Um Estudo da Instabilidade de Saffman-Taylor com Fluido Magnético e, ou Anisotrópico

Ataias Pereira Reis, Yuri Dumaresq Sobral, Francisco Ricardo da Cunha
Universidade de Brasília
Departamento de Matemática
Brasília, Brasil
ataiasreis@gmail.com

***Resumo*—Criação de algoritmos para resolver equações diferenciais parciais. Uso de diferenças finitas para resolver as equações de Laplace e Poisson, usando método ímplicito e explícito, usando bibliotecas de álgebra linear de código aberto. Análise da diferença de tempo de resolução entre os métodos implícito e explícito. Uso do método iterativo e de resolução direta de sistema linear.**

***Index Terms*—Ferrofluidos, Navier Stokes, EDP, Poisson, Laplace, Diferenças finitas, Eigen**

I. Introdução

A instabilidade de Saffman-Taylor, também chamada de endedamento, ou *fingering*, que pode ser visto na Figura 1, é um fenômeno que ocorre na superfície de contato entre dois fluidos sob circunstâncias específicas. Esse fenômeno ocorre quando um fluido menos viscoso é injetado para deslocar um outro mais viscoso (na situação inversa, do fluido mais viscoso usado para movimentar o outro, a interface é estavel, não ocorrendo o endedamento). Também pode ocorrer movida pela gravidade, ao invés de injeção de um fluido em outro. Neste caso, a interface separando os fluidos de diferentes densidades está direcionada na horizontal, e o mais pesado está em cima do outro. Este tipo de fenômeno é um problema ocorrente em petrolíferas marítimas. Em tais petrolíferas, ocorre a injeção de água nos tubos de extração de petróleo, no objetivo do óleo subir. Na interface entre água e petróleo, o endedamento ocorre, originando bolhas de óleo dentro de água, que tem um efeito negativo na extração do petróleo, causando perca de óleo quando a água é jogada fora.

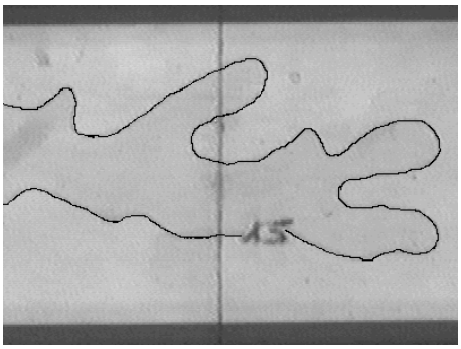

Figura 1. Demonstração de fingering

Para o estudo dessa instabilidade, que não é nada trivial, a solução se dá por meio de métodos numéricos auxiliados por computador. A equação de Navier Stokes deve ser discretizada e então resolvida numericamente. Ela é uma equação diferencial parcial altamente não-linear e de difícil resolução. No caso da instabilidade de Saffman-Taylor, no qual a fronteira está em movimento, que é a interface entre os dois fluidos, faz-se necessário algoritmos numéricos capazes de lidar com este movimento sem causar complicações extremas que impossibilitem a obtenção de soluções práticas.

Tais ferramentas numéricas e decisão de métodos/algoritmos a se utilizar são a primeira etapa neste projeto, para então, após se ter tais ferramentas, estudar a física do problema e propor soluções para o problema com a utilização de um ferrofluido. A apresentação no resto deste relatório mostrará até onde se alcançou na codificação dos algoritmos numéricos para resolução do problema proposto, que inicia-se com o estudo de equações diferenciais parciais e de diferenças finitas na forma contínua, e só então na forma discreta. Após isso, estuda-se em específico a equação de Navier Stokes, que é dividida em etapas, para facilitar a busca de resultados.

II. Metodologia

A metologia aqui proposta resume as tarefas realizadas, não seguindo um ordem cronológica, mas sim uma mais organizada, para estar bem divididas as seções.

A. Programação
B. Equação de Laplace com condições de fronteira de Dirichlet
C. Equação de Poisson com condições de fronteira de Dirichlet e Neumann
D. Equação de Navier Stokes

Considerações gerais sobre a equação

1. Desconsiderando a pressão
2. Obtendo a pressão
3. Avançando no tempo

III. Técnicas de Computação Básica

O fato do problema numérico em questão ser difícil torna o tempo uma questão fundamental na escolha da linguagem de programação a ser utilizada. De início a ferramenta

e linguagem de programação utilizada era o MATLAB®. O MATLAB inclui uma infinidade de algoritmos já prontos disponíveis para uso, ferramentas para plotagem de gráficos embutidas, ótimos sistemas de referências de funções, mas tem a desvantagem de ser muito caro, proprietário e consumir muita memória e processamento só de estar aberto, sem executar o próprio programa para resolver nossas equações em questão. Como o tempo é um fator crítico, como já mencionado, o custo de memória de um programa como o MATLAB, sendo executado por muitas horas ou dias, pode ser algo muito complicado, preferiu-se tomar outro rumo para o desenvolvimento dos nossos códigos.

Tais problemas levaram a uma escolha de uma linguagem de programação compilada e rápida, o C++. O fato do aluno ter experiência em C inicialmente, aliado ao fato de se haver encontrado uma biblioteca de álgebra linear chamada Eigen[1] que evita a necessidade de criação de inúmeros algoritmos básicos, e que é feita em C++, influenciou na escolha de tal linguagem.

A Eigen inclui algoritmos para resolução de sistemas lineares, criação de matrizes e alocação de memória, operações básicas de matrizes, resolução de sistemas de matrizes esparsas e vários outros algoritmos não explorados que estão à disposição, isso tudo de graça, pois é código aberto e gratuito.

Apesar de se ter escolhido o C++ e a Eigen, isto só não traz todas as ferramentas necessárias ao trabalho em questão. A plotagem de gráficos diretamente em C++ não é algo trivial, não foram encontradas bibliotecas de fácil uso que pudessem ser utilizadas de uma forma tão simples como o MATLAB. Após muita pesquisa, foi decidido utilizar o Python e suas bibliotecas de plotagem de gráficos. Os códigos em C++ passaram a ser compilados não em programas, mas em bibliotecas, e executados de dentro do ambiente Python, que possui poderosos recursos para salvar arquivos, plotar gráficos e tudo de fácil estudo para uso, completou-se o arsenal de ferramentas de programação que são utilizadas no presente trabalho.

