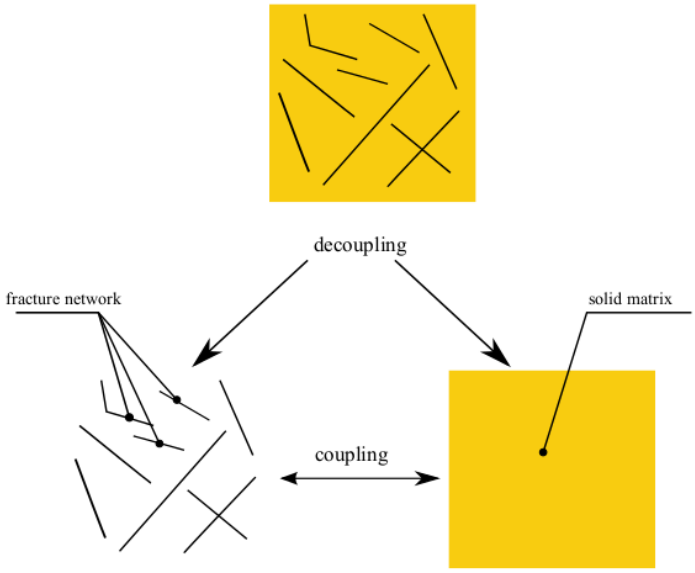


Figura 3.2: Sdoppiamento del modello intero in due modelli separati [41]

3.2.1 Modello per la matrice solida

Il modello per la matrice solida formulato sull'interno dominio Ω attraversato da fratture è costituito dal classico problema di Darcy:

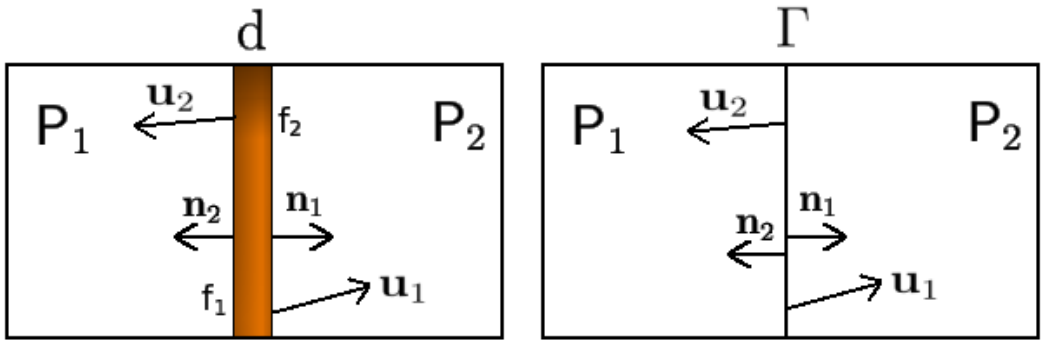
$$\begin{cases} \mathbf{u} + \mathsf{K}\nabla p = 0 & \text{in } \Omega \\ \text{div}\mathbf{u} = b & \text{in } \Omega \\ p = g^D & \text{su } \Gamma^D \\ \mathbf{u} \cdot \mathbf{n} = g^N & \text{su } \Gamma^N \ . \end{cases} \tag{3.1}$$

Le notazioni utilizzate sono quelle del capitolo precedente, inoltre continuano a valere le ipotesi fatte, in particolare si trascurano gli effetti dovuti alla gravità.

3.2.2 Modello per le fratture

Per calcolare la pressione nelle fratture che a priori non è nota, serve un ulteriore modello. L'ipotesi da cui partire è considerare lo spessore della frattura trascurabile rispetto al resto del mezzo; in questo modo la frattura si rappresenta come un'interfaccia approssimata con le facce delle celle della mesh. Nella frattura si definiscono una pressione p_Γ ed un flusso $\mathbf{u}_\Gamma$: quest'ultimo rappresenta l'integrale lungo la sezione trasversale del flusso netto, dimensionalmente si tratta di una velocità moltiplicata per una lunghezza.

Si consideri un dominio $\Omega \subset \mathbb{R}^2$ in cui, per semplicità, è presente una sola frattura, rappresentata da un segmento unidimensionale Γ, che divide Ω nei



(a) frattura con il suo spessore d (b) frattura come interfaccia

Figura 3.3: Dominio con una frattura

due sottodomini P_1 e P_2 (si veda, per esempio, la Figura 3.3). I parametri fisici che la caratterizzano sono lo spessore l_Γ e la permeabilità K_Γ. Nel definire la permeabilità si deve tenere che a causa della diversa conformazione geologica di matrice solida e frattura, il fluido può scorrere in due diverse direzioni: normale e tangenziale alla frattura, per questo motivo anche K_Γ deve essere scomposta nelle due componenti normale e tangenziale, ottenendo:

$$\mathsf{K}_\Gamma = \begin{pmatrix} \mathsf{K}_{\Gamma,n} & 0 \\ 0 & \mathsf{K}_{\Gamma,t} \end{pmatrix}.$$

L'idea che si sfrutta per derivare il modello si basa sulla scomposizione degli operatori differenziali e delle velocità lungo le due direzioni normale e tangenziale e sulla media integrale lungo la direzione trasversale alla frattura, in modo da ricavare un modello (n-1)dimensionale; per maggiori dettagli si veda [38]. Il problema che ne deriva è:

$$\begin{cases} \mathrm{div}_\Gamma \mathbf{u}_\Gamma = q_\Gamma + (\mathbf{u}_1 \cdot \mathbf{n} - \mathbf{u}_2 \cdot \mathbf{n}) & \text{su } \Gamma \\ \mathbf{u}_\Gamma + l_\Gamma \, \mathsf{K}_{\Gamma,t} \nabla_\Gamma p_\Gamma = 0 & \text{su } \Gamma \\ p_\Gamma = g_\Gamma^D & \text{su } \partial\Gamma, \end{cases} \tag{3.2}$$

dove gli operatori differenziali sono definiti lungo la frattura e corrispondono a:

$$\mathrm{div}_\Gamma \mathbf{v} = \mathrm{div}\mathbf{v} - \nabla(\mathbf{v} \cdot \mathbf{n}) \cdot \mathbf{n} \qquad \text{e} \qquad \nabla_\Gamma q = \nabla q - \nabla q \cdot \mathbf{n},$$

con $\mathbf{n} = \mathbf{n}_1 = -\mathbf{n}_2$ normale della frattura. La prima equazione di (3.2) è la conservazione della massa nella frattura, in essa compaiono due termini forzanti: il primo rappresenta una sorgente (o pozzo) che può essere presente all'interno, il secondo termine tiene conto del contributo di flusso netto dei sottodomini

rispetto alla frattura; si noti che essendo la normali uguali ed opposte, tale termine può essere alternativamente scritto come $(\mathbf{u}_1 \cdot \mathbf{n}_1 + \mathbf{u}_2 \cdot \mathbf{n}_2)$.
La seconda equazione è la legge di Darcy integrata lungo la direzione trasversale, per questo motivo compare lo spessore l_Γ nel termine di destra. Le due equazioni vanno completate dalle condizioni al bordo che possono essere di Dirichlet o Neumann.

3.2.3 Condizioni di accoppiamento

Affinché i modelli (3.1) e (3.2) non siano indipendenti tra loro occorre specificare delle condizioni di accoppiamento. Esse vanno fornite all'interfaccia che rappresenta la frattura. A tal scopo si consideri un caso semplice, rappresentato in Figura 3.3, in cui il dominio Ω è diviso in due sottodomini da un'unica frattura Γ. Su tale interfaccia, oltre ai vettori normali e la pressione p_Γ, si definiscono le velocità uscenti $\mathbf{u}_1$ e $\mathbf{u}_2$; nei sottodomini P_1 e P_2 si definiscono le pressioni p_1 e p_2. La frattura è caratterizzata dai parametri fisici l_Γ e K_Γ introdotti in precedenza.

Si definiscano il salto e la media attraverso la frattura come:

$$[\![v]\!] := v_1 - v_2, \qquad \{v\} := \frac{v_1 + v_2}{2};$$

allora una prima semplice strategia per accoppiare i modelli è la seguente:

1. La pressione nella frattura sia la media dei valori delle pressioni nelle due zone di matrice porosa ai suoi lati:

$$p_\Gamma = \{p\} \qquad \text{su } \Gamma.$$

2. La media della componente normale della velocità sia proporzionale al salto di pressione attraverso Γ:

$$\eta_\Gamma \{\mathbf{u} \cdot \mathbf{n}\} = [\![p]\!] \qquad \text{su } \Gamma;$$

in cui il coefficiente di proporzionalità è definito come $\eta_\Gamma = \frac{l_\Gamma}{\mathsf{K}_{\Gamma,n}}$.

Sommandole e sottraendole opportunamente, le condizioni possono essere riscritte come:

$$\begin{aligned} \frac{1}{2}\mathbf{u}_1 \cdot \mathbf{n} + \frac{1}{2}\mathbf{u}_2 \cdot \mathbf{n} &= \eta_\Gamma^{-1}(p_1 - p_\Gamma) && \text{su } \Gamma \\ \frac{1}{2}\mathbf{u}_1 \cdot \mathbf{n} + \frac{1}{2}\mathbf{u}_2 \cdot \mathbf{n} &= \eta_\Gamma^{-1}(p_\Gamma - p_2) && \text{su } \Gamma. \end{aligned}$$

Questo è un caso particolare ($\xi = \frac{1}{2}$) delle seguenti condizioni generali di interfaccia [38]:

$$\begin{aligned} \xi\mathbf{u}_1 \cdot \mathbf{n} + (1-\xi)\mathbf{u}_2 \cdot \mathbf{n} &= \eta_\Gamma^{-1}(p_1 - p_\Gamma) \qquad \text{su } \Gamma \\ (1-\xi)\mathbf{u}_1 \cdot \mathbf{n} + \xi\mathbf{u}_2 \cdot \mathbf{n} &= \eta_\Gamma^{-1}(p_\Gamma - p_2) \qquad \text{su } \Gamma; \end{aligned} \tag{3.3}$$

dove $\xi \in [0,1]$ è un parametro di chiusura del modello. Per $\xi > \frac{1}{2}$ queste condizioni si possono riscrivere come:

$$\begin{aligned} \eta_\Gamma[\![\mathbf{u} \cdot \mathbf{n}]\!] &= \frac{4}{2\xi - 1}(\{p\} - p_\Gamma) \qquad \text{su } \Gamma \\ \eta_\Gamma\{\mathbf{u} \cdot \mathbf{n}\} &= [\![p]\!] \qquad \text{su } \Gamma. \end{aligned}$$

Si osservi che, anche questa volta, è possibile sostituire l'unica normale $\mathbf{n}$ con le due normali riferite ai singoli sottodomini, ricordando che esse sono di segno opposto, $\mathbf{n}_1 = -\mathbf{n}_2$.

Osservazione 3.1. Il modello completo ora dipende solo da due parametri fisici, legati alle proprietà della frattura: il prodotto $\mathsf{K}_{\Gamma,t} l_\Gamma$ e il rapporto $\mathsf{K}_{\Gamma,n}/l_\Gamma$. Il primo coefficiente è relativo al salto della componente normale della velocità (discontinuità della velocità nelle facce di frattura); il secondo è relativo al salto di pressione (discontinuità della pressione nella frattura).

Il coefficiente $\mathsf{K}_{\Gamma,t} l_\Gamma$ rappresenta una permeabilità equivalente per il flusso lungo frattura. Quando esso è dello stesso ordine di grandezza della permeabilità nei sottodomini in cui la frattura è immersa ($\mathsf{K}_{\Gamma,t} l_\Gamma \simeq \mathsf{K}_i,\ i = 1,2$) e cioè quando la permeabilità nella frattura è sufficientemente grande, allora il fluido tende a scorrere lungo la discontinuità nella roccia. In questo caso, il salto di velocità nella frattura ($\mathbf{u}_1 \cdot \mathbf{n}_1 + \mathbf{u}_2 \cdot \mathbf{n}_2$) è in generale non nullo, perché il contributo della frattura al flusso del fluido non è trascurabile.

Il coefficiente $\mathsf{K}_{\Gamma,n}/l_\Gamma$ rappresenta una resistività equivalente che si incontra attraversando la frattura. Nel caso in cui esso sia dello stesso ordine di grandezza delle K_i e il coefficiente $\mathsf{K}_{\Gamma,t} l_\Gamma$ sia piccolo, il fluido non tende a scorrere lungo la frattura e il salto di velocità è quasi nullo. Le condizioni di accoppiamento si riducono a ([38]):

$$\begin{aligned} \mathbf{u}_1 \cdot \mathbf{n}_1 + \mathbf{u}_2 \cdot \mathbf{n}_2 &= 0 \qquad \text{su } \Gamma \\ \mathbf{u}_2 \cdot \mathbf{n}_2 &= \frac{\mathsf{K}_{\Gamma,t}}{l_\Gamma}(p_2 - p_1) \qquad \text{su } \Gamma. \end{aligned}$$

3.3 Volumi Finiti

La discretizzazione ai Volumi Finiti viene fatta a partire dalla formulazione primale del problema (3.1), cioè quella che non richiede l'uso della variabile duale $\mathbf{u}$; quindi, ricavando $\mathbf{u}$ dalla prima equazione e sostituendola nella seconda si ottiene:

$$\begin{cases} -\mathrm{div}(\mathsf{K}\nabla p) = b & \text{in } \Omega \\ p = g^D & \text{su } \Gamma^D \\ -\mathsf{K}\nabla p \cdot \mathbf{n} = g^N & \text{su } \Gamma^N. \end{cases} \tag{3.4}$$

In questo caso la permeabilità non è più considerata come un tensore 2×2, ma è una funzione scalare dello spazio, $\mathsf{K} = \mathsf{K}(\mathbf{x})$; così facendo si suppone che il mezzo poroso sia isotropo, se pur eterogeneo, [29].

Un approccio classico per risolvere (3.4) è il metodo degli Elementi Finiti, tuttavia essi non costituiscono un metodo conservativo per la velocità. Infatti non essendo la velocità discreta computata direttamente, bisogna ricavarla a partire dalla pressione discreta p_h: una strada percorribile è sfruttare la legge di Darcy e calcolare $\mathbf{u}_h = -\mathsf{K}\nabla p_h$. Tuttavia questo approccio presenta due problemi non trascurabili:

- La massa locale non si conserva, cioè non è vero che $\mathrm{div}(\mathbf{u}_h) = b$ su tutti i triangoli della mesh.
- Il flusso $\mathbf{u}_h \cdot \mathbf{n}$ lungo le facce dei triangoli è discontinuo, cioè detti K_1 e K_2 due triangoli del dominio che hanno in comune la faccia e, si ha che

$$(\mathbf{u}_h|_{K_1} \cdot \mathbf{n}_1 + \mathbf{u}_h|_{K_2} \cdot \mathbf{n}_2)_{|\mathsf{e}} \neq 0 \qquad \forall \mathsf{e}.$$

Per i problemi applicativi di interesse è necessaria la conservazione della massa e del flusso: i Volumi Finiti rappresentano un'alternativa che garantisce questa proprietà.

Il primo passo della discretizzazione ai Volumi Finiti consiste nel dividere il dominio Ω in un insieme di poligoni (o poliedri nel caso tridimensionale) $\Omega_i \subset \Omega$, detti *volumi di controllo*, tali che la loro unione coincida con il dominio e la loro intersezione sia vuota (si veda, per esempio [42]). L'equazione (3.4) viene integrata su ogni *volume* Ω_i, ottenendo il sistema di equazioni

$$-\int_{\Omega_i} \mathrm{div}(\mathsf{K}\nabla p) dV = \int_{\Omega_i} b \, dV, \qquad i = 1, ..., N;$$

dove N è il numero totale di volumi di controllo. Applicando il teorema della divergenza al primo termine e indicando con $\mathbf{n}_i$ il versore normale esterno a

$\partial\Omega_i$, si ottiene:

$$-\int_{\partial\Omega_i} \mathsf{K}\nabla p \cdot \mathbf{n}_i \, dS = \int_{\Omega_i} b \, dV, \qquad i = 1, ..., N;$$

Se si indica con L_i il numero di lati l_{ij} del volume di controllo e con $\mathbf{n}_{ij}$, $j = 1, ..., L_i$ i versori normali uscenti dai lati, si può riscrivere:

$$-\sum_{j=1}^{L_i} \int_{l_{ij}} \mathsf{K}\nabla p \cdot \mathbf{n}_{ij} \, dS = \int_{\Omega_i} b \, dV, \qquad i = 1, ..., N. \tag{3.5}$$

Occorre, a questo punto, stabilire come rappresentare la pressione p in ogni volume di controllo; per il caso in esame, essa viene collocata in un punto interno al volume, tipicamente il baricentro, portando ad un metodo *cell-centered*. Con questo approccio, poiché non c'è nessun nodo di pressione che giace sul bordo del dominio, l'imposizione delle condizioni al contorno richiederà degli accorgimenti particolari; alternativamente, l'incognita si può rappresentare nei vertici delle celle (metodi *vertex-centered*), in modo da aggirare questo problema, [28]. Un ulteriore aspetto che caratterizza la discretizzazione è come approssimare gli integrali (sulle celle e sulle facce): a seconda della forma dei volumi, si può scegliere un'opportuna formula di quadratura. Una volta operata tale scelta, il termine noto in (3.5) assume un valore ben definito su ogni cella, che si indicherà con b_i.

