Pontifícia Universidade Católica
do Rio de Janeiro

Rafael Ferrão Moura Cruz

Simulação do Escoamento em Meios Porosos Fraturados utilizando a Equação de Brinkman

Projeto de Graduação

Orientador : Prof. Márcio Carvalho
Coorientador: Dr. Frederico Gomes

Rio de Janeiro
Dezembro de 2017

Pontifícia Universidade Católica
do Rio de Janeiro

Rafael Ferrão Moura Cruz

Simulation of Fluid Flow Through Fractured Porous Media Using the Equation of Brinkman

Graduation Project

Adviser : Prof. Márcio Carvalho
Coadviser: Dr. Frederico Gomes

Rio de Janeiro
December 2017

Agradecimentos

Em primeiro lugar, gostaria de agradecer a minha familiar por toda estrutura e suporte que me deram durante essa trajetória.

Ao meu orientador Professor Márcio Carvalho pela oportunidade de trabalhar com ele e aos meus colegas de laboratório por terem me proporcionado um ambiente de estudo invejável.

Resumo

Cruz, Rafael; Carvalho, Márcio; Gomes, Frederico. **Simulação do Escoamento em Meios Porosos Fraturados utilizando a Equação de Brinkman**. Rio de Janeiro, 2017. 31p. Projeto de Graduação – Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Devido a sua complexa formação geológica as rochas carbonáticas possuem características únicas quem compõem a sua estrutura. Apresentando-se como uma preocupação para aqueles que se empenham na recuperação de petróleo desses reservatórios, essa formação geológica é composta por rochas classificadas como sedimentadas, que são formadas pela deposição e cementação de sedimentos e corpos de água. Devido a sua formação de natureza complexa, essas rochas possuem matrizes composta por diversas heterogeneidade gerando problemas relacionados a variação da sua porosidade de acordo com o trecho que está sendo analisando. Alguns dos problemas que surgem quando o escoamento nessas rochas é colocado em questão são relacionados a modelagem do comportamento desse escoamento, a medida que numa mesma rocha são encontrados meios porosos e espaços vazios. Devido a essa questão, criar uma metodologia de trabalho para lidar com a modelagem desse escoamento não é uma tarefa trivial. Atacando esse problema em uma escala menor, a particularidade equivalente a cada parcela de uma matriz de uma rocha carbonática pode ser modelada utilizando métodos mais sofisticados. Para esse trabalho, empregando o métodos de elementos finitos, foi desenvolvido uma metodologia para a modelagem desse escoamento atraves da formulação da equação de escoamento de Brinkman. E para cálculo da permeabilidade média desses segmentos foi utilizada a lei Darcy para meios porosos.

Palavras-chave

Escoamento em meios porosos; Rochas carbonáticas; Elementos finitos;

Abstract

Cruz, Rafael; Carvalho, Márcio; Gomes, Frederico. **Simulation of Fluid Flow Through Fractured Porous Media Using the Equation of Brinkman**. Rio de Janeiro, 2017. 31p. Graduation Project – Department of Mechanical Engineering, Pontifical Catholic University of Rio de Janeiro.

Due to their complex geological formation, the carbonate rocks have unique characteristics that form their stucture. Revealing as a concern for those who are engaged in oil recovery study from these reservoirs, this geological formation is composed of rocks classified as sedimented, which are formed by the deposition and cementation of sediments and bodies of water. Due to their complex formation, these rocks have matrices composed of diverse heterogeneity generating problems related to the variation of their porosity according to the section being analyzed. Some of the problems that arise when the flow in these rocks is studied are related to the modeling of the behavior of this flow, because in the same rock are found porous media and voids. Due to this issue, creating a working methodology to deal with the modeling of this flow is not a trivial task. By attacking this problem on a smaller scale, the particularity equivalent to each portion of an array of a carbonate rock can be modeled using more sophisticated methods. For this work, using a finite element method, a methodology was developed for the modeling of this flow through the formulation of the Brinkman flow equation. For the calculation of the average permeability of these segments the Darcy law for porous media was used.

Keywords

Flow through Porous Media; Brinkman Equation; Carbonate Rocks; Finite Elements;

Sumário

Lista de figuras

1
Introdução

As rochas carbonáticas, carbonatos, são rochas compostas majoritariamente por carbonatos sedimentares. Podem ser formados por fragmentos líticos, materiais biológicos provindos de restos de animais e ou precipitados químicos.

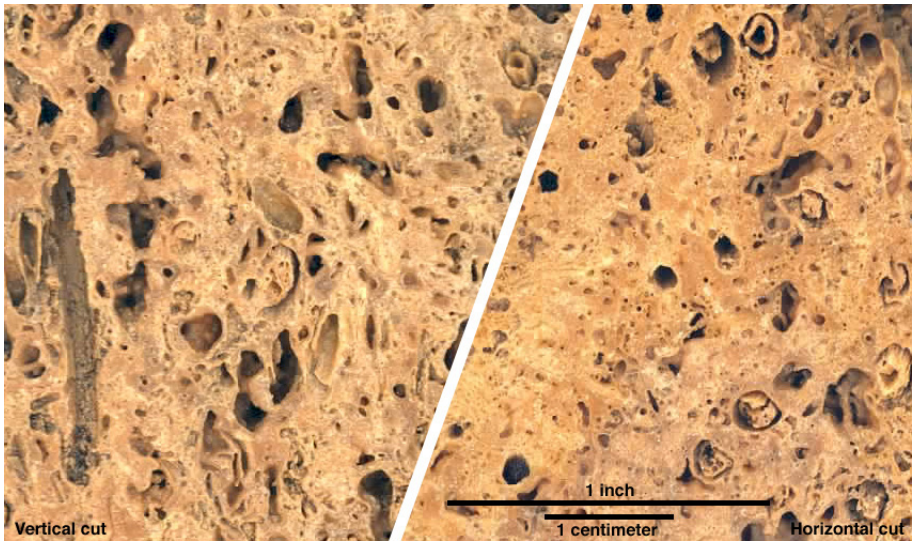


Figura 1.1: Rocha carbonática (1)

Essa rocha se torna interessante ao estudo de engenheiros, como nós, por se apresentar em abundância em reservatórios de petróleo. Devido a sua estrutura peculiar, a recuperação de petróleo dessas rochas se mostra difícil de ser previstas por modelos simplificados.Devido a natureza da sua formação, a porosidade dessas rochas não necessariamente se relacionam com os sedimentos que foram depositados nela. Também, as rochas carbonáticas apresentam uma gama de tipos de porosidade diferentes. Idealmente, o ataque a modelagem do escoamento nessas rochas deve ser feito de modo a abranger as suas mais variadas estruturas e, ao mesmo tempo, propor um modo simplificado de representar esse escoamento.

