UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
DEPARTAMENTO DE FÍSICA
CURSO DE BACHARELADO EM FÍSICA

PAULO VICTOR SOARES E SILVA

FÍSICA DO ESCOAMENTO MULTIFÁSICO

Uma abordagem computacional utilizando o software *OpenFOAM*®

Fortaleza

2014

PAULO VICTOR SOARES E SILVA

FÍSICA DO ESCOAMENTO MULTIFÁSICO : UMA ABORDAGEM COMPUTACIONAL UTILIZANDO O SOFTWARE *OPENFOAM*®

Monografia apresentada ao curso de Bacharelado em Física do Departamento de Física da Universidade Federal do Ceará, como requisito para a obtenção do Título de Bacharel em Física.

Orientador: Prof. Dr. Raimundo Nogueira da Costa Filho

Fortaleza

2014

PAULO VICTOR SOARES E SILVA

FÍSICA DO ESCOAMENTO MULTIFÁSICO: UMA ABORDAGEM COMPUTACIONAL UTILIZANDO O SOFTWARE *OPENFOAM*®

Monografia apresentada ao curso de Bacharelado em Física do Departamento de Física da Universidade Federal do Ceará, como requisito para a obtenção do Título de Bacharel em Física.

Aprovada em _____ / _____ /_____.

BANCA EXAMINADORA:

_____

Prof. Dr. Raimundo Nogueira da Costa Filho

_____

Prof. Dr. Murilo Pereira de Almeida

_____

Ms Rilder de Sousa Pires

À minha mãe, Maria Lila, por
todo apoio, carinho e incentivo.

AGRADECIMENTOS

Agradeço imensamente à todos familiares, amigos, colegas e funcionários do Departamento de Física por todo apoio nos anos de graduação na UFC. Agradeço também à todos os docentes do departamento que, de um modo ou de outro, contribuíram para minha formação como físico, bem como aos docentes da Universidade de Coimbra em especial à Prof. Drª Alexandra Pais, (Departamento de Física) e Prof. Drª Rita F. Lopes (Departamento de Eng. Civil) que também foram muito importantes para concretização desse trabalho.

Sou igualmente grato também às instituições públicas de fomento CNPq e Funcap pelos recursos que viabilizaram a minha formação e a conclusão desse trabalho.

“A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original.” (Albert Einstein)

RESUMO

O escoamento multifásico é de grande importância na investigação e no entendimento de uma grande série de fenômenos naturais e aplicações de engenharia. Processos tais como difusão gasosa em um meio, transporte de fármacos em vias aéreas ou sanguíneas e de combustão interna em motores são exemplos deste tipo de escoamento. Dada a natureza das equações envolvidas, a abordagem puramente analítica de problemas multifásicos enfrenta grande dificuldade em obter soluções satisfatórias. Sendo assim, é necessário fazer uso de estudos experimentais e computacionais para elucidar a dinâmica de tais processos. A Dinâmica dos Fluidos Computacional, ou no inglês, Computational Fluid Dynamics (CFD), se propõe a esclarecer escoamentos de tal complexidade. A simulação numérica de escoamento multifásico, fazendo uso do software livre OpenFOAM®, é o objeto de estudo nesse trabalho. Através dos métodos de Volumes Finitos e Volume de Fluido, determinou-se a superfície livre durante um escoamento envolvendo dois fluidos newtonianos (água e ar) bem como a distribuição dos campos de pressão e velocidade. Inicialmente, num caso bidimensional e em seguida um caso onde o escoamento se desenvolve em três dimensões. A fidelidade física obtida na determinação das propriedades de interesse evidencia a grande capacidade do software na solução de problemas de transporte. Contudo, a ausência de um modelo de turbulência para melhor determinar a interação entre as fases compromete em parte os resultados obtidos.

Palavras-chave: OpenFOAM. Escoamento Multifásico. Volumes Finitos. Método do Volume de Fluido. Dinâmica dos Fluidos Computacional. CFD.

ABSTRACT

The multiphase flow has a great importance in research and understanding of a wide range of natural phenomena and engineering applications. Processes such as gas diffusion in a medium, drug transport in air or blood pathways and internal combustion engines are just a few examples of this type of flow. Given the nature of the involved equations, the analytical approach of multiphase problems faces great difficulty in obtaining satisfactory solutions. Thus, it is necessary to make use of computational and experimental studies to elucidate the dynamics of these processes. The Computational Fluid Dynamics aims to clarify the flow of such complexity. The numerical simulation of multiphase flow, making use of free software OpenFOAM®, is the main goal of this work. By the methods of finite volume and fluid volume, the free surface for a flow involving two Newtonian fluids (water and air) was determined as well as the distribution of pressure and velocity fields. Initially, a two-dimensional case is solved and then a case where the flow develops in three dimensions. The physical fidelity obtained in determining the properties of interest shows the great ability of the software to solve transportation problems. However, the absence of a turbulence model to better determine the interaction between the phases compromises the results obtained.

Keywords: OpenFOAM. Multiphase Flow. Finites Volumes Method. Volume Of Fluid Volume. Computational Fluid Dynamics.

SUMÁRIO

1. INTRODUÇÃO

A Mecânica dos Fluidos é uma área da Física que estuda as características de um fluido em repouso ou em movimento. Tal análise é de grande utilidade para o entendimento de uma vasta gama de fenômenos desde o comportamento de correntes marítimas ou atmosféricas, funcionamento dos sistemas respiratórios e sanguíneos, mancha solares e até mesmo interação entre galáxias. A Mecânica dos Fluidos também ocupa lugar de destaque na solução de problemas práticos. Projetos aeronáuticos e navais, desenho de canais, barragens e sistemas de irrigação, de turbinas, exaustores e de geração de energia eólica são apenas alguns exemplos de engenharia nos quais o entendimento do escoamento do(s) fluido(s) em questão é fundamental para se obter êxito. As figuras 1.1 ilustram isso.

Os fluidos estão sujeitos às leis de conservação de massa, quantidade de movimento, de energia, e de entropia. A conservação de quantidade de movimento é expressa pelas Equações de Navier-Stokes. Estas equações são deduzidas a partir de um balanço de forças atuantes sobre um volume infinitesimal de fluido. Estes e outros conceitos importantes, bem como as equações envolvidas, serão discutidos com mais detalhes no capítulo 02.

Sendo uma ciência física, a Mecânica dos Fluidos (MF) faz uso intenso de análise matemática e experimental. Abordagens analíticas são muito produtivas no estudo de casos mais simples ou idealizados, onde se pode fazer considerações (tal como simetria ou viscosidade desprezível) a fim de tornar a solução mais acessível. Porém tais práticas tendem a nos distanciar da solução real de casos mais complexos. Para contornar essa questão lança-se mão de métodos numéricos de solução das equações envolvidas. Surge, portanto uma subárea da MF chamada de Dinâmica dos Fluidos Computacional, ou em inglês *Computational Fluid Dynamics* (CFD). Através de algoritmos apropriados as equações que descrevem o problema físico sofrem um processo de *discretização*. Daí então em vez de um sistema de equações diferenciais obtemos um sistema linear para ser solucionado. A resolução deste sistema nos dará a solução física desejada.

Existem disponíveis hoje uma série de softwares dedicados à solução numérica de problemas de CFD. Entre eles podemos destacar o ANSYS Fluent® e CFX®, Code Saturn e OpenFOAM®, todos possuindo suas vantagens e desvantagens em

determinados aspectos. O primeiro é largamente difundido tanto acadêmica quanto comercialmente, porém sendo um software privado é necessário o pagamento de licença para usá-lo. Já os demais são abertos e gratuitos podendo ser obtidos na internet e serem adaptados livremente. Ainda não há tanta documentação acerca deles na literatura e possuem uma comunidade de usuários relativamente pequena, mas essa é uma realidade que está rapidamente mudando. O número de usuários de OpenFOAM® (software usado no presente trabalho) cresce a cada ano e é cada vez mais comum encontrar publicações que o utilizaram para validar seus resultados.

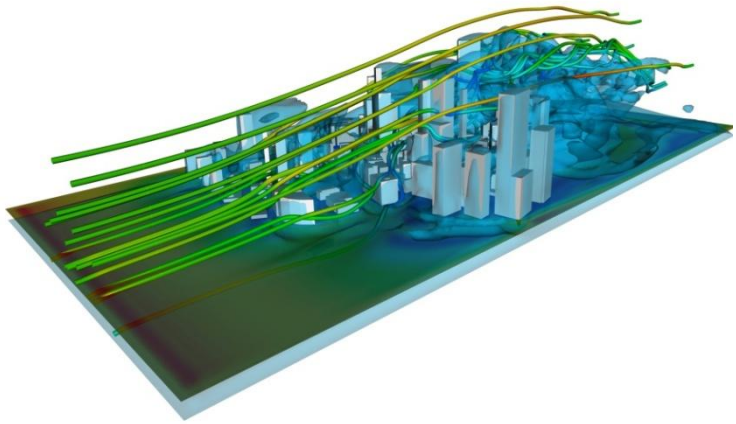


Figura 1.1(a) – Linhas de corrente incidindo sobre um conjunto de edifícios

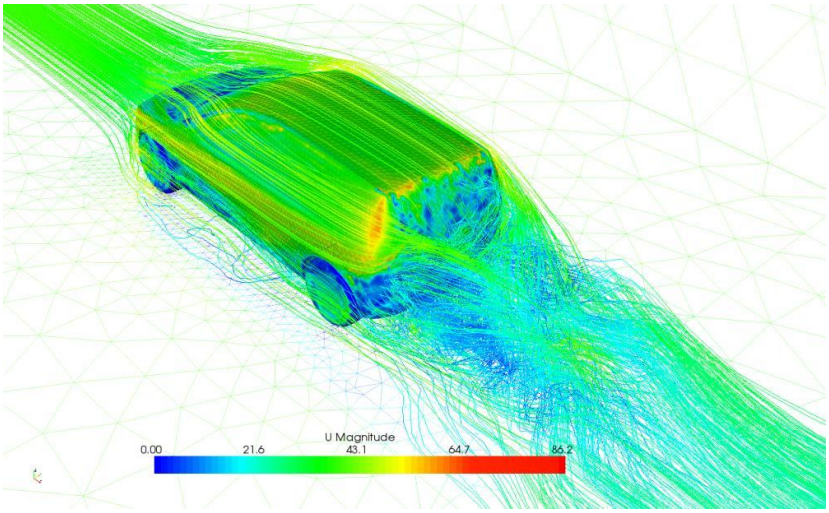


Figura 1.1(b) – Simulação de um escoamento sobre um automóvel.