Per approssimare il termine in pressione si applica un metodo di approssimazione a due punti, detto TPFA (*Two-Point Flux Approximation*), [32]. L'idea principale di tale metodo è esprimere il flusso uscente da una faccia e, definito da:

$$F_{\mathsf{e}} = -\int_{\mathsf{e}} \mathsf{K}\nabla p \cdot \mathbf{n}_{ij} \, dS, \tag{3.6}$$

come la differenza delle pressioni calcolate al centro delle celle che contengono la faccia.

A tale scopo occorre fornire qualche definizione: sia e_{12} la faccia su cui si vuole calcolare il flusso, mentre P_1 e P_2 sono le celle che condividono tale faccia. L'area della faccia, che nel caso 2D è una lunghezza, si indica con $A_1 = A_2 = |\mathsf{e}_{12}|$; C_1 e C_2 sono i baricentri delle celle, mentre C_0 è il centro della faccia; $\mathbf{f}_1$ e $\mathbf{f}_2$ sono i vettori che uniscono i baricentri delle celle con quello di e_{12} e D_1, D_2 sono le lunghezze di tali vettori. Infine si denotano con $\mathbf{n}_1$ e $\mathbf{n}_2$ le normali uscenti dalla faccia in questione rispetto alle due celle, che per costruzione saranno uguali e opposte. In Figura 3.4 sono riportati i parametri geometrici introdotti.

Facendo l'ipotesi che K è costante su ogni cella, si indicano con K_1 e K_2 i valori assunti nelle due celle. Si osservi, allora, che l'integranda in (3.6) non

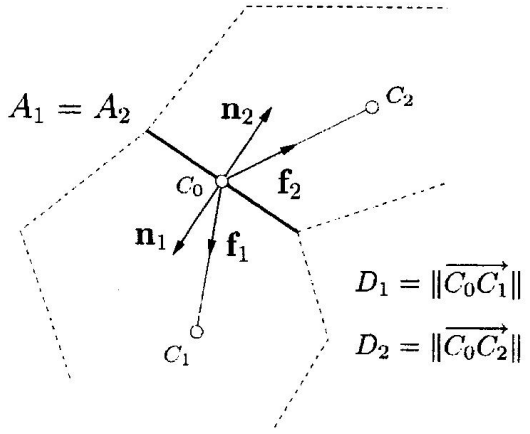


Figura 3.4: Grandezze geometriche in gioco nei VF

è altro che il gradiente di p calcolato nella direzione $\mathsf{K}\mathbf{n}$; se tale direzione è parallela ai vettori $\mathbf{f}_1$ e $\mathbf{f}_2$, si può sostituire ∇p con la differenza delle pressioni calcolate in C_1, C_2 e C_0:

$$p_1 - p(C_0) = \frac{F_{\mathsf{e}_{12}}}{A_1} \frac{D_1}{\mathbf{f}_1 \cdot \mathsf{K}_1 \mathbf{n}_1} \tag{3.7}$$

$$p(C_0) - p_2 = \frac{F_{\mathsf{e}_{12}}}{A_2} \frac{D_2}{\mathbf{f}_2 \cdot \mathsf{K}_2 \mathbf{n}_2}. \tag{3.8}$$

Ora, $F_{\mathsf{e}_{12}}$ deve essere uguale in (3.7) e (3.8), per la continuità del flusso lungo un lato; inoltre, sommando le due equazioni in modo da eliminare la $p(C_0)$ si ottiene:

$$p_1 - p_2 = \frac{F_{\mathsf{e}_{12}}}{A} \left(\frac{D_1}{\mathbf{f}_1 \cdot \mathsf{K}_1 \mathbf{n}_1} + \frac{D_2}{\mathbf{f}_2 \cdot \mathsf{K}_2 \mathbf{n}_2} \right). \tag{3.9}$$

L'equazione (3.9) è solitamente scritta come

$$F_{\mathsf{e}_{12}} = T_{12}(p_1 - p_2),$$

o, più in generale

$$F_{\mathsf{e}_{ij}} = T_{ij}(p_i - p_j),$$

dove

$$T_{ij} = A \left(\frac{D_i}{\mathbf{f}_i \cdot \mathsf{K}_i \mathbf{n}_i} + \frac{D_j}{\mathbf{f}_j \cdot \mathsf{K}_j \mathbf{n}_j} \right)^{-1} \tag{3.10}$$

è la *trasmissibilità* attraverso e_{ij}. Un altro modo per rappresentare la trasmissibilità è definire il coefficiente

$$\alpha_i = \frac{A_i \mathsf{K}_i}{D_i} \mathbf{f}_i \cdot \mathbf{n}_i \tag{3.11}$$

e, di conseguenza:

$$T_{ij} = \frac{\alpha_i \alpha_j}{\alpha_i + \alpha_j}. \tag{3.12}$$

In conclusione, si può riscrivere la (3.5) come

$$\sum_j T_{ij}(p_i - p_j) = b_i \qquad i = 1, ..., N, \tag{3.13}$$

che porta ad un sistema lineare simmetrico del tipo $\mathsf{A}\mathbf{p} = \mathbf{b}$. A tale sistema devono essere aggiunte le condizioni al bordo.

Osservazione 3.2. Si noti che l'approssimazione del primo ordine fatta in (3.9) è valida solo se i baricentri sono collegati da linee parallele a $\mathsf{K}\mathbf{n}$: una griglia con questa proprietà viene detta K-ortogonale. Se questo non si verifica il metodo non converge alla soluzione esatta ed è preferibile ricorrere ad altri metodi di discretizzazione.

Nel caso in cui il mezzo poroso presenti fratture, come precedentemente anticipato, esse non vengono modellizzate nella loro dimensione originale, ma ne viene trascurato lo spessore, come raffigurato in Figura 3.5.

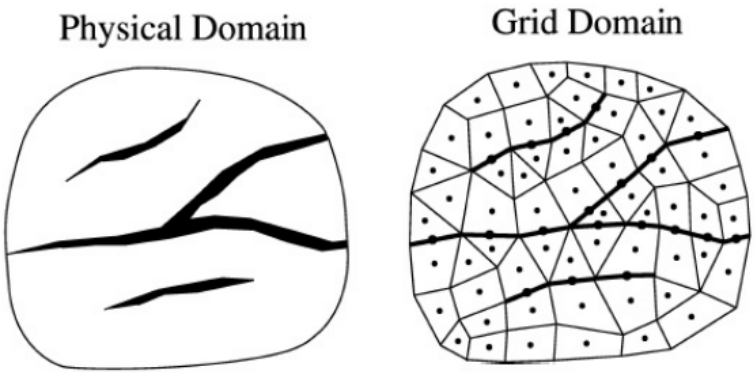


Figura 3.5: Esempio 2D di un mezzo poroso fratturato. Lo spessore delle fratture non viene riportato nella mesh: esse sono oggetti 1D composti da facce a cui è associato un valore di pressione [32].

Nell'approccio numerico, in fase di discretizzazione delle equazioni, le facce che rappresentano fratture si considerano come celle e quindi si associa loro un grado di libertà di pressione nel baricentro. In questo modo il flusso scambiato nel mezzo poroso fratturato può essere di tre diversi tipi:

- tra due celle di matrice solida,
- tra due celle di frattura,

- tra una cella di matrice solida e una cella di frattura.

Da ognuno di questi casi deriva un contributo di trasmissibilità, ottenuto discretizzando il corrispondente modello descritto in precedenza; occorre, a tal proposito fare alcune precisazioni. Infatti, a causa della formulazione primale dei Volumi Finiti, le velocità non sono incognite del problema ma possono essere calcolate solo a posteriori, una volta nota la pressione. Pertanto, bisogna sostituire il termine sorgente legato a $\mathbf{u}_1$, $\mathbf{u}_2$ in (3.2) con una differenza di pressioni moltiplicata per un'opportuna trasmissibilità, che si può ricavare sommando opportunamente le (3.3). Così facendo, per ogni cella di frattura, si calcola un contributo di trasmissibilità rispetto alle celle di frattura vicine e un contributo di trasmissibilità rispetto alla matrice solida circostante.

Il flusso scambiato tra le celle vere e proprie e quelle di frattura si ottiene applicando le formule (3.11)-(3.12). Nel caso specifico, si ha che $\mathbf{f}_i \cdot \mathbf{n}_i = 1$, dove i è l'indice relativo alla frattura. Il fatto di considerare solo implicitamente lo spessore delle celle di frattura, richiede una correzione ai volumi delle celle confinanti: infatti l'introduzione di un volume fittizio porta ad un volume totale che non è quello corretto e, per fare in modo che esso si conservi, è necessario togliere ai volumi di controllo vicini quanto si è aggiunto.

Diverso è il caso del contributo tra due facce di frattura, l'approccio comunemente usato è inserire un piccolo volume di controllo fittizio tra esse (Figura 3.6), allo scopo di calcolarne il flusso reciproco e poter considerare fratture con spessore diverso. In questo modo si calcolano i contributi T_{10} e T_{02} tra le

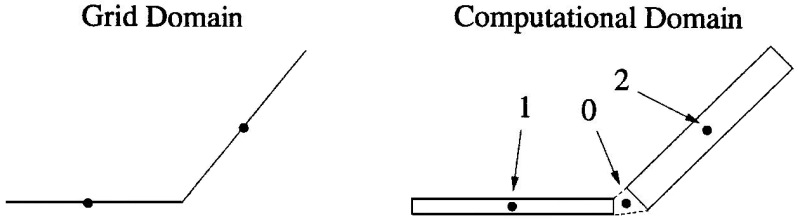


Figura 3.6: Connessione tra due facce di frattura.

fratture e la cella fittizia e la trasmissibilità totale sarà la loro media armonica:

$$T_{12} = \frac{T_{10}T_{02}}{T_{10} + T_{02}}.$$

Tuttavia, questa tecnica porta all'introduzione di celle fittizie molto piccole rispetto alle altre, con conseguenti problemi di mal condizionamento della matrice. Per questo motivo si fanno le seguenti ragionevoli semplificazioni: si

assume che $D_1 \gg D_0$ e $\mathsf{K}_1 \simeq \mathsf{K}_0$ e quindi $\alpha_1 \ll \alpha_0$ e $T_{10} \simeq \alpha_1$; lo stesso discorso vale per $T_{02} \simeq \alpha_2$ e, in conclusione:

$$T_{12} \simeq \frac{\alpha_1 \alpha_2}{\alpha_1 + \alpha_2} \qquad \text{dove} \quad \alpha_i = \frac{A_i \mathsf{K}_i}{D_i}.$$

Resta da considerare il caso in cui più di due fratture si intersechino nello stesso punto. In questo caso le formule (3.11)-(3.12) perdono di significato e bisogna apportare ad esse un'opportuna correzione. L'idea è la stessa del caso precedente, si inserisce una cella fittizia nel punto di intersezione e si calcolano i contributi di trasmissibilità tra essa e le fratture vicine; per esempio nel caso a tre rami si avrà $T_{10} \simeq \alpha_1$, $T_{20} \simeq \alpha_2$ e $T_{30} \simeq \alpha_3$, ma in questo caso non è più possibile rimuovere la cella fittizia con una semplice media armonica. A tal proposito, per eliminare il nodo introdotto, è conveniente fare un parallelismo con l'elettrotecnica e, il particolare, applicare la trasformazione *stella-triangolo* (si veda [32]), come riportato in Figura 3.7.

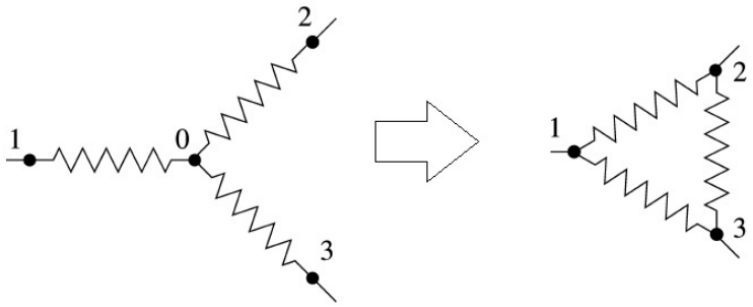


Figura 3.7: Trasformazione stella-triangolo

Allora la trasmissibilità tra due fratture i e j in un'intersezione di n fratture si può esprimere come:

$$T_{ij} = \frac{\alpha_i \alpha_j}{\sum_{k=1}^{n} \alpha_k}, \qquad \text{con } i, j = 1 : n. \tag{3.14}$$

In ultimo luogo si illustra in che modo si impongono le condizioni al bordo. Nel caso al bordo si imponga il flusso (condizione di Neumann), si avrà un'equazione del tipo $\nabla p \cdot \mathbf{n} = g^N$, allora si deve aggiungere al termine noto l'integrale di g^N sulla faccia di bordo, approssimato come la lunghezza della faccia moltiplicata per un'opportuna interpolazione di g^N. Più complicato è il caso in cui si imponga la pressione (condizione di Dirichlet): il metodo più comune consiste nel creare una cella fittizia fuori dal dominio e imporre il valore di pressione dato dalla condizione di Dirichlet. Il flusso tra le due celle viene quindi calcolato grazie alle formule (3.11)-(3.12).

3.4 Differenze Finite Mimetiche

La discretizzazione alle Differenze Finite Mimetiche si basa sulla formulazione mista del problema di Darcy; il vantaggio principale di tale formulazione risiede nel fatto che la velocità $\mathbf{u}$ è calcolata direttamente e non occorre ricostruirla a partire dalla pressione discreta, evitando tutti i problemi di conservazione della massa che ne potrebbero derivare.

Modello per la matrice solida

La tecnica di discretizzazione che si applica al modello (3.1) è costituita dalle Differenze Finite Mimetiche, descritte dettagliatamente nel Capitolo 2. Il sistema lineare che ne deriva è della forma:

$$\begin{bmatrix} \mathsf{M} & \mathsf{B}^T \\ \mathsf{B} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} g \\ -f \end{bmatrix},$$

dove $\mathbf{u}$ e p sono i vettori delle incognite di velocità e pressione.

Nel caso in cui il mezzo poroso sia fratturato, la strategia che si segue è creare una griglia conforme alle fratture; nella pratica, la frattura è rappresentata dalle interfacce fra le celle vicine. Si noti che in fase di generazione della griglia i metodi mimetici portano un enorme vantaggio, infatti non costituisce un problema il fatto che per seguire la frattura si vengano a creare celle di forme particolari, molto anisotrope ed eventualmente non convesse. Essendo le fratture rappresentate dalle facce bisogna apportare subito delle modifiche alla discretizzazione "classica"; infatti, il fatto che nella matrice solida siano presenti delle discontinuità, non garantisce più la conservazione del flusso lungo le facce: intuitivamente, quando il fluido incontra la frattura, sia nel caso funga da barriera che da condotto, subisce delle variazioni nel suo moto. La conseguenza è che data una faccia di frattura f, condivisa dagli elementi P_1 e P_2, in generale vale:

$$u^{\mathsf{f}}_{\mathsf{P}_1} + u^{\mathsf{f}}_{\mathsf{P}_2} \neq 0. \tag{3.15}$$

Pertanto, non ha più senso associare un unico grado di libertà alle facce di frattura, ma si rende necessario uno sdoppiamento dei gradi di libertà. Il vettore di incognite $\mathbf{u}$ viene quindi sostituito da un vettore $\widetilde{\mathbf{u}}$, che avrà dimensioni pari al numero delle facce più il numero delle facce di frattura. Con lo stesso criterio, si devono modificare anche la matrici derivanti dalla discretizzazione mimetica: $\widetilde{\mathsf{M}}$ in sostituzione di M, che continua ad essere simmetrica e definita positiva e $\widetilde{\mathsf{B}}$ in sostituzione di B; inoltre, anche il termine noto derivante dalle condizioni al bordo subisce lo sdoppiamento dei gradi di libertà sulle facce di

frattura. Si arriva dunque al seguente sistema lineare esteso:

$$\begin{bmatrix} \widetilde{\mathsf{M}} & \widetilde{\mathsf{B}}^T \\ \widetilde{\mathsf{B}} & 0 \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{u}} \\ p \end{bmatrix} = \begin{bmatrix} \widetilde{g} \\ -f \end{bmatrix}. \tag{3.16}$$

Si noti che non imporre nessuna condizione nelle facce di frattura, corrisponde ad imporre pressione nulla su tali facce; perciò, risolvendo il sistema (3.16) si nota un profilo di pressione che si annulla nelle fratture. In seguito, per alleggerire la notazione, si ometterà la *tilde* $\widetilde{(\cdot)}$ nell'indicare le matrici e i vettori dei gradi di libertà.