Dentre a variedade de defeitos nessas rochas carbonáticas que podem ser observadas, as mais comuns são as cavernas, fraturas, partículas internas e partículas conectadas. O estudo apresentado nesse trabalho se dedicou a análise dos defeitos do tipo cavidade (vuggy) e fratura (fracture).

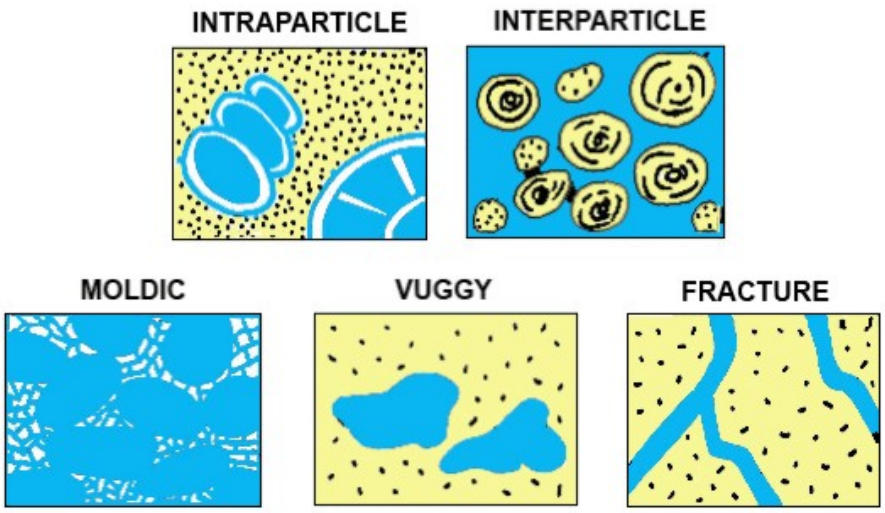


Figura 1.2: Tipos de porosidade - Rocha carbonática (2)

Assim como visto em Rodolfo 2014 (3), o estudo de estruturas miniaturizadas dessas rochas são de grande vália já que um coeficiente de permeabilidade médio pode ser encontrado e usado a fim de compreender, com mais facilidade, o escoamento nas pequenas porções das rochas que apresentarem essa estrutura específica. Posteriormente, caso um modelo simplificado de uma parede rochosa composta de carbonatos seja desejada, um coeficiente médio global de permeabilidade pode ser obtido a partir dos coeficientes médios locais.

Permeabilidade média:

$$k_{avg} = \frac{k_1 \cdot h_1 + \cdots + k_n \cdot h_n}{H} = \frac{\sum_{i=1}^{n} k_i \cdot h_i}{\sum_{i=1}^{n} h_i} \tag{1-1}$$

Onde h_i representa o comprimento dessa camada, k_i o coeficiente médio dessa camada, H o comprimento total.

2
Formulação Numérica

A modelagem numérica de um meio poroso complexo, como as rochas carbonáticas, tem um desafio no que tange uma estrutura formada por trechos em que o fluido escoa em um meio poroso, obedecendo a lei de Darcy, e outros em que ele escoa livremente, obedecendo a equação de Navier-Stokes. Esse problema pode ser contornado através da equação de Brinkman que, através de um termo chamado de viscosidade de Brinkman, consegue ajustar a sua equação de forma a se comportar como a equação de Navier-Stokes quando as forças viscosas são significantes e com a equação de Darcy quando as forças viscosas puderem ser negligenciadas.
Mesmo utilizando uma equação tão flexível quanto a de Brinkman um problema ainda é presente, a resolução analítica de um escoamento tão complexo como esse não é praticável.

2.1
Elementos Finitos

Haja visto que o problema proposto não será resolvido analiticamente devido a sua complexidade, o método de elementos finitos mostrou-se um bom caminho a ser seguido. Esse procedimento numérico é utilizado quando é permitida a obtenção de uma solução aproximada subdividindo o domínio do problema em sub-regiões que são submetidas a condições de contorno previamente definidas. Para utilizar esse método é necessário reformular a equação diferencial parcial que representa o comportamento do sistema para a sua forma discreta. Essa equação discreta será obtida através da formulação fraca de Galerkin.

2.2
Escoamento de Stokes

Para o caso especial em que o comportamento estudado é de um fluido Newntoniano incompressível, a tensão é proporcional a viscosidade e a deformação infinitesimal do fluido com o tempo:

$$\boldsymbol{\tau} = \mu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^{\mathrm{T}} \right) \tag{2-1}$$

Para o τ como o tensor das tensões, μ a viscosidade do fluido e u a velocidade. ' Por fim, o escoamento estudado por stokes é aquele que é caracterizado por ter as suas forças inerciais pequenas se comparadas as forcas viscosas. Esse tipo de escoamento é identificado por ter um numero de Reynolds muito pequeno devido a sua baixa velocidade. À vista disso, substituindo os temos explicitados e aplicando os conceitos apresentados, a equação de conservação da quantidade de movimento, derivada da 2ª lei de newton, para um regime permanente pode ser reduzida a:

$$\boldsymbol{f} + \mu\nabla^2\boldsymbol{u} = \nabla p \tag{2-2}$$

Para o τ como o tensor das tensões, μ a viscosidade do fluido, u a velocidade e $\boldsymbol{f}$ uma força externa aplicada ao fluido.
'