Figuras 1.1 – A Mecânica dos Fluido/CFD encontra lugar de destaque na aplicação em vários ramos da engenharia. Desde a construção civil e arquitetura até a aerodinâmica de meios de transporte.

O OpenFOAM®, abreviação para *Open source Field Operation And Manipulation*, é um software de CFD desenvolvido pela *ESI* que pode ser gratuitamente adquirido através do site http://www.openfoam.org/. Tendo uma grande gama de recursos para solução de casos desde escoamentos complexos envolvendo reações químicas, turbulência e transferência de calor até mecânica dos sólidos e eletromagnetismo. Estando sob os termos da licença *General Public License* (GNU), o OpenFOAM® oferece ao usuário completa liberdade de adaptar e estender as funcionalidades existentes no software. Dentre os seus recursos destacam-se os *solvers*, algoritmos desenvolvidos para cada tipo de problema a ser resolvido, e os *utilities* que são ferramentas para manipulação de dados de cada caso. A introdução dos dados do problema a ser desenvolvido (condições iniciais, de contorno etc) é realizada através de arquivos "*Dict*", também chamados de dicionários, que serão interpretados pelo software para a solução. Esse método é considerado uma das desvantagens do OpenFOAM® pois não há uma interface *Graphical User Interface* (GUI) para o input dos dados, como ocorre *ANSYS Fluent*® por exemplo.

Se tratando de um pacote completo, é dotado de ambientes de pré e pós-processamento.

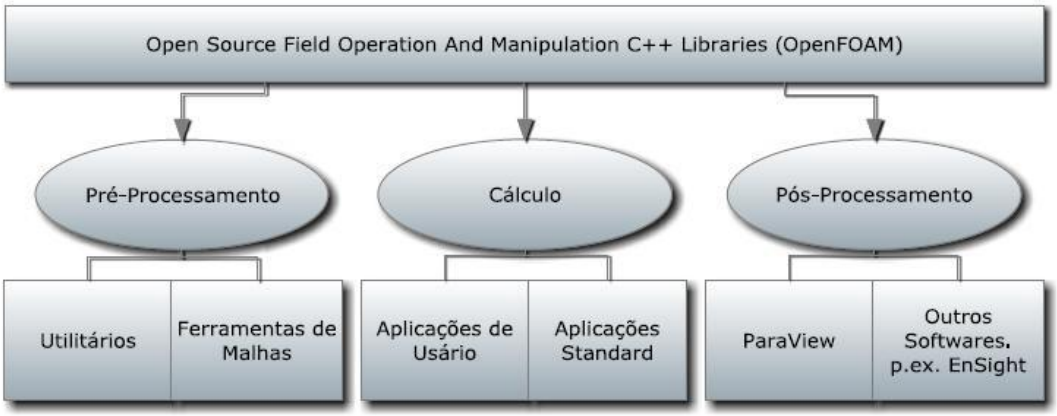


Figura 1.2 - Estrutura do OpenFOAM® (adaptado do User Guide)

Estes recursos bem como o seu desenvolvimento, sua estrutura de funcionamento, métodos de discretização utilizados e outros detalhes acerca do software serão introduzidos no capítulo 03.

Dentre os escoamentos de interesse físico e tecnológico está o escoamento multifásico. Trata-se de um escoamento composto por dois ou mais fluidos, ou fases, distintos. Uma técnica consagrada na solução de problemas multifásicos e utilizada pelo OpenFOAM® (OF) é o método conhecido como *Volume Of Fluid* (*VOF)* [seção 3.2]. Esse método consiste na determinação da quantidade das fases em cada volume computacional dum dado escoamento, através da determinação da função de fase. As equações de Navier-Stokes e da continuidade são resolvidas para ambos os fluidos e as propriedades físicas de interesse (viscosidade, densidade etc.) são calculadas como médias ponderadas da função de fase numa célula computacional do domínio. Obteremos um mapeamento de cada fluido em todo domínio computacional em cada instante de tempo.

No capítulo 04 faremos, através do estudo de dois casos, a determinação da superfície livre entre as fases, bem como dos campos de velocidade e pressão, em um escoamento multifásico por meio do solver *interFoam*. No primeiro, denominado "*bottle*", temos um exemplo mais pedagógico, com o intuito de introduzir como se realiza a solução através do OF para um recipiente inicialmente repleto de ar que num dado instante começa a ser preenchido por água. Já o segundo, denominado "*DAM 3D*", possui cunho prático, se tratando da simulação da ruptura da parede de uma represa.

2 ELEMENTOS DE MECÂNICA DOS FLUIDOS.

2.1 – Conceitos Fundamentais.

Uma substância é denominada de fluido caso ela possua a característica de fluir, ou seja, de escoar facilmente, quando sob tensões de *cisalhamento*. Quando um material sólido é exposto à tensões de cisalhamento, também chamadas de tensões tangenciais, este se deforma até que sejam produzidas *tensões tangenciais internas* que equilibrem as forças externas. Tais forças são proporcionais à deformação sofrida pelo sólido no processo, e caso não sejam excessivamente grandes, o sólido volta à sua forma original uma vez retiradas as forças externas. Isto é, os sólidos de uma maneira geral apresentam um *comportamento elástico*, mesmo que às vezes imperceptível.

No caso de líquidos e gases, não há esse equilíbrio oriundo de tensões internas de cisalhamento. Logo não há qualquer resistência à deformação. Enquanto houver tensões tangenciais externas aplicadas sobre essas substâncias, estas continuarão a se deformar. Mesmo após o término do estimulo externo, os fluidos não mais irão voltar à sua forma anterior. Contudo, mesmo não resistindo à deformação os fluidos resistem à velocidade com que se deformam. Os fluidos reais oferecem resistência ao deslizamento relativo de suas camadas internas adjacentes, sendo tanto maior quanto for a taxa de variação espacial da velocidade relativa de deslizamento. A essa resistência dá-se o nome de *viscosidade*. Em 1687 Isaac Newton postulou uma relação entre à tensão tangencial viscosa aplicada τ e a taxa de deformação resultante, para uma série de fluidos nos quais se incluem o ar e a água, dada por:

$$\tau = \mu \frac{du}{dy} \tag{2.1a}$$

onde a constante de proporcionalidade de μ é identificada como *viscosidade dinâmica*, e u é a componente do campo de velocidade na direção $\hat{\boldsymbol{x}}$.

Aos fluidos que obedecem tal relação, damos o nome de fluidos *Newtonianos*. Há ainda outra forma de representar a viscosidade, dada por:

$$\nu = \frac{[?]}{\rho} \tag{2.1b}$$

em que ρ é a massa específica do fluido. Quando representada pela equação (2.1b) é chamada de *viscosidade cinemática***.**

Existe uma ampla gama de substâncias as quais possuem propriedades entre os sólidos e fluidos, dependendo da natureza da força externa, da sua intensidade bem como do tempo o qual o escoamento está sob a ação de tais forças. Um exemplo bem conhecido é o piche. Este sofre fratura tal qual um sólido sob um impacto brusco mas escoa como um fluido mesmo que em extrema lentidão.

Os fluidos possuem moléculas cujas dimensões características são muito inferiores à distância que as separam. Daí, tem-se que um fluido é basicamente composto por espaços vazios, com uma distribuição discreta de massa através do volume ocupado por ele. A princípio isso não seria um empecilho para o estudo da física dos fluidos, porém ao fazê-lo em tal escala microscópica enfrentam-se dois grandes inconvenientes:

- Do ponto de vista teórico, uma vez que temos uma distribuição discreta de matéria, seria impossível usar a abordagem do cálculo diferencial, que exige que em cada ponto do espaço e em cada instante de tempo as propriedades em estudo sofram variações continuas.
- Do ponto de vista experimental os instrumentos de pesquisa precisariam ter uma imensa resolução na aferição de suas medidas de modo que permitisse acesso a essa escala microscópica.

Portanto, surge a necessidade de adotar uma escala macroscópica que permita contornar tais barreiras, sendo pequena o suficiente para que suas propriedades físicas possam ser consideradas uniformes, porém ainda grande o bastante para conter um grande número de moléculas do fluido. Essa é a chamada *Hipótese do Contínuo*.

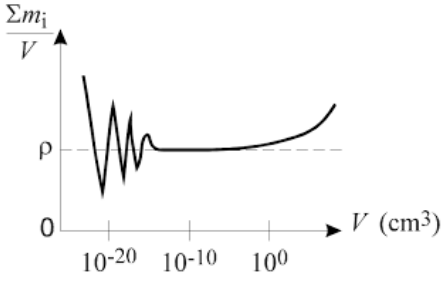


Figura 2. 1 Escalas micro e macroscópicas para V

Na figura 2.1, vemos que para dimensões muito pequenas temos uma oscilação bastante significativa da quantidade de matéria $\sum m_i$ encerrada por um certo volume V. No entanto, a medida que se aumenta o volume de referência a quantidade de matéria encerrada por ele tende a ser constante. Isto é, ρ tende a ser constante à medida que aumentos a escala. É nesta região onde concentraremos nossos esforços para o entendimento dos fenômenos físicos envolvendo as propriedades de interesse, pois assim elas variam continuamente de ponto a ponto em ambos os domínios espacial e temporal. A substância, essencialmente discreta, passa a ser contínua.