Modello per le fratture

Nel presente lavoro si è considerato unicamente il caso 2D, perciò nello scegliere il metodo con cui discretizzare (3.2) si è tenuto conto che la frattura è un oggetto unidimensionale e non è necessaria una tecnica troppo sofisticata per ottenere una buona soluzione discreta. Inoltre, nel caso 1D, tutti gli operatori differenziali si riducono a semplici derivate prime, fatte lungo la direzione tangenziale alla frattura. Tenendo conto di tutti questi aspetti, si è scelto di utilizzare i Volumi Finiti descritti nella sezione precedente, più facili di implementare rispetto ai metodi mimetici, ma comunque garanti di buoni risultati. Inoltre la velocità del fluido nelle facce di frattura può essere calcolata risolvendo il modello per la matrice solida nelle celle vicine, di conseguenza l'uso di una formulazione primale non è penalizzante in questo senso.

Eliminando $\mathbf{u}_\Gamma$ in (3.2) tramite sostituzione si ottiene la seguente equazione:

$$-l_\Gamma \ \mathrm{div}_\Gamma \left(\mathsf{K}_{\Gamma,t} \nabla_\Gamma p_\Gamma\right) = q_\Gamma + (\mathbf{u}_1 \cdot \mathbf{n} - \mathbf{u}_2 \cdot \mathbf{n}) \qquad \text{su } \Gamma,$$

che nel caso unidimensionale, supponendo che non ci siano termini sorgenti nella frattura, diventa:

$$-l_\Gamma \ \frac{d}{ds}\left(\mathsf{K}_{\Gamma,t} \frac{dp_\Gamma}{ds}\right) = (\mathbf{u}_1 \cdot \mathbf{n} - \mathbf{u}_2 \cdot \mathbf{n}) \qquad \text{su } \Gamma, \tag{3.17}$$

dove s è la direzione tangenziale alla frattura.

La griglia che è stata generata è conforme alla frattura, quindi quest'ultima viene divisa in tante facce l_i, in cui sono definite una pressione e due velocità (per lo splitting dei gradi di libertà descritto precedentemente). Integrando la (3.17) sui volumi, che in questo caso si riducono a segmenti, si ha:

$$\int_{l_i} -l_\Gamma \ \frac{d}{ds}\left(\mathsf{K}_{\Gamma,t} \frac{dp_\Gamma}{ds}\right) d\Gamma = \int_{l_i} (\mathbf{u}_1 \cdot \mathbf{n} - \mathbf{u}_2 \cdot \mathbf{n}) \ d\Gamma, \qquad i = 1 : N_\Gamma.$$

Ora, applicando il teorema della divergenza, che nel caso 1D è la formula di integrazione per parti, e ricordando che il salto di velocità è costante lungo le facce, si ottiene:

$$-l_\Gamma \left(\mathsf{K}_{\Gamma,t} \frac{dp_\Gamma}{d\mathbf{n}_\Gamma} \right) \Bigg|_{l_i} = (\mathbf{u}_1 \cdot \mathbf{n} - \mathbf{u}_2 \cdot \mathbf{n}) |l_i|, \qquad i = 1 : N_\Gamma, \tag{3.18}$$

dove $|l_i|$ è la lunghezza della faccia.

Il flusso a sinistra dell'uguale in (3.18) viene approssimato con una matrice di trasmissibilità, secondo le (3.11)-(3.12); tuttavia, essendo gli operatori differenziali definiti in direzione tangenziale alla frattura, a differenza del caso di discretizzazione ai Volumi Finiti ovunque, non va considerato il contributo tra le facce di frattura e le celle della matrice solida, che arriverà dalle condizioni di accoppiamento.

La formulazione matriciale derivante dalla discretizzazione ai Volumi Finiti è, pertanto:

$$\mathsf{C}\mathbf{u} + \mathsf{T}\pi = 0; \tag{3.19}$$

infatti nel modello completo, le velocità $\mathbf{u}_1$, $\mathbf{u}_2$ sulle facce sono delle incognite. In questo caso C è una matrice di adiacenza che associa $\mathbf{u}_1 \cdot \mathbf{n}$ e $\mathbf{u}_2 \cdot \mathbf{n}$ ad una specifica faccia di frattura, tale matrice è riscalata per la lunghezza della faccia. π è il vettore contenente tutte le pressioni definite nelle facce di frattura.

Le condizioni al bordo che tipicamente si impongono nel modello sono di tipo Neumann omogeneo, questo è suggerito dal senso fisico di tali condizioni: imporre $\nabla p_\Gamma \cdot \mathbf{n} = 0$ ai bordi della frattura significa imporre flusso nullo agli estremi. Inoltre, essendo le condizioni di Neumann imposte in modo essenziale nello spazio funzionale, implementativamente non occorre apportare alcuna modifica. Si noti che per fratture totalmente immerse nel dominio non è necessaria nessuna condizione al bordo.

Osservazione 3.3. Si noti come lo sdoppiamento delle velocità sulle facce di frattura sia fondamentale per l'assemblaggio di C, in caso contrario il grado di libertà sarebbe unico e non si potrebbero rappresentare i flussi netti non nulli attraverso la frattura.

Condizioni di accoppiamento

Una volta definite le condizioni (3.3), bisogna implementarle in modo tale che accoppino i problemi (3.16) e (3.19). La scelta che è stata fatta consiste nell'imporre in forma forte le (3.3) nel modello per la matrice solida. Questo significa che le righe corrispondenti alle facce di frattura (che per lo splitting dei gradi di libertà sono doppie) vengono cancellate e sostituite dalle corrispettive condizioni all'interfaccia. Tale scelta comporta una modifica alle matrici

M e B^T nel sistema lineare (3.16) e l'aggiunta di una nuova matrice $\hat{\mathsf{C}}^T$. Quest'ultima è simile come pattern alla trasposta di C ma come si può notare dalle equazioni, i coefficienti delle matrici sono diversi. Il sistema lineare che ne deriva è della forma:

$$\begin{bmatrix} \mathsf{M} & \mathsf{B}^T & \hat{\mathsf{C}}^T \\ \mathsf{B} & 0 & 0 \\ \mathsf{C} & 0 & \mathsf{T} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \\ \pi \end{bmatrix} = \begin{bmatrix} g \\ -f \\ 0 \end{bmatrix}, \tag{3.20}$$

dove si ricorda che per alleggerire la notazione è stata omessa la $\tilde{\cdot}$.

E' importante notare che l'implementazione delle condizioni di accoppiamento modifica la struttura della matrice M, che perde la sua buona proprietà di simmetria, con tutti gli svantaggi che ne derivano in fase di risoluzione del sistema lineare; inoltre le matrici B e B^T non sono più una la trasposta dell'altra. Un'alternativa, non implementata in questo lavoro, è l'imposizione in forma debole delle condizioni, si veda per esempio [25].

Come nel caso di mezzo poroso senza fratture, le strade per risolvere (3.20) sono due: assemblare una matrice totale e risolvere il sistema monolitico o, tramite sostituzione, risolvere le equazioni per le singole incognite:

$$\begin{cases} \mathsf{M}\mathbf{u} + \mathsf{B}^T p + \mathsf{C}^T \pi = g \\ \mathsf{B}\mathbf{u} = -f \\ \mathsf{C}\mathbf{u} + \mathsf{T}\pi = 0 \ . \end{cases}$$

L'approccio preferibile è il primo, infatti risolvere il sistema per le singole matrici richiede molte più operazioni, anche se con matrici di dimensione ridotta, e quindi tempi computazionali più elevati. Nel caso si risolva il sistema completo, l'unica operazione richiesta è la fattorizzazione di una matrice di dimensioni superiori all'altro caso, ma comunque computazionalmente vantaggiosa per i problemi considerati, che essendo 2D hanno dimensioni non troppo elevate.

3.5 Risultati numerici

In questa sezione vengono riportati i principali risultati numerici relativi al modello precedentemente ricavato, ottenuti sfruttando la libreria `C++` che verrà descritta nel prossimo capitolo. La libreria, dopo aver generato una griglia conforme alla frattura, assembla le matrici del sistema lineare (3.20) e, in ultimo luogo, risolve tale sistema.

Come precedentemente riportato, la scelta che si compie in fase di discretizzazione è costruire una griglia conforme alla frattura, in modo che la frattura

sia rappresentata dalle facce delle celle. Una conseguenza evidente di questa scelta è l'impossibilità di generare griglie strutturate, perciò le simulazioni sono state compiute su griglie triangolari e miste (con elementi di tre, quattro e cinque lati).

Caso test 1. Per validare il metodo numerico, il primo test da svolgere è finalizzato all'analisi dell'errore e al relativo studio di convergenza. A tal scopo si considera un dominio $\Omega = [-1,1] \times [-1,1]$ e un tensore di permeabilità pari all'identità, $\mathsf{K} = \mathsf{I}$. Nel dominio è immersa una frattura di spessore l_Γ, in modo tale che il dominio della frattura sia $\Omega_F = [-1,1] \times [-\frac{l_\Gamma}{2}, \frac{l_\Gamma}{2}]$. La frattura è isotropa, e ciò si traduce in una permeabilità del tipo:

$$\mathsf{K}_\Gamma = \begin{pmatrix} k_F & 0 \\ 0 & k_F \end{pmatrix}.$$

La soluzione esatta considerata è:

$$p_{ex} = \begin{cases} k_F \cos(x)\cosh(y) + (1-k_F)\cosh(\frac{l_\Gamma}{2})\cos(x) & \text{in } \Omega \backslash \Omega_F \\ \cos(x)\cosh(y) & \text{in } \Omega_F \ , \end{cases}$$

ottenuta imponendo la seguente forzante:

$$b(x,y) = \begin{cases} (1-k_F)\cosh(\frac{l_\Gamma}{2})\cos(x) & \text{in } \Omega \backslash \Omega_F \\ 0 & \text{in } \Omega_F \ . \end{cases}$$

Le condizioni al bordo nel modello per la matrice solida sono di Dirichlet, mentre in quello per la frattura sono di Neumann omogenee, che significa imporre flusso nullo agli estremi della frattura. Il problema che si va a risolvere è, pertanto:

$$\begin{cases} \mathbf{u} + \nabla p = 0 & \text{in } \Omega \backslash \Omega_F \\ \mathrm{div}\mathbf{u} = (1-k_F)\cosh(\frac{l_\Gamma}{2})\cos(x) & \text{in } \Omega \backslash \Omega_F \\ p = p_{ex} & \text{su } \partial(\Omega \backslash \Omega_F) \\ \mathrm{div}_\Gamma \mathbf{u}_\Gamma = (\mathbf{u}_1 \cdot \mathbf{n} - \mathbf{u}_2 \cdot \mathbf{n}) & \text{in } \Omega_F \\ \mathbf{u}_\Gamma + l_\Gamma \ k_F \nabla_\Gamma p_\Gamma = 0 & \text{in } \Omega_F \\ \nabla p_\Gamma \cdot \mathbf{n} = 0 & \text{su } \partial\Omega_F \\ \xi \mathbf{u}_1 \cdot \mathbf{n} + (1-\xi)\mathbf{u}_2 \cdot \mathbf{n} = \eta_\Gamma^{-1}(p_1 - p_\Gamma) & \text{su } \Gamma \\ (1-\xi)\mathbf{u}_1 \cdot \mathbf{n} + \xi \mathbf{u}_2 \cdot \mathbf{n} = \eta_\Gamma^{-1}(p_\Gamma - p_2) & \text{su } \Gamma \end{cases} \tag{3.21}$$

dove Γ è l'interfaccia tra la frattura e il mezzo poroso e $\xi = 0.75$. Discretizzando (3.21) con le Differenze Finite Mimetiche si arriva al sistema lineare (3.20).

Tabella 3.1: Ordini di convergenza, calcolati prendendo gli errori a due livelli di raffinamento: $\mathbf{N}_i$ e $\mathbf{N}_{i-1}$

$\mathbf{N}_i$-$\mathbf{N}_{i-1}$	$k_F = 1$	$k_F = 1000$	$k_F = 0.001$
2-1	1.7792	1.8250	2.0301
4-2	1.8305	1.8852	1.8803
8-4	1.7699	1.8294	1.6681
16-8	1.7332	1.9089	1.8383
32-16	1.7829	1.9631	2.1786

Come precedentemente riportato, la scelta che si compie in fase di discretizzazione è trascurare lo spessore della frattura, in modo che il dominio Ω_F si riduca alla sola interfaccia Γ; un esempio di griglia che si ottiene è mostrato in Figura 3.8. Nelle simulazioni si è mantenuto lo spessore della frattura costante e pari a $l_\Gamma = 0.01$ e si è fatta variare la permeabilità nella frattura. I casi considerati sono $k_F = 1$, $k_F = 1000$ e $k_F = 0.001$.

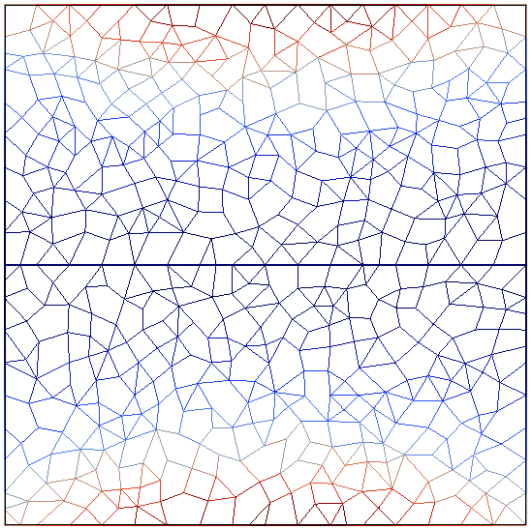


Figura 3.8: Griglia con elementi misti in presenza di frattura: la frattura coincide con le facce degli elementi

L'errore viene calcolato per la pressione p nel dominio $\Omega \backslash \Omega_F$, che rappresenta il mezzo poroso senza la frattura. L'errore viene valutato come:

$$err_p = \frac{|||p^I - p_h|||}{|||p^I|||} = \sqrt{\frac{\sum_{\mathsf{P}\in\Omega} \frac{1}{\mathsf{P}} \int_{\mathsf{P}} (p^I_{\mathsf{P}} - p_{h,\mathsf{P}})^2 dV}{\sum_{\mathsf{P}\in\Omega} \frac{1}{\mathsf{P}} \int_{\mathsf{P}} (p^I_{\mathsf{P}})^2 dV}}.$$

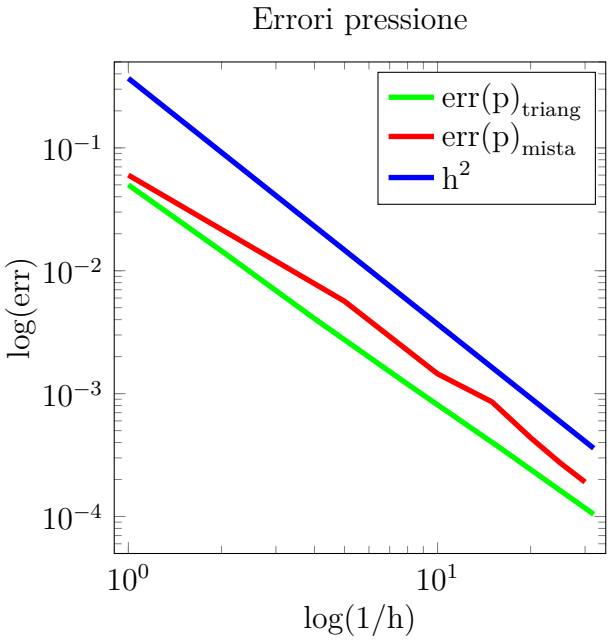


Figura 3.9: Errori di approssimazione $k_F = 1$

Tabella 3.2: Errori per la pressione a diversi valori di permeabilità (griglia triangolare)

N	$k_F = 1$	$k_F = 1000$	$k_F = 0.001$
1	0.0499765	0.185306	0.0230769
2	0.0145006	0.0485009	0.00567969
4	0.0040728	0.0128769	0.00154462
8	0.00119378	0.00360331	4.8617e-4
16	3.5905e-4	9.58182e-4	1.35967e-4
32	1.04336e-4	2.45666e-4	3.00352e-5

Come si nota dalla Tabella 3.1 e dai grafici 3.9, 3.10 e 3.11, l'ordine di convergenza della pressione è di poco inferiore a quello quadratico mostrando, di fatto, il fenomeno della superconvergenza anche nel caso con fratture.