2.3
Escoamento de Darcy

Tendo sida formulada inicialmente por Darcy durante os seus estudos relacionados ao escoamento de água por pacotes de areia, a lei de Darcy descreve o comportamento de um fluido quando imposto à uma região porosa. Embora a lei de Darcy tenha sido obtida através de experimentos empíricos, ela hoje em dia é comumente obtida através da equação de stokes. Considerando que o termo que representa a viscosidade da equação de stokes ($\mu\nabla^2\boldsymbol{u}$) é proporcional ao campo de velocidade($\mu\boldsymbol{k^{-1}u}$), o escoamento estudado por Darcy é pode ser deduzido para:

$$\mu\boldsymbol{k^{-1}u} + \boldsymbol{f} - \nabla p = 0 \tag{2-3}$$

É váido salientar que para um meio poroso isotrópico, com o escoamento perpendicular a gravidade, o tensor $\boldsymbol{k}$, que representa o coeficiente de permeabilidade do meio, se reduz a uma matriz diagonal resultando em uma equação simplificada:

$$u = \frac{k}{\mu}\nabla p \tag{2-4}$$

2.4
Equação de Brinkman

Visto que foram apresentadas equações que explicitam o comportamento de um fluido em dois meios diferentes, a equação de Brinkman provém de uma mistura dessas últimas duas. A equação de Brinkman descreve o escoamento em um meio poroso que consegue acumular velocidade o suficiente para gerar energia cinética. Ela, geralmente, é aplicada a meios que possuem uma

estrutura de transição entre um meio poroso e um outo meio que as forças inerciais podem ser ignoradas. Essa equação pode ser obtida através da adição do termo viscoso de Brinkman $\mu^*\nabla^2\boldsymbol{u}$ a equação de Darcy:

$$\nabla p = -\mu\boldsymbol{k^{-1}u} + \mu^*\nabla^2\boldsymbol{u} + \boldsymbol{f} \tag{2-5}$$

Onde μ^* representa a viscosidade efetiva de Brinkman e geralmente é aproximada para a viscosidade do fluido. A utilidade desse termo à equação vem da sua característica de adaptar-la de acordo com a posição da estrutura que está sendo varrida. Para $k- > \infty$ o termo $\mu\boldsymbol{k^{-1}u}$ é zerado e a equação de Stokes é obtida enquanto que para $\mu^*- > 0$ o termo $\mu^*\nabla^2\boldsymbol{u}$ é zerado e a equação de Darcy é obtida.

2.5 Formulação Variacional

A fim de utilizar o método de elementos finitos, através do pacote FEniCS, a equação de Brinkman deve ser modificada de forma a ser compreendida pelo programa. A equação de Brinkman, para um meio homogêneo e isotrópico, deverá ser adaptada da forma de equação diferencial parcial para uma formulação discreta (4), definida no método de elementos finitos. Como visto na equação (2-5), a equação do escoamento de Brinkman é descrita por:

$$-\mu^*\Delta\boldsymbol{u} + \nabla p + \mu\boldsymbol{k^{-1}u} = \boldsymbol{f} \quad \forall\ x\ \in\ \Omega \tag{2-6a}$$

$$\nabla\cdot\boldsymbol{u} = 0 \quad \forall\ x\ \in\ \Omega \tag{2-6b}$$

Onde o termo $\nabla\cdot u = 0$ representa representa a incompressibilidade do fluido. As condições de contorno sâo definidas por:

$$\boldsymbol{u}\cdot\boldsymbol{n} = u_0 \quad \text{on } \Gamma_D \tag{2-7a}$$

$$p = p_{in} \quad \text{on } \Gamma_{p_{in}} \tag{2-7b}$$

$$p = p_{out} \text{ on } \Gamma_{p_{out}} \tag{2-7c}$$

Onde Γ representa a fronteira do domínio, o índice D a fronteira definida para as condições de borda de Dirichlet, p_{in} a pressão de entrada, p_{out} a pressão de saída e $\boldsymbol{n}$ o vetor normal a essa fronteira. A fim de obter a formulação fraca da equação de Brinkman, a equação (2-5) será multiplicada por uma função peso vetorial v, contínua e contida no subespaço das funções que assumem o valor de $v = 0$ nas fronteiras onde as condições essenciais do sistema são definidas. Após essa multiplicação a equação é integrada no domínio:

$$\int_\Omega -\mu^*\Delta\boldsymbol{u}\cdot\boldsymbol{v}\ d\Omega + \int_\Omega \nabla p\cdot\boldsymbol{v}\ d\Omega + \int_\Omega \mu\boldsymbol{k^{-1}u}\cdot\boldsymbol{v}\ d\Omega = \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}\ d\Omega \tag{2-8}$$

Integrando por partes os dois primeiros termos da equação e aplicando o teorema de Gauss as igualdades são obtidas:

$$-\int_{\Omega} \mu^* \Delta \boldsymbol{u} \cdot \boldsymbol{v}\ d\Omega = \int_{\Omega} \mu \nabla \boldsymbol{u} : \nabla \boldsymbol{v} d\Omega - \int_{\Gamma} \mu (\nabla \boldsymbol{u} \cdot \boldsymbol{v}) \cdot \boldsymbol{n} dS \tag{2-9a}$$

$$\int_{\Omega} \nabla p \cdot \boldsymbol{v}\ d\Omega = -\int_{\Omega} p(\nabla \cdot \boldsymbol{v}) d\Omega + \int_{\Gamma} p \cdot \boldsymbol{v} \cdot \boldsymbol{n} dS \tag{2-9b}$$

Para as igualdades acima, o segundo termo da primeira equação (2-9a) será igualado a *zero* devido a hipótese de escoamento desenvolvido para as fronteiras de entrada e saída do fluido.
Substituindo as igualdades na equação (2-8):

$$\begin{aligned} \int_{\Omega} \mu^* \nabla \boldsymbol{u} : \nabla \boldsymbol{v}\ d\Omega - \int_{\Omega} p\ (\nabla \cdot \boldsymbol{v})\ d\Omega + \int_{\Omega} \mu \boldsymbol{k^{-1}} \boldsymbol{u} \cdot \boldsymbol{v}\ d\Omega = \\ \int_{\Omega} \boldsymbol{f} \cdot \boldsymbol{v}\ d\Omega - \int_{\Gamma} p \cdot \boldsymbol{v} \cdot \boldsymbol{n} dS \end{aligned} \tag{2-10}$$