A hipótese do Contínuo é valida para uma ampla gama de problemas práticos e teóricos, porém, como se trata de uma aproximação, possui limitações. Problemas envolvendo difusão gasosa em meios nano-porosos, por exemplo, não podem ser tratados com tal abordagem. No entanto, nas aplicações de CFD ela é sempre válida e conduz a resultados de precisão extremamente satisfatória.

Quando um líquido entra em contato com outra substância, tal com um sólido, um gás ou um outro líquido (desde que não sejam imiscíveis), há a formação de um interface entre os meios. As moléculas mais afastadas dessa interface gozam de certa liberdade de movimento. Uma vez que estão sendo solicitadas igualmente em todas as direções pelas moléculas adjacentes. No entanto, as moléculas mais próximas à superfície entre os meios são mais solicitadas para o interior. A figura 2.2 ilustra esse fato.

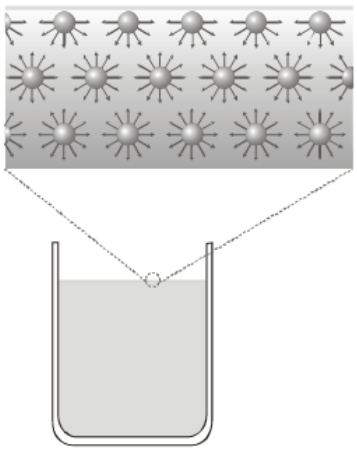


Figura 2.2 Balanço de forças moleculares na interface.

Esta interface, então, se encontra sob ação de uma força de atração molecular que atua ao longo dela. Como resultado, a superfície livre adquire um comportamento de uma fina membrana. A esta tensão sobre a superfície livre entre dois meios dá-se o nome de *tensão superficial*. Matematicamente, define-se tensão superficial, que é característica de um par de meios e não de apenas um deles, como sendo:

$$\gamma = \frac{W_t}{A} \tag{2.2a}$$

isto é, o trabalho W_t realizado pelas forças tangentes à superfície necessário para um aumento A da área da interface. Ou se preferirmos, podemos representar γ como:

$$\gamma = \frac{\boldsymbol{F_t}}{l} \tag{2.2b}$$

Onde l é magninutide da distensão sofrida pela superfície livre sob efeito da força tangencial $\boldsymbol{F_t}$.

2.2 – Distribuição de pressão em um fluido.

Formalmente define-se pressão, p, como sendo a força exercida por unidade de área:

$$p = \frac{dF}{dA} \tag{2.3}$$

Tradicionalmente na mecânica dos fluidos considera-se que se a razão (2.3) for positiva, isto é, $p > 0$ e um $\Delta V < 0$, chama-se p de *compressão.* Ao passo que se $p < 0$, e $\Delta V > 0$ chama-se de *descompressão*.

As forças atuantes sobre um elemento de fluido podem ser classificadas da seguinte forma:

- *Forças de Campo (ou de Corpo)*: são forças que atuam à longa distância, sem a necessidade de contato. Tais como; as força gravitacional e a de Lorentz.
- *Forças de Superfície (ou de Contato)*: são forças que são transmitidas a curta distância através de uma superfície por contato direto com

as partículas de fluido adjacentes. Alguns exemplos são as forças oriundas de gradientes de pressão e forças de atrito/viscosas.

A figura 2.3 mostra um elemento de fluido retangular sob um determinado campo de pressão p.

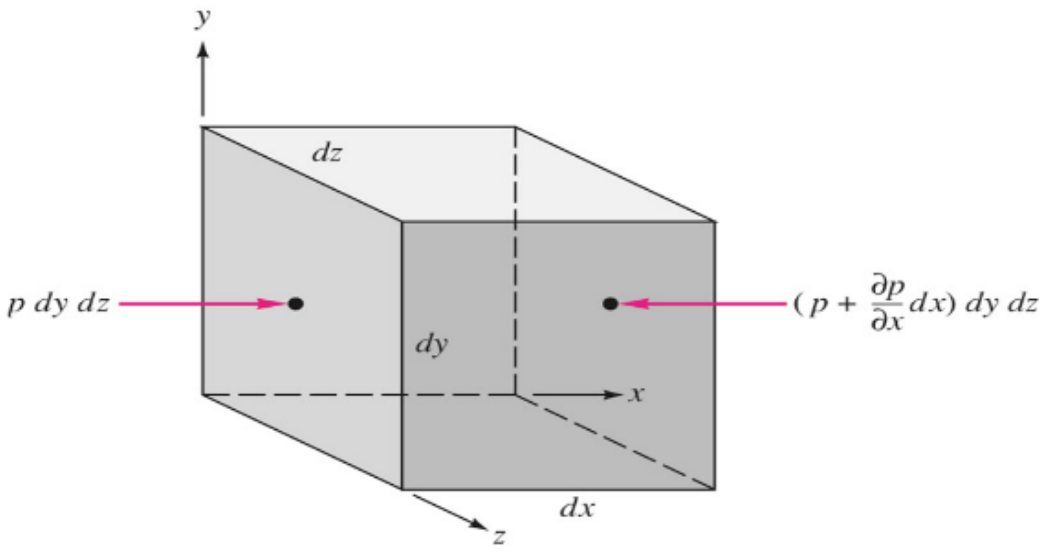


Figura 2.3 - Forças de pressão e gradiente de p num elemto de fluido retangular

Fazendo o balanço de forças de pressão sobre as faces do elemento é para a direção $\hat{x}$, temos uma força líquida dada por:

$$df_x = pdydz - \left(p + \frac{\partial p}{\partial x}dx\right)dydz = -\frac{\partial p}{\partial x}dxdydz \tag{2.4a}$$

Procedendo analogamente para as direções y e z, temos:

$$df_y = pdxdz - \left(p + \frac{\partial p}{\partial y}dy\right)dxdz = -\frac{\partial p}{\partial y}dxdydz \tag{2.4b}$$

$$df_z = pdxdy - \left(p + \frac{\partial p}{\partial z}dz\right)dxdy = -\frac{\partial p}{\partial z}dxdydz \tag{2.4c}$$

Portanto, se reagruparmos os termos temos:

$$df_p = \left(-\frac{\partial p}{\partial x}\hat{x} - \frac{\partial p}{\partial y}\hat{y} - \frac{\partial p}{\partial z}\hat{z}\right)dxdydz \tag{2.5}$$

Ou por unidade de volume:

$$f_p = -\nabla p \tag{2.6}$$

O sinal negativo em (2.6) indica que a força f_p, que recebeu o subscrito p para indicar sua origem, é orientada segundo a direção decrescente da pressão.

A mesma formulação pode ser usada para determinar a força viscosa (2.7a) atuante num elemento de fluido, desde que haja um deslocamento relativo entre as faces. Fazendo tal suposição obtemos na direção $\hat{x}$ o seguinte:

$$df_{vx} = \lim_{dy \to 0} \left[\mu \left(\frac{du}{dy} \right)_{y+dy} - \mu \left(\frac{du}{dy} \right)_{y} \right] dxdz = \frac{d}{dy} \left(u \frac{du}{dy} \right) dxdydz \tag{2.7a}$$

Então, supondo μ constante, temos por unidade de volume:

$$f_v = \mu \frac{\partial^2 u}{\partial y^2} \tag{2.7b}$$

No caso mais geral, haverá componentes de tensão viscosas nas três direções espaciais. Uma vez que μ e ρ possam ser considerados constantes, segundo **[2]** podemos obter a expressão na forma tridimensional:

$$\boldsymbol{f}_v = \mu \left(\frac{\partial^2 \boldsymbol{U}}{\partial x^2} + \frac{\partial^2 \boldsymbol{U}}{\partial y^2} + \frac{\partial^2 \boldsymbol{U}}{\partial z^2} \right) = \mu \nabla^2 \boldsymbol{U} \tag{2.8}$$

em que **U**, cujos componentes (u, v, w), é o vetor velocidade do escoamento.

A segunda Lei de Newton agora nos permite obter a resultante das forças, de campo e de superfícies, que atuam sobre um elemento de fluido e assim determinar seu movimento. Considerando ρ$\boldsymbol{g}$ ($\boldsymbol{g} = -g\hat{z}$) como a única força de corpo atuando sobre o elemento de fluido, e fazendo o somatório as forças viscosas e as oriundas de gradientes de pressão, temos:

$$\rho \boldsymbol{a} = \rho \, {}^{D\boldsymbol{U}}/_{Dt} = -\nabla p + \mu \nabla^2 \boldsymbol{U} + \rho \boldsymbol{g} \tag{2.9}$$

Em que ${}^{D\boldsymbol{U}}/_{Dt}$ é definido como:

$${}^{D\boldsymbol{U}}/_{Dt} = \frac{\partial \boldsymbol{U}}{\partial t} + (\boldsymbol{U} \cdot \nabla)\boldsymbol{U} \tag{2.10}$$

Esta é chamada de *Derivada Material* de **U**. O termo $\partial \boldsymbol{U}/\partial t$ se refere à variação ao longo do tempo, num determinado ponto do espaço. Ao passo que $(\mathbf{U} \cdot \nabla)\mathbf{U}$, chamado de termo advectivo, mede a taxa de variação espacial de **U**.

A equação (2.9) é chamada de *Equação de Navier-Stokes* para fluidos newtonianos com ρ e μ constantes. Ela descreve a taxa de variação da quantidade de movimento do fluido, por unidade de volume, em função do conjunto de forças atuantes sobre o mesmo. Na grande maioria de casos práticos, a equação (2.9) apresenta comportamento não linear. O que requer uso de técnicas numéricas para sua solução. Podendo-se obter uma solução puramente analítica em apenas alguns casos mais simples.

Um dos casos no quais (2.9) se torna de fácil resolução é quando se tem $\mathbf{U} = 0$. Isto é, quando não há um campo de escoamento. Imediatamente temos que:

$$^{D\boldsymbol{U}}/_{Dt} = \nabla^2\boldsymbol{U} = 0 \tag{2.11}$$

Tendo assim se reduzindo à:

$$\nabla p = \rho\boldsymbol{g} \tag{2.12a}$$

A equação (2.2) é chamada de *Equação Fundamental da Hidrostática*, pois mostra que para um fluido em repouso o campo de forças resultante é aquele oriundo do seu próprio peso.