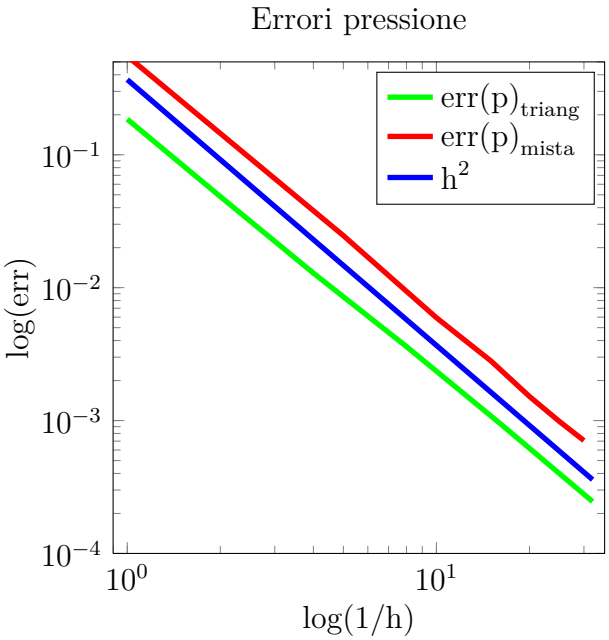


Figura 3.10: Errori di approssimazione $k_F = 1000$

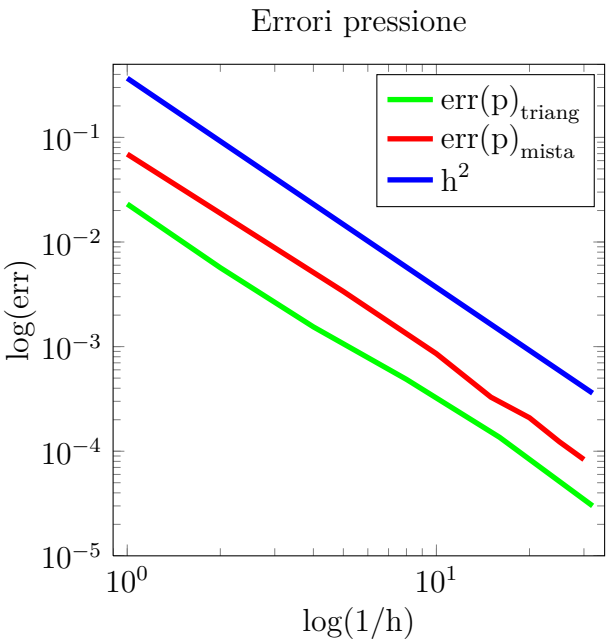


Figura 3.11: Errori di approssimazione $k_F = 0.001$

Caso test 2. L'obiettivo del secondo caso test è valutare l'effetto della permeabilità nella frattura. Sia $\Omega = [0,1]\times[0,1]$ un dominio attraversato dalla frattura $\Gamma = \{(x,y) \in \Omega : y + 2x = 1.4\}$; i bordi del dominio sono $\Gamma_D = \{0,1\} \times [0,1]$, su cui si applica un condizione del tipo $p(x,y) = y$, e $\Gamma_N = [0,1] \times \{0,1\}$, su cui si applica una condizione di Neumann omogenea. La forzante del sistema è pari a $b(x,y) = 4$ e il tensore di permeabilità nel mezzo è uguale all'identità. La frattura ha spessore $l_\Gamma = 0.01$ e permeabilità K_Γ anisotropa:

$$\mathsf{K}_\Gamma = \begin{pmatrix} \mathsf{K}_{\Gamma,n} & 0 \\ 0 & \mathsf{K}_{\Gamma,t} \end{pmatrix},$$

che varia nei casi che si considereranno. Il problema che ne deriva è:

$$\begin{cases} \mathbf{u} + \nabla p = 0 & \text{in } \Omega \\ \text{div}\mathbf{u} = 4 & \text{in } \Omega \\ p = y & \text{su } \Gamma_D \\ \mathbf{u}\cdot\mathbf{n} = 0 & \text{su } \Gamma_N \\ \text{div}_\Gamma \mathbf{u}_\Gamma = (\mathbf{u}_1\cdot\mathbf{n} - \mathbf{u}_2\cdot\mathbf{n}) + 4l_\Gamma & \text{su } \Gamma \\ \hat{\eta}\mathbf{u}_\Gamma + \nabla_\Gamma p_\Gamma = 0 & \text{su } \Gamma \\ \nabla p_\Gamma \cdot \mathbf{n} = 0 & \text{su } \partial\Gamma \\ \xi\mathbf{u}_1\cdot\mathbf{n} + (1-\xi)\mathbf{u}_2\cdot\mathbf{n} = \eta_\Gamma^{-1}(p_1 - p_\Gamma) & \text{su } \Gamma \\ (1-\xi)\mathbf{u}_1\cdot\mathbf{n} + \xi\mathbf{u}_2\cdot\mathbf{n} = \eta_\Gamma^{-1}(p_\Gamma - p_2) & \text{su } \Gamma \end{cases} \tag{3.22}$$

in cui $\xi = 0.75$. I parametri $\hat{\eta} = \frac{1}{\mathsf{K}_{\Gamma,t} l_\Gamma}$ e $\eta_\Gamma = \frac{l_\Gamma}{\mathsf{K}_{\Gamma,n}}$ sono quelli che caratterizzano il comportamento della frattura all'interno del mezzo poroso (si veda l'Osservazione (3.1)), facendo variare questi due parametri si riscontrano comportamenti diversi:

1. $\hat{\eta} = 1$ e $\eta_\Gamma = 0.01$. In questo caso la pressione è quasi continua lungo la frattura Γ, questo perché la permeabilità in direzione normale è uguale a quella del mezzo circostante e il fluido non risente della presenza della frattura (Figura 3.12).

2. $\hat{\eta} = 1$ e $\eta_\Gamma = 1$. Ora la permeabilità normale è bassa rispetto a quella tangenziale, il fluido si muove lungo la frattura e si viene a creare un salto di pressione (Figura 3.13).

3. $\hat{\eta} = 0.01$ e $\eta_\Gamma = 1$. Nell'ultimo caso la frattura è molto permeabile in direzione tangenziale e il fluido è attratto dalla frattura, conseguenza di ciò è che la pressione è quasi lineare nella frattura e le linee di flusso sono dirette verso la frattura (Figura 3.14).

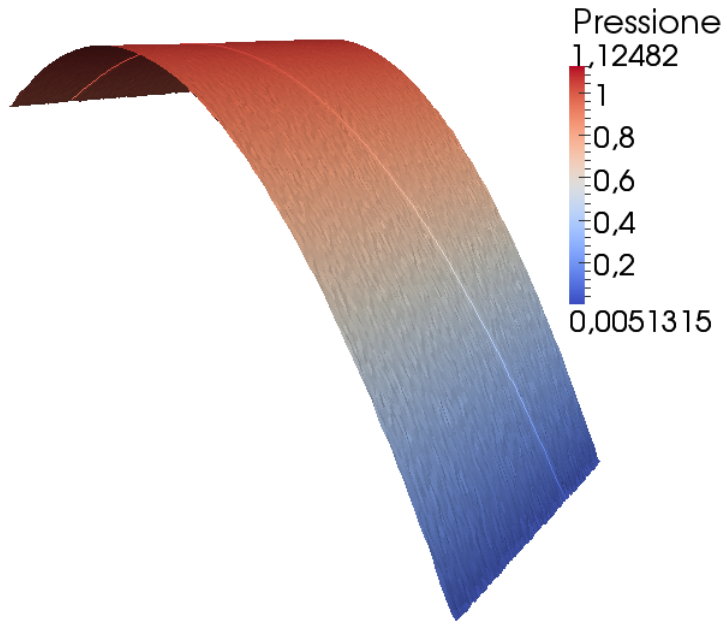


Figura 3.12: Grafico della pressione nel mezzo poroso fratturato, caso 1

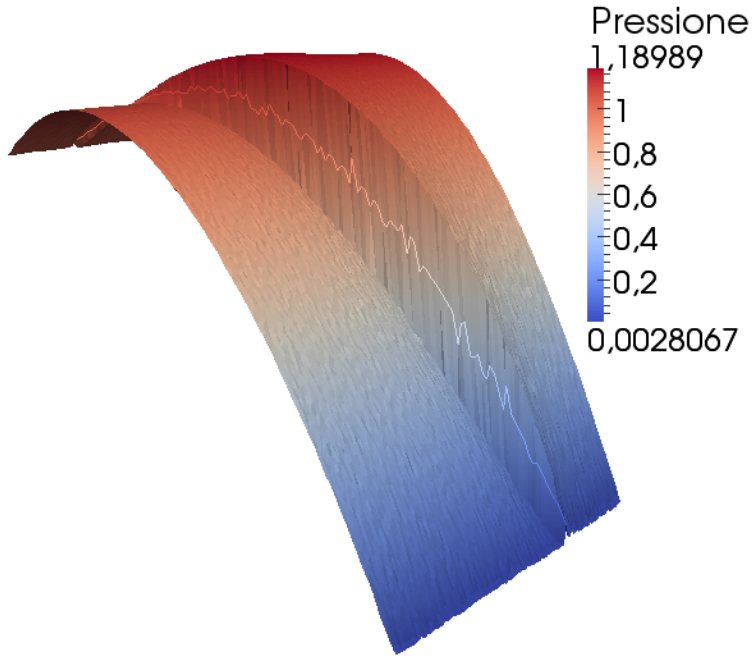


Figura 3.13: Grafico della pressione nel mezzo poroso fratturato, caso 2

Si noti che la pressione nella frattura è una funzione definita a tratti, che in fase di visualizzazione viene interpolata dal software grafico creando l'effetto di linea spezzata che si ha nelle immagini.

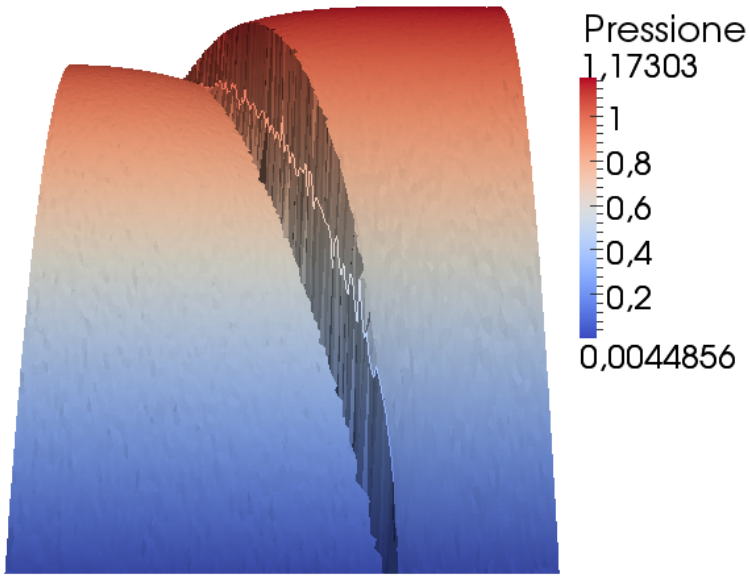


Figura 3.14: Grafico della pressione nel mezzo poroso fratturato, caso 3

Caso test 3. Nell'ultimo caso test si propone un confronto tra Differenze Finite Mimetiche (DFM) e Volumi Finiti (VF) per il calcolo della pressione in un mezzo poroso in cui siano presenti molte fratture. Si consideri quindi un dominio $\Omega = [0,1]^2$, ai bordi del quale sono applicate condizioni di Dirichlet omogenee. Il mezzo poroso viene, per semplicità, considerato isotropo e omogeneo e cioè $\mathsf{K} = \mathsf{I}$, questo per permettere il confronto con i VF, la cui implementazione prevede una permeabilità scalare. Le fratture hanno spessore $l_\Gamma = 0.01$ e permeabilità k_Γ isotropa.
La forzante che si impone al problema è:

$$b(x,y) = \begin{cases} 10 & \text{per } (x-0.1)^2 + (y-0.1)^2 \leq 0.04 \\ -10 & \text{per } (x-0.9)^2 + (y-0.9)^2 \leq 0.04 \end{cases},$$

che rappresenta una sorgente nell'angolo in basso a sinistra e un pozzo in quello in alto a destra. Si considerano tre casi:

1. Dominio senza fratture, (Figura 3.15).

2. Permeabilità delle fratture alta $k_\Gamma = 1000$, (Figura 3.16).

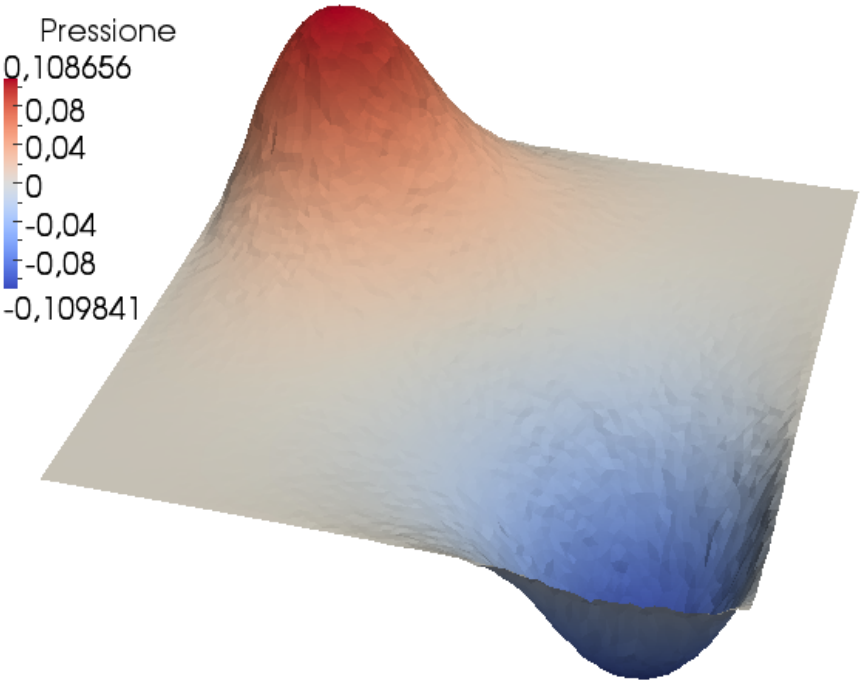
3. Permeabilità della frattura bassa $k_\Gamma = 0.001$, (Figura 3.17).

Dalle immagini si può notare come i risultati prodotti dalle DFM siano coerenti con quelli prodotti dai VF.

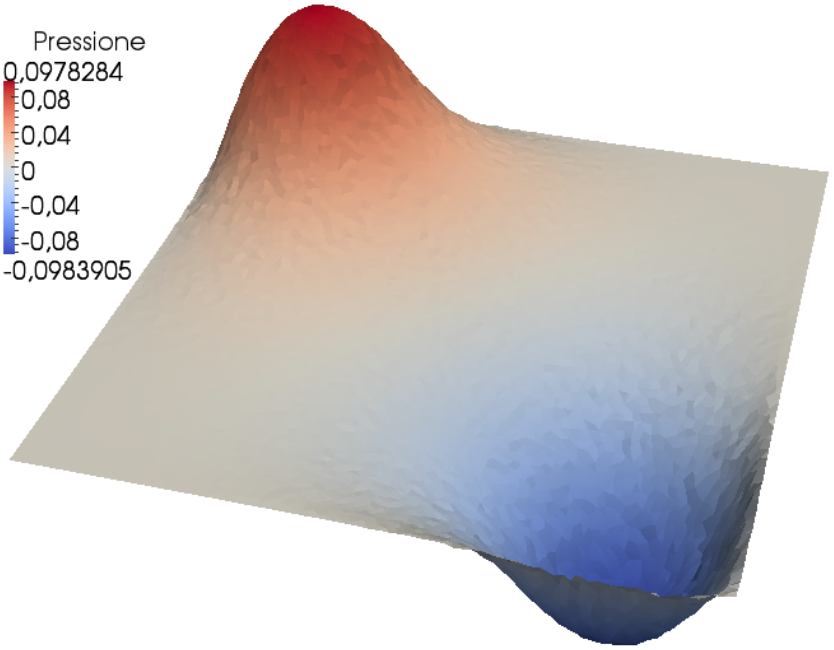
Le figure mostrano che, anche in assenza di fratture, le soluzioni derivanti dai VF hanno picchi di pressione maggiori rispetto a quelle prodotte delle DFM. Questo è dovuto al fatto che il metodo a due punti su cui si basano i VF non è accurato nel caso di griglie non cartesiane (si veda l'Osservazione 3.2); per questo motivo, se le normali delle facce non sono dirette lungo il segmento che unisce i baricentri delle celle, il flusso netto dalle facce è minore di quello previsto e quindi si ha un innalzamento del picco di pressione (per dettagli si veda [4], [18]).

Nel caso con permeabilità alta, la pressione non risente particolarmente della presenza delle fratture. L'effetto più evidente è una diminuzione del massimo e del minimo di pressione, questo perché il fluido tende, se pur di poco, a scorrere lungo le fratture, che si comportano da condotti. In questo caso, le DFM mostrano un andamento più regolare rispetto ai VF: questi ultimi tendono a mantenere costante la pressione nelle fratture, mostrando discontinuità ai bordi della frattura.

Se le fratture sono poco permeabili, il fluido non riesce ad attraversarle e questo si traduce in un salto di pressione attraverso di esse. Conseguenza di ciò è un aumento del massimo e una diminuzione del minimo della pressione, infatti non potendo scorrere attraverso le fratture, il fluido tende a ristagnare, provocando tale effetto.

