Desenvolvimentos matemáticos e numéricos em escoamentos bifásicos aplicados a processos de refino

Documentação

Adenilso Silva Simão
Antônio Castelo Filho
Fabricio Simeoni de Sousa
Leandro Franco de Souza

Sumário

Capítulo 1

Documentação científica

Este capítulo descreve os métodos utilizados para resolução numérica de equações diferenciais parciais (em especial as equações de Navier-Stokes) em uma malha hierárquica do tipo "HiG-tree". São descritos detalhes sobre este tipo de malha, bem como os métodos utilizados para discretização por diferenças finitas generalizadas e interpolações necessárias entre os diferentes níveis de refinamento. Finalmente são apresentados os mecanismos necessários para a resolução de uma equação de Poisson em um domínio genérico.

1.1 Malhas hierárquicas

O desenvolvimento dos métodos apresentados neste documento é baseado em uma malha cartesiana do tipo hierárquica, contendo elementos de tamanhos diferentes. De agora em diante, tais malhas serão denotadas como `HiG` (do inglês, *"Hierarchical Grid"*) e a estrutura de dados utilizada para representá-la será denominada `HiG-Tree`. Tal estrutura é baseada em subdivisões espaciais recursivas arbitrárias, e pode ser representada por um árvore $m-$tree. Um exemplo clássico deste tipo de malha são quadtree em 2D e octree em 3D.

A malha `HiG` é baseada em uma matriz regular de elementos de mesmo tamanho, e, a partir destes, subdivisões arbitrárias podem ser definidas de forma recursiva, obtendo-se uma malha irregular (não-estruturada). Um exemplo desta malha e sua respectiva estrutura de árvore podem ser vistos na figura 1.1. Apesar de o exemplo da figura 1.1 ser ilustrado em duas dimensões, esta representação é genérica o suficiente para ser utilizada em qualquer número de dimensões.

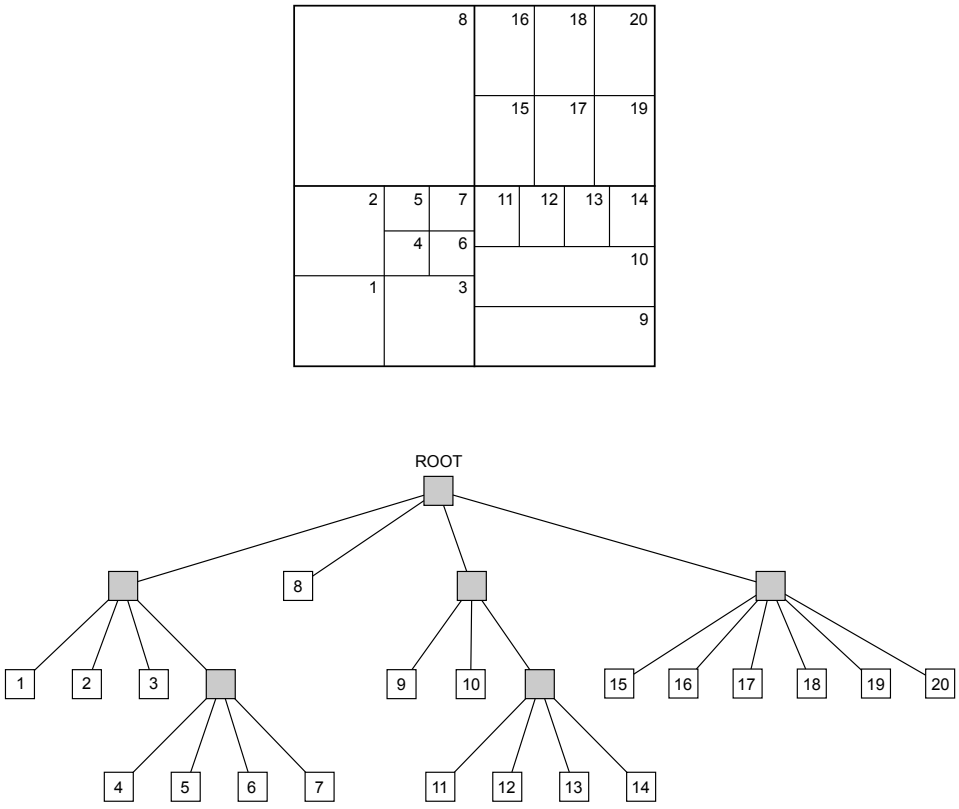

Figura 1.1: Exemplo de uma malha tipo HiG e sua representação na forma de árvore (HiG-Tree).

1.2 Diferenças finitas generalizadas

O método adotado para discretização de equações diferenciais é o método de diferenças finitas. Tal método é baseado na expansão em série de Taylor da função a ser aproximada e de suas derivadas, cuja omissão de termos de alta ordem desta série resulta em fórmulas adequadas para as aproximações.

Um exemplo clássico é a discretização de uma equação de Poisson do tipo

$$\nabla^2 u = f,$$

com condições de contorno adequadas. Simplificando o domínio por um retângulo em um espaço de dimensão dois $\Omega = [a, b] \times [c, d]$, e utilizando uma malha cartesiana regular (do tipo grid) com espaçamentos δx e δy, obtém-se a clássica fórmula do laplaciano de 5 pontos, dada por

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\delta y^2} + \mathcal{O}(\delta x^2 + \delta y^2) = f_{i,j},$$

ou ainda, desprezando-se os termos de ordem 2,

$$\frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{\delta x^2} + \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{\delta y^2} = f_{i,j},$$

onde $U_{i,j}$ representa uma aproximação para a função $u_{i,j}$, avaliada em $(x_i, y_j) = (a + i\delta x, b + j\delta y)$, e $f_{i,j} = f(x_i, y_j)$. Com esta fórmula, mais as condições de contorno, é possível montar um sistema da forma

$$L\mathbf{u} = \mathbf{f},$$

cuja a solução é o vetor de aproximações $\mathbf{u} = (U_{1,1}, U_{1,2}, \ldots)$ para a função u original, avaliada nos vértices da malha cartesiana.

Como visto na seção anterior, o objetivo é utilizar diferenças finitas em uma malha cartesiana irregular, então tal aproximação deve ser modificada, pois uma discretização por um laplaciano de 5 pontos resultaria o envolvimento de valores não conhecidos no interior da malha. Desta forma é preciso realizar um procedimento de interpolação que seja genérico o suficiente para englobar qualquer configuração de células (como no exemplo da figura 1.1), e que seja preciso o suficiente para não deteriorar a ordem de convergência do método numérico.