O segundo termo da equação de Brinkman, relacionado ao divergente do campo de velocidades, é multiplicado por outra função teste q porém, agora escalar. Essa função teste é contínua e contida no subespaço das funções que assumem o valor de $q = 0$ nas fronteiras onde as condições essenciais, de pressão, do sistema são definidas. O procedimento que subcede é o mesmo realizado na equação diferencial anterior:

$$\int_{\Omega} (\nabla \cdot \boldsymbol{u})\ q\ d\Omega = 0 \tag{2-11}$$

Finalmente, a integral obtida acima é subtraída da equação (2-10) e a forma variacional da equação de Brinkman é obtida:

$$a_B(\boldsymbol{u}, \boldsymbol{v}) + b(\boldsymbol{v}, p) + b(\boldsymbol{u}, q) = L(\boldsymbol{v}) \quad \forall (\boldsymbol{v}, q) \in \boldsymbol{V} \chi\ Q \tag{2-12}$$

Para os valores de a_B, b e L expostos abaixo:

$$a_B(\boldsymbol{u}, \boldsymbol{v}) = \int_{\Omega} \mu \nabla \boldsymbol{u} : \nabla \boldsymbol{v}\ d\Omega + \int_{\Omega} \frac{\mu}{k} \boldsymbol{u} \cdot \boldsymbol{v}\ d\Omega \tag{2-13a}$$

$$b(\boldsymbol{v}, q) = \int_{\Omega} \nabla \cdot \boldsymbol{v}\ q\ d\Omega \tag{2-13b}$$

$$L(\boldsymbol{v}) = \int_{\Omega} \boldsymbol{f} \cdot \boldsymbol{v}\ d\Omega + \int_{\Gamma} P\ \boldsymbol{v} \cdot \boldsymbol{n}\ ds \tag{2-13c}$$

Onde P, visto em $L(v)$, representa um função linear, definidas nas fronteiras de entrada e saída do sistema. Por conseguinte, P assume o valor de p_{in} na parede de entrada e p_{out} na parede de saída da integral de linha $\int_{\Gamma} P\ \boldsymbol{v} \cdot \boldsymbol{n}\ ds$. O passo a passo do programa implementado será apresentado a seguir explicitando a função de cada bloco do programa assim como a resolução da equação variacional de Brinkman.

3
Desenvolvimento do Programa

3.1
FEniCS Project

FEniCS project é um amalgama de softwares livres que tem a finalidade de resolver equações diferenciais parciais. Dentro dele se encontram diversos componentes que funcionam através do ambiente de interface dolfin, utilizando a linguagem Python. Esses componentes são:

- FIAT
  É um módulo de python dedicado a geração dos mais diversos elementos das funções de base.
- UFL
  Abreviação para Unified Form Language, UFL é uma linguagem embutida em python com a finalidade de discretizar equações diferenciais parciais em relação a sua forma variacional.
- FFC
  Manipulando a entrada vinda do UFL, o FFC se comporta como um compilador do FEniCS gerando um output no formato UFC
- UFC
  Baseado na linguagem C++, essa componente empenha-se nos comando de low-level que resolvem as formas variacionais dos elementos finitos.

Embora o programa não seja trivial para um usuário que não esteja familiarizado com a linguagem de programação Python, o guia disponibilizado no site oficial do projeto (5) serve como um guia essencial para aqueles que estão dando começando a utiliza-lo.
Obviamente, devido a simplicidade característica da linguagem Python, caso seja do interesse do usuário, outros módulos podem ser exportados de acordo com a necessidade do programa.

3.2 Implementação do Programa

Inicialmente o programa é iniciado importando o modulo backend do pacote FEniCS, dolfin, e as malhas que serão utilizadas no programa:

```
from dolfin import *
import Fractures, Vugs, Connected
```

As malhas, representadas pelas funções *Fractures*, *Vugs*, *Connected* foram geradas em programas a parte e podem ser chamadas de acordo com a necessidade do usuário, devido a generalização do programa. O módulo dolfin carrega todas as funções (exceto as malhas) que serão usadas no decorrer do programa.

A implementação do programa se inicia com a definição das condição de contorno Dirichlet para os subdomínios referentes as paredes da malha. Essas classes funcionam de modo a retornar um dado boolean de acordo com a posição da malha que está sendo varrida. Caso o ponto se encontre na posição desejada a classe retorna o valor **True**, caso não esteja ela retorna o valor **False**. É bom notar que a fim de evitar erros numéricos uma tolerância $DOLFIN\_EPS = 10^{-14}$ é adicionada.

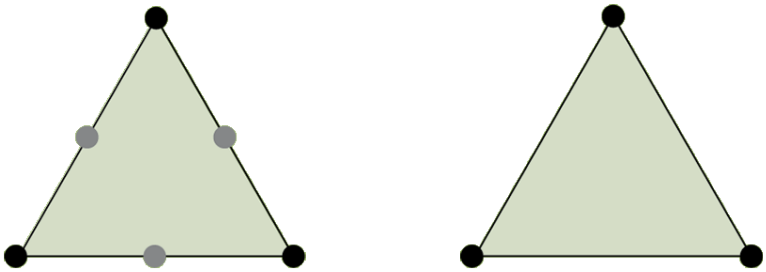
```
class Left ( SubDomain ) :
    def inside (self,x,on_boundary) :
        return x[0] < DOLFIN_EPS

class Right ( SubDomain ) :
    def inside ( self , x , on_boundary ) :
        return x[0] > 1.0 - DOLFIN_EPS

class Bottom ( SubDomain ) :
    def inside ( self , x , on_boundary ) :
        return x[1] < DOLFIN_EPS

class Top ( SubDomain ) :
    def inside ( self , x , on_boundary ) :
        return x[1] > 1.0 - DOLFIN_EPS
```

Sequencialmente, a malha é chamada de acordo com a vontade do usuário. As malhas geradas foram escritas em função do tamanho do lado do domínio que contem as geometrias complexas propostas. A função que segue é de extrema

importância para o programa, através dela todas as células que compõem os subdomínios relacionados as fraturas e as cavidades que compõem as malhas são marcados com o índice 1 e as células que estão fora são marcadas com o índice 0.

```
#definindo malha
l=1
mesh = Connected.Malha(l)

#Funcao que marca as celulas do dominio
boundary_markers = MeshFunction('size_t', mesh, 2, mesh.domains
    ↪ ())
```

Sequencialmente o espaço funcional, as funções base e as funções peso são criadas a partir dos elementos finitos inicializados para a velocidade e para pressão. A fim de prezar pela estabilidade do método, o elemento referente a velocidade possui seis graus de liberdade enquanto o referente a pressão possui três. Nas funções que definem os elementos finitos o termo $'CG'$ se refere a $ContinuosGalerkin$, o que implica na família de funções base definidas por polinômios linear, para a pressão, e quadráticos para a velocidade.