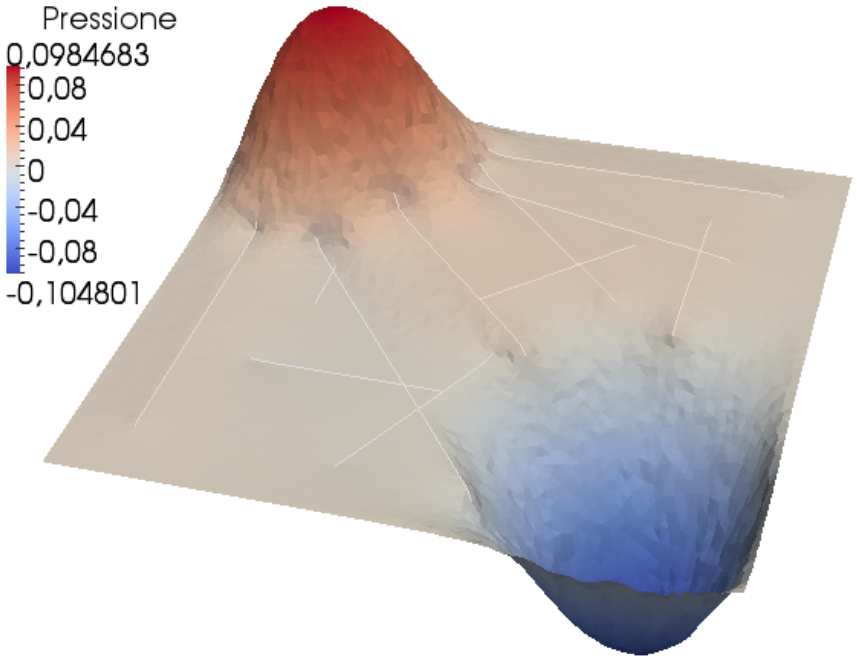


(a) VF

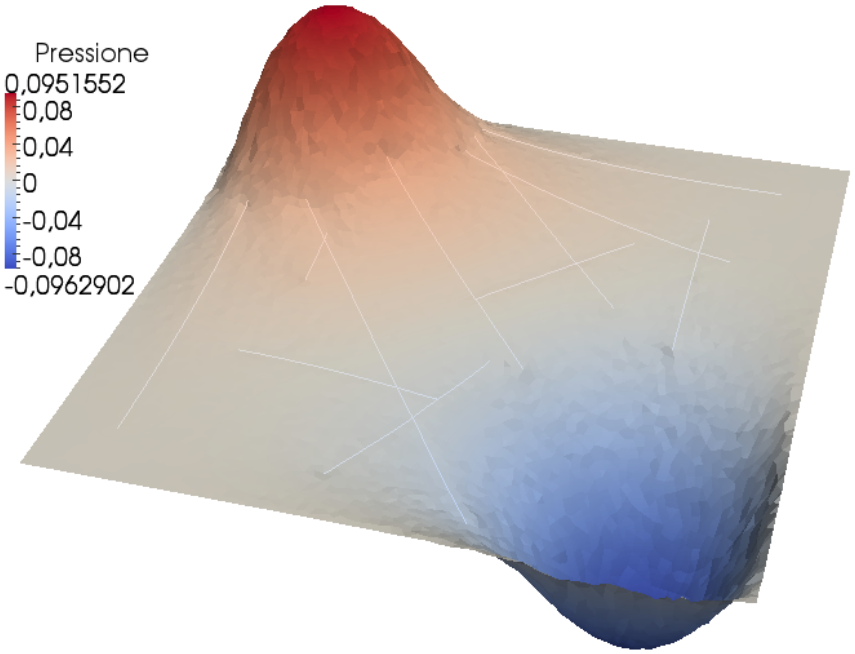


(b) DFM

Figura 3.15: Dominio senza fratture

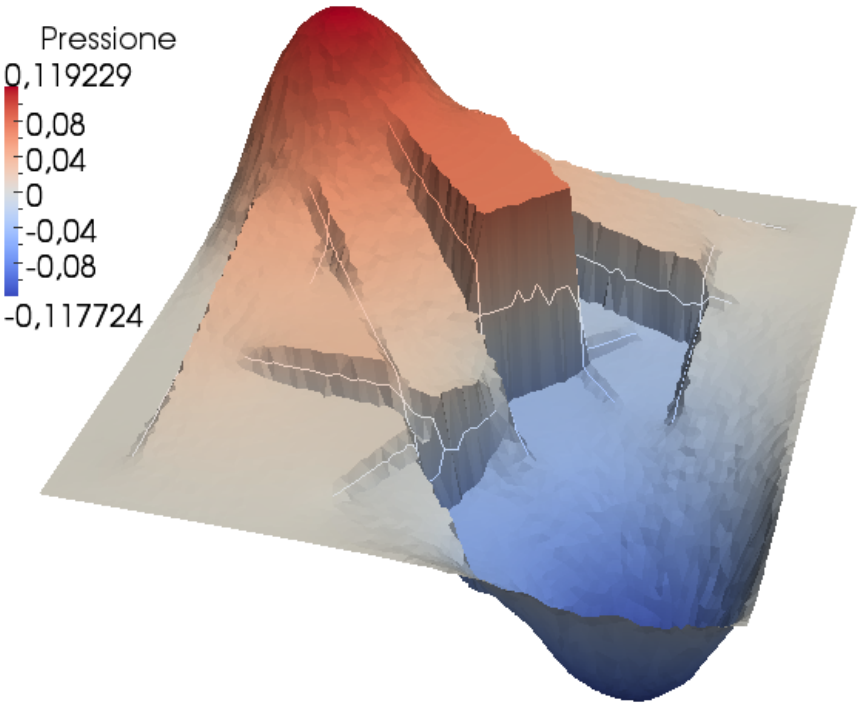


(a) VF

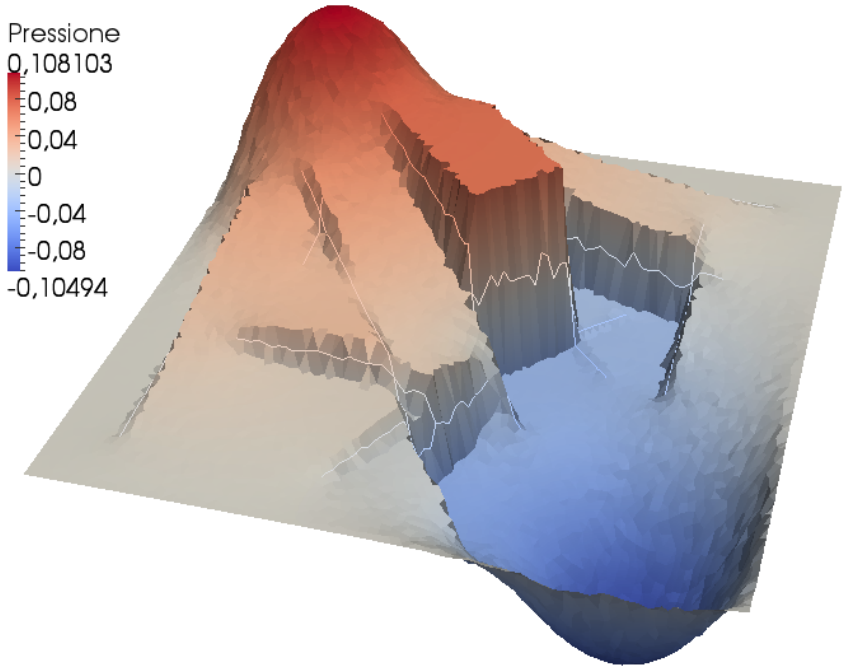


(b) DFM

Figura 3.16: Permeabilità alta



(a) VF



(b) DFM

Figura 3.17: Permeabilità bassa

Capitolo 4

Il codice C++

In questo capitolo viene presentata l'implementazione degli algoritmi per la risoluzione dei problemi descritti in precedenza. L'obiettivo è la creazione di una libreria, scritta nel linguaggio `C++`, che permetta di risolvere il flusso di un fluido in un mezzo poroso fratturato. Il codice è stato sviluppato partendo da uno già esistente, un solutore dell'equazione di Darcy tramite la tecnica dei Volumi Finiti; ad esso è stato aggiunta l'implementazione del metodo delle Differenze Finite Mimetiche.

La prima operazione che deve essere eseguita dal codice è la creazione della griglia sulla quale verrà risolto il problema differenziale. Entrambi i solutori sono stati sviluppati nel caso bidimensionale, tuttavia, per permettere l'estensione al caso tridimensionale più agevole, la griglia fa uso di un parametro template, che ne identifica la dimensione. L'implementazione della griglia prende spunto dalla libreria Mesh2D che, sfruttando la libreria esterna `CGAL` [1], si occupa della generazione di una prima griglia, che è poi adattata al problema di Darcy. Dopo la creazione della griglia, una volta noti tutti gli altri parametri fisici, vengono assemblate le matrici dello schema mimetico. Come ultimo passaggio, il sistema lineare viene risolto e si compiono le opportune operazioni per la visualizzazione della soluzione e il post-processing; queste ultime hanno richiesto l'uso di software esterni, come `Matlab` e `ParaView`.

4.1 Implementazione della griglia

La progettazione di una mesh che si adatti agli scopi preposti è una parte fondamentale e decisiva del lavoro. E' importante che il codice abbia una struttura che permetta di rendere veloce il riempimento delle matrici derivanti dalle due discretizzazioni: Volumi Finiti (VF) e Differenze Finite Mimetiche (DFM); esse hanno strategie implementative differenti, che hanno richiesto

caratteristiche della mesh diverse, andando a penalizzare, in alcuni punti, l'ottimizzazione del codice. Un esempio è la modalità di accesso alle facce degli elementi: per assemblare la matrice dei VF è sufficiente ciclare su tutte le facce, indipendentemente dall'elemento a cui appartengono, per questo motivo può bastare avere tra i membri della griglia un vettore contenente le facce; tuttavia per le DFM è indispensabile sapere a che elemento appartiene la faccia che si sta considerando, per poterne calcolare l'orientazione e, quindi, si è reso necessario poter associare le facce alle corrispondenti celle.

Tutto ciò che è relativo alla griglia è contenuto nel namespace `Geometry`, principalmente per due motivi, il primo è accorpare tutte le classi che si occupano della gestione della mesh sotto lo stesso scope e il secondo è mantenere coerenza con le librerie su cui si basano. Inoltre, nel namespace, vengono ridefiniti, tramite `typedef` i `double` (`Real`) e i `long unsigned int` (`UInt`), in questo modo se si vuole cambiare tipo in tutto il codice è sufficiente farlo in un solo punto.

Il prodotto finale è la classe template `Rigid_Mesh<T>`, che contiene tutte le caratteristiche necessarie per le operazioni successive; prima di passare in rassegna i singoli oggetti della griglia, viene presentata la sua struttura generale.

```
1  template <class T>
2  class Rigid_Mesh{
3  public:
4  //typedef typename
5  class Facet{};
6  class Cell{};
7  class Facet_ID{};
8  class Regular_Facet : public Facet_ID {};
9  class Border_Facet: public Facet_ID {};
10 class Fracture_Facet: public Facet_ID {};
11 public:
12 //Costruttori
13 //Metodi di tipo get
14 //Altri metodi
15 protected:
16 //Metodi protetti usati da costruttore o da altri metodi
17 protected:
18 std::vector<Point> M_nodes;
19 std::vector<Facet> M_facets;
20 std::vector<Cell> M_cells;
21 std::vector<Regular_Facet> M_internalFacets;
22 std::vector<Border_Facet> M_borderFacets;
23 std::vector<Fracture_Facet> M_fractureFacets;
```

```
24 std::vector<Generic_Vector> M_fracturesNormal;
25 Domain<T> M_domain;};
```

L'approccio usato nella progettazione di `Rigid_Mesh<T>` è di tipo *nested-classes*, cioè all'interno della classe principale sono state implementate altre classi che rappresentano oggetti raramente usati esternamente. In particolare, ci sono le celle (`Cell`) e le facce (`Facet`). Le `Facet` hanno una struttura generale, sia che siano interne, di bordo o di frattura. Questa scelta è stata fatta affinché la classe possa adattarsi anche ad altri problemi differenziali; per il caso di Darcy è stato scelto inoltre di creare altre tre classi, una per ogni tipologia di faccia, che derivano dalla stessa classe base. Questa strategia rende molto efficiente l'implementazione dei VF, perché, dovendo ciclare una sola volta su tutte le facce, lo si fa direttamente sui membri protetti `M_internalFacets`, `M_borderFacets` e `M_fractureFacets`, evitando, nell'assemblaggio delle matrici, di ciclare sulle facce generiche e conoscere il tipo della faccia tramite degli `if` . Per le DFM, invece, il ciclo viene fatto sulle celle e poi sulle sue facce e, quindi, si è reso necessario fornire ogni `Facet` del suo tipo (`Internal, Border, Fractured`). Questa doppia mappatura è necessaria nel momento in cui si vogliono mantenere entrambe le discretizzazioni.

Si noti, infine, che le strutture di dati utilizzate sono gli `std::vector`, questo perché essi consentono un accesso agli elementi poco costoso e sono piuttosto facili da utilizzare. Inoltre, poiché l'aggiunta o l'eliminazione di nodi, facce o celle risulta computazionalmente molto costosa, si è scelto di non consentire all'utente nessuna modifica, da qui il nome `Rigid_Mesh`.

4.1.1 La classe Dimension

Tutte le classi che riguardano la griglia hanno come capostipite la classe `Geometry::Dimension`. Il suo scopo è permettere la gestione del caso 2D e 3D nello stesso modo, infatti, nonostante i due casi abbiamo molti tratti in comune, differiscono per alcuni elementi. Come strategia si è scelto di introdurre un type-trait, in modo tale da decidere già a livello del compilatore la dimensione; infatti, i type-trait si basano sul fatto che gli `enum` vengono valutati al momento della compilazione. Nello specifico, è stata creata una classe vuota che ammette come parametro template un intero `N`, ed esistono due specializzazioni, il caso 2D (`N` = 2) e il caso 3D (`N` = 3). Si riporta, a titolo di esempio, la specializzazione per il caso bidimensionale.

```
1 //Classe vuota
2  template<int N>
3 struct Dimension {};
4 //Specializzazione 2D
```

```
template<>
struct Dimension<2>
{
typedef Geometry::Point2D Generic_Point;
typedef Geometry::Mesh2D::Edge2D Generic_Facet;
typedef Geometry::Mesh2D::Cell2D Generic_Cell;
typedef Geometry::Mesh2D Generic_Mesh;
//altri typedef
enum {dim=2};
};
```

Definite tali specializzazioni, il parametro template di `Rigid_Mesh<T>`, sarà uno tra `Dimension<2>` e `Dimension<3>` e, verrà scelto a livello di compilazione quale dei due casi chiamare.

La seconda utile implicazione della classe `Dimension` è dovuta ai `typedef` che contiene e che permettono di chiamare lo stesso costruttore per entrambe le dimensioni, in modo tale da diminuire il codice specifico da implementare e, soprattutto renderlo più efficiente.

4.1.2 La classe Facet

La struttura generale di questa classe è la seguente:

```
class Facet{
public:
//Costruttori
Facet() = delete;
//Metodi di tipo get
protected:
Geometry::Rigid_Mesh<T> * M_mesh;
UInt M_Id;
std::vector<UInt> M_Vertexes_Ids;
std::vector<UInt> M_separatedCells_Ids;
Real M_size;
Generic_Point M_center;
Generic_Vector M_UnsignedNormal;
F_Type M_fType;
protected:
//Metodi di tipo protetto
};
```

Per prima cosa si osservi che il costruttore vuoto è stato eliminato; questa scelta viene ripetuta in tutte le altre classi, in modo non dover fornire loro di metodi che possano cambiare i membri protected dall'esterno. Il costruttore

di `Geometry::Facet`, che non è qui riportato, fa uso degli oggetti della libreria Mesh2D, mentre i metodi di tipo *get* permettono all'utente di vedere le variabili protette.

La parte principale è rappresentata proprio dai membri protected della classe: alcuni forniscono informazioni sulla topologia della griglia (i vertici, le celle condivise dalla faccia), altri contengono parametri che saranno utilizzati in fase di assemblaggio delle matrici (la normale, il baricentro), altri ancora identificano la faccia (l'indice id, l'oggetto `F_Type`, ossia una `std::tuple` che ha come primo campo il tipo di faccia e come secondo il relativo id).

Ci sono, infine, i metodi di tipo protetto, che sono metodi chiamati dal costruttore per calcolare il centroide e la normale alla faccia; si noti che la normale contenuta in `Geometry::Facet` non ha segno, infatti per stabilire se un vettore è entrante o uscente, bisogna associare una cella alla faccia. In questo caso, le due discretizzazioni hanno procedure diverse: i VF non tengono conto del segno della normale, infatti il prodotto scalare in (3.11) è sempre positivo; le DFM, invece, tengono conto del segno della normale e, dove serve, la normale con segno viene calcolata con un metodo pubblico della classe `Geometry::Cell`. A tal proposito, gioca un ruolo importante il metodo protetto `changeVertexesOrder()`, esso viene chiamato dal costruttore di `Rigid_Mesh<T>`, nel momento in cui vengono associate le facce alle rispettive celle. Nello svolgere questa operazione è fondamentale che l'orientazione della faccia corrisponda a quella della cella e, nel caso in cui questo non si verifichi, tale funzione viene chiamata per riordinare opportunamente i vertici.