Uma clara desvantagem da utilização de malhas irregulares como as exemplificadas anteriormente é que uma numeração do tipo matricial (i, j) não é mais possível, sendo necessária a definição de uma enumeração dos nós válidos da malha. Tal enumeração é fornecida pela estrutura `HiG-Tree`, contudo, diferentes

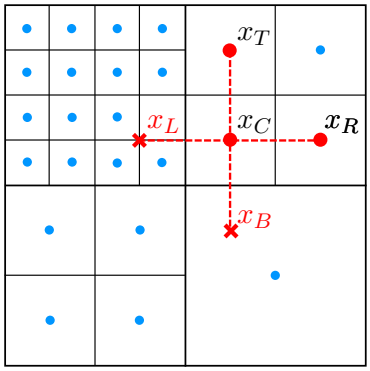

Figura 1.2: Exemplo de um estêncil resultando em pontos a serem interpolados (x_L e x_B).

formas de enumeração podem ser implementadas, com impacto direto na estrutura da matriz do sistema resultante, e consequentemente no desempenho dos métodos numéricos utilizados para resolvê-lo.

Considere o caso ilustrado na figura 1.2, onde um estêncil de cinco pontos é utilizado para discretizar as derivadas no ponto x_C. É fácil notar que, devido a configuração irregular da malha, não existe nó da malha que armazene um valor de u para os pontos x_L e x_B, ou seja, não existem incógnitas U_L e U_B para o sistema linear. Neste exemplo, é necessário obter então uma interpolação dos nós vizinhos, pertencentes à malha, mais próximos às incógnitas não existentes. Digamos que uma interpolação adequada resulte na seguinte expressão para U_L

$$U_L \approx w_1 U_{j_1} + w_2 U_{j_2} + \ldots + w_n U_{j_n},$$

sendo $U_{j_1}, U_{j_2}, \ldots, U_{j_n}$ valores conhecidos na malha. Escrevendo a fórmula usual do laplaciano de 5 pontos para o ponto x_C, obtém-se uma equação para U_C, a saber,

$$\frac{U_L - 2U_C + U_R}{\delta x^2} + \frac{U_T - 2U_C + U_B}{\delta y^2} = f_C,$$

que pode ainda ser reescrita como

$$a_L U_L + a_R U_R + a_T U_T + a_B U_B + a_C U_C = f_C,$$

onde

$$a_L = \frac{1}{\delta x^2}, \quad a_R = \frac{1}{\delta x^2}, \quad a_T = \frac{1}{\delta y^2}, \quad a_B = \frac{1}{\delta y^2}, \quad a_C = -\frac{2}{\delta x^2} - \frac{2}{\delta y^2}.$$

Substituindo o valor interpolado de U_L na equação, obtém-se

$$a_L \left(w_1 U_{j_1} + w_2 U_{j_2} + \ldots + w_n U_{j_n} \right) + a_R U_R + a_T U_T + a_B U_B + a_C U_C = f_C,$$

gerando-se novos coeficientes e uma equação. Note que o mesmo deve ser feito para U_B, para que a equação envolva finalmente somente valores conhecidos da malha. Note que este processo pode ser repetido arbitrariamente para qualquer valor que ainda não esteja definido na malha, e que incógnitas repetidas podem ser agrupadas em um único coeficiente multiplicativo. Desta forma, obtém-se uma expressão que é uma combinação linear formada somente por incógnitas verdadeiras, para valores definidos na malha, que define a equação para a incógnita U_C no sistema linear.

A próxima seção explica como obter uma interpolação que seja suficientemente genérica e precisa, e como obter os pesos $w_{j_1}, w_{j_2}, \ldots, w_{j_n}$ necessários.

1.3 Interpolações

1.3.1 Aproximação polinomial

Um dos objetivo do método descrito aqui é obter uma aproximação para o valor de uma propriedade em um dado ponto (x, y, z) em função dos valores da propriedade em m outros pontos $(x_1, y_1, z_1), ..., (x_m, y_m, z_m)$. Os valores nos m pontos não são conhecidos a priori. Denotamos o valor em um ponto (x, y, z) por $P(x, y, z)$ um polinômio de grau no máximo r,

$$P(x, y, z) = \alpha_{i_1, j_1, k_1} x^{i_1} y^{j_1} z^{k_1} + \cdots + \alpha_{i_n, j_n, k_n} x^{i_n} y^{j_n} z^{k_n} = c^t \alpha,$$

onde $0 \leq i_s + j_s + k_s \leq r$, para $s = 1, \ldots, n$. Observe que poderia ser uma combinação de funções não polinomiais.

Assim, o método deve obter os coeficientes w_i, para $1 \leq i \leq m$, de forma que

$$P(x, y, z) = \sum_{i=1}^{m} w_i \; P(x_i, y_i, z_i) = w^t b.$$

Assim,

$$b = \begin{pmatrix} P(x_1, y_1, z_1) \\ \vdots \\ P(x_m, y_m, z_m) \end{pmatrix} = \begin{pmatrix} x_1^{i_1} y_1^{j_1} z_1^{k_1} & \cdots & x_1^{i_n} y_1^{j_n} z_1^{k_n} \\ \vdots & & \vdots \\ x_m^{i_1} y_m^{j_1} z_m^{k_1} & \cdots & x_m^{i_n} y_m^{j_n} z_m^{k_n} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} =$$

$$= \begin{pmatrix} a_1^t \\ \vdots \\ a_m^t \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} = A\ \alpha$$

Logo,

$$w^t\ b = w^t\ A\ \alpha = c^t\ \alpha$$

e uma solução é dada por $w^t\ A = c^t$, ou

$$A^t\ w = c.$$

Se $m \geq n$ e A tem posto máximo n, w pode ser obtido pelo método dos mínimos quadrados.

1.3.2 Aproximação pelo Método dos Mínimos Quadrados

Fazendo a decomposição QR de A, temos:

$$\begin{aligned} A_{m\times n} &= \begin{pmatrix} A_{1n\times n} \\ A_{2m-n\times n} \end{pmatrix} = Q_{m\times m} R^*_{m\times n} \\ &= Q_{m\times m} \begin{pmatrix} R_{n\times n} \\ 0_{m-n\times n} \end{pmatrix} = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix} \end{aligned}$$

Assim,

$$w = A^+ c^t = A\left(A^t A\right)^{-1} c^t = QR^*\left(R^{*t} Q^t Q R^*\right)^{-1} c^t = QR^*\left(R^t R\right)^{-1} c^t =$$

$$= QR^* R^{-1} (R^t)^{-1} c^t = Q \begin{pmatrix} R \\ 0 \end{pmatrix} R^{-1} (R^t)^{-1} c^t = Q \begin{pmatrix} (R^t)^{-1} \\ 0 \end{pmatrix} c^t$$

Fazendo $u = Q^t\ w$ e $u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$, temos:

$$u = Q^t w = \begin{pmatrix} (R^t)^{-1} \\ 0 \end{pmatrix} c^t$$

$$u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} (R^t)^{-1} \\ 0 \end{pmatrix} c^t$$

$$\begin{cases} u_1 &= (R^t)^{-1} c^t \\ u_2 &= 0 \end{cases}$$

Daí, resolvendo o sistema triangular inferior $R^t u_1 = c^t$, temos $w = Qu$, com $u_2 = 0$.

Resumindo, são dados A e c^t e queremos w, então são feitos três passos:

1. Fazer a decomposição QR de A, isto é, $A = Q\ R^*$;
2. Resolver o sistema triangular inferior $R^t u_1 = c^t$;
3. Fazer $w = Qu$ com $u_2 = 0$.