(a) Elemento com 6 graus de liberdade

(b) Elemento com 3 graus de liberdade

Figura 3.1: Elementos Finitos (6)

```
#Espaco das funcoes relacionadas aos elementos inicializados/
    ↪ Funcoes base e peso
V = VectorElement('CG',mesh.ufl_cell(),2)
Q = FiniteElement('CG',mesh.ufl_cell(),1)
Element = V*Q
W = FunctionSpace(mesh,Element)
(u,p) = TrialFunctions(W)
(v,q) = TestFunctions(W)
```

Através das classes relacionadas as condições de contorno Dirichlet, previamente vistas, os entornos das malhas são marcados de acordo com a sua posição. A função iniciada transcreve todos os lados dos elementos e as suas devidas coordenadas para posteriormente serem marcadas de acordo com a sua posição.

```
#Marcacao das fronteiras da malha
boundaries = FacetFunction ('size_t', mesh)
boundaries.set_all(0)
left.mark(boundaries,1)
top.mark(boundaries,2)
```

A fim de rodar o programa, os dados iniciais também são necessários. Como o problema proposto não tem a ambição de estudar a influência de forças externas assim como da velocidade inicial, esses valores foram definidos como zero. A expressão P é necessária por representar a condição de contorno associada a pressão de entrada e saída do sistema, ela é representada por uma função linear em x porém, P só será computado nas fronteiras de entrada e saída do sistema, como poderá ser visto mais a frente.

```
#Dados de entrada
mu = 0.001003 #Viscosidade da agua [Pa.s]
k = 1E-11 #Coeficiente de permeabilidade do meio poroso [m^2]
pin = Constant(10.0) #Pressao de entrada [Pa]
pout = Constant(0.0) #Pressao de saida [Pa]
dp = pin-pout
P = Expression('b-a*x[0]' , degree=1 ,a = Constant(pin-pout), b
    ↪ = pin) #g = div(u)
u_in = Constant((0.0)) #Velocidade inicial em x [m/s]
f = Constant((0.0,0.0)) #Forca externa
```

Para as fronteiras do domínio superior e inferior as condições de contornos essenciais, relacionadas a componente vertical da velocidade, foram definidas como 0. $W.sub(0)$ chama o elemento finito referente a velocidade e $W.sub(0).sub(1)$ aponta para a coordenada y desse elemento. Nota-se que caso fosse do interesse do usuário impor uma velocidade inicial na direção x ao sistema, uma condição de contorno Dirichlet teria que chamar o seguinte termo; $W.sub(0).sub(0)$.

```
#Condicao de contorno Dirichlet nas fronteiras
```

```
bc2 = DirichletBC(W.sub(0).sub(1),Constant(0.0),boundaries,2)
bc4 = DirichletBC(W.sub(0).sub(1),Constant(0.0),boundaries,4)

bcs = [bc2,bc4]
```

Para efetuar as integrações de superfície e de linha definidas no problema variacional, as parcelas infinitesimais devem ser iniciadas com os seus devidos índices. Apontando para as paredes do domínio o elemento *ds* foi iniciado para o subdomínio referente as paredes da malha, o índice 1 representa a fronteira de entrada, 2 a parede superior, 3 a fronteira de saída e 4 a parede inferior do sistema. Para identificar os subdomínios relacionados à dentro e fora das geometrias complexas iniciadas, o elemento *dx* foi iniciado, o índice 1 presenta a região interna aos defeitos do da rocha enquanto o 2 a externa a esses defeitos.

```
#Demarcacao dos indices das regioes
ds = Measure('ds',domain=mesh, subdomain_data = boundaries)
dx = Measure('dx', domain=mesh, subdomain_data =
    ↪ boundary_markers)
```

Finalmente, a equação de Brinkman é transposta para o problema variacional. Ela é resolvida utilizando a função *solve* que além da igualdade referente ao problema variacional (2-8) também requer os termos referente ao espaço funcional definido e as condições de contorno escolhidas. Como o problema foi resolvido pela função U definida sobre o espaço W, ele contém tanto o mapa de pressão quanto o campo de velocidade. Para obtê-los separadamente, a função *.split* foi utilizada.

```
#Equacao variacional de Brinkman
a =(mu*inner(grad(u),grad(v))*dx(1)+(mu/k)*inner(u,v)*dx(0)-div(
    ↪ v)*p*dx(1)-div(v)*p*dx(0)-div(u)*q*dx(1)-div(u)*q*dx(0))

L = (inner(f,v)*dx(1)+inner(f,v)*dx(0)-P*dot(v,n)*ds(1)-P*dot(v,
    ↪ n)*ds(3))

#Solucao do problema
U = Function (W)
solve(a==L,U,bcs)

#Separar do espaco U
```

```
u,p=U.split()
```

Por fim, a permeabilidade média do meio foi calculada. O ataque a esse problema se deu da seguinte forma, os nós dos elementos de velocidade que se encontram na fronteira de saída do fluxo ($x = 1$) foram projetos em x a fim de apenas computar o termo da velocidade que está atravessando a fronteira. Esses componentes foram somatizada para todos os vértices de todos os elemento que cumprem a condição definida, Essa velocidade somatizada foi então dividida pelo número de vértices que contribuíram para essa soma. O resultado dessa conta foi substituído na equação de Darcy com o intuito de achar um coeficiente médio de permeabilidade que representasse o sistema, caso ele fosse homogêneo.