4.1.3 La classe Cell

La struttura della classe `Cell` è molto simile a quella della `Facet`:

```
1  class Cell{
2  public:
3  //Costruttori
4  Cell() = delete;
5  //Metodi di tipo get
6  //Metodi pubblici
7  void showMe (std::ostream& out=std::cout) const;
8  bool hasNeighborsThroughFacet
9          (const UInt& facet_Id, const UInt& idNeighbor);
10 Generic_Vector outerNormalToEdge
11         (const UInt& idv1, const UInt& idv2 );
12
13 protected:
14 Geometry::Rigid_Mesh<T> * M_mesh;
```

```
15 UInt M_Id;
16 std::vector<UInt> M_Vertexes_Ids;
17 std::vector<UInt> M_Neighbors_Ids;
18 Generic_Point M_centroid;
19 Real M_volume;
20 std::vector<Facet> M_CFacets;
21
22 friend class Rigid_Mesh<T>;
23
24 protected:
25 //Metodi di tipo protetto
26 };
```

Anche per questa classe, valgono le stesse considerazioni sui costruttori e sui metodi *get* fatte per `Geometry::Facet`. Volendo evitare la presenza di metodi pubblici che possano modificare i membri protetti della classe, si è deciso di rendere `friend` la classe `Rigid_Mesh<T>`, in modo che essa possa accedere in scrittura ai membri protected di `Cell`.

Tra i metodi pubblici, oltre a una funzione che stampa informazioni sulla cella e un metodo che consente di determinare se la cella ha un vicino di indice `idNeighbor` attraverso la faccia di indice `facet_Id`, c'è il metodo `outerNormalToEdge` ed è quello a cui si accennava in precedenza. Tale funzione è largamente usata nel caso di DFM, dato che le matrici N_P sono costruite a partire dalla normali uscenti dalle facce dell'elemento P; essa prende in ingresso gli indici della faccia e ne calcola la normale uscente, per questo motivo è fondamentale che l'orientazione della faccia quella della cella siano coerenti.

4.1.4 La classe Facet Id e le sue derivate

Come è noto, al fine di rendere la struttura utilizzabile in altre applicazioni, la classe `Geometry::Facet` non è differenziata a seconda delle possibili tipologie di faccia. Tuttavia, nel caso in esame, le facce hanno implicazioni diverse a seconda che rappresentino una frattura, un lato del bordo del dominio o una faccia interna normale. Una possibile soluzione a tal richiesta è raggruppare gli indici di una diversa tipologia di faccia in un vettore distinto; in realtà per permettere un semplice accesso ad altre informazioni che risultano utili, è stata creata la classe `Facet_ID` che, oltre ad un indice di faccia, nel caso di facce fratturate e di bordo, contiene tali informazioni.

```
1 class Facet_ID{
2 public:
3
4 Facet_ID (const UInt facet_id,
```

```
5            Geometry::Rigid_Mesh<T> *const mesh);
6  Facet_ID () = delete;
7
8  //metodi di tipo get
9
10 protected:
11 UInt Facet_Id;
12 Rigid_Mesh<T>* M_mesh;
13 };
```

I metodi di tipo *get* danno l'accesso diretto alla `Facet` e ai suoi metodi, tuttavia, pur essendo una scelta comoda, risulta non essere molto efficiente. Da questa classe derivano le tre classi che caratterizzano una diversa tipologia di faccia:

- `Regular_Facet`. Serve a contenere gli indici delle facce che non sono né di bordo né di frattura e, di fatto, è uguale alla classe base.
- `Border_Facet`. Questa classe derivata presenta qualche modifica rispetto alla classe base: infatti essa contiene l'indice del bordo del dominio al quale la faccia appartiene, tale indice viene recuperato tramite la `Rigid_Mesh<T>` a cui punta l'oggetto della classe.
- `Fracture_Facet`. La classe che rappresenta le facce di frattura è, fra quelle derivate, la più complessa. Infatti, contiene al suo interno tutte le informazioni della frattura, oltre che un Id che identifica la `Facet` di frattura. Per questo motivo è utile riportare la sua struttura generale:

```
1  class Fracture_Facet: public Facet_ID {
2  public:
3  //Costruttore
4  //Costruttori di tipo copy constructor
5  //Metodi di tipo get
6
7  friend class Rigid_Mesh<T>;
8
9  private:
10 UInt Cells_number;
11 UInt M_Id;
12 Real M_permeability_tang;
13 Real M_permeability_norm;
14 Real M_aperture;
15 std::vector<UInt> Fracture_Ids;
16 std::map<Fracture_Juncture, std::vector<UInt> >
```

```
17                      Fracture_Neighbors;
18 };
```

Alle facce di frattura, si associa un ulteriore Id, che permette di identificarle come celle. Giocano un ruolo molto importante i membri che rappresentano i parametri fisici della frattura, il suo spessore (o apertura) e la sua permeabilità: quest'ultima è descritta dalle sue componenti tangenziale e normale, per poter considerare anche il caso di fratture anisotrope. Va sottolineato che tali parametri vanno settati con opportuni metodi pubblici una volta definite le fratture.

La mappa `Fracture_Neighbors` indica quali sono le facce di frattura confinanti con l'oggetto a cui appartiene: essa associa un vettore di Id ad una `Fracture_Juncture`. Quest'ultima rappresenta, appunto, la giunzione tra due facce di frattura ed è un punto in 2D e un segmento in 3D. La mappa permette di assemblare in maniera rapida il contributo frattura-frattura nella matrice dei VF, grazie alla formula (3.14).

Per come è stato impostato il lavoro, non è previsto che una faccia possa essere sia di bordo che di frattura, ma la struttura del codice è facilmente generalizzabile in modo da poter risolvere questo problema.

4.1.5 La classe Rigid_Mesh

I metodi della classe `Rigid_Mesh<T>` si possono dividere in:

- Costruttori
- Metodi di tipo *get*
- Altri metodi pubblici
- Metodi protetti

Per quanto riguarda i costruttori, vale lo stesso discorso fatto in precedenza: il costruttore senza argomenti è stato eliminato. Esiste, oltre a quello di copia, un costruttore che prende in ingresso un oggetto della Mesh2D (o della Mesh3D) e un dominio di tipo `Geometry::Domain<T>`, utilizzato per fornire le informazioni necessarie alle facce di bordo.

```
Rigid_Mesh (Generic_Mesh & generic_mesh, Domain<T> & domain);
```

I metodi di tipo *get* sono utilizzati quando si vuole avere accesso in lettura ai membri protetti della classe, per esempio se si vogliono informazioni sui nodi, sulle facce, sulle celle, sulle normali delle fratture. A questi metodi se ne aggiungono altri che permettono di conoscere ulteriori parametri della

griglia, come la dimensione della faccia più grande, quella della più piccola e la dimensione media.

Molto importanti sono i metodi che permettono la visualizzazione dei risultati ottenuti, in particolare, essi sono due: il primo salva la griglia in formato `".vtk"`, mentre il secondo associa la soluzione discreta ad ogni cella della griglia.

```
bool exportVtk(const std::string & fileName) const;

bool appendSolutionToVtk (Darcy::Vector& sol,
  const std::string & fileName,
  const std::string & label = "pressure",
  const std::string & solType = "CELL") const;
```

E' importante, al fine di ottenere una corretta visualizzazione, chiamare i due metodi nell'ordine in cui sono stati riportati. Per permettere generalità e adattare la classe a diversi tipi di discretizzazione, la funzione che aggiunge al file vtk la soluzione calcolata ha, fra gli argomenti, l'oggetto `solType` che stabilisce se essa deve essere salvata per punti o, come nel caso in esame, per celle; inoltre viene associata una soluzione anche alle facce di frattura. Il prodotto finale è `fileName.vtk`, compatibile con il software `ParaView` che si occupa molto bene del plotting di griglia e soluzione.

Ci sono, infine, i metodi protetti. Essi sono chiamati dal costruttore e si differenziano nel caso 2D e 3D nel caso richiedano una diversa implementazione. I loro scopi sono molteplici: convertire gli oggetti e le informazioni delle librerie esterne nei membri protetti della `Rigid_Mesh`; associare le facce alle celle corrispondenti; calcolare la topologia della griglia; salvare le informazioni sulle singole celle in modo da poterle esportare. E' conveniente soffermarsi sul metodo `Rigid_Mesh<T>::ComputeNormal`, ossia quello che costruisce il vettore `M_fracturesNormal`. Esso, per ogni fattura del dominio, stabilisce una direzione (tipicamente perpendicolare alla frattura) e un verso di riferimento; tale vettore, pertanto, avrà dimensione pari al numero di fratture presenti nel dominio. La direzione di riferimento che viene stabilita, verrà usata nelle DFM per decidere in che modo splittare i gradi di libertà delle celle fratturate (più avanti verranno forniti maggiori dettagli).

4.1.6 Le classi Domain e BoundaryConditions

Parallelamente alla `Rigid_Mesh` vengono implementate altre due classi che rappresentano il dominio $\partial\Omega$ e le condizioni al bordo del problema differenziale che si vuole risolvere. Senza entrare troppo nei dettagli, `Domain` serve per

attribuire un indice di bordo alle `Border_Facet` ed essa contiene tanti lati di bordo quante sono le condizioni al contorno da imporre; essendo il dominio un oggetto della griglia, esso compare sotto il namespace `Geometry`. Per associare ai lati del dominio le rispettive condizioni al contorno, è stata creata `BoundaryConditions`: essa contiene una referenza ad un oggetto `Domain` ed è dotata di una `enum` che permette di distinguere diverse condizioni:

```
enum BCType{Dirichlet, Neumann};
```

L'altro oggetto indispensabile è la funzione imposta come BC al bordo, essa viene fornita tramite un *function-wrapper* che prende in ingresso un punto e ne restituisce il valore calcolato nella funzione:

```
std::function<D_Real(Generic_Point)> bc);
```

poiché questa classe non è un oggetto della geometria, ma è legata al problema differenziale che si sta risolvendo, essa fa parte del namespace `Darcy`, che racchiude anche tutte le matrici.

4.2 Assemblaggio delle matrici

In questa sezione vengono riportati i dettagli implementativi che portano all'assemblaggio delle matrici di (3.20). Si noti che, per i mezzi porosi in cui ci sono fratture, la matrici da implementare sono sei: M, B, B^T, C, C^T e T; le ultime tre, invece, non sono presenti nel caso in cui il mezzo non abbia fratture. In questo secondo caso, per come è stato pensato il codice, non è necessario fare alcuna specializzazione degli algoritmi, semplicemente essi riconoscono l'assenza di fratture e viene risolto il sistema (2.22).

4.2.1 La classe MatrixHandler

La classe `MatrixHandler` è una classe di tipo astratto, ossia una classe base che contiene al suo interno dei metodi *pure virtual*. Questo tipo di oggetto non può essere costruito, ma rappresenta una base da cui ereditare. Gli oggetti che derivano da `MatrixHandler` assemblano una matrice derivante da una qualsiasi discretizzazione di un problema differenziale generico definito su una griglia del tipo `Rigid_Mesh`, per questo motivo, essendo quest'ultima una classe template, anche `MatrixHandler` lo sarà. Prima di procedere, va notato che anche il `namespace Darcy` ha i suoi tipi ridefiniti: `D_Real` e `D_UInt`.

La struttura è la seguente:

```
enum DiscretizationType {D_Cell, D_Facet, D_Fracture};
typedef DiscretizationType DType;

template <class T>
class MatrixHandler {
public:
MatrixHandler(const Geometry::Rigid_Mesh<T> &rigid_mesh,
              DType dtype_row, DType dtype_col,
              BoundaryConditions<T> &bc);
//Metodi di tipo get
virtual void assemble()=0;
virtual void fillMatrix(SpMat &MAT)=0;

protected:
const Geometry::Rigid_Mesh<T> & M_mesh;
DType M_policy_row;
DType M_policy_col;
D_UInt M_size_rows;
D_UInt M_size_cols;
std::unique_ptr<SpMat> M_Matrix;
BoundaryConditions<T>& m_Bc;
};
```

Come si può notare, per fare in modo che tutte le matrici di (3.20) possano ereditare da `MatrixHandler`, le dimensioni della matrice non sono fissate, ma possono seguire tre policy diverse: celle (`D_Cell`), facce (`D_Facet`) o facce fratturate (`D_Fracture`); inoltre, le matrici non dovranno essere necessariamente quadrate, ma righe e colonne possono avere diversa dimensione. Va ricordato che, nel caso ci siano fratture, le facce che le rappresentano hanno due gradi di libertà per la velocità e, di conseguenza, se la matrice deve essere dimensionata con il numero delle facce, in realtà, la dimensione sarà il numero delle facce più il numero delle facce fratturate. In tabella (4.1) si trovano le dimensioni di tutte le matrici; esse non sono dei parametri template perché il dimensionamento viene fatto comunque al momento dell'esecuzione.

Questa classe (e, quindi, anche tutte quelle che da essa derivano) ammette un solo costruttore, infatti, oltre ad eliminare il costruttore vuoto, si elimina anche quello di copia, dato che tale operazione non ha senso su tutta la classe, ma solamente sulla matrice stessa. Il costruttore, come prima cosa, stabilisce le dimensioni della matrice, nel modo seguente:

```
M_size ((dtype == D_Cell)*(Cells.size() ) +
(dtype == D_Facet)*(Facets.size()+FractureFacets.size())+
```

Tabella 4.1: Dimensioni delle matrici

Matrice	Dimensione righe	Dimensione colonne
M	D_Facet	D_Facet
B	D_Cell	D_Facet
B^T	D_Facet	D_Cell
C	D_Fracture	D_Facet
C^T	D_Facet	D_Fracture
T	D_Fracture	D_Fracture

```
3 (dtype == D_Fracture)*(FractureFacets.size() ) );
```

Si rimarca che la dimensione delle righe e delle colonne può essere diversa e che, come si vede dal codice riportato, le matrici dimensionate con `D_Facet` contano due volte le eventuali facce fratturate. Una volta definite le dimensioni, viene creata una matrice vuota:

```
1 M_Matrix(new SpMat(this->M_size_rows,this->M_size_cols))
```

In questo contesto `SpMat` è una matrice sparsa della libreria `Eigen`; essa è implementata con tecniche di programmazione avanzata e, perciò, risulta essere molto efficiente. Tuttavia, per come sono costruite, tali oggetti non possono essere tenuti in una classe a meno che non occupino un preciso numero di byte o un suo multiplo. Per non rinunciare all'efficienza delle `Eigen` si è scelto di aggirare questo problema usando uno `unique_pointer`, ossia un puntatore introdotto nell'ultimo standard, che ha il grande vantaggio di liberare al momento della sua eliminazione la memoria da lui puntata.

Il riempimento delle matrici è affidato al metodo `assemble()` che, nella classe base è dichiarato virtuale, ma avrà la sua implementazione specifica nelle classi che ereditano. Inoltre, è stato dichiarato un secondo metodo virtuale: `fillMatrix(SpMat &MAT)`, esso ha lo scopo di creare la matrice completa del metodo, in modo da risolvere l'unico sistema $\mathsf{A}x = b$. Tale metodo prende in ingresso una referenza alla matrice completa e, nelle classi derivate, riempirà solo gli elementi che gli competono.

Infine, il costruttore della classe prende in ingresso anche un oggetto del tipo `BoundaryConditions`, il motivo risiede nel fatto che quasi tutte le matrici hanno bisogno di conoscere le condizioni al bordo per poter essere assemblate in modo corretto ed è conveniente che queste siano contenute nella classe base.

4.2.2 Le classi Mloc e Mglob

Le classi `Mloc` e `Mglob` forniscono l'implementazione delle matrici M_P e M, rispettivamente. La classe che costruisce la matrice locale è diversa da tutte le altre, essa infatti è definita solo sulla cella P e per questo motivo è l'unica che non eredita da `MatrixHandler`; inoltre essendo per il momento implementata solo nel caso 2D, non è necessario introdurre alcun parametro template. La struttura di `Mloc` è la seguente:

```
class Mloc_2D {
public:
Mloc_2D (const Rigid_Mesh<Dimension<2> >::Cell &cell,
         func kappa_11, func kappa_12, func kappa_22);
//Metodi di tipo get e altri
void assemble();

private:
void assemble_N();
void assemble_R();

private:
const Rigid_Mesh<Dimension<2> >::Cell M_Cell;
D_UInt M_size;
Matrix M_Matrix_M;
Matrix M_Matrix_R;
Matrix M_Matrix_N;
Matrix K_E; }
```

Pur non ereditando da `MatrixHandler`, la classe `Mloc` adotta la sua stessa strategia implementativa per quanto riguarda i costruttori: vengono eliminati quello vuoto e quello di copia e l'unico costruttore ammissibile prende in ingresso una cella `Cell` e il tensore simmetrico di permeabilità che, nel caso bidimensionale, richiede il passaggio in input di tre *lambda-function*. Tale costruttore assembla le matrici locali N_P e R_P grazie alle formule (2.15) e (2.18), tramite i metodi `assemble_N()` e `assemble_R()`, che non possono essere chiamati dall'esterno, dato che l'unica matrice che interessa è quella finale; il costruttore, inoltre, calcola il tensore permeabilità nel centroide della cella. Il metodo `assemble()` è chiamato da `Mglob`, che si occuperà del passaggio da locale a globale; la matrice che ne deriva è quella della formula (2.21).