Além das ferramentas que lidam direto com código, é utilizado o CMake[2] para criação de Makefiles para compilar o código e o git[3] é utilizado como gerenciador de versões do projeto, de forma a sempre se ter as versões antigas dos códigos salvas, e bem organizadas.

O projeto tem sido desde certo ponto hospedado num servidor git gratuito, que pode ser obtido facilmente online[4]. Neste site do projeto há um histórico de todas as versões dos programas do projeto, e os códigos C++, Python e instruções de como compilar e executar em sistema operacional Ubuntu.

[1]Eigen: http://eigen.tuxfamily.org/
[2]Cmake: http://www.cmake.org/
[3]Git: http://git-scm.com/
[4]Github projeto: https://github.com/ataias/ff

IV. Equação de Laplace com condições de fronteira de Dirichlet

Primeiramente, o aluno se familiarizou com a resolução de equações diferenciais parciais de Laplace, Poisson, Calor e Onda. O método de resolução aprendido foi o método de Fourier. Tal método consiste na separação de variáveis. Tendo-se uma função incógnita $u(x, y)$, propõe-se uma solução do tipo $H(x)G(y)$ e este termo é então trabalhado na equação diferencial, de forma a obter-se equações diferenciais ordinárias que podem ser resolvidas mais facilmente. As soluções muitas vezes envolvem o uso da série de Fourier, onde partes das EDPs são representadas como uma soma de senos e cossenos, por necessidade pelo uso do método.

Após isso, o primeiro problema proposto foi resolver uma destas equações numericamente, a escolhida foi a equação de Laplace com condições de fronteira de Dirichlet em duas dimensões, num quadrado 1×1, conforme se verifica abaixo:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{1}$$
$$u(x,0) = f_1(x),\ u(x,1) = f_2(x)$$
$$u(0,y) = g_1(y),\ u(1,y) = g_2(y)$$

A primeira etapa na resolução numérica desta equação de maneira é discretizá-la, e isto implica em tornar finito o número de pontos em cada dimensão, e representar a equação como operações mais simples e que o computador seja capaz de entender, operações diferentes da derivada, tais como soma, subtração e multiplicação. O método escolhido para realizar a discretização foi diferenças finitas. Este método basicamente faz a seguinte aproximação para a derivada:

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h} = \frac{d}{dx} f(x) \approx \frac{f_i - f_{i-1}}{\Delta x} \tag{2}$$

Os valores de f_i são valores discretizados da função $f(x)$, ou seja, $f_i = f(i\Delta x)$. Agora a distância entre dois pontos passa a ser de Δx, e daí a derivada se calcula como a diferença entre um ponto e outro, dividida pela distância entre eles.

Há vários tipos de diferenças finitas: progressivas, regressivas e centrais. Além disso, elas podem envolver mais de dois pontos, de forma a se obter diferentes ordens de erro. O erro também muda de acordo com quão grande é Δx, que está diretamente ligado ao número de pontos da malha. A malha é uma matriz bidimensional que contém pontos representando posições do espaço. Para resolver a equação de Laplace, utilizou-se diferenças centrais de segunda ordem, cujo erro é da ordem de Δx^2:

$$f''(x) \approx \frac{f(x+\Delta x) - 2f(x) - f(x-\Delta x)}{\Delta x^2} \tag{3}$$

Utilizando essa discretização da segunda derivada, chega-se na seguinte discretização da equação de Laplace, considerando $\Delta x = \Delta y$, ou seja, uma malha quadrada, e que a variável x varia de acordo com o índice i e a variável y com o índice j:

$$\frac{u_{i+1j} + u_{i-1j} + u_{ij+1} + u_{ij-1} - 4u_{ij}}{\Delta x^2} = 0 \tag{4}$$

Reordenando essa equação álgebrica para encontrar o valor de u_{ij}:

$$u_{ij} = \frac{u_{i+1j} + u_{i-1j} + u_{ij+1} + u_{ij-1}}{4} \tag{5}$$

A equação da fórmula 5 está na forma explícita, na qual um ponto central é obtido diretamente a partir dos pontos adjacentes. No entanto, quando a equação é aplicada ao ponto central, ele influencia no cálculo de outros, que faz com que esta fórmula deva ser aplicada muitas vezes, até se alcançar a convergência de todos os pontos do domínio. Entretanto, se essa equação é aplicada a todos os pontos do domínio, já se nota um problema, pois ela requere pontos adjacentes em todo local no qual ela é aplicada. Por isso, além da equação, necessita-se das condições de contorno para o problema, onde-se são determinados os valores nas fronteiras do domínio, de forma direta ou indireta, dependendo se as condições são de Dirichlet, Neumann ou uma mistura. No caso atual, as condições são de Dirichlet, que indica o valor de forma direta que a fronteira deve ter. O domínio escolhido é um quadrado, como já mencionado, e abaixo tem-se o exemplo de uma malha 6x6.

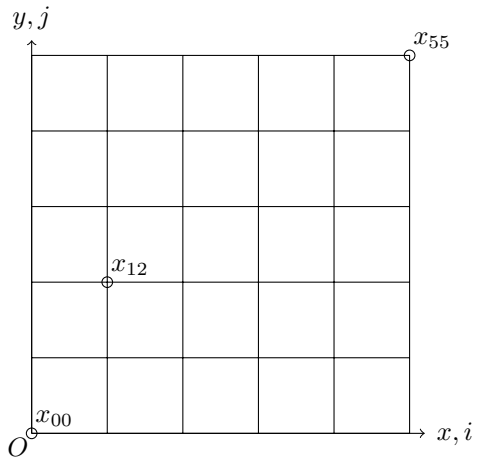

Figura 2. Malha padrão para discretizar Laplace e Poisson

Cada cruzamento de linha indica um ponto no domínio, e a Equação 5 será aplicada em quase todos estes postos, com exceção dos de fronteira, que são determinados pelas condições de Dirichlet.