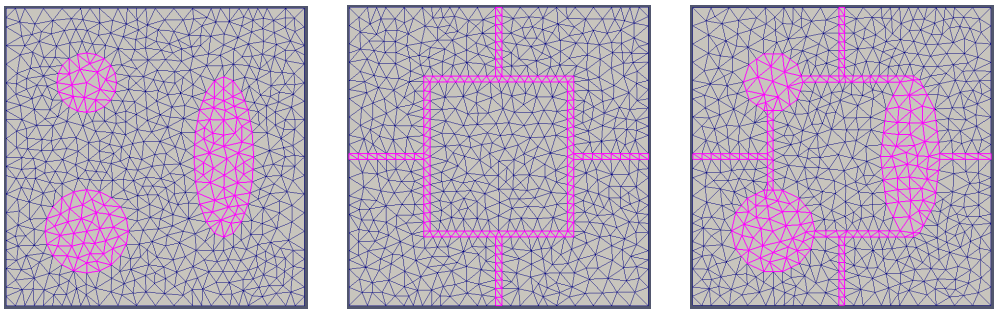
$$k_{medium} = u \cdot \frac{\mu L}{dp} \tag{3-1}$$

```
#Calculo Permeabilidade media
ux, uy = u.split(deepcopy=True)
num_border=0
utot=0

for i in range(len(mesh.coordinates())):
    if mesh.coordinates()[i,0] == 1:
        utot += ux.compute_vertex_values(mesh)[i]
        num_border += 1

k_medium = (utot/num_border)*(mu*l)/(dp)
```

O programa completo assim como todas as malhas se encontram em anexo nas últimas páginas desse trabalho.

4
Resultados

De inicio, é interessante validar o programa através de uma ideia geral de como o sistema deveria se comportar. As três matrizes apresentadas podem ser subdividias em duas regiões, uma representada pela cor rosa que indica a região fraturada da rocha carbonática e uma outra que não se apresenta destacada, representando a região porosa. A primeira será regida pela lei de Stokes enquanto a outra pela equação de Darcy para meios porosos.
As três malhas foram escolhidas com o intuito de representar estruturas que são encontradas com regularidade(figura 1) em rochas carbonáticas. A primeira representa uma estrutura formada por duas cavidades circulares e uma outra em forma de elipse, a segunda foi elaborada de forma a representar uma estrutura complexa formada por rachaduras e a terceira é basicamente a união das duas últimas. As três malhas foram geradas pelo pacote *mshr* dedicado a criação das mesmas, o código pode ser visto em A.1, A.2 e A.3.



(a) Malha com três cavidades

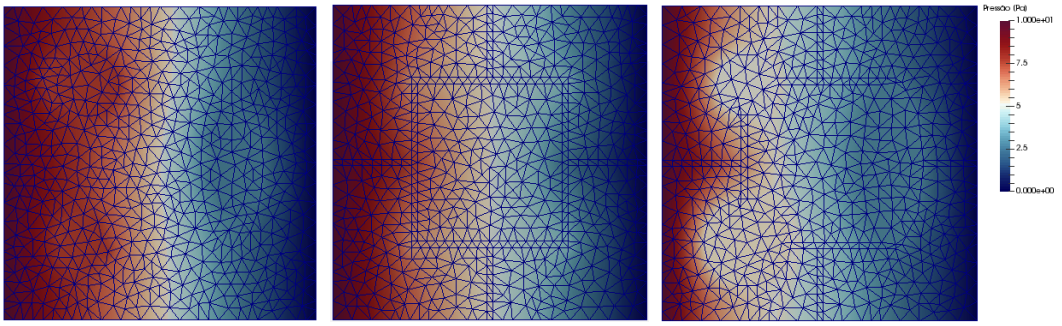
(b) Malha com fraturas

(c) Malha com cavidades e fraturas

Figura 4.1: Malhas

Antes de começar a expor os resultados obtidos através desse trabalho, os dados de entrada utilizados serão apresentados: A viscosidade do fluido escolhido foi $\mu = 0.001003[Pa.s]$, o coeficiente de permeabilidade do meio poroso $k = 1E-11[m^2]$, a pressão de entrada $pin = 10.0[Pa]$, pressão de saída $pout = 0.0[Pa],]$. Os valores referentes as forças externas e a velocidade inicial da água foram igualadas a zero já que estudar a influencia deles nesse sistema não era o interesse do projeto.
O primeiro grupo de dados gerados foram os mapas de pressão de cada malha.

A fim de melhor compreender a influência da malha nos dados pressão obtidos, os dois foram superpostos.



(a) Malha com três cavidades (b) Malha com fraturas (c) Malha com cavidades e fraturas

Figura 4.2: Mapas de Pressão

Com o auxílio visual da malha é possível notar a influência da mesma. Nos mapas de pressão são perceptíveis a formação de bolhas de pressão constante nas cavidades propostas, isso acontece devido a velocidade não variante nessa região. Fato esse não observado com clareza nas rachaduras devido a sua área diminuta e estrutura paralela, em alguns pontos, a direção da velocidade.
O segundo grupo de mapas gerados foram dos campos de velocidade. Além dos vetores que representam tanto a direção quanto a tendencia de escoamento de cada vértice de cada elemento finito gerado, subposto a eles também está presente o mapa da magnitude da velocidade.

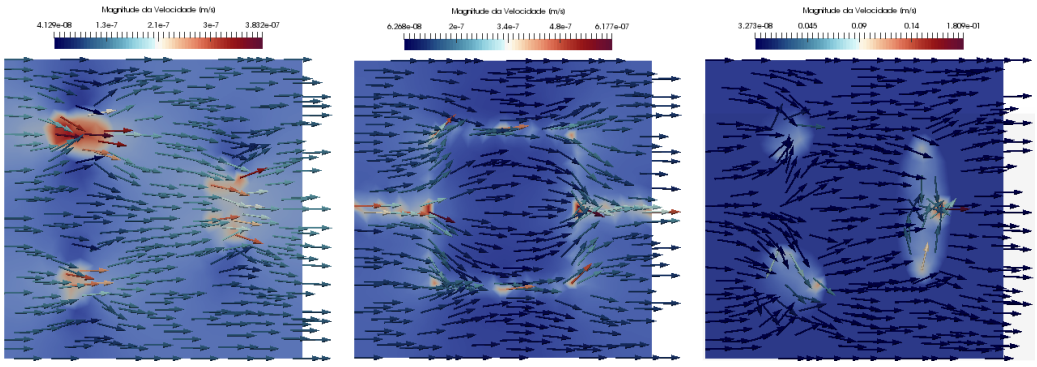


Figura 4.3: Campos de Velocidade

Por efeito das regiões com permeabilidade infinita, regidas pela equação de Stokes, já era esperado essa tendência preferencial do escoamento a essas áreas. Outro ponto que pode ser observado é o aumento considerável da velocidade nas regiões livres. Isso se da devido ao principio de conservação de momento linear, assim como pode ser observado o efeito de um escoamento através de um tubo estrangulado, a concentração do escoamento, devido ao caminho