Observe que para o caso de $m = n$, a aproximação polinomial P será o polinômio interpolador.

1.3.3 Cálculo dos coeficientes

Seja

$$C = \begin{pmatrix} A_1 & B_1 \\ A_2 & B_2 \end{pmatrix} = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \begin{pmatrix} R & 0 \\ 0 & I \end{pmatrix} = \begin{pmatrix} Q_{11}R & Q_{12} \\ Q_{21}R & Q_{22} \end{pmatrix}.$$

Desde que,

$$I_m = Q^t Q = \begin{pmatrix} Q_{11}^t & Q_{21}^t \\ Q_{12}^t & Q_{22}^t \end{pmatrix} \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} =$$

$$= \begin{pmatrix} Q_{11}^t Q_{11} + Q_{21}^t Q_{21} & Q_{11}^t Q_{12} + Q_{21}^t Q_{22} \\ Q_{12}^t Q_{11} + Q_{12}^t Q_{22} & Q_{12}^t Q_{12} + Q_{22}^t Q_{22} \end{pmatrix} = \begin{pmatrix} I_n & 0 \\ 0 & I_{m-n} \end{pmatrix},$$

Temos que,

$$A^t B = \begin{pmatrix} A_1^t & A_2^t \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} = A_1^t B_1 + A_2^t B_2 = R\left(Q_{11}^t Q_{12} + Q_{21}^t Q_{22}\right) = 0,$$

$$B^t B = \begin{pmatrix} B_1^t & B_2^t \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} = B_1^t B_1 + B_2^t B_2 = Q_{12}^t Q_{12} + Q_{22}^t Q_{22} = I_{m-n},$$

e

$$C^t C = \begin{pmatrix} A_1^t & A_2^t \\ B_1^t & B_2^t \end{pmatrix} \begin{pmatrix} A_1 & B_1 \\ A_2 & B_2 \end{pmatrix} = \begin{pmatrix} R^t R & 0 \\ 0 & I \end{pmatrix}.$$

Logo $Det(C) = \pm Det(R) = R_{1,1} \cdots R_{n,n}$, e se A tem posto máximo $R_{i,i} \neq 0$ e

$$C^t\ w = \begin{pmatrix} A_1^t & A_2^t \\ B_1^t & B_2^t \end{pmatrix} w = \begin{pmatrix} c \\ 0 \end{pmatrix},$$

tem solução única.

Se $m = n$, $C^t = A_1^t = (a_1 \cdots a_n)$. Logo se $x = x_i (i = 1, \ldots, n)$ tem-se $c = a_i$ e $C^t w = d$, segue que $A_1 w = c$ que implica que

$$(a_1 \cdots a_i \cdots a_n)\, w = a_i,$$

tem solução $w_k = 1$ se $i = k$ e $w_k = 0$ se $i \neq k$.

Se $m > n$ e $x = x_i (i = 1, \ldots, n)$,

$$C^t\ w = \begin{pmatrix} A_1^t & A_2^t \\ B_1^t & B_2^t \end{pmatrix} w = \begin{pmatrix} a_1 & \cdots & a_i & \cdots & a_m \\ b_1 & \cdots & b_i & \cdots & b_m \end{pmatrix} w = \begin{pmatrix} c \\ 0 \end{pmatrix} = \begin{pmatrix} a_i \\ 0 \end{pmatrix}.$$

Como

$$C^t = \begin{pmatrix} A_1^t & A_2^t \\ B_1^t & B_2^t \end{pmatrix} = \begin{pmatrix} R^t & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} Q_{11}^t & Q_{21}^t \\ Q_{12}^t & Q_{22}^t \end{pmatrix} = \begin{pmatrix} R^t & 0 \\ 0 & I \end{pmatrix} Q^t,$$

$$\begin{pmatrix} A_1^t & A_2^t \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} R^t & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} Q_{11}^t & Q_{21}^t \\ Q_{12}^t & Q_{22}^t \end{pmatrix} = \begin{pmatrix} R^t & 0 \\ 0 & 0 \end{pmatrix} Q^t,$$

que implica que

$$\begin{pmatrix} a_i \\ 0 \end{pmatrix} = \begin{pmatrix} A_1^t & A_2^t \\ 0 & 0 \end{pmatrix} e_i = \begin{pmatrix} R^t & 0 \\ 0 & 0 \end{pmatrix} Q^t\ e_i.$$

Logo,

$$C^t w = d$$

$$\begin{pmatrix} R^t & 0 \\ 0 & I \end{pmatrix} Q^t\ w = \begin{pmatrix} R^t & 0 \\ 0 & 0 \end{pmatrix} Q^t\ e_i.$$

$$w = Q \begin{pmatrix} R^{-t} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} R^t & 0 \\ 0 & 0 \end{pmatrix} Q^t\ e_i.$$

$$w = Q \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} Q^t\ e_i.$$

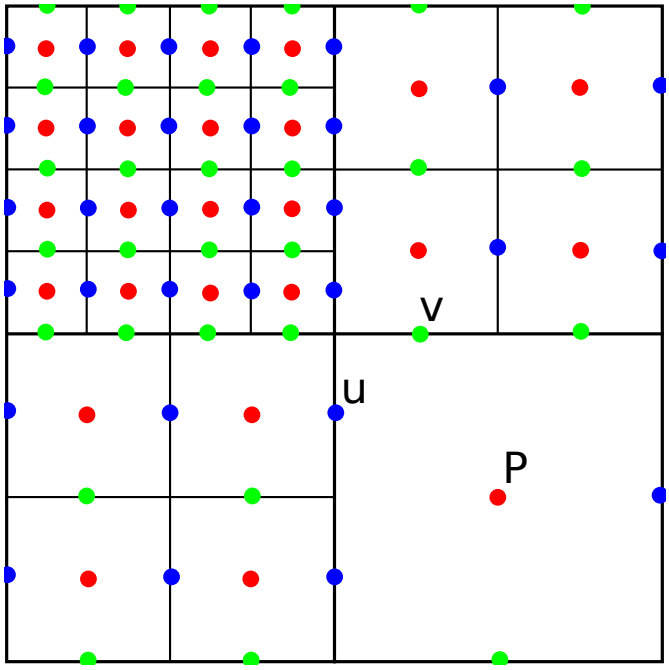

Figura 1.3: Malha deslocada.

1.3.4 Interpolações por MMQ com peso

Complementar com MMQ com peso.

1.4 Propriedades variáveis

O que muda para ρ variável.

1.5 Distribuição de variáveis

Com o objetivo de evitar o aparecimento de oscilações não físicas nas simulações numéricas, foram desenvolvidas técnicas de distribuição de variáveis de forma deslocada, ou seja, pode-se atribuir variáveis no centro da célula, ou nas facetas da mesma. O esquema mais adotado pelos pesquisadores da área de simulação de escoamento de fluidos, quando se adota formulação com variáveis primitivas, em escoamentos incompressíveis, é a distribuição da pressão nos centros das células, e das velocidades nas facetas da mesma, conforme pode ser visto na figura 1.3.