Pelo fato do método ser iterativo, com um *loop* sendo repetido muitas vezes, até a convergência, é natural escolher um método de parada. Isto poderia ser: (1) escolher um tempo limite de execução, (2) ver a diferença entre duas matrizes após uma iteração completa em seus pontos internos, para analisar se a diferença entre elas é desprezível e se indica convergência, ou ainda (3) pode-se calcular os valor da equação de Laplace em cada ponto interno, que seria o melhor a se fazer, e analisar se a equação está dentro de uma faixa de erro desejada. A segunda maneira de se fazer isto foi escolhida na primeira parte, e foi uma difícil decisão escolher qual utilizar, pois ainda não se compreendia como aplicar o terceiro método.

Um valor que não está sendo utilizado agora, mas será posteriormente, é a distância Δx. Não é utilizada pelo fato do lado direito da equação de Laplace ser zero, e assim simplificou-se o problema.

$$\Delta x = 1/(n-1) \tag{6}$$

Agora, apresenta-se o algoritmo que se utilizou para resolver a equação:

input : Matriz com as condições de contorno de tamanho $n \times n$
output: Matriz com cada ponto interno satisfazendo a equação de Laplace e a fronteira igual à entrada

$STOP = 0$;
$err \leftarrow 10^{-6}$;
while $STOP \neq 1$ **do**
  $u^{\text{old}} \leftarrow u$;
  for $i \leftarrow 1$ **to** $n-2$ **do**
    for $j \leftarrow 1$ **to** $n-2$ **do**
      $u_{ij} \leftarrow (u_{i+1j} + u_{i-1j} + u_{ij+1} + u_{ij-1})/4$;
    end
  end
  $ERROR \leftarrow Norma(u^{\text{old}} - u)$;
  if $ERROR < err$ **then**
    $STOP \leftarrow 1$;
  end
end

Algoritmo 1: Resolvendo a equação de Laplace Iterativamente

O método de resolução apresentado no algoritmo acima é o explícito, no qual o valor de cada ponto no método é obtido diretamente dos pontos adjacentes, e se faz necessária uma resolução iterativa. Isso tem um custo de processamento razoável, no qual o tempo aumenta grandemente conforme o número de pontos n escolhido. Tendo em vista uma resolução mais rápida e direta, pode-se utilizar o método implícito.

Anteriormente, a partir da equação 4, isolou-se u_{ij} e então resolveu-se iterativamente nosso problema. Outra forma de se resolver o mesmo problema é pegar essa mesma equação e trocar os índices (i, j) por números de fato, obtendo várias equações, e isolando na parte da direita os termos de fronteira. Assim, para um sistema $n \times n$, dos n^2 pontos da malha dos índices $(0, 0) \rightarrow (n-1, n-1)$, $(n-1)^2$ pontos são internos, correspondendo aos índices de $(1, 1) \rightarrow (n-2, n-2)$. A seguir temos as equações para um sistema 5×5:

$$\begin{aligned}
u_{12} + u_{21} - 4u_{11} &= -u_{01} - u_{10} \\
u_{11} + u_{22} + u_{31} - 4u_{21} &= -u_{20} \\
u_{21} + u_{32} - 4u_{31} &= -u_{41} - u_{30} \\
u_{22} + u_{13} + u_{11} - 4u_{12} &= -u_{02} \\
u_{32} + u_{12} + u_{23} + u_{21} - 4u_{22} &= 0 \\
u_{22} + u_{33} + u_{31} - 4u_{32} &= -u_{42} \\
u_{23} + u_{12} - 4u_{13} &= -u_{03} - u_{14} \\
u_{33} + u_{13} + u_{22} - 4u_{23} &= -u_{24} \\
u_{23} + u_{32} - 4u_{33} &= -u_{43} - u_{34}
\end{aligned}$$

O conjunto de equações acima é um sistema linear, e é possível representá-lo na forma matricial $Ax = b$. O que torna este problema interessante é que a matriz A e b tem padrões muito bem definidos. A matriz A toma a seguinte forma:

$$\begin{pmatrix}
-4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4
\end{pmatrix}$$

Observando o padrão de formação da matriz A pentadiagonal e sabendo-se também obter o vetor de valores conhecidos b, tem-se um sistema linear para obter os pontos internos x. A solução é $x = A^{-1}b$.

Como se nota, a resolução é muito mais direta, mas o algoritmo para se resolver este sistema linear pode ser muito mais complexo. Não se preocupou em criar um algoritmo para isso, mas sim em utilizar um pronto da biblioteca Eigen. Mas há um problema para este método, que um sistema pode ficar extremamente grande com um n relativamente pequeno, tão grande, ao ponto de necessitar de mais memória do que o computador tem. Isso se resolve utilizando matrizes esparsas, que só salvam na memória elementos diferentes de zero. A matriz A é uma matriz esparsa, na qual a maioria de seus elementos é igual a 0. Também utilizaram-se métodos prontos para resolução de sistemas esparsos provenientes da Eigen.

V. Equação de Poisson

A equação de Poisson é muito similar a de Laplace, mas com uma diferença, que é a de haver uma função forçamento existente no lado direito da equação.

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) \tag{7}$$

Este foi o segundo problema proposto, e o fato de ter essa função $f(x, y)$ complica o problema porque muita coisa que antes era desconsiderada com o 0 na equação de Laplace não é mais ignorado, mas sim levado em consideração. No caso explícito, tem-se a seguinte discretização obtida:

$$u_{ij} = \frac{1}{4}[(u_{i+1j} + u_{i-1j} + u_{ij+1} + u_{ij-1}) - \Delta x^2 f_{ij}] \tag{8}$$

O algoritmo iterativo para este caso é o mesmo anterior, substituindo a equação 5 por 8 no algoritmo.

Para o caso implícito, o problema é muito similar, bastando mudar o vetor b, que passa a apresentar termos $\Delta x^2 f_{ij}$.

A equação de Poisson realmente complicou quando se preparou o algoritmo numa versão mais similar àquela que é utilizada em Navier Stokes. Esta versão resolve o problema de Poisson com as quatro fronteiras com condições de Neumann, isto é:

$$\frac{\partial u}{\partial n} = f \tag{9}$$

Isto resulta em maior processamento devido à esta equação gerar condições na fronteira que não são inicialmente fixas, mas que também são incógnitas. O problema foi resolvido e é apresentado na seção de resultados.