Uma vez que $\boldsymbol{g} = -g\hat{z}$, p se torna função apenas de uma direção. Logo:

$$\frac{dp}{dz} = -\rho g \tag{2.12b}$$

Assim,

$$dp = -\rho g\, dz \Rightarrow p = -\rho g \int_{z_0}^{z} dz \tag{2.13}$$

Das expressões (2.12) podemos inferir:

- A pressão varia independentemente das formas de onde está o fluido.
- A pressão cresce verticalmente para baixo.
- Dois pontos situados num mesmo nível estão sob a mesma pressão.
- A interface entre dois fluidos em equilíbrio, não miscíveis e de diferentes ρ é horizontal.

Todos estes são conceitos fundamentais da hidrostática.

Os fluidos, em repouso ou em movimento, estão sujeitos às leis de conservação da física. A mais imediata delas é a conservação da massa. Num escoamento, não há criação ou perda de fluido. Matematicamente, a conservação da massa é descrita pela *equação da continuidade*:

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \boldsymbol{U}) \tag{2.14}$$

Para o caso de escoamentos incompressíveis, usualmente quando **U** não excede um 1/3 da velocidade do som no fluido, temos ρ constante em (2.14). Logo:

$$\nabla \cdot \mathbf{U} = 0 \tag{2.15}$$

A equação (2.15) implica além da conservação da massa, na conservação do volume envolvido no escoamento. Geometricamente, o escoamento incompressível é representado através de linhas de correntes paralelas, não havendo fontes nem sorvedouros.

Além da massa é preciso verificar a conservação da energia no escoamento. Assim como na mecânica dos sólidos, a energia nos fluidos deve se conservar na ausência de forças dissipativas. Desconsiderando as há tensões viscosas em (2.9) obtém-se:

$$\rho \left[\frac{\partial \boldsymbol{U}}{\partial t} + (\boldsymbol{U} \cdot \nabla)\boldsymbol{U}\right] = -\nabla p + \rho \boldsymbol{g} \tag{2.16a}$$

A equação (2.16a) por vezes é chamada de *Equação de Euler*. Sendo $\boldsymbol{g} = -g\hat{z}$ e utilizando a propriedade $\hat{z} = \nabla z$, podemos reescrever os termos da seguinte forma:

$$\left[\frac{\partial \boldsymbol{U}}{\partial t} + (\boldsymbol{U} \cdot \nabla)\boldsymbol{U}\right] = -\nabla(p + \rho g z) \tag{2.16b}$$

Segundo **[2]**, considerando uma dada coordenada curvilínea s, de maneira que o versor seja $\widehat{\boldsymbol{ds}}$, o vetor deslocamento elementar $\boldsymbol{ds} = ds\,\hat{\mathbf{s}}$, $\nabla = (\partial/\partial s)\hat{s}$ e $\hat{\mathbf{s}}$ // **U**. Temos:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \left(U\frac{\partial}{\partial s}\right)\boldsymbol{U} = -\frac{1}{\rho}\frac{\partial p}{\partial s}\hat{s} - g\frac{\partial z}{\partial s}\hat{s} \tag{2.16c}$$

Multiplicando todos os membros de (2.16c) por $\boldsymbol{ds}$ e considerando:

$$\left(\frac{\partial \boldsymbol{U}}{\partial t}\cdot ds\right) \cong \frac{\partial U}{\partial t}ds \tag{2.16d}$$

Resulta em:

$$\frac{\partial U}{\partial t}ds + U\frac{\partial U}{\partial s}ds = -\frac{1}{\rho}\frac{\partial p}{\partial s}ds - g\frac{\partial z}{\partial s}ds \tag{2.16e}$$

Uma vez que $UdU = d(U^2/2)$, temos:

$$\frac{\partial U}{\partial t}ds + \frac{\partial p}{\rho} + d\left(\frac{U^2}{2}\right) + gdz = 0 \tag{2.17a}$$

A equação (2.17a) acima é a *Equação de Bernoulli* para escoamentos dependentes do tempo e com ρ variável. Ela exprime a conservação da energia em escoamentos sem tensões viscosas. Integrando-se (2.17a) entre dois pontos 1 e 2 da linha de corrente, e considerando o escoamento estacionário ($\partial U/\partial t = 0$) e incompressivel, temos a sua forma mais conhecida:

$$p_2 - p_1 + \rho\frac{1}{2}(U_2^2 - U_1^2) + g\rho(z_2 - z_1) = 0 \tag{2.17b}$$

2.3 Regimes de Escoamento e Turbulência.

Quando num escoamento as camadas adjacentes de fluido se deslocam com a mesma velocidade, ou deslizam entre si de maneira ordenada quando há pequenas variações da velocidade relativa entre elas, os elementos de fluido seguem trajetórias bem definidas e contínuas. Não há alterações súbitas em seu percurso, jamais se interceptando. Neste caso, dizemos que o escoamento está em *regime laminar*. Porém, caso haja um deslocamento brusco entre as camadas, com os elementos de fluido se interceptando em suas trajetórias de forma aleatória, o escoamento é classificado como *regime turbulento.*

A turbulência é um fenômeno altamente dissipativo no qual surgem estruturas espirais, que acompanham as linhas de corrente no escoamento, chamadas de *vórtices*. Possuindo caráter caótico, a turbulência exige para seu estudo detalhado o uso de complexas técnicas de estatística, de maneira que aqui nos limitaremos a uma abordagem introdutória ao assunto. Uma vez que os escoamentos a serem analisados no capítulo 04 são tratados como laminar.

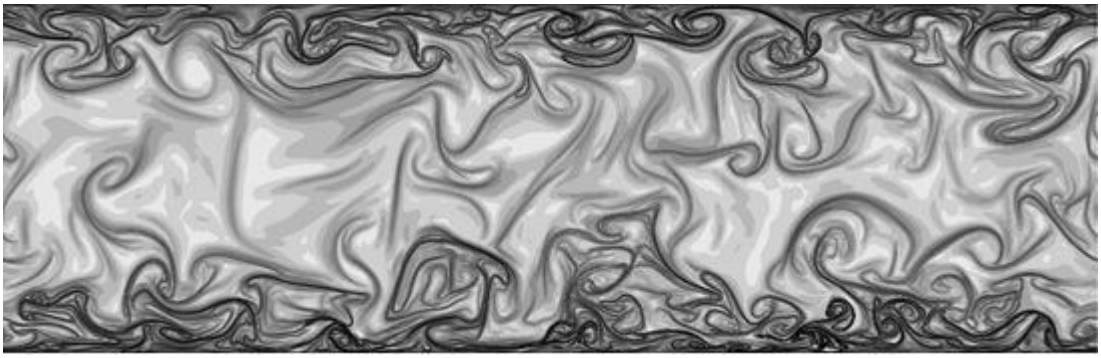


Figura 2.4- A formação de vórtices evidencia a intensa interação entre as camadas fluidas durante o escoamento.

Um parâmetro para medir o grau de turbulência de um escoamento é o *número de Reynolds* do mesmo, definido pela expressão:

$$Re = \frac{UL\rho}{\mu} \tag{2.18a}$$

onde usando a definição de viscosidade cinemática (2.1b):

$$Re = \frac{UL}{\nu} \tag{2.18b}$$

onde $\boldsymbol{U}$ e L são respectivamente a magnitude do vetor velocidade e o comprimento característico do escoamento.

Sendo um parâmetro adimensional, Re é uma relação entre as forças de inércia e as viscosas. Dessa maneira, num escoamento laminar, qualquer perturbação tende a ser amortecida por ação viscosa o que implica em baixos valores para Re. No entanto, para altos valores de U e/ou fluidos poucos viscosos, as forças inerciais sobrepujam as forças viscosas dando origem a instabilidades no

escoamento e consequentemente a formação de vórtices. Nesse caso, teremos um alto Re.

Um escoamento inicialmente laminar pode passar a ser turbulento à medida que se aumenta Re. O escoamento passará por uma fase intermediária chamada de *transição de regime* onde surgirão as primeiras instabilidades na forma de vórtices de pequena intensidade. Nessa condição, o escoamento possui um Re chamado *crítico*. À medida que Re aumenta o escoamento deixa o comportamento laminar passando a se desenvolver com grandes vórtices, sendo agora turbulento.

Durante a formulação da equação de Navier-Stokes (2.9), nada foi dito em relação ao regime de escoamento em questão. Logo elas permanecem válidas independentemente do Re. Porém, pelo fato da turbulência possuir grande caráter dissipativo faz-se necessário fazer correções em (2.9) na forma da adição de um termo extra e uso dos valores médios das variáveis físicas.

O termo adicionado é definido como *tensor das tensões de Reynolds:*

$$\overleftrightarrow{Re} = \begin{pmatrix} \overline{u'^2} & \overline{u'v'} & \overline{u'w'} \\ \overline{v'u'} & \overline{v'^2} & \overline{v'w'} \\ \overline{w'u'} & \overline{w'v'} & \overline{w'^2} \end{pmatrix} \tag{2.19}$$

Onde u', v', w' são os valores das flutuações das componentes u, v, w de **U**. Neste contexto, se faz necessário a substituição de U por uma aproximação estatística (figura 2.5) na forma:

$$U = \bar{U} + U' \tag{2.20}$$

Onde $\bar{U}$ é a média temporal de U dada por:

$$\bar{U} = \frac{1}{\Delta t}\int_{t}^{t+dt} U dt \tag{2.21}$$

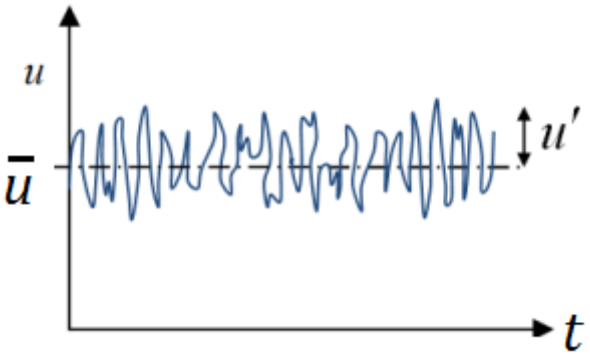


Figura 2.5 Evolução temporal da componente u e sua decomposição em $\bar{u}$ e u´.