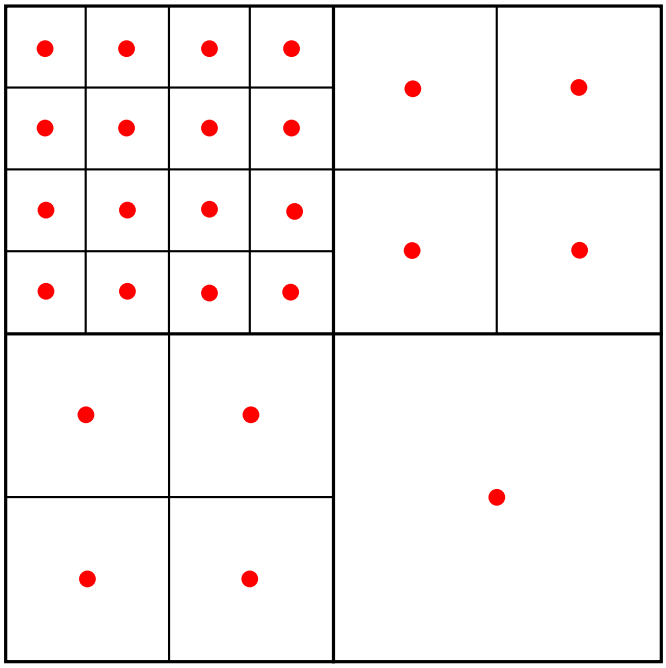

Figura 1.4: Distribuição da Pressão no centro das células.

1.5.1 Variáveis no centro da célula

A pressão, densidade, entalpia total, etc, são normalmente distribuídas nos centros das células. A figura 1.4 ilustra a distribuição da pressão em uma malha com 3 níveis de refinamento.

1.5.2 Variáveis nas facetas da célula

As variáveis provenientes da decomposição da velocidade em vetores (u_i) são distribuídas nas facetas das células. A distribuição da componente da velocidade u_1 em uma malha com 3 níveis de refinamento é apresentada na figura 1.5 e a componente da velocidade u_2 nesta mesma malha é apresentada na figura 1.6.

1.6 Cálculo de derivadas espaciais

Com a utilização da malha deslocada, temos que o cálculo das derivadas espaciais de cada variável distribuída em uma localização diferente tem uma forma diferente de cálculo. Abaixo são apresentadas as variações possíveis numa simulação de

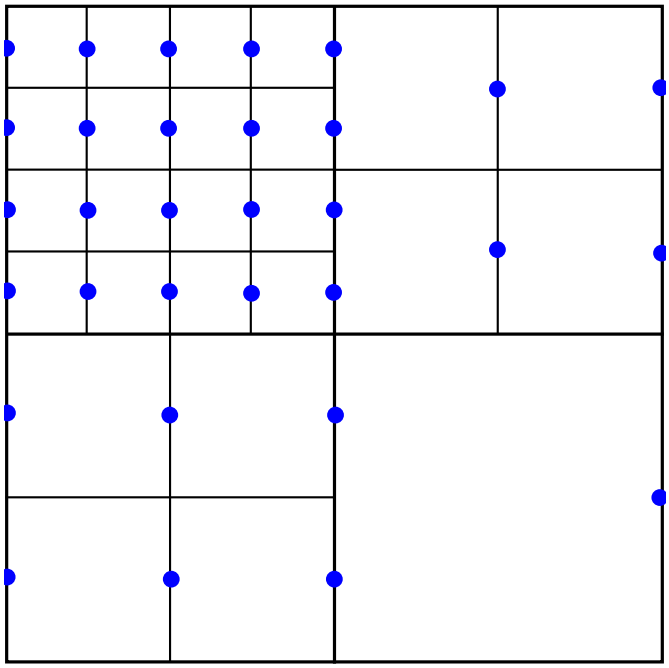

Figura 1.5: Componente da velocidade u_1 nas facetas da células.

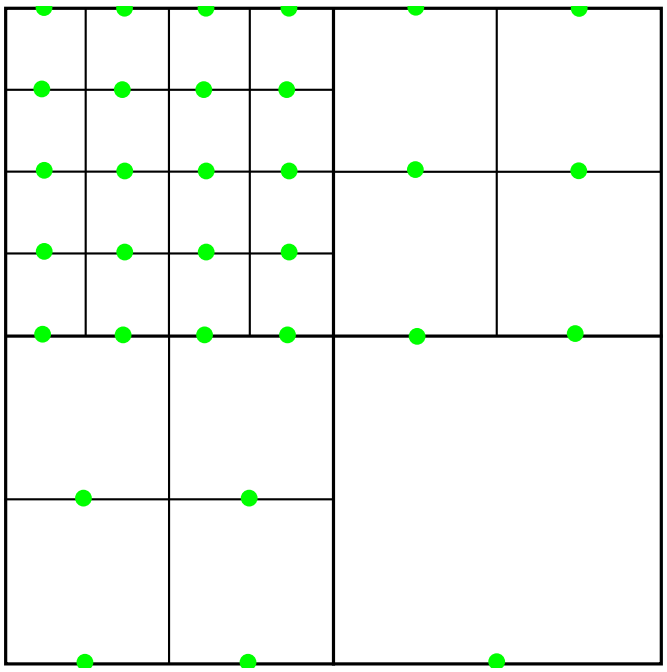

Figura 1.6: Componente da velocidade u_2 nas facetas da células.

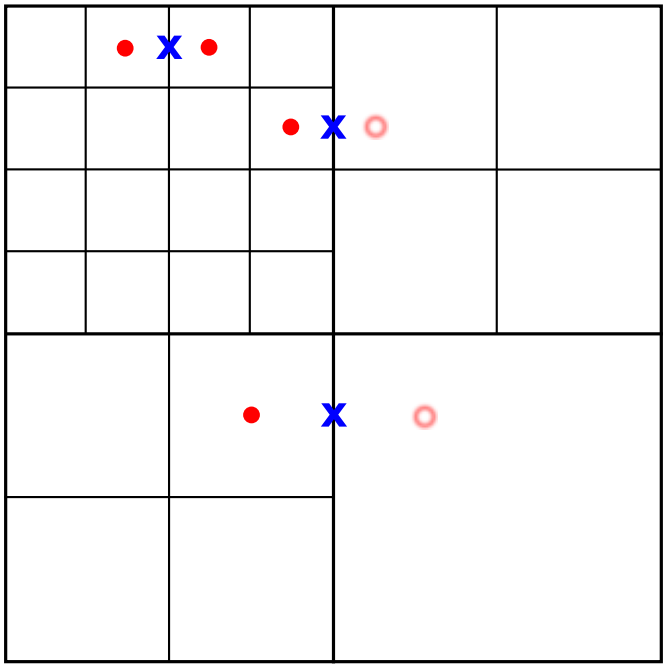

Figura 1.7: Derivada de p na direção x_1.

escoamento.