VI. Equação de Navier Stokes

Considerações gerais

Após resolver-se as equações de Laplace e Poisson com condições de Dirichlet, pelos métodos explícito e implícito, tem-se já uma base para partir para a próxima etapa: Navier Stokes. Essa equação tem muitas complicações, e está levando ainda certo tempo para ter-se um programa que a resolva, e esteja de fato livre de bugs. A complexidade do programa é muitas maior, tem várias etapas, e tem ainda uma evolução no tempo que não foi lidada nos problemas anteriores. A equação de Navier Stokes é a seguinte:

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}\right) = -\nabla p + \mu \nabla^2 \mathbf{v} + \mathbf{f} \tag{10}$$

$$\nabla \cdot \mathbf{v} = 0 \tag{11}$$

A discretização não é simples e é um pouco diferente das que foram feitas anteriormente. Em primeiro lugar, tem-se de saber que a discretização de Navier Stokes que será feita aqui é a fórmulação em variáveis primitivas. Isso quer dizer que o problema será trabalhado com as variáveis originais de velocidade u, v (em x e y) e pressão p. Outra forma de resolver é usando uma formulação com a vorticidade, que não será trabalhada aqui.

No que se aprendeu lendo sobre tal formulação do processo de resolução em variáveis primitivas, sabe-se que utilizando uma malha como a da Figura 2, ocorrem modos espúrios de pressão que causam um perfil de velocidades que contém oscilações não esperadas no resultado do problema. O resultado, obtido com tal malha, desta forma, não é satisfatório, e para resolver este problema usa-se uma malha escalonada.

A malha escalonada é uma malha na qual valores guardados num ponto se referem à informação em um outro

ponto, próximo. No ponto $P = (i, j)$ são salvos cinco valores: força horizontal, força vertical, pressão, velocidade horizontal e vertical. Essas são representadas pelos símbolos f_x, f_y, p, u e v. Um elemento da malha escalonada é representado na figura a seguir, na qual as forças não estão mostradas, mas elas são localizadas nas mesmas posições que as velocidades.

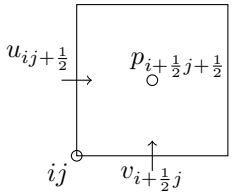

Figura 3. Elemento de malha escalonada

Diferente dos problemas de Laplace e Poisson, no qual um elemento u_{ij} que estava salvo no computador na posição ij realmente é o elemento u_{ij}, agora o elemento u_{ij} salvo na memória do computador é fisicamente o elemento $u_{ij+\frac{1}{2}}$, enquanto o elemento v_{ij} se refere ao elemento $v_{i+\frac{1}{2}j}$. De maneira similar ocorre para as forças. Para a pressão, p_{ij} corresponde ao ponto $p_{i+\frac{1}{2}j+\frac{1}{2}}$. Isso influencia diretamente na discretização, e na posterior criação dos gráficos.

Uma malha escalonada completa é apresentada na Figura a seguir:

Na malha escalonada da Figura 4, tem-se ainda uma nova adição: uma fronteira imaginária. Quando se discretiza a equação de Navier Stokes, faz-se ainda necessários outros pontos além do internos e fronteira. As equações apresentadas são discretizadas em domínios similares ao de cima, onde a única coisa que muda é o tamanho do domínio, que pode ser maior.

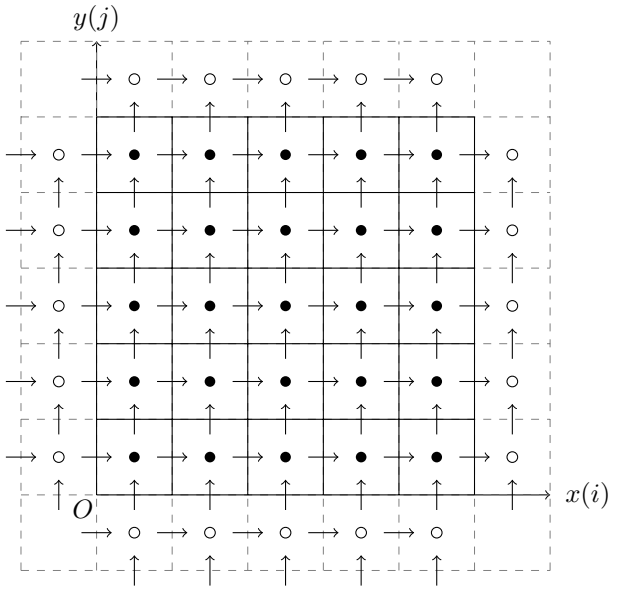

Figura 4. Malha escalonada. Círculos são pontos internos e de fronteira enquanto circunferências indicam fronteira imaginária adicionada

O problema de Navier Stokes é trabalhado em três etapas principais:

1) Desconsiderando a pressão
2) Obtendo a pressão
3) Avanço no tempo

Inicia-se agora a descrição da metodologia empregada em tais etapas.

A. Desconsiderando a pressão

A primeira etapa consiste literalmente de desconsiderar a pressão. Há um termo de pressão na Equação de Navier Stokes (eq. 10) que é o termo ∇p, chamado de gradiente de pressão. Elimina-se ele da equação e a partir dessa simplificação, faz-se a discretização da equação. Pelo fato da malha ser escalonada, termos de velocidade estão em pontos diferentes da malha, e há momentos nos quais é necessário elementos de velocidade que não estão salvos na malha, fazendo-se necessária obter uma média de velocidades próximas para obter as velocidades necessárias na discretização.