Essendo la dimensione della matrice locale pari al numero di facce dell'elemento, non vale la pena usare le `SpMat`, ma si è scelto di usare le matrici dense delle `Eigen`, che permettono operazioni come l'inversa, la traccia o l'accesso diretto agli elementi. Questa scelta non influisce sull'efficienza del codice.

Tra i membri protetti di `Mglob`, oltre a tutte le variabili ereditate da `MatrixHandler`, ce ne sono di ulteriori, sempre non modificabili dall'esterno:

```
1 std::unique_ptr<Vector> _g;
2 func K11;
3 func K12;
4 func K22;
```

L'oggetto `_g` è uno *smart pointer*, che punta ad un vettore della libreria `Eigen` contenente il vettore delle condizioni al bordo; esso corrisponde a g nella formula (3.20); è presente, a tal proposito, un metodo che, conoscendo le `BoundaryConditions` e la `Rigid_Mesh`, costruisce il termine noto. I restanti oggetti sono delle funzioni che rappresentano il tensore di permeabilità del problema:

$$\begin{pmatrix} K11 & K12 \\ K12 & K22 \end{pmatrix}$$

esso viene passato in input alla classe `Mloc`, a cui serve per comporre la matrice del prodotto scalare locale.

I metodi `assemble()` e `fillMatrix(SpMat &MAT)` hanno il compito di assemblare la matrice globale M: il primo fornisce la singola matrice, mentre il secondo riempie una matrice più grande nelle righe e colonne giuste. Nonostante i prodotti finali siano diversi, l'implementazione alla base è identica, pertanto l'algoritmo interno è lo stesso per entrambi. Per rendere le cose più chiare, viene presentato, in primo luogo, il caso dei mezzi porosi non fratturati, pur ricordando che esso rappresenta un caso particolare dei mezzi con fratture. Supponendo, allora, che il vettore contenente le facce di frattura sia vuoto, la matrice M viene così assemblata:

1. Si cicla sulle `Cell` della griglia e per ognuna si calcola la matrice locale; esse sono matrici quadrate di piccole dimensioni, pari al numero di facce della cella. Gli indici di riga e di colonna delle M_P corrispondono agli indici locali delle facce di P, per questo ad ogni elemento della matrice locale si possono associare due facce (per i termini diagonali la faccia è la stessa).

2. Si itera su tutti gli elementi diversi da zero della matrice locale e si calcola la loro corretta posizione nella matrice globale. Tale posizione è data dagli indici globali delle facce in questione, proprio come avviene per i gradi di libertà nei metodi agli Elementi Finiti.

3. Si moltiplicano tutti gli elementi da inserire per i due `segni` corrispondenti alle facce in questione: il `segno` della faccia è positivo nel caso

l'orientazione locale corrisponda a quella globale, in caso contrario è negativo. Per stabilire se si è nel primo o nel secondo caso, è sufficiente fare un confronto sull'ordine dei vertici della faccia: si ha orientazione positiva se nella faccia locale (vale a dire la `Facet` presente nella `Cell`) si ha lo stesso ordine di vertici di quella presente in `Rigid_Mesh`. A tal proposito si ricorda che è fondamentale, quando si associano le `Facet` alle `Cell`, che esse si presentino con la stessa connettività.

Ben più complessa è la procedura da eseguire in presenza di fratture. La difficoltà nasce principalmente dal fatto che nelle facce di frattura avviene uno sdoppiamento del grado di libertà di velocità e la matrice M subisce un aumento di righe e colonne. Conseguentemente, ad ogni faccia di frattura corrispondono due righe e due colonne e, in fase di assemblaggio, bisogna decidere a che riga e colonna associare l'elemento su cui si sta ciclando. Per la convenzione che è stata adottata, le righe e colonne vengono aggiunte in fondo alla matrice, cioè detti N_F il numero totale di facce, N_f il numero di facce fratturate e N_{M} la dimensione della matrice, gli indici da $N_F + 1$ a N_{M} sono quelli associati ai gradi di libertà derivanti dallo sdoppiamento. In altre parole il vettore delle incognite risulta pari a:

$$\begin{bmatrix} u \\ \widetilde{u} \\ p \\ \pi \end{bmatrix}$$

dove $\widetilde{u}$ è il vettore delle velocità nelle facce fratturate.

Concretamente, bisogna stabilire una convenzione per associare $(\mathbf{u}_1, p_1)$ e $(\mathbf{u}_2, p_2)$ alle cella a destra e a sinistra della frattura nelle formule (3.3). La scelta che è stata fatta è la seguente: la cella chiamata 1 (e i suoi corrispondenti gradi di libertà) è quella che ha la normale di faccia concorde a quella di riferimento in `Rigid_Mesh`, presente nel vettore `M_fracturesNormal`; l'altra sarà la cella 2.

Dato che questo procedimento deve essere ripetuto più volte, per tutte le classi, si è scelto di aggiungere ai membri protetti di `Rigid_Mesh` un vettore contenente gli indici delle celle 1 (`M_CritRef`); per ottenere questa informazione, basterà accedere a tale vettore.

Una volta che tutte le M_{P} sono state inserite nella matrice globale, le righe di M corrispondenti a fratture vengono sostituite dalle equazioni di accoppiamento che sono, in questo modo, imposte in forma forte.

Continua a valere la regola dei `segni` descritta in precedenza; tuttavia in questo caso è bene fare qualche considerazione aggiuntiva. Detti s_1 e s_2 i segni

delle facce, le condizioni di accoppiamento da implementare sono:

$$\eta_\Gamma[\xi \mathbf{u}_1 s_1 s_1 - (1-\xi)\mathbf{u}_2 s_1 s_2] + p_1 s_1 - p_\Gamma s_1 = 0$$

$$\eta_\Gamma[(1-\xi)\mathbf{u}_1 s_1 s_2 - \xi \mathbf{u}_2 s_2 s_2] - p_2 s_2 + p_\Gamma s_2 = 0;$$

ma, essendo i `segni` sempre riferiti ad una stessa faccia, se uno è positivo, l'altro sarà necessariamente negativo, pertanto $s_1 s_2 = -1$ e $s_1 s_1 = s_2 s_2 = 1$ e le condizioni diventano:

$$\eta_\Gamma[\xi \mathbf{u}_1 + (1-\xi)\mathbf{u}_2] + p_1 s_1 - p_\Gamma s_1 = 0 \tag{4.1}$$

$$\eta_\Gamma[-(1-\xi)\mathbf{u}_1 - \xi \mathbf{u}_2] - p_2 s_2 + p_\Gamma s_2 = 0. \tag{4.2}$$

Quindi i segni con cui inserire gli elementi nella matrice M sono definiti a priori e non è necessario calcolarli ogni volta.

4.2.3 La classe DivOperator

La classe `DivOperator` è stata creata allo scopo di assemblare le matrici B e B^T, che sono la versione algebrica dell'operatore divergenza (1.19) e del suo operatore duale (1.21). La classe eredita da `MatrixHandler` e non è necessario aggiungere altri membri protected, si deve solo procedere all'implementazione dei metodi virtuali della classe base.

In questo caso, dovendo calcolare una matrice e la sua trasposta, i metodi `assemble()` e `fillMatrix(SpMat &MAT)` seguono una strategia leggermente diversa:

- `assemble()` calcola solamente la matrice B, tramite l'algoritmo che verrà riportato in seguito. Per ottenere la trasposta, si definisce il metodo pubblico `getMatrix_transpose()` che, in primo luogo calcola la trasposta di B, sfruttando gli efficienti metodi delle `Eigen` e dopo corregge le righe delle facce fratturate, secondo le (4.1)-(4.2). La matrice che ne risulta è l'output del metodo.

- `fillMatrix(SpMat &MAT)` riempie gli elementi di `MAT` che gli competono; le parti relative alle due matrici vengono quindi riempite contemporaneamente, tramite lo stesso algoritmo del primo caso; così facendo, in un unico ciclo si vanno a riempire due blocchi di matrice diversi, riducendo i tempi computazionali. A posteriori, le righe relative alle facce fratturate vengono corrette.

Per calcolare gli elementi della matrice B in assenza di fratture, analogamente ad M, si itera su tutte le `Cell` del dominio e per ogni `Facet` della cella, tramite la formula (1.19), si ricava l'elemento da inserire e lo si moltiplica per il `segno` opportuno che, essendo solo le colonne numerate con `D_Facet`, è soltanto uno.

Nel caso il mezzo poroso presenti delle fratture, le colonne di B aumentano di numero, per lo splitting dei gradi di libertà; allora, si usa lo stesso criterio della classe `Mglob` per decidere chi è la cella chiamata 1 e chi è quella chiamata 2 e l'implementazione delle condizioni di accoppiamento deve essere coerente con tale criterio.

4.2.4 La classe MatrixFract

La classe `MatrixFract` si utilizza solo nel caso di mezzi porosi fratturati; il suo scopo è assemblare le matrici C e C^T usate, rispettivamente, nel modello di frattura (3.2) e nelle condizioni di accoppiamento (3.3). Si noti che esse sono fortemente sparse, il numero degli elementi non nulli è pari al doppio delle facce fratturate, più precisamente, detto F il numero totale di facce e f il numero di facce fratturate (e tipicamente $F \gg f$), allora C ha dimensione $f \times (F + f)$ e i suoi elementi diversi da zero sono $2f$.

Nella sua struttura, `MatrixFract` è analoga a `MatrixDivOperator`: non ci sono membri protetti aggiuntivi a quelli ereditati e i due metodi `virtual` da implementare seguono la stesso strategia: `assemble()` assembla C, poi `getMatrix_transpose()` ne calcola la trasposta e la corregge secondo le (4.1)-(4.2); mentre `fillMatrix(SpMat &MAT)` riempie i blocchi che gli competono di `MAT` contemporaneamente, senza dover fare due cicli. In questo secondo caso, essendo gli elementi calcolati ad hoc per le due matrici, non è necessario apportare nessuna correzione posteriore.

Per assemblare le matrici, pertanto, è sufficiente iterare sulle facce di frattura (quindi sugli elementi del vettore `M_fractureFacets` presente nella griglia) e sfruttando le formule sopra citate calcolare gli elementi da inserire. Inoltre, come per tutte le altre matrici, anche gli elementi di C e C^T devono essere moltiplicati per un opportuno `segno`, derivante dall'orientazione della faccia e ricavato come descritto in precedenza.

Infine, come si può notare dalle condizioni di accoppiamento, la pressione nelle facce di frattura, rappresentata dal vettore π è, ovviamente, unica, quindi non avviene nessuno sdoppiamento del grado di libertà.

4.2.5 Le classi per i Volumi Finiti

Oltre alle classi che assemblano le matrici delle Differenze Finite Mimetiche, nel codice sono presenti anche quelle che implementano i Volumi Finiti. In particolare, la classe `MatrixStiffness` ha il compito di fornire la matrice di trasmissibilità della formula (3.13); gli scopi sono, principalmente, due: permettere un confronto tra le due discretizzazioni e assemblare la matrice T delle DFM. Infatti, si ricorda che per calcolare la pressione nelle fratture, si è scelto di risolvere il modello con la tecnica dei VF. Esistono, quindi, due versioni di questa classe: una che calcola la matrice di trasmissibilità nella sua completezza e una che si preoccupa solo del contributo frattura-frattura (per i dettagli si veda il Capitolo 3). Va sottolineato che solo la versione relativa alle DFM eredita da `MatrixHandler`, infatti, le matrici relative ai VF hanno una strategia di dimensionamento diversa, dovuta alla loro formulazione primale; per esempio, non vengono sdoppiati i gradi di libertà delle velocità nelle facce fratturate, che sono considerate vere e proprie celle, e il sistema viene risolto solo in pressione. Per questo motivo, la classe base da cui ereditare è un'altra: `MatrixHandlerVF`, di cui non si riportano i dettagli implementativi, (si veda [23]).

Un'altra importante differenza risiede nel tensore di permeabilità. Nei VF esso si riduce ad una funzione scalare, che viene calcolata nel baricentro di ogni cella, inoltre, la classe `MatrixStiffness` prende in input anche la viscosità μ del fluido; invece, nelle fratture la permeabilità è considerata costante e il mezzo isotropo. Per le DFM, la permeabilità nel mezzo poroso è rappresentata da un tensore (2×2 nel caso 2D), che racchiude anche la viscosità del fluido, mentre, nelle fratture, la permeabilità è costante e scomposta nelle componenti tangenziale e normale, per poter considerare anche i mezzi anisotropi.

E' presente un'ulteriore classe, `MatrixMass`, che calcola la matrice di massa del problema, cioè una matrice diagonale che ha in posizione i-esima il volume della cella i-esima. Essa può risultare utile nel caso si voglia calcolare l'avanzamento temporale dell'equazione di Darcy o se è presente un termine di reazione con coefficiente costante. Analogamente alla matrice di stiffness, anche per quella di massa esistono due versioni: per VF e per DFM, le quali ereditano dalla rispettiva classe base. Oltre alle dimensioni, le matrici risultanti differiscono per un altro motivo: nei VF le facce di frattura sono considerate vere e proprie celle, di cui si calcola il volume, allora si rende necessaria una correzione al volume delle celle confinanti, per far sì che quello totale si conservi. Nelle DFM, invece, le facce di frattura sono trattate in modo differente e la matrice di massa è assemblata solo per le celle effettive.

L'implementazione di queste classi esula da questo lavoro e, per tale motivo,

non vengono riportati ulteriori dettagli.

4.2.6 Condizioni al bordo e termine noto

Si è già parlato di come si ottiene il vettore contenente le condizioni al bordo, resta da spiegare come esso viene riempito. Nel caso sul bordo si imponga la pressione (condizione di Dirichlet), essa compare esplicitamente nella formulazione debole mimetica (2.10), questo significa che non bisogna fare nessuna modifica alle matrici, ma il vettore g si compone calcolando l'integrale della funzione imposta sulle facce di bordo. Diverso è il caso in cui si imponga la velocità (condizione di Neumann); tale condizione va imposta in forma forte e questo significa che la riga corrispondente alla faccia di bordo diventa nulla, tranne che per un 1 sulla diagonale. Nella corrispondente riga del vettore g, anche questa volta va messo l'integrale della funzione imposta sul bordo. Operando in questo modo, la matrice M perde la simmetria e, quindi, sarebbe preferibile adottare tecniche di penalizzazione per imporre la velocità al bordo, se si usano solver iterativi per matrici simmetriche e definite positive.

La forzante b del sistema (2.4), secondo la teoria riportata, deve essere interpolata nello spazio mimetico, il che significa integrarla sulle celle e riempire il vettore termine noto in modo opportuno. A tal proposito, si accenna brevemente alle due classi che permettono di compiere questa operazione: `QuadratureRule` e `Quadrature`.

La classe `QuadratureRule` è una classe astratta, che contiene al suo interno un metodo `clone()` e un metodo `apply()`. Da essa può derivare una qualsiasi regola di quadratura, che permetta di calcolare l'integrale di una funzione data su un dominio dato. La firma del metodo `apply()` è la seguente:

```
1 D_Real apply(const std::vector<Point2D>& pointVector,
2   const D_Real _volume,
3   const std::function<D_Real(Point2D)>& Integrand) const;
```

e cioè in input al metodo bisogna dare il vettore contenente i vertici del dominio su cui si vuole integrare, il volume del dominio e la funzione che si vuole integrare.

Le regole di quadratura che sono state implementate sono una regola a tre punti per triangoli 2D (del secondo ordine) e una regola che approssima la funzione con il suo valore nel baricentro, valida per elementi poligonali generici.

La classe `Quadrature` ha la seguente struttura:

```
1 template <class T>
2 class Quadrature {
3 //alcuni typedef
```

```
4  public:
5  Quadrature (const Geometry::Rigid_Mesh<T> &rigid_mesh);
6  Quadrature (const Geometry::Rigid_Mesh<T> &rigid_mesh,
7              const QuadratureRule<T>& quadrature);
8  Quadrature (const Geometry::Rigid_Mesh<T> &rigid_mesh,
9              const QuadratureRule<T>& quadrature,
10             const QuadratureRule<T>& fracturequadrature);
11
12 D_Real Integrate (const function& Integrand);
13 Vector CellIntegrate (const function& func);
14 Vector CellFractureIntegrate (const function& func);
15 D_Real Integrate (const Vector& Integrand);
16 D_Real L2Norm (const Vector& Integrand);
17
18 protected:
19 const Geometry::Rigid_Mesh<T> & M_mesh;
20 QR_Handler M_quadrature;
21 QR_Handler M_fractureQuadrature;
```

In ingresso al costruttore si deve dare una griglia e, opzionalmente, una regola di quadratura del tipo `QuadratureRule` per le celle e una per le facce di frattura. Le funzionalità di questa classe sono molteplici:

- `D_Real Integrate (const Vector& Integrand)` restituisce l'integrale di una funzione costante rappresentata da `Integrand` sul dominio. L'integrale si calcola moltiplicando ogni elemento del vettore per il rispettivo volume di cella e sommando tutti i contributi.
- `D_Real L2Norm (const Vector& Integrand)` svolge lo stesso compito, ma elevando al quadrato gli elementi del vettore e calcolando la radice quadrata del risultato ottenuto.
- `D_Real Integrate (const function& Integrand)` restituisce l'integrale di una funzione continua, data in input al metodo.
- `Vector CellIntegrate (const function& func)` utilizza la regola di quadratura per calcolare l'integrale sulle singole celle e restituisce un vettore che ha in posizione i-esima l'integrale della funzione nella cella i-esima. Questa funzione è quella da chiamare per calcolare l'interpolata del termine sorgente da usare in (2.22) e (3.20).
- `Vector CellFractureIntegrate (const function& func)` svolge lo stesso compito della funzione precedente, ma ora l'integrale viene fatto sulle facce di frattura.