1.6.1 Derivada primeira de p na direção x_i

Neste caso a variável é definida no centro da célula e pretende-se calcular a derivada espacial na direção x_i, na faceta da célula. A figura 1.7 ilustra as possibilidades de cálculo da derivada na posição marcada com um x de uma variável distribuída no centro da célula na direção x_1. A forma equivalente para o cálculo da derivada na direção x_2, é apresentada na figura 1.8.

1.6.2 Derivada primeira de u_i na direção x_i

Nas equações de Navier-Stokes faz-se necessário o cálculo das primeiras derivadas de variáveis distribuídas nas facetas das células. A figura 1.9 ilustra o cálculo da derivada na posição marcada com um x da componente u_1 na direção x_1. A malha adotada tem 3 níveis de refinamento para ilustrar as diversas possibilidades neste cálculo. O cálculo da componente u_2 na direção x_2 e apresentado na figura 1.10.

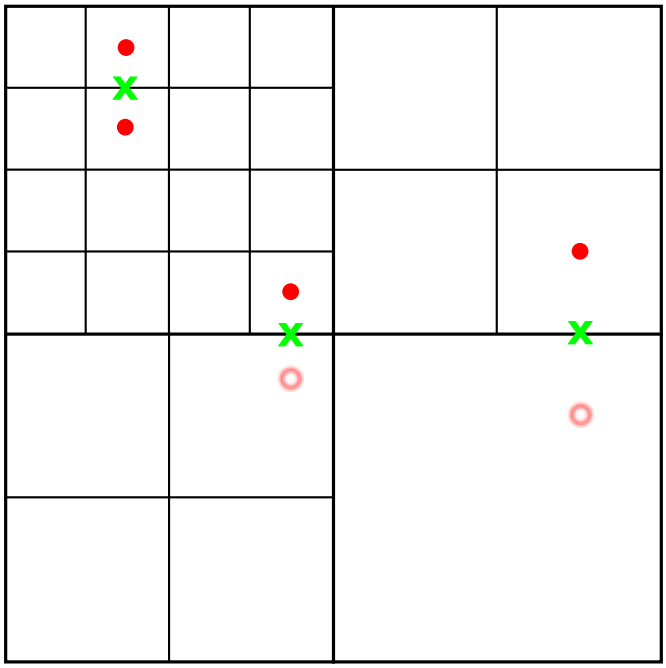

Figura 1.8: Derivada de p na direção x_2.

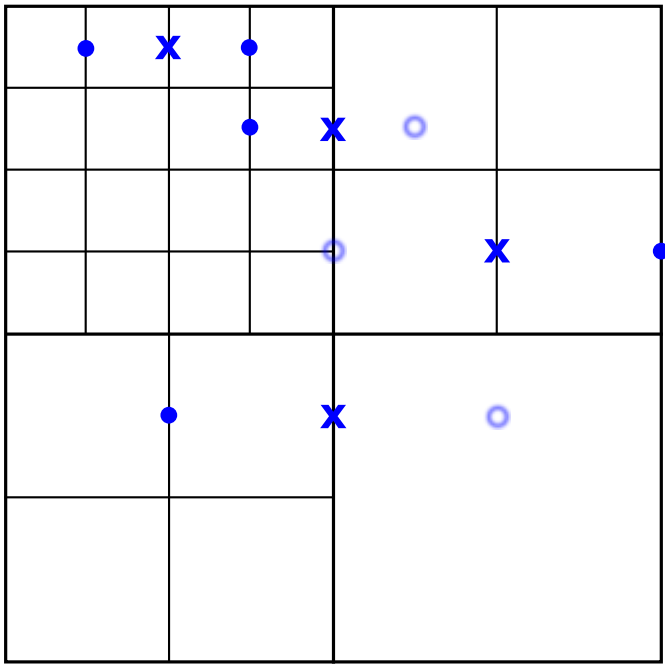

Figura 1.9: Derivada de u_1 na direção x_1.

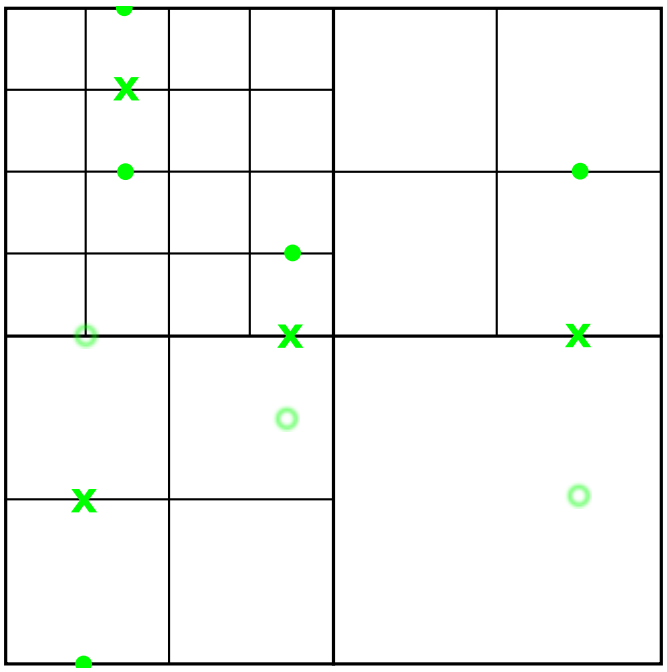

Figura 1.10: Derivada de u_2 na direção x_2.

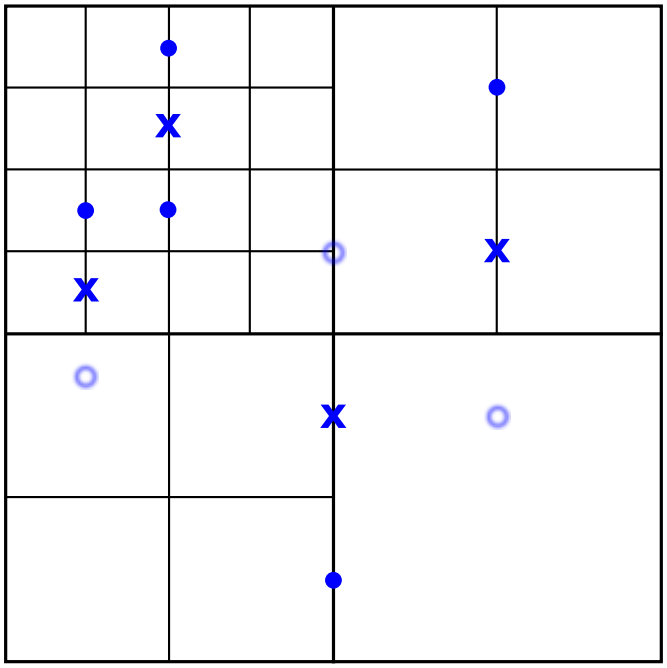

Figura 1.11: Derivada de u_1 na direção x_2.

1.6.3 Derivada primeira de u_i na direção x_j

A figura 1.11 ilustra o cálculo da derivada na posição marcada com um x da componente u_1 na direção x_2. O cálculo da componente u_2 na direção x_1 e apresentado na figura 1.10.