$$\begin{aligned}\mathbf{v}^s_{ij} &= (u_{i+1j} + u_{i-1j} + u_{ij+1} + u_{ij-1})\mathbf{i} \\ &+(v_{i+1j} + v_{i-1j} + v_{ij+1} + v_{ij-1})\mathbf{j} \qquad (12)\\ \mathbf{v}^t_{ij} &= 0.25(u_{ij} + u_{i+1j} + u_{i+1j-1} + u_{ij-1})\mathbf{i} \\ &+0.25(v_{ij} + v_{i-1j} + v_{i-1j+1} + v_{ij+1})\mathbf{j} \qquad (13)\end{aligned}$$

$$\begin{aligned}u^*_{ij} &= \left[\frac{\mu}{\rho}\left(\frac{u^s_{ij} - 4u_{ij}}{\Delta x^2}\right) + \frac{1}{\rho}f_{x,ij}\right] + u_{ij} \\ &- \left[u_{ij}\frac{u_{i+1j} - u_{i-1j}}{2\Delta x} - v^t_{ij}\frac{u_{ij+1} - u_{ij-1}}{2\Delta x}\right]\Delta t \qquad (14)\end{aligned}$$

$$\begin{aligned}v^*_{ij} &= \left[\frac{\mu}{\rho}\left(\frac{v^s_{ij} - 4v_{ij}}{\Delta x^2}\right) + \frac{1}{\rho}f_{y,ij}\right] + v_{ij} \\ &- \left[u^t_{ij}\frac{v_{i+1j} - v_{i-1j}}{2\Delta x} - v_{ij}\frac{v_{ij+1} - v_{ij-1}}{2\Delta x}\right]\Delta t \qquad (15)\end{aligned}$$

Com isto, tem-se:

$$\mathbf{v}^* = (u^*, v^*)$$

As equações para u^*_{ij} e v^*_{ij} são utilizadas para obter a solução de velocidades da equação de Navier Stokes sem a pressão, portanto, não é a solução real, mas está relacionada com a solução.

B. Obtendo a pressão

A forma de obter a pressão é utilizando as Equações 10 e 11. Primeiro é obtido o divergente da Eq. 10 e então, utilizando a Equação 11, se simplifica a equação obtida, obtendo-se o seguinte:

$$\nabla^2 p = \frac{\rho}{\Delta t}\nabla \cdot \mathbf{v}^* = \frac{\rho}{\Delta t}\left(\frac{\partial u^*}{\partial x} + \frac{\partial v^*}{\partial y}\right) \qquad (16)$$

O termo da direita na equação acima será chamado de DIV, e é discretizado na equação seguinte:

$$DIV_{ij} = \frac{\rho}{\Delta t}\left(\frac{u^*_{i+1j} - u^*_{ij}}{\Delta x} + \frac{v^*_{ij+1} - v^*_{ij}}{\Delta x}\right) \quad (17)$$

A equação que estamos tratando é a de Poisson, e resultará na pressão desejada para resolvermos o problema de Navier Stokes. A Equação 8 é usada para pontos internos da malha. Já para a fronteira, a situação muda em relação à anterior. A partir da equação de Navier Stokes podem ser obtidas as condições de contorno para a pressão. Veremos que tais condições são todas de Neumann, fazendo com que não haja uma única solução de pressão, mas como na equação de Navier Stokes só aparece gradiente de pressão, qualquer solução de pressão pode ser usada para então obter esse gradiente. Destrinchando a Equação 10 em duas coordenadas, tem-se:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left(\frac{\partial^2 u}{\partial^2 x} + \frac{\partial^2 u}{\partial^2 y}\right) + f_x \quad (18)$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left(\frac{\partial^2 v}{\partial^2 x} + \frac{\partial^2 v}{\partial^2 y}\right) + f_y \quad (19)$$

Se fizermos uma análise dos pontos na parede da esquerda ou direita, e aplicarmos as condições de impenetrabilidade e não-deslizamento, que dizem que $u = 0$ e $v = 0$ na parede, temos a simplificação da equação, pois vários termos se tornam zero. As equações para as duas paredes verticais é a seguinte:

$$\left.\frac{\partial p}{\partial x}\right|_{\text{parede}} = \rho\nu\left.\frac{\partial^2 u}{\partial x^2}\right|_{\text{parede}} + \rho f_x \quad (20)$$

Para as paredes que ficam na horizontal tem-se:

$$\left.\frac{\partial p}{\partial y}\right|_{\text{parede}} = \rho\nu\left.\frac{\partial^2 v}{\partial y^2}\right|_{\text{parede}} + \rho f_y \quad (21)$$

As discretizações das Equações 20 e 21 devem ser feitas considerando-se a malha escalonada, da forma como os dados são salvos no computador. O seguinte esquema mostra a origem O do nosso domínio e o ponto no computador que representa a origem imaginária x_{00}:

Esta malha é mostrada como na figura acima porque em C++ não é possível ter elementos de matrizes com índices negativos, como em Fortran. Discretizando as equações e isolando os termos da fronteira imaginária, que se deseja obter, tem-se o seguinte:

$$p_{i-1j}|_{i=1} = p_{ij} - \frac{\mu}{\Delta x}\left(2u_{ij} - 5u_{i+1j} + 4u_{i+2j} - u_{i+3j}\right) - \rho\Delta x f_{ij} \quad (22)$$

$$p_{i+1j}|_{i=n} = p_{ij} + \frac{\mu}{\Delta x}\left(2u_{ij} - 5u_{i-1j} + 4u_{i-2j} - u_{i-3j}\right) + \rho\Delta x f_{ij} \quad (23)$$

$$p_{ij-1}|_{j=1} = p_{ij} - \frac{\mu}{\Delta x}\left(2v_{ij} - 5v_{ij+1} + 4u_{ij+2} - u_{ij+3}\right) - \rho\Delta x f_{ij} \quad (24)$$

$$p_{ij+1}|_{j=n} = p_{ij} + \frac{\mu}{\Delta x}\left(2v_{ij} - 5v_{ij-1} + 4u_{ij-2} - u_{ij-3}\right) + \rho\Delta x f_{ij} \quad (25)$$