Essa aproximação também é valida para as demais variáveis do escoamento $(v, w, p \ldots)$ de maneira que para (2.15) temos:

$$\nabla \cdot \bar{U} = \frac{\partial \bar{u}}{\partial x} + \frac{\partial \bar{v}}{\partial y} + \frac{\partial \bar{w}}{\partial z} = 0 \qquad (2.22a)$$

$$\nabla \cdot U' = \frac{\partial u'}{\partial x} + \frac{\partial v'}{\partial y} + \frac{\partial w'}{\partial z} = 0 \qquad (2.22b)$$

Assim (2.9) é reescrita na forma:

$$\rho \, {}^{D\bar{\boldsymbol{U}}}/_{Dt} = -\nabla \bar{p} + \mu \nabla^2 \bar{\boldsymbol{U}} + \rho \boldsymbol{g} - \rho \nabla \cdot \overleftrightarrow{Re} \qquad (2.23)$$

Os termos de $\overleftrightarrow{Re}$ na prática atuam como mecanismos extras de troca de quantidade de movimento entre diferentes regiões do escoamento, chegando em muitos casos a serem mais relevantes, em termos de dissipação de energia, do que as forças viscosas.

Sendo simétrico, o tensor de Reynolds adiciona, seis incógnitas ao sistema de equações diferenciais parciais do escoamento. Assim temos uma número de incógnitas maior que os de equações, o que impossibilita a resolução direta do problema. Para se contornar tal inconveniente são adotadas expressões destinadas a relacionar as novas incógnitas com as variáveis do escoamento. Tais funções são chamadas de *modelos de turbulência*. Entre os tipos de modelos mais utilizados em

CFD podemos destacar o *Reynolds-averaged Navier–Stokes* (RANS), de abordagem similar à exposta acima, o *Large Eddy Simulation* (LES) e o *Direct Numerical Simulation* (DNS). Maiores informações a respeito do fenômeno da turbulência e sua modelagem computacional podem ser encontrada em **[1].**

3. O OpenFOAM

3.1 – Visão geral do software.

O intuito deste capítulo é apresentação dos principais aspectos para o entendimento de como o software funciona e como deve ser operado. Iniciamos falando acerca de sua origem, em seguida tratamos de como os casos a serem solucionados são estruturados e por fim analisamos os métodos de discretização e de mapeamento de campos para escoamento multifásicos.

Muitos dos pontos mostrados a seguir, bem como outros aspectos relevantes do OpenFOAM®, podem ser encontrados no *User Guide* **[6]** e no *Programmer's Guide* **[7]** bem como no website *Notas em CFD* **[8].**

3.1. 1 - Um breve histórico.

Em 1993 os então alunos de doutorado do Imperial College, Henry Weller e Hrvoje Jasak decidiram combinar esforços para desenvolver um código chamado FOAM (*Field Operation And Manipulation*) para operação e manipulação de campos tensoriais visando a sua aplicação à fluidodinâmica computacional. A tecnologia do FOAM é baseada em um conjunto flexível de módulos escritos em C++ que são usados com o intuito de construir *solvers* para resolver problemas específicos de engenharia que envolvam campos tensoriais, *utilities* para realizar tarefas de pré e pós-processamento, que vão de uma simples manipulação de dados à visualização, construção e processamento de malhas e biblioteca necessárias.

Em 12 de dezembro de 2004 o código do FOAM se tornou de domínio público sobre a licença GNU passando a ser chamado de OpenFOAM®, uma vez que o código passara a ser aberto. Desde que seu código foi liberado, a comunidade científica passou a trata-lo com muito interesse utilizando-o em projetos de pesquisa e propondo inúmeras colaborações para o seu desenvolvimento. Foram criados grupos de discussão na internet para debater aspectos de uso, implementação de código e análise numérica do software. Proporcionando assim inúmeras colaborações e interações com o meio acadêmico. Hoje o OpenFOAM® é um pacote bastante robusto capaz de solucionar casos de alta complexidade (como escoamentos turbulentos, envolvendo troca de calor, multifásicos, etc), multiplataforma (rodando em Linux, Solaris, MacOS...) e capaz de

fazer uso de processamento em paralelo para realização de tarefas de grande demanda computacional.

3.2 – Casos no OpenFOAM®

3.2.1 – Estrutura dos casos.

A preparação de um caso para solução no OpenFOAM® implica necessariamente na criação de um diretório próprio, onde ficarão armazenados os arquivos necessários para a solução.

A figura abaixo mostra como devem ser organizadas as pastas, subpastas e demais arquivos.

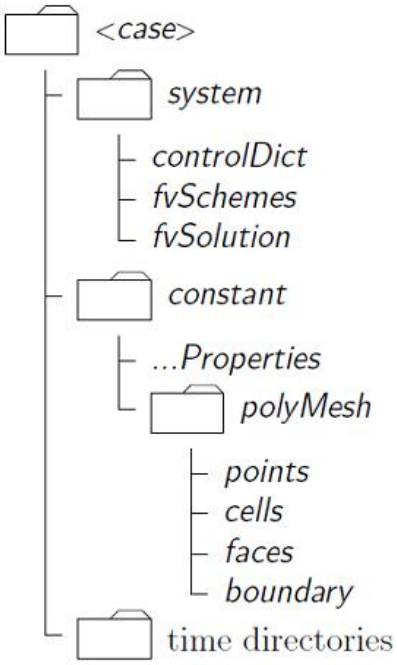


Figura 3.1 Estrutura da Pasta de um Caso no OF. (fonte: User Guide)

A descrição de cada diretório e seus componentes é dada a seguir.

3.2.1.1- O subdiretório *System*

No subdiretório *system* devem estar configurados os procedimentos de solução propriamente ditos.

O arquivo *controlDict* estabelece os parâmetros que controlam a resolução; tais como tempo inicial e final da solução, tamanho do passo de tempo , output e intervalo da escrita dos dados, número máximo de Courant e etc.

A seguir destacamos os principais pontos de um típico arquivo *controlDict* .

```
application     interFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         20;

deltaT          1e-03;

writeControl    adjustableRunTime;

writeInterval   0.05;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression compressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;

adjustTimeStep  yes;

maxCo           0.5;
```

Logo após o cabeçalho (aqui omitido) é encontrado o item *application* o qual determina para qual *solver* o *controlDict* esta sendo escrito. Os próximos cinco itens do código são usados para determinar como a solução irá evoluir com o tempo. O tempo de início da simulação é definido em *startFrom* por um valor numérico ou por uma

variável que neste caso é definida como *startTime* igual à zero. O tempo final pela variável *stopAt* definida pela outra variável *endTime,* igual à vinte segundos, que especifica que a simulação termina quando atingidos 20 segundos do tempo computacional.

O incremento no passo de tempo é definido pela variável *deltaT.* No exemplo acima temos um Δt igual a 0.001, mas pode-se adotar um valor variável ajustando o *adjustTimeStep* com *yes*. Dessa forma, o passo de tempo é ajustado de modo a cumprir o valor dado do número de Courant (*maxCo*). Sendo este o critério de estabilidade da solução numérica adotado no OpenFOAM®. Os dois outros itens destinados ao tratamento temporal da solução são *timeFormat* e *timePrecision*. Nestes o usuário determina se será usada notação científica e com quantas casas decimais deseja a escrita dos resultados.

O intervalo de tempo o qual os resultados serão escritos é definido pela variável *writeInterval,* aqui fixado em 0.05 segundos computacionais. A variável *purgewrite* limita a quantidade de dados que podem ser reescritos por vez. O formato dos arquivos a serem escritos (ASCII ou binário) são determinados em *writeFormat*. Os dados obtidos das soluções ainda podem ser escritos compactados (no formato *.gzip*) conforme a função *writeCompression* esteja configurada.

O próximo passo é a configuração do arquivo denominado *fvSchemes*. Nele são ditados os esquemas numéricos de discretização à serem utilizados nas equações governantes. Aqui o software permite uma grande liberdade de escolha dos esquemas de interpolação, de cálculo de gradientes, divergentes, laplacianos, derivadas temporais etc.

A figura 3.3 a seguir mostra a sintaxe utilizada para cada categoria de termos matemáticos seguido um fragmento de código *fvSchemes* ilustrativo respectivamente.

Keyword	Category of mathematical terms
interpolationSchemes	Point-to-point interpolations of values
snGradSchemes	Component of gradient normal to a cell face
gradSchemes	Gradient ∇
divSchemes	Divergence $\nabla \bullet$
laplacianSchemes	Laplacian ∇^2
timeScheme	First and second time derivatives $\partial/\partial t, \partial^2/\partial^2 t$
fluxRequired	Fields which require the generation of a flux

Figura 3.2 Sintaxe usada no fvSchemes (fonte:User Guide)

```
        ddtSchemes
{
   default       Euler;
}
  gradSchemes
{
   default       Gauss linear;
   grad(p)       Gauss linear;
}
  divSchemes
{
   default       none;
   div(phi,U)     Gauss linear;
}
  laplacianSchemes
{
   default       Gauss linear orthogonal;
}
  interpolationSchemes
{
   default       linear;
}
  snGradSchemes
{
   default       corrected;
}
  fluxRequired
{
   default       no;
    p           ;

          }
```

Pode-se notar no exemplo acima que o código possui seis funções do tipo *schemes* nas quais estão definidas quais métodos numéricos serão utilizados. O tempo será discretizado usando método de Euler implícito:

$$\frac{\partial}{\partial t}\int_V \rho\Phi\, dV = \frac{(\rho\Phi V)^{n+1} - (\rho\Phi V)^n}{\Delta t} \tag{3.1}$$

Os termos diferencias $\nabla\Phi$, $\nabla\cdot\Phi$,$\nabla^2\Phi$ serão submetido ao método de Gauss. Logo temos respectivamente:

$$\int_V \nabla\Phi\, dV = \int_S dS\,\Phi = \sum_f S_f \Phi_f \tag{3.2}$$

$$\int_V \nabla\cdot\Phi\, dV = \int_S dS\cdot\Phi = \sum_f S_f \cdot \Phi_f \tag{3.3}$$

$$\int_V \nabla\cdot(\Gamma\nabla\Phi)\, dV = \int_S dS\cdot(\Gamma\nabla\Phi) = \sum_f \Gamma S_f \cdot (\nabla\Phi_f) \tag{3.4}$$

Com Φ e Γ sendo a propriedade física de interesse e sua difusidade respectivamente.