1.6.4 Derivada segunda de u_i na direção x_i

O cálculo da segunda derivada de variáveis definidas nas facetas das células são apresentadas nas figuras 1.13 a 1.16. A posição onde a derivada é calculada é marcada com um x. As diversas configurações dentro da malha com 3 níveis de refinamento são ilustradas.

1.7 Resolução da equação de Poisson

Dada uma variável $p(x, y)$ distribuída no centro das células e um campo de velocidade dado por $u^*(x, y)$, distribuído nas facetas das células, pretende-se resolver a equação de Poisson abaixo:

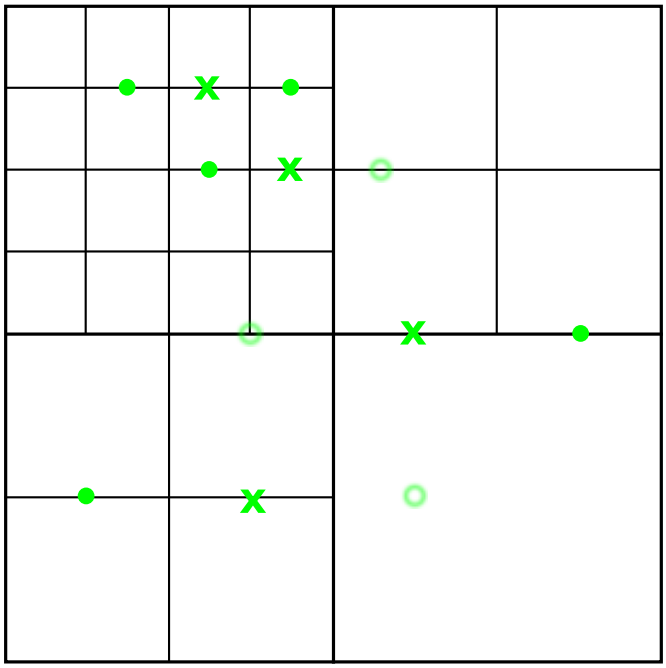

Figura 1.12: Derivada de u_2 na direção x_1.

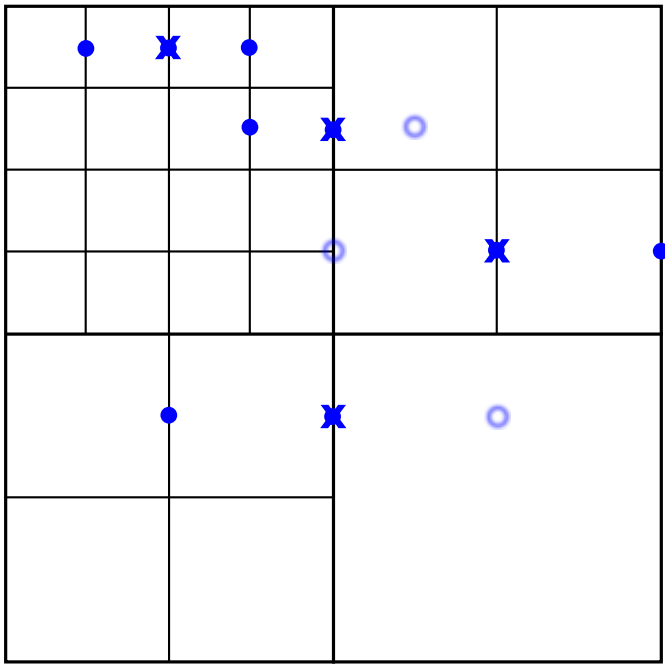

Figura 1.13: Derivada segunda de u_1 na direção x_1.

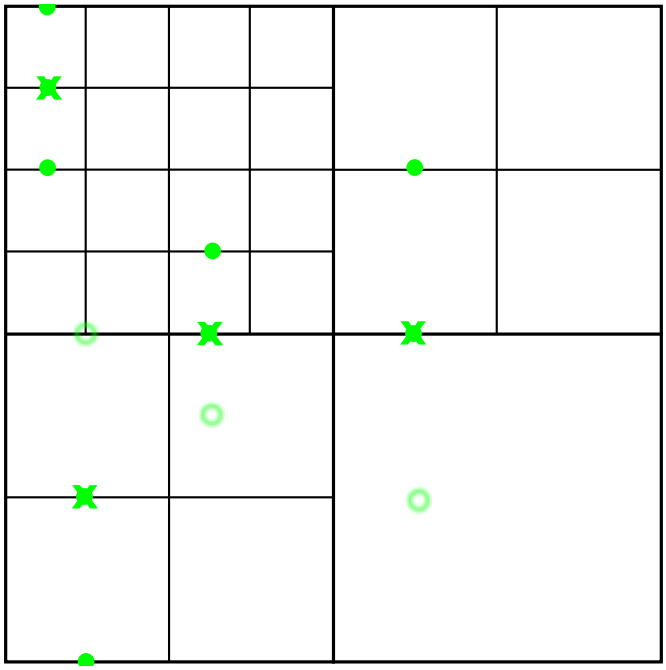

Figura 1.14: Derivada segunda de u_2 na direção x_2.

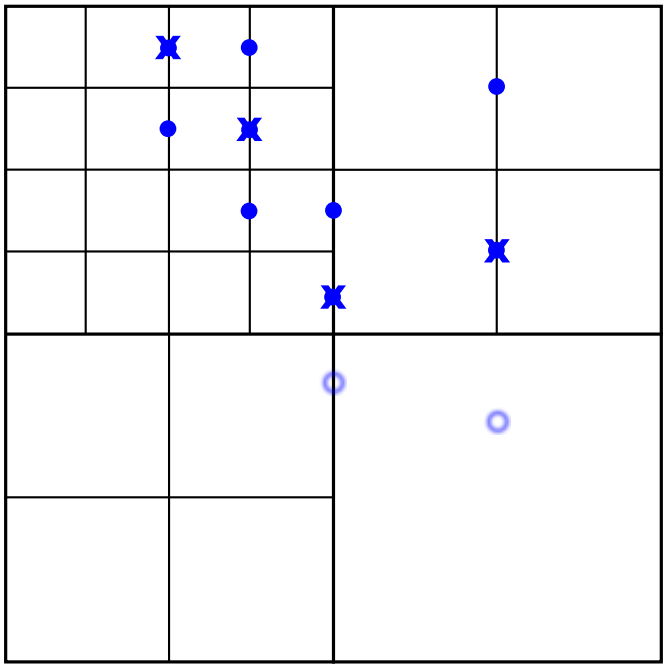

Figura 1.15: Derivada segunda de u_1 na direção x_2.