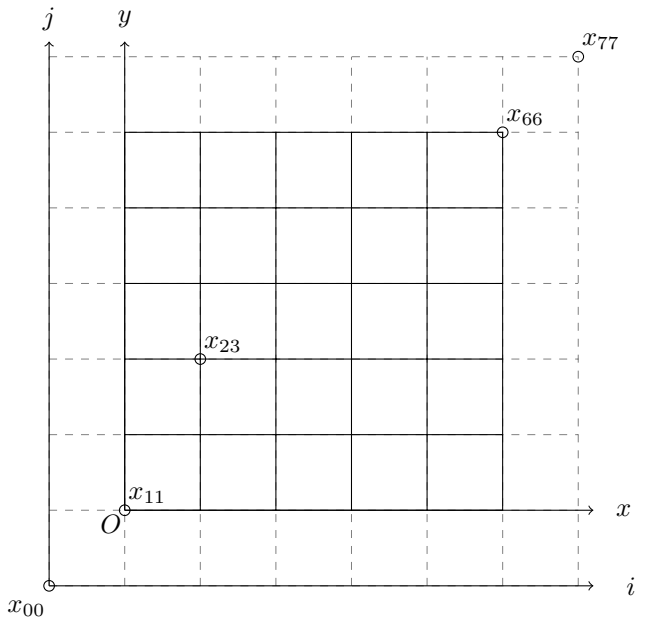

Figura 5. Mostra dos índices dos pontos do domínio e pontos imaginários

Tais equações devem então ser utilizadas no algoritmo para resolver a equação de Poisson, lembrando que essas condições são de Neumann, e então a fronteira não é fixa, mas irá variar até o momento que o algoritmo iterativo utilizado convergir.

C. Avanço no tempo

Agora faz-se a relação entre a velocidade obtida ignorando-se a pressão com o gradiente da pressão obtida no item anterior para então obter-se o próximo passo de tempo da velocidade. Finalmente, tem-se:

$$\frac{\mathbf{v}^{n+1} - \mathbf{v}^*}{\Delta t} = -\frac{1}{\rho}\nabla p \quad (26)$$

Expandindo para as equações escalares, tem-se:

$$u^{n+1} = u^* - \frac{\Delta t}{\rho}\frac{\partial p}{\partial x} \quad (27)$$

$$v^{n+1} = v^* - \frac{\Delta t}{\rho}\frac{\partial p}{\partial y} \quad (28)$$

Basta discretizar os termos das derivadas parciais agora:

$$p_{x,ij} = \frac{p_{ij} - p_{i-1j}}{\Delta x} \quad (29)$$

$$p_{y,ij} = \frac{p_{ij} - p_{ij-1}}{\Delta x} \quad (30)$$

E finalmente tem-se a solução para o tempo $n+1$ a partir do tempo passado n:

$$u^{n+1} = u^* - \frac{\Delta t}{\rho}p_x \quad (31)$$

$$v^{n+1} = v^* - \frac{\Delta t}{\rho}p_y \quad (32)$$

Cada uma das varíaveis acima é uma matriz. Com isso, tem-se as fórmulas necessárias para resolver a equação de Navier Stokes. No entanto, mais processos são necessários computacionalmente, é necessário escolher uma boa condição de parada para as etapas que consideram convergência, trabalhar com arquivos para salvar os resultados, entre outras coisas, que realmente complicam a solução do problema.

VII. Resultados

A. Equação de Laplace e Poisson com fronteira de Dirichlet

A equação de Laplace foi resolvida pelo método explícito iterativo, algumas vezes usando paralelismo e outras não, e pelo método ímplicito. Serão mostradas várias figuras com os resultados destas etapas.

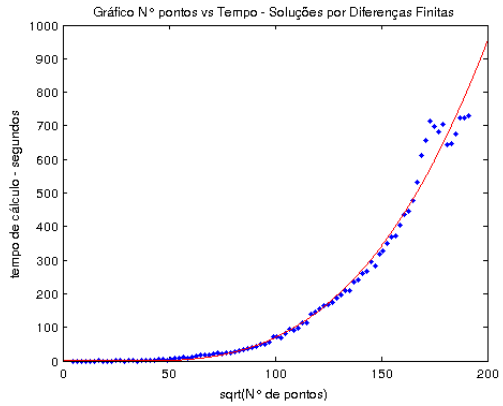

Figura 6. Pontos vs Tempo de Cálculo - 1 thread - Método Explícito

Nas Figuras 6 e 7 tem-se uma visão geral do tempo que o programa criado gasta para resolver a equação de Laplace com condições de Dirichlet, pelo método explícito iterativo. Os conjuntos de pontos obtidos em cada gráfico foram melhor aproximados por polinômios cúbicos. A diferença entre os dois se vê no tempo de cálculo quando o número de pontos passa de 100. Pelo fato de serem polinômios cúbicos, ambos crescem muito rápido com o número de pontos, mas o cálculo com 4 threads foi mais rápido pelo fato de estar usando mais poder de processamento do computador nos cálculos.

As Figuras 8 a 15 são um conjunto de resultados da equação de Laplace com condições de fronteira de Dirichlet, resolvidas pelo método iterativo explícito, tanto com 1 thread como 4 threads. As condições de contorno em todas são as mesmas, como se pode notar pelo formato similar de cada uma. Os lados tem valores de condições de Dirichlet que são $u(x,0) = 0$, $u(x,1) = 50$, $u(0,y) = 75$ e $u(1,y) = 100$. Nota-se que cada gráfico se parece mais com uma função contínua, sendo a malha 500×500 praticamente toda contínua em questão dos pontos coloridos, que parecem estar em todo local. Para se ter estes resultados próximos do contínuo, o gasto computacional é enorme, conforme se nota nas descrições das figuras.