A função do tipo *fluxRequired* especifica os campos os quais serão gerados fluxos durante a execução do solver. No exemplo do código acima, o campo em questão é o campo de pressão p. Encontra-se ainda a função *interpolationSchemes*. Nela, estão configurados os esquemas de interpolação calculados desde o centro da célula computacional até suas faces. O termo *linear* se refere à interpolação através do método de diferença central, porém há várias opções como esquemas de alta resolução *Total Variation Diminishing* (TVD) dentre outros. A *snGradSchemes* se refere

aos termos normais de gradiente. Estes são calculados nas faces da célula; são as componentes, normais à face, do gradiente dos valores no centro de duas células adjacentes conectadas por uma de suas faces. Assim, no exemplo acima, temos uma correção explícita não-ortogonal dada por *corrected*.

O arquivo *fvSolution* contém informação acerca da solução do sistema linear, algoritmos de cálculo, tolerâncias e fatores de relaxação. Para a solução de cada campo de interesse num determinado caso, é necessário que sejam configurados os métodos de solução das matrizes (simétricas e assimétricas) e os critérios de convergência numérica.

```
        solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }
      U
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-05;
        relTol          0;
    }
}
   PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}
```

No exemplo acima, serão resolvidas equações paras os campos de pressão p e velocidade U. Em *solver* é definido o método de solução da matriz. Para p usa-se o *Preconditioned Conjugate Gradient* (PCG), uma vez que as matrizes são simétricas, ao passo que em U usa-se o *Preconditioned Bi-Conjugate Gradient* (PBC) devido à assimetria de suas matrizes. O pré-condicionamento das matrizes é dado pelo

preconditioner de cada campo. Acima temos *Diagonal incomplete-Cholesky* (DIC) e *Diagonal incomplete-LU* (DILU) para p e $\boldsymbol{U}$ respectivamente.

Note que o termo *solver* aqui não se refere à um aplicação do software destinado à resolver um problema particular. Neste contexto, este se refere ao método de solução de um sistema linear. O sentido é diferente do que já foi visto anteriormente.

Antes de resolver uma equação para um campo específico, o resíduo inicial é avaliado com base nos valores atuais do campo. Após cada iteração, o resíduo é reavaliado. Tal ciclo é interrompido quando o resíduo atinge um valor menor do que a tolerância dada por *tolerance* ou a tiver sido reduzida, em cada intervalo de tempo, por um valor dado por *relTol* (*relative tolerance*).

A tolerância relativa representa o nível o qual o resíduo é suficientemente pequeno para que a precisão da solução possa ser considerada satisfatória. A variável *relTol* limita a melhoria relativa entre a solução inicial e final. Em simulações transientes, é usual definir a tolerância relativa nula. Os parâmetros *relTol* e *tolerance* são necessários em todos os *solvers* dos campos a serem resolvidos.

Finalmente, na parte inferior de *fvSolution* há a declaração de parâmetros para um algoritmo denominado PISO, ou *Pressure-Implicit Split-Operator*. Este algoritmo, assim como o *Semi-Implicit Method for Pressure-Linked Equations* (SIMPLE), é destinado à solução de problemas cujo as equações envolvam acoplamento pressão-velocidade. Sendo o PISO mais destinado a problemas transientes e o SIMPLE a casos estacionários. Ambos se baseiam na avaliação das soluções iniciais e realizam suas posteriores correções. O método SIMPLE realiza apenas uma correção ao passo que o PISO realiza múltiplas correções. O usuário deve portanto determinar quantas correções deverão ser feitas através da variável *nCorrectors* .

Uma correção adicional deve ser realizada para o casos de uma malha não-ortogonal. Esta é feita através do valor de *nNonOrthogonalCorrectors* que depende do grau de não ortogonalidade da malha. Para uma malha perfeitamente ortogonal teríamos um valor nulo, como mostrado. Porém, para o caso de um escoamento em torno de um cilindro por exemplo, é necessário fazer tal correção para que o valor da solução

numérica seja coerente com a solução analítica. As figuras 3.2 a seguir ilustram tal afirmação.

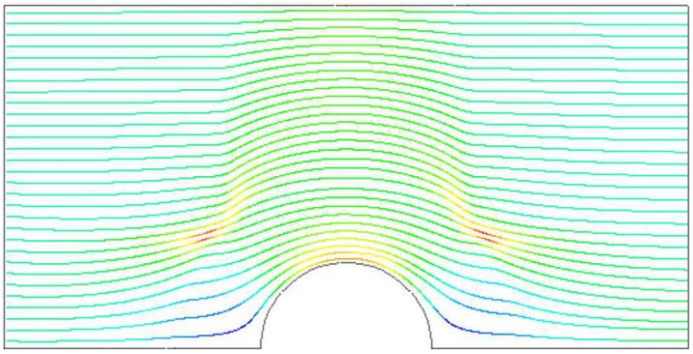


Figura 3.2(a) - Sem correção não-ortogonal

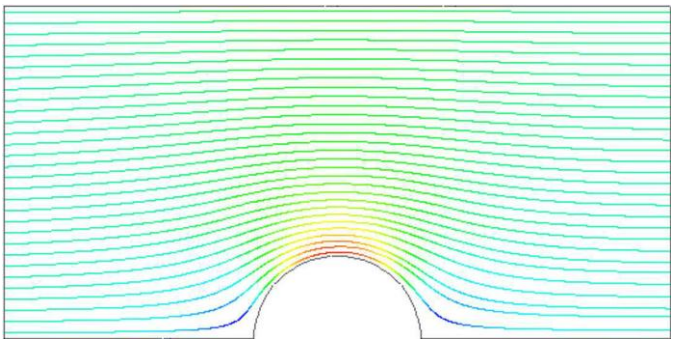


Figura 3.2(b) - Com correção não-ortogonal

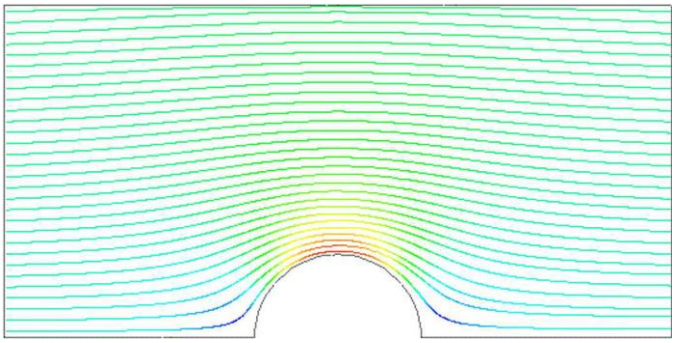


Figura 3.2(c) - Solução Analítica

Figuras 3.2 – Linhas de Fluxo em torno de um cilindro.

Finalizando, temos ainda as variáveis *pRefCell* e *pRefValue*. Estas são artifícios usados para estabilizar a solução numérica para casos incompressíveis. Uma vez que neles a pressão relativa é que é relevante, não seu valor absoluto.

3.2.1.2- O subdiretório *Constant*

Neste diretório ficarão armazenados os dados acerca da geometria do caso, a descrição das propriedades físicas (densidade, viscosidade, modelos de turbulência...) envolvidas, bem como das condições de contorno. Todas essas informações serão declaradas e configuradas através de rotinas chamadas de dicionários ou *Dicts*. No OpenFOAM® são utilizadas as unidades do Sistema Internacional (SI) para a descrição das devidas propriedades físicas para massa, volume etc.

Um *Dict* de uma certa propriedade possui as dimensões físicas desta e a declaração de seu valor. Definindo-se assim entre colchetes o expoente de cada dimensão em uma ordem bem definida, mostrada na tabela abaixo.

Ordem	**Propriedade**	**Unidades SI**
1	Massa	Quilograma (Kg)
2	Comprimento	Metro (m)
3	Tempo	Segundo (s)
4	Temperatura	Kelvin (K)
5	Quantidade	Quilograma-mole (Kgmol)
6	Corrente	Ampère (A)
7	Intensidade Luminosa	Candela (cd)

Tabela - 3.1

Por exemplo, caso desejássemos declarar o valor da aceleração da gravidade g como sendo 9.81 m/s^2 na direção $\hat{\boldsymbol{z}}$, a sintaxe deve ser:

```
g                    g [0 1 -2 0 0 0 0] (0 0 -9.81)
```

A sequencia limitada pelos colchetes indica a potência das unidades dimensionais da propriedade em questão, no caso $M \cdot T^{-2}$. Já a sequência entre colchetes define o vetor aceleração gravitacional $\boldsymbol{g}$, sendo *0,0* e *-9.81* as respectivas coordenadas nos eixos cartesianos. Desta maneira, em casos em que se fizer necessário, podemos descrever a viscosidade cinemática *v* da água como:

```
nu                   nu[0 2 -1 0 0 0 0] 1e-06
```

Uma vez que no SI temos que as dimensões de ν são $M^2 \cdot T^{-1}$.