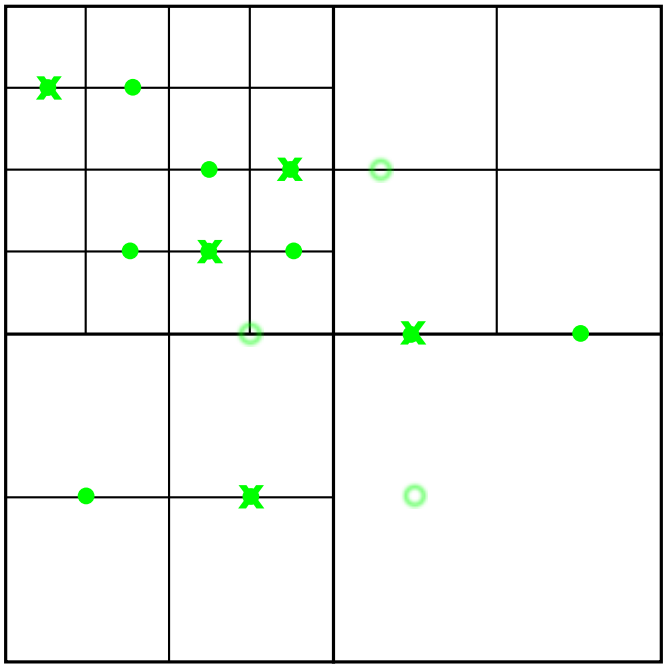

Figura 1.16: Derivada segunda de u_2 na direção x_1.

$$\nabla^2 p = \nabla \cdot u^*$$

Para analisar a convergência do método propõe-se um Método de Solução Manufaturada (MMS - do inglês *Method of Manufactured Solution*) onde as componentes da velocidade são dadas por:

$$u_1(x, y) = -sen(x)cos(y)$$

e

$$u_2(x, y) = -cos(x)sen(y)$$

utilizando as componentes de velocidade acima descritas, em um domínio $0 \leq x, y \leq \pi$ teremos que a solução desta equação para a pressão p é dada por:

$$p(x, y) = cos(x)cos(y)$$

a adoção desta solução garante que a condição de contorno do tipo Neumann é aplicada em todos os contornos do domínio. Para se obter uma solução única, é fixado o valor da pressão em um ponto do contorno. Desta forma a solução desta equação é bem próxima ao que desejamos para aplicação em Navier-Stokes.

1.8 Simulação de escoamento incompressível 2D

A dinâmica de um escoamento bidimensional incompressível de um único fluido sem transferência de calor, pode ser modelado pelas equações de Navier-Stokes e da continuidade dados abaixo:

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j^2} + s_i, \tag{1.1}$$

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{1.2}$$

onde ν é viscosidade cinemática, ρ é a densidade, u_i o componente da velocidade, p é pressão, t é o tempo, x_i as coordenadas espaciais e s_i o termo fonte (gravidade, coriollis, etc.).

Por razões didáticas, propõe-se inicialmente a utilização do método de Euler explícito para avançar a Eq. (1.1) no tempo. O sistema de EDPs Eqs. (1.1) e (1.2) é resolvido utilizando o método da projeção de Chorin [1] descrita abaixo.

1.8.1 Método da projeção de Chorin

Com o objetivo de se obter um campo de velocidade que respeite a continuidade, e como não há uma equação para a pressão p, as equações de Navier-Stokes (Eq. (1.1)) podem ser discretizada da seguinte forma [1]:

$$\frac{u_i^* - u_i^n}{\Delta t} = -u_j^n \frac{\partial u_i^n}{\partial x_j} + \nu \frac{\partial^2 u_i^n}{\partial x_j^2} + s_i,$$

onde o valor de u_i^* é um valor intermediário que deverá ser corrigido através da equação:

$$\frac{u_i^{n+1} - u_i^*}{\Delta t} = -\frac{1}{\rho} \frac{\partial p^{n+1}}{\partial x_i},$$

ou seja,

$$u_i^{n+1} = u_i^* - \Delta t \frac{1}{\rho} \frac{\partial p^{n+1}}{\partial x_i}.$$

Para que isto funcione, pode-se obter a equação de Poisson abaixo, que deverá ser resolvida para podermos avançar com a discretização temporal:

$$\frac{\partial^2 p^{n+1}}{\partial x_i^2} = \frac{\rho}{\Delta t} \frac{\partial u_i^*}{\partial x_i}$$

Utilizando-se este método obtém-se uma distribuição de velocidade e pressão que respeitam a condição de incompressibilidade. Desta forma pode-se realizar uma simulação até atingir-se o estado estacionário (quando as propriedades não variam mais com o tempo), ou até que se atinja um determinado tempo ou mesmo um estado periódico no tempo.

1.8.2 Método da solução manufaturada

Para verificação do método numérico que simula o escoamento bidimensional incompressível propõe-se resolver o problema clássico de fluxo laminar dentro de uma cavidade quadrada de dimensões $0 < x_1, x_2 < 1$, na qual a tampa se move. Para este caso, existe uma solução analítica dada por [2]. As condições de contorno para as componentes de velocidades u_i são do tipo Dirichlet, sendo nula em todos os pontos, exceto em $x_2 = 1$, onde temos:

$$u_1(x_1, 1) = 16(x_1^4 - 2x_1^3 + x_1^2).$$

Um termo fonte s_i está presente somente na equação da quantidade de movimento na direção x_2 e é dado como:

$$\begin{aligned} s_1 &= 0 \\ s_2 &= -8\nu[24F(x_1) + 2f^{(1)}(x_1)g^{(2)}(x_2) + f^{(3)}(x_1)g(x_2)] \\ &\quad -64[F_2(x_1)G_1(x_2) - g(x_2)g^{(1)}(x_2)F_1(x_1)], \end{aligned}$$

onde

$$\begin{aligned} f(x_1) &= x_1^4 - 2x_1^3 + x_1^2, \\ g(x_2) &= x_2^4 - x_2^2, \\ F(x_1) &= \int f(x_1)dx_1 = 0.2x_1^5 - 0.5x_1^4 + x_1^3/3, \\ F_1(x_1) &= f(x_1)f^{(2)}(x_1) - [f^{(1)}(x_1)]^2 = -4x_1^6 + 12x_1^5 - 14x_1^4 + 8x_1^3 - 2x_1^2, \\ F_2(x_1) &= \int f(x_1)f^{(1)}(x_1)dx_1 = 0.5[f(x_1)]^2, \\ G_1(x_2) &= g(x_2)g^{(3)}(x_2) - g^{(1)}(x_2)g^{(2)}(x_2) = -24x_2^5 + 8x_2^3 - 4x_2. \end{aligned}$$

A solução analítica é dada por

$$\begin{aligned} u(x_1, x_2) &= 8f(x_1)g^{(1)}(x_2) = 8(x_1^4 - 2x_1^3 + x_1^2)(4x_2^3 - 2x_2), \\ v(x_1, x_2) &= -8f^{(1)}(x_1)g(x_2) = -8(4x_1^3 - 6x_1^2 + 2x_1)(x_2^4 - x_2^2), \\ p(x_1, x_2) &= 8\nu[F(x_1)g^{(3)}(x_2) + f^{(1)}(x_1)g^{(1)}(x_2)] + \\ &\quad 64F_2(x_1)\{g(x_2)g^{(2)}(x_2) - [g^{(1)}(x_2)]^2\}. \end{aligned}$$

Sabendo-se a solução analítica do escoamento pode-se então verificar o correto funcionamento do mesmo. Além disto, caso sejam realizadas simulações com malhas com diferentes números de pontos, pode-se verificar a ordem de precisão do código como um todo.