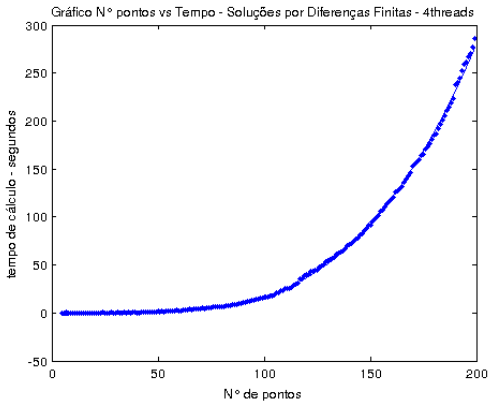

Figura 7. Pontos vs Tempo de Cálculo - 4 threads - Método Explícito

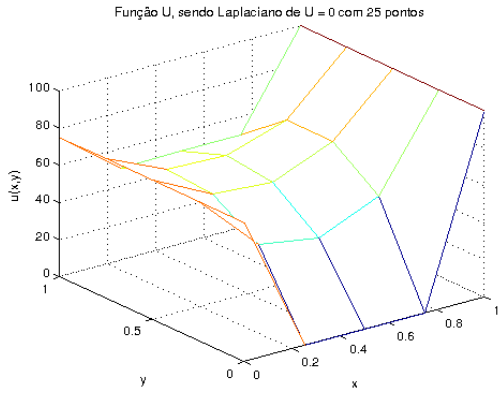

Figura 8. Malha 5x5 - 1 thread - Tempo: 0s - Método Explícito

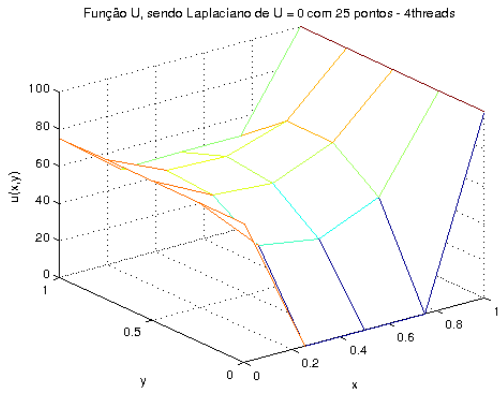

Figura 9. Malha 5x5 - 4 threads - Tempo: 0s - Método Explícito

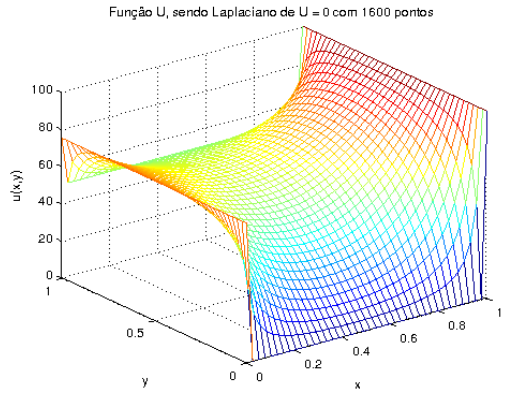

Figura 10. Malha 40x40 - 1 thread - Tempo: 2s - Método Explícito

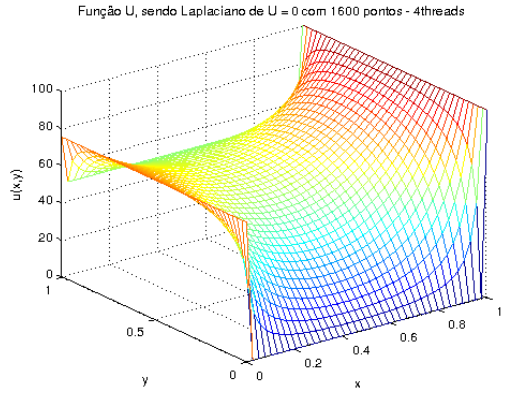

Figura 11. Malha 40x40 - 4 threads - Tempo: 0s - Método Explícito

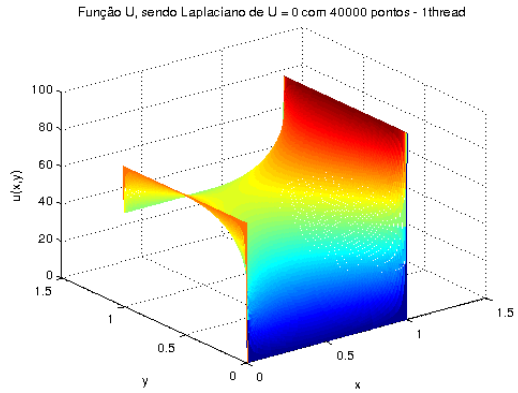

Figura 12. Malha 200x200 - 1 thread - Tempo: 1235s - Método Explícito

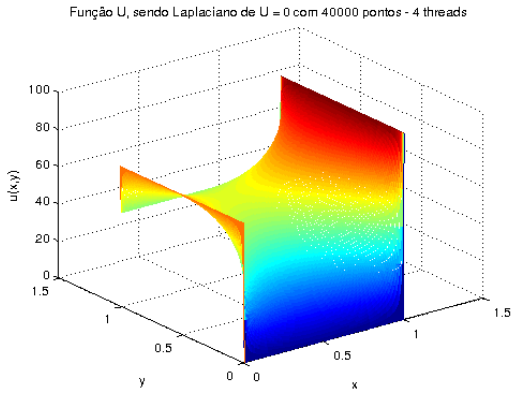

Figura 13. Malha 200x200 - 4 threads - Tempo: 274s - Método Explícito

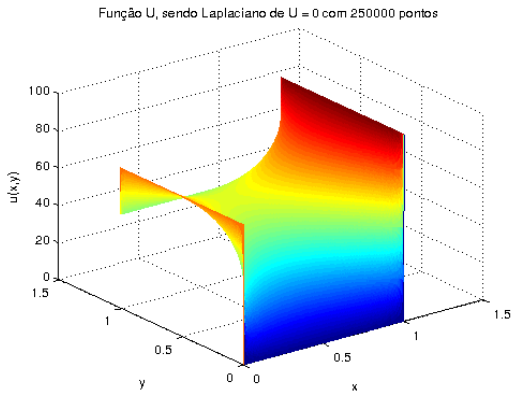

Figura 14. Malha 500x500 - 1 thread - Tempo: 36083s - Método Explícito

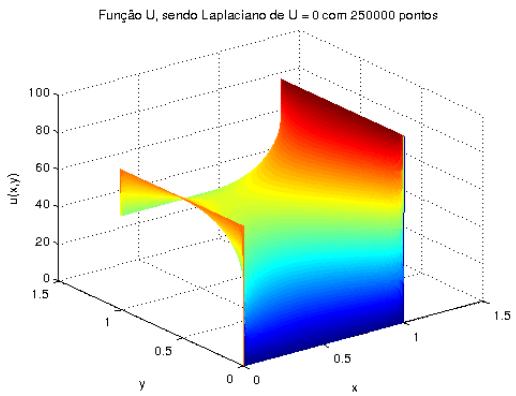

Figura 15. Malha 500x500 - 4 threads - Tempo: 9610s - Método Explícito

O uso de paralelismo começou por causa do elevado custo computacional para malhas maiores e, sabendo-se que computadores atuais tem múltiplos threads, fez-se uso deste recurso para analisar como ficava a relação entre tempo e número de pontos[5]. O mesmo resultado foi obtido

[5]Pontos: o eixo de número de pontos nos gráficos do método explícito na verdade é a ordem da malha, que é a raiz quadrada do número de pontos.

por meio do uso de múltiplos threads que no caso não-paralelo.