No OpenFOAM® grandezas tensoriais são declaradas de maneira semelhante, através de suas nove coordenadas. O tensor simétrico identidade

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

tem a sua sintaxe dada por (1 0 0 0 1 0 0 0 1) descrevendo as respectivas linhas do mesmo.

Na subpasta *polyMesh* estão guardadas as informações sobre geometria e a malha do caso. O domínio computacional é considerado como a composição de blocos definidos nas três dimensões cartesianas por vértices, arestas e faces.

O *blockMeshDict* contém definições da malha através de instruções *convertToMeters, vertices, edges, blocks, patches, mergePatchPairs*. A seguir exemplificaremos as instruções necessárias para criação de uma geometria de um cubo e definição da malha.

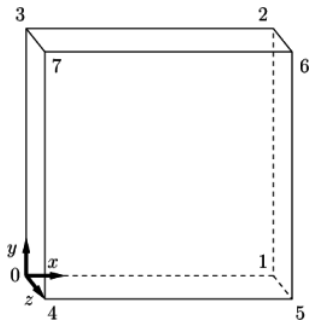


Figura 3.3 Determinação dos vértice de um cubo no *blockMeshDict*

```
FoamFile
{
    version    2.0;
    format     ascii;
    class      dictionary;
    object     blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
 convertToMeters 0.1;
 vertices
(
    (0 0 0)  // 0
    (1 0 0)  // 1
    (1 1 0)  // 2
```

```
    (0 1 0)  // 3
    (0 0 1)  // 4
    (1 0 1)  // 5
    (1 1 1)  // 6
    (0 1 1)  // 7
  );
   blocks
  (
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
  );
   edges
  (
  );
   boundary
  (
    movingWall
    {
      type wall;
      faces
      (
        (3 7 6 2)
      );
    }
    fixedWalls
    {
      type wall;
      faces
      (
        (0 4 7 3)
        (2 6 5 1)
        (1 5 4 0)
      );
    }
    frontAndBack
    {
     type empty;
      faces
      (
        (0 3 2 1)
        (4 5 6 7)
      );
    }
  );
   mergePatchPairs
  (
  );
   //
*********************************************************************//
```

Por padrão, as dimensões são medidas em metros. Caso se pretenda uso de outras unidades utiliza-se na instrução *convertToMeters* um valor proporcional de conversão. No código ilustrado acima, a dimensão trabalhada é o decímetro.

Os vértices são representados através de suas coordenadas cartesianas *x, y, z*. Uma vez definidos os vértices, estes são numerados a partir do zero. Em *blocks* é efetuada a construção e divisão do bloco geométrico em células computacionais com o número e o espaçamento escolhido. A instrução *hex* indica que as células serão hexaedricas, mas o OpenFOAM® também trabalha com células prismáticas, tetraédricas, piramidais dentre outras. A sequencia *(0 1 2 3 4 5 6 7)* define o bloco através dos seus oito vértices. Aqui é necessário ter cuidado com a declaração na sequencia dos pontos, tendo que se manter a mesma para cada face. A divisão do bloco em células é descrita imediatamente a seguir; sendo assim dividiu-se o bloco em 20 células nas direções $\hat{\boldsymbol{x}}$ e $\hat{\boldsymbol{y}}$ e nenhuma na direção $\hat{\boldsymbol{z}}$.

O refinamento da malha é feito usando uma relação de expansão/contração de cada direção. O termo *simpleGrading* implica num vetor que relacionará as dimensões relativas entre as células finais iniciais em cada direção. Para o caso de *simpleGrading*(1 2 3), temos que em $\hat{\boldsymbol{x}}$ a célula final tem dimensões iguais a inicial. Já em $\hat{\boldsymbol{y}}$ e $\hat{\boldsymbol{z}}$, as últimas células terão respectivamente o dobro e o triplo das iniciais. Porém, no caso aqui ilustrado temos isonomia no tamanho das células. Em *edges,* são configurados os comandos para criação de arestas curvas a partir de dois vértices. No caso de arestas retas, deixa-se vazio. O próximo passo é a descrição das fronteiras da malha, determinando e nomeando as faces que as compõe. A descrição geométricas das faces é feita através dos seus quatro pontos. Estes deverão ser introduzidos no sentido horário ou anti-horário dependendo da orientação desejada para o vetor normal à face Assim começamos a descrever o objeto *boundary* com a sequência de nomes dados a cada face, seu tipo, bem como os pontos que as definem.

```
fixedWalls
    {
      type wall;
      faces
      (
        (0 4 7 3)
        (2 6 5 1)
        (1 5 4 0)
      );
     }
```

As faces são nomeadas de *fixedWalls*. Em *type* define-se o tipo de face *wall* indica que a face se trata de uma parede. Pode-se, aqui, definir planos de simetria, condições periódicas etc. A seguir, lista-se os pontos que darão origem às faces desejadas com essas características. Assim sucessivamente pra todas as demais faces. A função *mergePatchPairs* é utilizada quando se deseja fundir duas ou mais células que dividem uma mesma face interna. Quando não declarada, as faces internas em contato são ignoradas.

Ao se gerar malhas através do *blockMesh,* criam-se os arquivos *points, faces, cells, boundary*. Listando respectivamente os pontos ordenados correspondentes aos vértices das células enumerados a partir do zero, as faces que correspondem à lista de pontos, as células com os respectivos índices e as faces definidas pelos procedimentos acima.

3.2.1.3 - Os subdiretórios tempo

Os diretórios de tempo contêm os arquivos e dados individuais de cada variável física envolvida na solução do caso. O conteúdo neste diretório se refere às informações acerca das condições iniciais e de contorno que necessitam ser declaradas antes do inicio da solução, bem como os resultados obtidos durante a resolução ao longo do tempo. O nomes com os quais são gravados os dados aos diretórios oriundos do processos de solução, assim como o intervalo entre eles, serão aqueles especificados em *system/controlDict*. Uma vez que a solução seja iniciada em $t = 0$, as condições iniciais do problema serão aquelas que constam na pasta *0* , ou *0.00000e+00* dependendo de como o formato tenha sido especificado. Da mesma maneira, informações a respeito do campo de velocidade $\boldsymbol{U}$e pressão p são encontradas em $0/U$ e $0/p$ respectivamente.

Os dados de entrada e saída no OpenFOAM® são em sua maioria campos tensoriais que são lidos/escritos a partir dos diretórios de tempo. O registro deles é feito inicialmente declarando as dimensões físicas da variável, seu valor inicial no interior do domínio computacional e a declaração das suas condições de contorno.

Abaixo examinaremos as declarações para os campos de velocidade $\boldsymbol{U}$ e pressão p respectivamente para o caso *cavity* presente nos tutoriais do software.

```
FoamFile
{
   version     2.0;
   format      ascii;
   class       volVectorField;
   object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);
boundaryField
{
   movingWall
     {
      type          fixedValue;
      value         uniform (1 0 0);
      }
   fixedWalls
     {
      type          fixedValue;
      value         uniform (0 0 0);
      }
   frontAndBack
     {
      type          empty;
     }
}
```

```
FoamFile
{
   version     2.0;
   format      ascii;
   class       volScalarField;
   object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];
internalField   uniform 0;
boundaryField
{
   movingWall
     {
      type          zeroGradient;
      }
   fixedWalls
     {
      type          zeroGradient;
      }
   frontAndBack
     {
      type          empty;
     }
```

```
}
```

Neste caso, o contorno consiste apenas em paredes, divididas em dois conjuntos. Um primeiro denominado *fixedWalls* que são as laterais fixas e a parte inferior da geometria (mostrada abaixo), e outro chamado de *movingWall* que caracteriza o topo móvel.

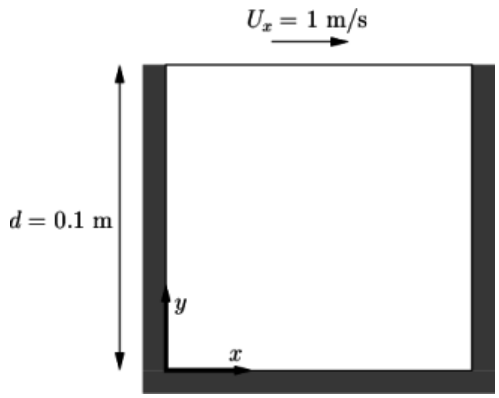


Figura 3.4 Geometria do caso *Cavity*

Logo após o cabeçalho, encontra-se a declaração das dimensões dos respectivos campos. Sendo $L \cdot T^{-1}$ para a velocidade e $M \cdot L^{-1} \cdot T^{-2}$ para a pressão. A seguir, temos a definição do valor interno de cada campo. O termo *uniform* implica que um mesmo valor inicial é definido pra cada elemento computacional do campo, convenientemente define-se como zero. Nota-se que em **U** tem que ser declarar tal valor nulo em suas coordenadas uma vez que se trata de um campo vetorial.

O campo de velocidade na parte superior é definido como sendo fixo possuindo um valor constante de 1 m/s na direção $\hat{x}$. Nas laterais e no fundo ele possui valor inicial nulo. O contorno frontAndBack se refere as parte frontal e traseira da geometria. Como se trata de um caso bidimensional, assume-se que haja simetria em $\hat{z}$. Tal configuração é dada pelo comando *empty* neste contorno em ambos os campos. Na descrição das condições de contorno para a pressão temos o comando *zeroGradient*. Este indica que a normal de ∇p nas paredes é nula.

Com isso, termina-se a descrição de como um caso é estruturado no OpenFOAM®. Considera-se uma das grandes desvantagens do software essa fragmentação da inserção dos diferentes parâmetros para a solução de um caso. Uma interface gráfica nesta etapa seria de grande valia. No capítulo 04 faremos o estudo de

caso dos dois problemas descritos na introdução do trabalho. Veremos seus pormenores e também a analise dos resultados da simulação através do software Paraview®.