Capítulo 2

Documentação do usuário

Esta parte da documentação é destinada a usuários do código e foi subdividida em tutoriais. Estes tutoriais tem como objetivo familiarizar o usuário com os comandos utilizados no código, começando por exemplos bem simples até os casos mais complexos. Cada tutorial é acompanhado de um código executável.

2.1 Tutorial 1 – Gerando uma malha 2D

Nesta seção iremos gerar uma malha bidimensional, em um domínio retangular, o qual damos o nome de *root*, e subdividi-lo em células. Esta será a malha raiz e à partir dela poder-se-á refiná-la nas regiões aonde o usuário achar conveniente. Iniciamos com uma malha bidimensional, portanto onde referencia-se a variavel *DIM*, ela terá valor 2. O código completo se encontra no diretório *tutoriais*, com o nome de *tutorial01.c*. A seguir será descrito o que é realizado em cada comando do código.

Primeiramente é necessário incluir as definições minimamente necessárias para a execução do código, as bibliotecas padrão do sistema e da HiG-Tree:

```
#include<stdio.h>
#include<stdlib.h>
#include "mtree.h"
#include "mtree-io.h"
```

A seguir, inicializa-se o bloco principal

```
int main(int argc, char *argv[])
```

A malha será gerada em um retângulo, desta forma é necessário declarar então os pontos inferior esquerdo *lp* e superior direito *hp*, além de atribuí-los um valor

```
Point lp, hp;
POINT_ASSIGN_SCALAR(lp, -1.0);
POINT_ASSIGN_SCALAR(hp,  1.0);
```

Veja que neste caso atribuimos $lp = (-1, -1)$ e $hp = (1, 1)$ através da macro `POINT_ASSIGN_SCALAR}`, que substitui o código

```
lp[0] = -1.0;
lp[1] = -1.0;
hp[0] =  1.0;
hp[1] =  1.0;
```

Com os pontos lp e hp declarados, basta criar a malha com o comando

```
mt_cell *root = mt_create_root(lp, hp);
```

Com este comando, a célula endereçada pela variável *root* foi criada. Agora é necessário subdividi-la em uma malha regular. Para este exemplo, vamos gerar uma malha de 2×2 elementos. Primeiro, declara-se o número de células desejadas em cada direção (lembrando que *DIM=2*)

```
int num_cell[DIM];
POINT_ASSIGN_SCALAR(num_cell, 2);
```

e invocar o comando

```
mt_refine_uniform(root, num_cell);
```

para criar a malha desejada.

Para visualizar a malha, basta criar um arquivo *vtk* e imprimir a malha no arquivo. Isto pode ser realizado da seguinte forma:

```
FILE *fd = fopen("tut01.vtk", "w");
mtio_print_in_vtk2d(fd, root);
fclose(fd);
```

Uma imagem da malha gerada pode ser vista na figura 2.1.

Após realizados todos os cálculos a raiz tem que ser destruir para não ficar ocupando espaço em memória, o que é feito através do comando:

```
mt_destroy(root);
```

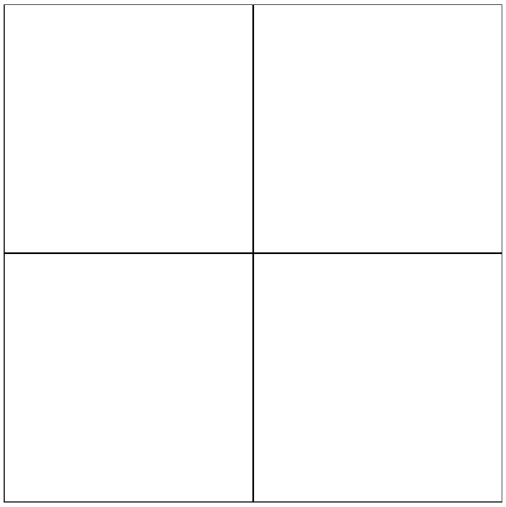

Figura 2.1: Malha gerada pelo tutorial 1.

2.2 Tutorial 2 – Refinamento de células

Nesta seção iremos tomar uma malha bidimensional gerada como no exemplo anterior, mas com 4 células na direção x e 5 na direção y, e realizar o procedimento de refinamento, ou seja, tomar uma ou mais células da malha e dividi-la em partes menores. O código a ser adotado se encontra no diretório *tutoriais*, com o nome de *tutorial02.c*. A seguir será descrito o que é realizado em cada comando do código.

Após inicializada a nova malha, subdivide-se o domínio da forma descrita acima

```
int num_cell[DIM];
num_cell[0] = 4;
num_cell[1] = 5;
mt_refine_uniform(root, num_cell);
```

Digamos que seja necessário o refinamento de células próximo ao ponto $(0.5, 0.5)$. Definamos o ponto p

```
Point p;
p[0] = 0.5;
p[1] = 0.5;
```

Em seguida, selecionamos a célula a ser refinada através do comando

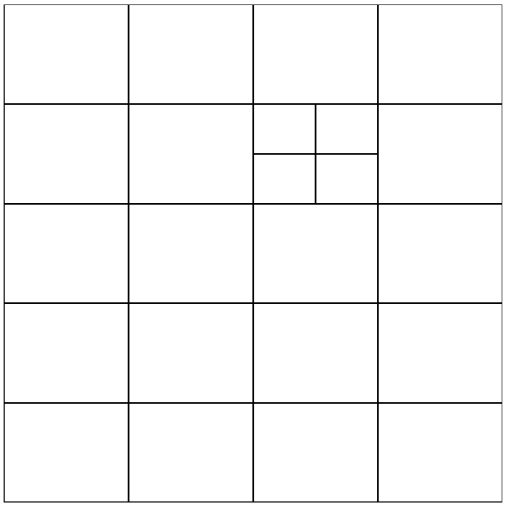

Figura 2.2: Malha gerada pelo tutorial 2.

```
mt_cell *c = mt_get_cell_with_point(root,p);
```

Para realizar o refinamento da célula que contém o ponto *p*, define-se o número de células que será utilizado para este refinamento nas duas direções e em seguida é dado o comando de refinamento uniforme na célula *c*.

```
num_cells[0] = 2;
num_cells[1] = 2;
mt_refine_uniform(c,num_cells);
```

Como feito anteriormente, a malha pode ser impressa em arquivo *vtk* da seguinte forma

```
FILE *fd = fopen("tut02.vtk", "w");
mtio_print_in_vtk2d(fd, root);
fclose(fd);
```

A malha resultante pode ser vista na figura 2.2.

Capítulo 3

Documentação do programador

Referências Bibliográficas

[1] Alexandre Joel Chorin. Numerical solution of the Navier–Stokes equations. *Mathematics of Computation*, 22(104):745–762, 1968.

[2] T. M. Shih, C. H. Tan, and B. C. Hwang. Effects of grid staggering on numerical schemes. *International Journal for Numerical Methods in Fluids*, (2):193–212, 1989.