4.3 Il `main`

4.3.1 Mezzo poroso non fratturato

Nel main `test.cpp` sono implementati i tre esempi riportati nel Capitolo 2. L'obiettivo è calcolare gli errori e gli ordini di convergenza del metodo delle Differenze Finite Mimetiche applicato ad un mezzo poroso non fratturato.

Il flusso di operazioni svolte è il seguente:

1) Definizione dei vertici del dominio
2) Definizione delle condizioni al bordo
3) Creazione della Rigid_Mesh
4) Calcolo delle matrici
    4a) M
    4b) B
    4c) B^T
5) Calcolo del vettore delle condizioni al bordo
6) Definizione di una Quadrature
7) Discretizzazione del termine noto tramite la Quadrature
8) Risoluzione del sistema lineare
9) Interpolazione delle soluzioni esatte negli spazi mimetici
10) Calcolo degli errori
11) Esportazione dei dati in formato .vtk

Nei paragrafi precedenti sono stati riportati i dettagli su come viene creata la griglia e su come vengono assemblate la matrici, non si è ancora parlato di come si risolve il sistema lineare. In un mezzo poroso privo di fratture, la matrice M del prodotto interno mimetico è sempre simmetrica e definita positiva; a tal proposito si veda il Lemma 2.3. Per questa parte, si è scelto di non assemblare il sistema completo, ma di risolvere (2.23), le matrici vengono quindi computate con il metodo `assemble()`. Data la struttura della matrice, il metodo utilizzato è `SimplicialCholesky` delle `Eigen`, che si basa sulla fattorizzazione di Cholesky. Le operazioni da svolgere per arrivare a calcolare p e u sono le risoluzioni dei seguenti sistemi lineari:

1. $\mathsf{M}v = g$
2. $\mathsf{MS} = \mathsf{B}^T$
3. $\mathsf{BS}p = f + \mathsf{B}v \qquad \Longrightarrow p = ...$
4. $\mathsf{M}u = g - \mathsf{B}^T p \qquad \Longrightarrow u = ...$

Si noti che l'operazione computazionalmente più onerosa è la seconda, in quanto l'incognita del sistema non è un vettore ma una matrice. I metodi delle `Eigen` sono progettati per essere efficienti anche in questi casi.

4.3.2 Mezzo poroso fratturato

Il main `fractured_test.cpp` implementa i casi test riportati nel Capitolo 3.

Il flusso di operazione svolte è il seguente:

1) Definizione dei vertici del dominio
2) Definizione delle condizioni al bordo
3) Definizione delle fratture
4) Creazione della Rigid_Mesh
5) Calcolo delle matrici
    - 5a) M
    - 5b) B
    - 5c) B^T
    - 5d) C
    - 5e) C^T
    - 5f) T
6) Calcolo del vettore delle condizioni al bordo
7) Definizione di una Quadrature
8) Discretizzazione del termine noto tramite la Quadrature
9) Risoluzione del sistema lineare completo
10) Interpolazione delle soluzioni esatte negli spazi mimetici
11) Calcolo degli errori
12) Esportazione dei dati in formato .vtk

In questo secondo caso, si è scelto di assemblare la matrice globale del sistema (3.20), tramite il metodo `fillMatrix(SpMat &MAT)`, questo perché per dimensioni non elevate delle matrici è preferibile risolvere il sistema monolitico. Si arriva, quindi, ad un sistema lineare del tipo $Ax = b$ dove, per le considerazioni fatte in precedenza, A non è necessariamente simmetrica e definita positiva, per questo motivo non si può più usare il metodo `SimplicialCholesky` e bisogna ripiegare su metodi diretti per matrici non SDP; è stato scelto il metodo `SparseLU` delle `Eigen` che, nonostante la matrice perda le buone qualità, permette un calcolo efficiente della soluzione.

Conclusioni

L'obiettivo di questa tesi è stata l'analisi, sia dal punto di vista modellistico che da quello numerico, del flusso di un fluido in un mezzo poroso in cui sono presenti delle fratture. Si è scelto di descrivere il fenomeno con la legge di Darcy, sia per il flusso nel mezzo poroso, sia nelle fratture. Il metodo numerico che è stato analizzato è quello alle Differenze Finite Mimetiche (DFM), che presenta alcuni vantaggi rispetto metodi tradizionali come i Volumi Finiti o gli Elementi Finiti: in primo luogo vi è la possibilità di generare una griglia con elementi del tutto generici, ciò è molto utile nei casi applicativi, infatti è possibile generare griglie che si adattino bene alle fratture, non preoccupandosi dell'eventuale formazione di elementi allungati e anisotropi; in secondo luogo i metodi mimetici presentano buone proprietà di convergenza dell'errore.

La prima analisi che è stata svolta riguarda il caso di un mezzo poroso privo di fratture: dopo aver descritto in dettaglio la discretizzazione tramite DFM del problema di Darcy si è passati alla validazione del metodo numerico. Sono state effettuate simulazioni per alcuni casi test di complessità crescente, su griglie strutturate di quadrati, non strutturate triangolari e griglie formate da elementi generici di tre, quattro e cinque lati. Il calcolo dell'errore ha mostrato risultati in accordo con le stime teoriche: per la pressione si è notato un decadimento quadratico dell'errore per tutti i tipi di griglia (fenomeno di superconvergenza); per la velocità il decadimento quadratico si è presentato solo per le griglie strutturate, mentre per le altre tipologie si è riscontrato un ordine di convergenza poco più che lineare.

Il passaggio successivo è stato estendere il modello al caso di mezzo poroso con fratture. Si è proposto, a tal scopo, un modello matematico per il flusso nelle fratture, opportunamente accoppiato a quello della matrice solida circostante. Dopo aver introdotto il modello è stata proposta una strategia di discretizzazione che consiste nell'utilizzare le Differenze Finite Mimetiche per la matrice solida e i Volumi Finiti per le fratture. L'approssimazione numerica è stata validata con l'analisi dell'errore: si è riscontrato anche per questo caso un abbattimento quadratico dell'errore per la pressione. Infine, sono stati risolti alcuni casi test per verificare la giusta dipendenza del metodo dai pa-

rametri fisici: i risultati hanno mostrato un comportamento qualitativamente corretto sia nel caso di fratture molto permeabili, sia nel caso di fratture quasi impermeabili. Inoltre, i risultati sono coerenti con quelli prodotti da una discretizzazione ai Volumi Finiti.

Per poter effettuare le simulazioni è stato implementato un codice `C++` originale che, dopo aver generato la griglia, si occupa dell'assemblaggio e della risoluzione del sistema lineare derivante dalla discretizzazione mimetica.

Da questo lavoro si possono trarre alcuni spunti per ulteriori sviluppi. Uno dei principali è l'estensione del codice al caso 3D: nel dominio tridimensionale le fratture diventano oggetti a due dimensioni e si possono rappresentare con le facce dei poliedri che formano la griglia. Il modello matematico per mezzi porosi fratturati si può estendere al caso 3D, così come il metodo delle DFM. In questo caso il modello di frattura può essere discretizzato con i Volumi Finiti, oppure, essendo un problema 2D, si può proporre un approccio mimetico. Naturalmente, l'implementazione 3D delle DFM presenta difficoltà maggiori rispetto al caso 2D, sia dal punto di vista geometrico (generazione della griglia, scelta dell'orientazione delle facce) sia per la scelte implementative da compiere per assemblare le matrici in maniera corretta, inoltre il costo computazionale del caso 3D cresce notevolmente e occorre usare un solver adatto per calcolare le soluzioni dei sistemi lineari.

Un ulteriore campo di indagine futuro è rappresentato dalla ricostruzione della velocità nel dominio: ad oggi esistono operatori di ricostruzione teorici (si veda [21]) ma è ancora oggetto di studio la loro implementazione concreta.

Infine il modello matematico può essere arricchito: per esempio si può considerare l'effetto della gravità nella legge di Darcy, per tenere conto anche dell'effetto delle variazioni di quota, inoltre si può considerare un fluido bifase, tenendo conto delle interazioni diverse che si possono avere o si può considerare il termine in tempo nell'equazione della conservazione della massa, modellizzando questo modo un fluido comprimibile. Ulteriori estensioni si possono fare sul modello per la frattura, per esempio non trascurandone lo spessore o accoppiandolo al modello per la matrice porosa con condizioni diverse da quelle proposte.

Bibliografia

[1] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[2] EIGEN v3. http://eigen.tuxfamily.org.

[3] Jørg E Aarnes, Stein Krogstad, and Knut-Andreas Lie. Multiscale mixed/mimetic methods on corner-point grids. *Computational Geosciences*, 12(3):297–315, 2008.

[4] Ivar Aavatsmark. Interpretation of a two-point flux stencil for skew parallelogram grids. *Computational geosciences*, 11(3):199–206, 2007.

[5] Pierre M. Adler, Jean-François Thovert, and Valeri V. Mourzenko. *Fractured Porous Media*. Oxford, University of Oxford, 2013.

[6] C. Alboin, J. Jaffré, J. E. Roberts, and C. Serres. Modeling fractures as interfaces for flow and transport in porous media. In *Fluid Flow and Transport in Porous Media, Mathematical and Numerical Treatment*, volume 295, page 13. American Mathematical Soc., 2002.

[7] Philippe Angot, Franck Boyer, and Florence Hubert. Asymptotic and numerical modelling of flows in fractured porous media. *M2AN Math. Model. Numer. Anal.*, 43(2):239–275, 2009.

[8] PF Antonietti, N Bigoni, and M Verani. Mimetic finite difference approximation of quasilinear elliptic problems. Technical report, Technical Report MOX-Preprint, 38/2012. Submitted for publication, 2012.

[9] PF Antonietti, L Beirão da Veiga, N Bigoni, and M Verani. Mimetic finite differences for nonlinear and control problems. *Mathematical Models and Methods in Applied Sciences*, 2012.

[10] Douglas N Arnold, Franco Brezzi, Bernardo Cockburn, and L Donatella Marini. Unified analysis of discontinuous galerkin methods for elliptic problems. *SIAM journal on numerical analysis*, 39(5):1749–1779, 2002.

[11] Jacob Bear, Chin-Fu Tsang, and G de Marsily. *Flow and contaminant transport in fractured rock*. Academic Press, San Diego, 1993.

[12] Luca Bergamaschi, Stefano Mantica, and Fausto Saleri. Mixed finite element approximation of darcy's law in porous media. *Report CRS4 AppMath-94-20, CRS4, Cagliari, Italy*, 1994.

[13] Franco Brezzi, Annalisa Buffa, and Konstantin Lipnikov. Mimetic finite differences for elliptic problems. *ESAIM: Mathematical Modelling and Numerical Analysis*, 43(02):277–295, 2009.

[14] Franco Brezzi and Michel Fortin. *Mixed and hybrid finite element methods*. Springer-Verlag, New York, 1991.

[15] Franco Brezzi, Konstantin Lipnikov, and Mikhail Shashkov. Convergence of the mimetic finite difference method for diffusion problems on polyhedral meshes. *SIAM Journal on Numerical Analysis*, 43(5):1872–1896, 2005.

[16] Franco Brezzi, Konstantin Lipnikov, and Valeria Simoncini. A family of mimetic finite difference methods on polygonal and polyhedral meshes. *Mathematical Models and Methods in Applied Sciences*, 15(10):1533–1551, 2005.

[17] Andrea Cangiani and Gianmarco Manzini. Flux reconstruction and solution post-processing in mimetic finite difference methods. *Computer Methods in Applied Mechanics and Engineering*, 197(9):933–945, 2008.

[18] Yufei Cao, Rainer Helmig, and Barbara I Wohlmuth. Geometrical interpretation of the multi-point flux approximation l-method. *International journal for numerical methods in fluids*, 60(11):1173–1199, 2009.

[19] José E Castillo and Guillermo F Miranda. *Mimetic Discretization Methods*. CRC Press, 2013.

[20] Zhangxin Chen, Guanren Huan, and Yuanle Ma. *Computational Methods for Multiphase Flows in Porous Media*. SIAM, 2006.

[21] Lourenço Beirao da Veiga, Konstantin Lipnikov, and Gianmarco Manzini. *The Mimetic Finite Difference Method for Elliptic Problems*. Springer, 2014.

[22] Durga Dalal, Frédéric Hecht, and Olivier Pironneau. Implementation of a low order mimetic elements in freefem++. *Journal of Numerical Mathematics*, 20(3-4):183–194, 2012.

[23] F. DellaPorta. Implementazione di un solutore per flussi in mezzi porosi con fratture, 2013. Politecnico di Milano.

[24] Daniele Antonio Di Pietro, Martin Vohralík, et al. A review of recent advances in discretization methods, a posteriori error analysis, and adaptive algorithms for numerical modeling in geosciences. *Oil and Gas Science and Technology*, 2013.

[25] Carlo D'Angelo and Anna Scotti. A mixed finite element method for darcy flow in fractured porous media with non-matching grids. *ESAIM: Mathematical Modelling and Numerical Analysis*, 46(02):465–489, 2012.

[26] Alessio Fumagalli and Anna Scotti. An efficient XFEM approximation of darcy flow in arbitrarly fractured porous media. *Oil and Gas Sciences and Technologies - Revue d'IFP Energies Nouvelles*, 2013. In printing.

[27] Alessio Fumagalli and Anna Scotti. A numerical method for two-phase flow in fractured porous media with non-matching grids. *Advances in Water Resources*, 62:454–464, 2013.

[28] H Hægland, A Assteerawatt, HK Dahle, GT Eigestad, and R Helmig. Comparison of cell-and vertex-centered discretization methods for flow in a two-dimensional discrete-fracture–matrix system. *Advances in water resources*, 32(12):1740–1755, 2009.

[29] Hadi Hajibeygi, Dimitris Karvounis, and Patrick Jenny. A hierarchical fracture model for the iterative multiscale finite volume method. *Journal of Computational Physics*, 230(24):8729–8743, 2011.

[30] J Hyman, J Morel, M Shashkov, and Stanly Steinberg. Mimetic finite difference methods for diffusion equations. *Computational Geosciences*, 6(3-4):333–352, 2002.

[31] Jérôme Jaffré, Vincent Martin, and Jean E. Roberts. Generalized cell-centered finite volume methods for flow in porous media with faults. In *Finite volumes for complex applications, III (Porquerolles, 2002)*, pages 343–350. Hermes Sci. Publ., Paris, 2002.

[32] M Karimi-Fard, LJ Durlofsky, K Aziz, et al. An efficient discrete-fracture model applicable for general-purpose reservoir simulators. *SPE Journal*, 9(02):227–236, 2004.

[33] SPE Kok-Thye Lim, SPE Mun-Hong Hui, and SPE Bradley Mallison. A next-generation reservoir simulator as an enabling technology for a complex discrete fracture modeling workflow. 2009.

[34] Yu Kuznetsov, K Lipnikov, and M Shashkov. The mimetic finite difference method on polygonal meshes for diffusion-type problems. *Computational Geosciences*, 8(4):301–324, 2004.

[35] Knut-Andreas Lie, Stein Krogstad, Ingeborg Skjelkvåle Ligaarden, Jostein Roald Natvig, Halvor Møll Nilsen, and Bård Skaflestad. Open-source matlab implementation of consistent discretisations on complex grids. *Computational Geosciences*, 16(2):297–322, 2012.

[36] I. S. Ligaarden. Well models for mimetic finite difference methods and improved representation of wells in multiscale methods. Master thesis, University of Oslo, 2008.

[37] S.B. Lippman, J. Lajoie, and B.E. Moo. *C++ Primer (IV ed)*. Addison Wesley, 2005.

[38] Vincent Martin, Jérôme Jaffré, and Jean E Roberts. Modeling fractures and barriers as interfaces for flow in porous media. *SIAM Journal on Scientific Computing*, 26(5):1667–1691, 2005.

[39] H. M. Nilsen, K-A Lie, J. R. Natvig, and S. Krogstad. Accurate modelling of faults by multipoint, mimetic and mixed methods. *SPE Journal*, 17:568–579, 2013.

[40] C. Ossola. Differenze finite mimetiche per equazioni ellittiche a coefficienti altamente variabili. Master thesis, Dipartimento di Matematica, Politecnico di Milano, 2012.

[41] M. Oueslati. Comparison of monolithic and iterative solvers for coupled fracture-network rock-matrix porous media problems. Bachelor thesis, Universität Stuttgart, 2013.

[42] Alfio Quarteroni. *Modellistica Numerica per Problemi Differenziali*. Springer, Milan, 2008.