No caso do método ímplicito, utilizando matrizes esparsas, os resultados foram espantosamente mais rápidos. Na maior malha que foi calculada no método explícito, foram gastas 10h[6] no cálculo com 1 thread e 2h40min[7] no cálculo com 4 threads. No método ímplicito com matrizes esparsas, o mesmo resultado foi obtido em menos de 1min. O método ímplicito não é utilizado em Navier Stokes devido ao problema da pressão nesta equação ter fronteiras de Neumann, e o método ímplicito se torna muito complicado. Os resultados para Poisson são similares, só mudando o tempo no caso do método explícito.

B. Solução da Equação de Poisson com fronteira de Neumann pelo método iterativo

São apresentados os resultados para o problema abaixo:

$$\nabla^2 u = 1 \tag{33}$$

$$\frac{\partial u}{\partial n} = 0 \,\text{na fronteira, exceto} \tag{34}$$

$$\frac{\partial u(x,0)}{\partial y} = 2x \tag{35}$$

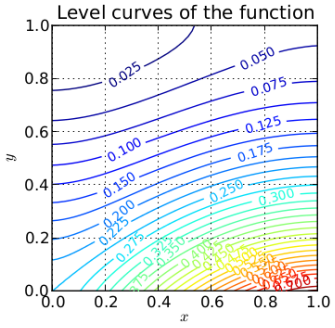

Figura 16. Poisson com 4 fronteiras de Neumann - Linhas de contorno

[6] 36083s
[7] 9610s

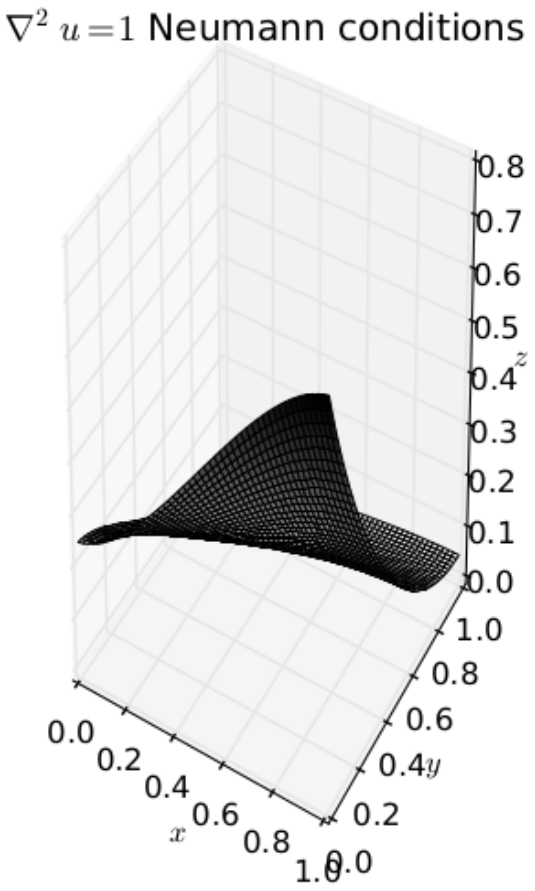

Figura 17. Poisson com 4 fronteiras de Neumann

C. Equação de Navier Stokes

Não há resultados para a equação de Navier Stokes, pois está muito complicado descobrir os bugs ainda existentes no programa. Muita coisa já foi corrigida, mas não há resultado nenhum.

VIII. Conclusão

O problema proposto inicialmente ainda não foi alcançado, que é trabalhar com a instabilidade de Saffman-Taylor com fluido magnético e, ou Anisotrópico. Antes de trabalhar com tal problema, teve-se uma etapa inicial que demandou um certo trabalho. Essa primeira etapa deste projeto consistiu então do estudo da resolução de equações diferenciais parciais, por meio de sua discretização e codificação em um programa de computador.

Foi bem sucedida a resolução das equações de Laplace e Poisson numa malha quadrada, pelos métodos implícito e explícito, usando diferenças finitas. No caso de Poisson, ainda teve-se a resolução bem sucedida com condições de fronteira de Neumann, que seria essencial para trabalhar com a pressão na equação de Navier Stokes.

Chegando na equação de Navier Stokes, houve muitas equações a serem discretizadas, e a malha escalonada foi algo difícil de ser plenamente entendido, tendo muitas vezes ocorrido erros no caminho por causa disso.

O trabalho foi um aprendizado razoavelmente complexo, e que teve muitas consequências positivas. As habilidades de programação de programas numéricos pelo aluno aumentaram bastante, na realidade, de qualquer programa, pelo fato de se ter treinado bastante o uso do git e da ferramenta de compilação cmake. Apesar da programação ainda ser um problema, pela complexidade do programa, o código[8] está muito mais organizado do que se esperava, e pra isso ele foi refeito algumas vezes.

O programa de Navier Stokes ainda está em fase de desenvolvimento, e se espera ter resultados em breve, com mais um tempo de dedicação e, futuramente, continuar trabalhando com esse programa para um problema físico com ferrofluido.

Agradecimentos

Primeiramente agradeço a Deus pela oportunidade de ter feito parte deste trabalho. Agradeço também ao professor Yuri Dumaresq pelas orientações e ânimo no ensino que eu vi nele e que também me animaram na resolução destes problemas numéricos. Agradeço também ao CNPq pela bolsa de incentivo à pesquisa.

Referências

[1] Getting Started with Ubuntu 10.10

[8] Código está acessível online em https://github.com/ataias/ff