preferencial, irá aumentar a velocidade do mesmo.
Finalmente, através do script justificado no final do capítulo 3.2, que calcula a componente horizontal do velocidade que atravessa a fronteira referente a parede direita do domínio, a permeabilidade média foi calculada aproximando o meio complexo a um simplesmente poroso através da lei de Darcy:

$$k_{medium} = u \cdot \frac{\mu L}{dp} \tag{4-1}$$

Malha	Permeabilidade Média k_{medium} $[mD]$	$(k - k_{medium})/k \times 100\%$
Fraturada	14678.05	44,86%
Cavidades	12134.94	19,76%
Conectadas	16534.93	63,18%

*mD representa miliDarcy, $1mD \simeq 9.869233 \times 10^{-16}\ m^2$

Lembrando que o coeficiente de permeabilidade do meio poroso foi definido como $k = 10^{-11}[m^2]$ (valor médio achado nos reservatórios de petróleo) para todas malhas, a influência das fraturas assim como das cavidades no escoamento não podem ser ignorada. Por definição esse coeficiente de permeabilidade em solos consiste na capacidade que o material poroso tem de permitir o escoamento por ele. Numericamente, quanto maior o coeficiente de permeabilidade menor é a resistência do material poroso ao escoamento. Logo, o aumento desse coeficiente de permeabilidade faz todo sentido a medida que as fraturas e as cavidades representam um meio com esse coeficiente tendendo ao infinito $k- > \infty$.

5
Conclusão

Neste trabalho foi implementado um programa que lida com o escoamento em meios que apresentam uma estrutura complexa. Para vias de um estudo mais simplificado, as malhas geradas são simples e obviamente fogem de um caso real de um rocha carbonática. Porém, no que esse trabalho falhou em representar a realidade ele teve êxito aplicar uma metodologia de cálculo que é capaz de generalizar o escoamento em uma estrutura complexa.
Como foi transcorrido na equação (2-5) sobre escoamentos nessas estruturas, a equação de Brinkman se adapta ao comportamento do fluido. De acordo que com a posição que esta sendo estudada os parâmetros referentes ao coeficiente de permeabilidade e a viscosidade mudam. Posto isso, o programa gerado tem a sua relevância a medida que representa, através de um coeficiente médio de permeabilidade, o que se deve esperar do comportamento do fluido no contorno do seu domínio.

Referências bibliográficas

[1] *www.sandatlas.org.*

[2] *http://spe-sc.ft.ugm.ac.id/w/the-two-main-important-thing-in-reservoir-rock/.*

[3] Rodolfo Oliveira *Finite Element Method Applied to Flow in Heterogeneous Porous Media*

[4] Lin Mu., Junping Wang and Xio Ye *A stable numerical algorith for the Brinkman equations by weak Galerkin Finite Element Methods*

[5] Hans Petter Langtangen, Ander Logg *Solving PDEs in Python - The FEniCS Tutorial Volume I*

[6] *https://femtable.org/*

A
Scripts

Código A.1: Malha composta por Vugs

```
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 20 11:41:26 2017

@author: rafael
"""
from dolfin import *
from mshr import *

def Malha(l):

        xmin = 0
        xmax = 1*l
        ymin = 0
        ymax = 1*l

        xcenter1 = (.25+.02)*l
    ycenter1 = 0.25*l
    xcenter2 = (.25+.02)*l
    ycenter2 = 0.75*l
    xcenter3 = (0.75-0.02)*l
    ycenter3 = 0.5*l
        radius1 = 0.14
        radius2 = 0.1
        a=0.1*l
        b=0.27*l

        domain = Rectangle(Point(xmin,ymin),Point(xmax,ymax))
        domain.set_subdomain(1,Circle(Point(xcenter1,ycenter1),
            ↪ radius1))
```

```
30        domain.set_subdomain(2,Circle(Point(xcenter2,ycenter2),
              ↪ radius2))
31        domain.set_subdomain(3,Ellipse(Point(xcenter3,ycenter3),a
              ↪ ,b))
32        mesh = generate_mesh(domain,20)
33        return mesh
```

Código A.2: Malha composta por Fraturas

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 22 15:57:39 2017

@author: rafael
"""

from dolfin import *
from mshr import *

def Malha(l):

        xmin = 0
        xmax = 1*l
        ymin = 0
        ymax = 1*l

        xcenter1 = 0.25*l
        ycenter1 = 0.75*l
        xcenter2 = 0.25*l
        ycenter2 = 0.25*l
        xcenter3 = 0.75*l
        ycenter3 = 0.5*l
        radius1 = 0.2
        radius2 = 0.1

        domain = Rectangle(Point(xmin,ymin),Point(xmax,ymax))
        R1 = Rectangle(Point(0,(.5-.01)*l),Point(.25*l,(.5+.01)*l
            ↪ ))
        R2 = Rectangle(Point(.75*l,(.5-.01)*l),Point(l,(.5+.01)*l
            ↪ ))
        R3 = Rectangle(Point(.25*l,.75*l),Point((.75)*l,(.75+.02)
            ↪ *l))
        R4 = Rectangle(Point(.25*l,(.25-.02)*l),Point((.75)*l
            ↪ ,.25*l))
        R5 = Rectangle(Point((.5-.01)*l,(.75+.02)*l),Point
            ↪ ((.5+.01)*l,l))
        R6 = Rectangle(Point((.5-.01)*l,0),Point((.5+.01)*l
```

```
        ↪ ,(.25-.02)*l))
34      R7 = Rectangle(Point(.25*l,.25*l),Point((.25+.02)*l,.75*l
        ↪ ))
35      R8 = Rectangle(Point(.75*l-.02,.25*l),Point(.75*l,.75*l))
36      tot = R1+R2+R3+R4+R5+R6+R7+R8
37
38      domain.set_subdomain(1,tot)
39      mesh = generate_mesh(domain,20)
40
41      return mesh
```

Código A.3: Malha composta pela união dos Vugs com as fraturas

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 27 11:17:40 2017

@author: rafael
"""

from dolfin import *
from mshr import *

l=1

def Malha(l):

    xmin = 0
    xmax = 1*l
    ymin = 0
    ymax = 1*l

    xcenter1 = (.25+.02)*l
    ycenter1 = 0.25*l
    xcenter2 = (.25+.02)*l
    ycenter2 = 0.75*l
    xcenter3 = (0.75-0.02)*l
    ycenter3 = 0.5*l
    radius1 = 0.14
    radius2 = 0.1
    a=0.1*l
    b=0.27*l

    domain = Rectangle(Point(xmin,ymin),Point(xmax,ymax))
    R1 = Rectangle(Point(0,(.5-.01)*l),Point(.25*l,(.5+.01)*l))
    R2 = Rectangle(Point(.75*l,(.5-.01)*l),Point(l,(.5+.01)*l))
    R3 = Rectangle(Point(.25*l,.75*l),Point((.75-0.01)*l
        ↪ ,(.75+.02)*l))
    R4 = Rectangle(Point(.25*l,(.25-.02)*l),Point((.75-0.01)*l
        ↪ ,.25*l))
    R5 = Rectangle(Point((.5-.01)*l,(.75+.02)*l),Point((.5+.01)*
```

```
        ↪ l,l))
37      R6 = Rectangle(Point((.5-.01)*l,0),Point((.5+.01)*l
        ↪ ,(.25-.02)*l))
38      R7 = Rectangle(Point(.25*l,.25*l),Point((.25+.02)*l,.75*l))
39      C1 = Circle(Point(xcenter1,ycenter1),radius1,14)
40      C2 = Circle(Point(xcenter2,ycenter2),radius2,10)
41      E1 = Ellipse(Point(xcenter3,ycenter3),a,b,10)
42      tot = R1+R2+R3+R4+R5+R6+R7+C1+C2+E1
43
44      domain.set_subdomain(1,tot)
45      mesh = generate_mesh(domain,20)
46
47      return mesh
```

Código A.4: Programa Principal

```
# -*- coding: utf-8 -*-
"""
Created on Wed Nov 25 18:00:00 2017

@author: rafael
"""

from dolfin import *
import Fractures, Vugs, Connected

class Left ( SubDomain ) :
    def inside (self,x,on_boundary) :
        return x[0] < DOLFIN_EPS

class Right ( SubDomain ) :
    def inside ( self , x , on_boundary ) :
        return x[0] > 1.0 - DOLFIN_EPS

class Bottom ( SubDomain ) :
    def inside ( self , x , on_boundary ) :
        return x[1] < DOLFIN_EPS

class Top ( SubDomain ) :
    def inside ( self , x , on_boundary ) :
        return x[1] > 1.0 - DOLFIN_EPS

#definindo malha
l=1
mesh = Connected.Malha(l)

#Funcao que marca as celulas do dominio
boundary_markers = MeshFunction('size_t', mesh, 2, mesh.domains
    ↪ ())

#Espaco das funcoes relacionadas aos elementos inicializados/
    ↪ Funcoes base e peso
V = VectorElement('CG',mesh.ufl_cell(),2)
Q = FiniteElement('CG',mesh.ufl_cell(),1)
```

```
Element = V*Q
W = FunctionSpace(mesh,Element)
(u,p) = TrialFunctions(W)
(v,q) = TestFunctions(W)

#Ativar os sub-dominios de fronteira
left, right = Left(), Right()
top, bottom = Top() , Bottom()

#Marcacao das fronteiras da malha
boundaries = FacetFunction ('size_t', mesh)
boundaries.set_all(0)
left.mark(boundaries,1)
top.mark(boundaries,2)
right.mark(boundaries,3)
bottom.mark(boundaries,4)


#Dados de entrada
mu = 0.001003 #Viscosidade da agua [Pa.s]
k = 1E-11 #Coeficiente de permeabilidade do meio poroso [m^2]
pin = Constant(10.0) #Pressao de entrada [Pa]
pout = Constant(0.0) #Pressao de saida [Pa]
dp = pin-pout
P = Expression('b-a*x[0]' , degree=1 ,a = Constant(pin-pout), b
    ↪ = pin) #g = div(u)
u_in = Constant((0.0)) #Velocidade inicial em x [m/s]
f = Constant((0.0,0.0)) #Forca externa

#Condicao de contorno Dirichlet nas fronteiras
bc2 = DirichletBC(W.sub(0).sub(1),Constant(0.0),boundaries,2)
bc4 = DirichletBC(W.sub(0).sub(1),Constant(0.0),boundaries,4)

bcs = [bc2,bc4]

#Demarcacao dos indices das regioes
ds = Measure('ds',domain=mesh, subdomain_data = boundaries)
dx = Measure('dx', domain=mesh, subdomain_data =
    ↪ boundary_markers)
```

```
74
75  #vetor normal a malha
76  n = FacetNormal(mesh)
77
78  #Equacao variacional de Brinkman
79  a =(mu*inner(grad(u),grad(v))*dx(1)+(mu/k)*inner(u,v)*dx(0)-div(
        ↪ v)*p*dx(1)-div(v)*p*dx(0)-div(u)*q*dx(1)-div(u)*q*dx(0))
80
81  L = (inner(f,v)*dx(1)+inner(f,v)*dx(0)-P*dot(v,n)*ds(1)-P*dot(v,
        ↪ n)*ds(3))
82
83  #Solucao do problema
84  U = Function (W)
85  solve(a==L,U,bcs)
86
87  #Separar do espaco U
88  u,p=U.split()
89
90  #Calculo Permeabilidade media
91  ux, uy = u.split(deepcopy=True)
92  num_border=0
93  utot=0
94
95  for i in range(len(mesh.coordinates())):
96      if mesh.coordinates()[i,0] == 1:
97          utot += ux.compute_vertex_values(mesh)[i]
98          num_border += 1
99
100 k_medium = (utot/num_border)*(mu*l)/(dp)
101 print(k_medium)
102
103 #Gera os graficos
104 plot(u,title = 'Velocity')
105 plot(p,title = 'Pressure')
106 interactive ()
```