3.2 Método dos Volumes Finitos: Uma breve introdução.

O método dos volumes finitos (MVF) é um dos mais versáteis métodos de discretização usados em CFD. Comumente confundido com o método das diferenças finitas, pelo fato que as equações discretizadas resultantes de ambos serem semelhantes. O MVF é baseado em princípios físicos de conservação; ao passo que o MDF usa aproximações de derivadas a partir de séries de Taylor. Devida à sua grande complexidade, faremos aqui uma abordagem unicamente introdutória ao assunto, com o intuito meramente ilustrativo. Maiores detalhes a respeito do método e suas peculiaridades podem ser encontrados em **[1]** e **[8]**.

Quando uma grandeza física (tal como massa, energia, momento...) é expressa matematicamente num volume de controle, obtemos uma equação diferencial que determina a conservação da grandeza envolvida. Tal procedimento apoia-se no fato que no MVF os domínios de cálculo são divididos num certo número de subdomínios nos quais as leis de conservação serão impostas. A figura 3.5 representa um subdomínio bidimensional por onde há um escoamento $\boldsymbol{U} = u\hat{x} + v\hat{y}$ onde estão representados o seu centro P, a posição de suas quatro face n, s, e e w $(north, south, east, west)$ e o fluxo que às atravessa num dado intervalo de tempo Δt.

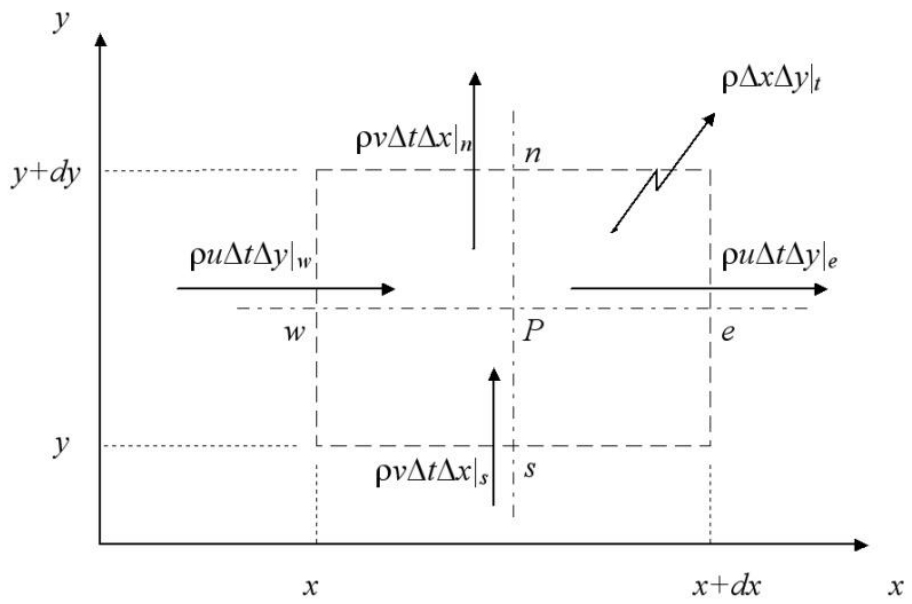


Figura 3.5 Subdomínio bidimensional atravessado por um fluxo ρU

A quantidade de massa dentro desse volume de controle será dada por:

$$(\rho\Delta x\Delta y\Delta t)_{t+\Delta t} - (\rho\Delta x\Delta y\Delta t)_t = (\rho u\Delta y\Delta t)_w - (\rho u\Delta y\Delta t)_e + (\rho v\Delta x\Delta t)_s - (\rho uvx\Delta t)_n \quad (3.5)$$

Dividindo a quantidade acima por $\Delta x\Delta y\Delta t$ obtemos:

$$\frac{\rho_{t+\Delta t} - \rho_t}{\Delta t} = \frac{(\rho u)_w - (\rho u)_e}{\Delta x} + \frac{(\rho v)_s - (\rho v)_n}{\Delta y} \quad (3.6)$$

Rearranjando os termos, nos fornece:

$$\frac{\rho_{t+\Delta t} - \rho_t}{\Delta t} + \frac{(\rho u)_e - (\rho u)_w}{\Delta x} + \frac{(\rho v)_n - (\rho v)_s}{\Delta y} = 0 \quad (3.7)$$

Esta é a forma discretizada da equação da continuidade (2.14) para o volume finito em questão.

É interessante notar que esse é o resultado exato para a conservação da massa dentro do volume. Não foi feita qualquer aproximação de nenhuma natureza. Agora ao se falar de fluxo médio através das faces e de densidade médias no volume deverão ser feitas aproximações fazendo uso de seus valores em pontos discretos da malha para um dado instante de tempo. Tal prática é uma das fontes de erros na aproximação numérica.

Na figura 3.6 é mostrada a projeção unidimensional de um volume de controle. Usualmente os valores das grandezas físicas de interesse são armazenadas nos centros dos volumes (W, P, E) ou no centro das faces adjacentes (w, e). Então se fazendo uso de esquemas de interpolação são obtidos os valores destas grandezas em outros pontos do domínio.

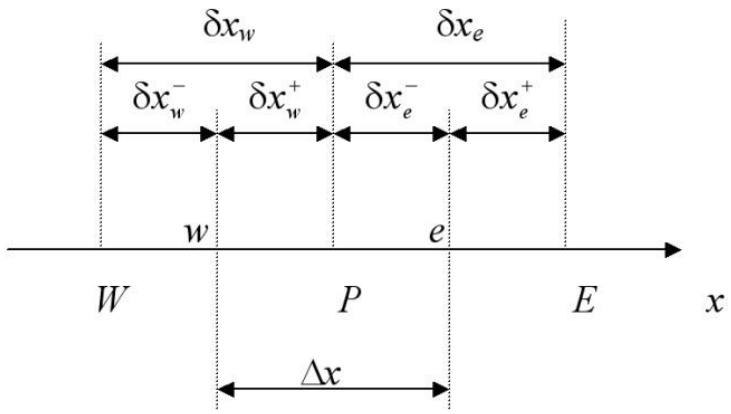


Figura 3.6 Centros, faces e incrementos δx de três células computacionais consecutivas.

Pode-se ainda, armazenar valores nos centroides dos volumes em certas regiões da malha enquanto que em outras tais valores são armazenados nos centros das faces. Assim, o MVF permite usar malhas com estruturas diferentes num mesmo domínio computacional. O que representa uma grande vantagem para problemas de elevada complexidade. O mesmo princípio aqui mostrado é aplicado às demais equações relevantes à problemas de CFD, tais como a Equação do Transporte, Navier-Stokes etc.

3.3 Método *Volume Of Fluid* (VOF)

O método de VOF é o método utilizado pelo OpenFOAM® para problemas de escoamento multifásico onde se tem interesse em determinar a superfície livre entre as fases. A ideia principal é a definição da *função de fase* α. Esta nos permite determinar quais células estão ocupadas por uma determinada fase do escoamento e quais estão sendo ocupadas por ambas. No caso de um escoamento com duas fases distintas (líquido e gás, por exemplo) atribui-se o valor unitário à função de fase ($\alpha = 1$) caso uma certa célula computacional num dado instante de tempo esteja preenchida pelo líquido. Caso uma célula esteja repleta de gás defini-se $\alpha = 0$. Na interface entre as duas fases, isto é, nas células onde há a presença de ambas as fases a função α assume valores no intervalo $0 < \alpha < 1$. Resumidamente temos,

$$\alpha = \begin{cases} 1, \text{ para a fase líquida} \\ 0, \text{ para a fase gasosa} \\ 0 < \alpha < 1, \text{ para a interface} \end{cases}$$

Os dois fluidos (imiscíveis) serão tratados daqui em diante como um único fluido através de todo domínio computacional. As propriedades físicas deste novo fluido serão calculadas a partir da média ponderada da distribuição de α. Assim nas células completamente preenchidas por uma das fases tais propriedades se mantêm, variando assim somente na interface. As equações (3.8) quantificam esse raciocínio para a massa específica ρ e para viscosidade μ;

$$\rho = \rho_l \alpha + \rho_g (1 - \alpha) \tag{3.8a}$$

$$\mu = \mu_l \alpha + \mu_g (1 - \alpha) \tag{3.8b}$$

onde ρ_l, ρ_g, μ_l e μ_g são as massas específicas e a viscosidade das fases líquidas e gasosas respectivamente. Observa-se que no limite $\alpha \rightarrow 1$ as equações acima nos dão ρ e μ da fase líquida bem como para $\alpha \rightarrow 0$ obtemos ρ e μ da fase gasosa.

No método VOF resolvemos as equações de Navier-Stokes (2.9) e da continuidade (2.14) para ambas as fases. Mas assim como foi feito na seção 2.3,

também se faz necessário corrigir a equação (2.9) adicionando um termo oriundo da tensão superficial. Então com ρ e μ assumindo as formas dadas pelas equações (3.8), a equação para o movimento (2.9) é reescrita na forma:

$$\rho\, {D\boldsymbol{U}}/{Dt} = -\nabla p + \nabla^2 \boldsymbol{U} + \rho \boldsymbol{g} - \boldsymbol{F}_s \tag{3.9}$$

onde $\boldsymbol{F}_s$ é o termo extra adicionado de força da tensão superficial atuante somente na superfície livre.

Assim como ρ e μ foram definidas em função de α, o campo de velocidade **U** também o é. Portanto, da mesma forma temos:

$$\boldsymbol{U} = U_l \alpha + U_g (1 - \alpha) \tag{3.10}$$

onde U_l e U_g são os campos de velocidade de fases líquida e gasosa respectivamente.

Um dos grandes problemas em simulações numéricas de superfícies livre usando o VOF é a conservação da função de fase. Um cálculo preciso da função de fase em todo o domínio é crucial para uma solução apropriada da curvatura da superfície, uma vez que esta é necessária para a determinação da tensão superficial e o gradiente de pressão correspondente através da superfície livre.

Para garantir fidelidade física aos resultados o OF (através do solver *interFoam*) usa uma abordagem que consiste na adição de um termo convectivo, a partir da nova definição de **U** dada em (3.10), na equação da continuidade para α, permitindo assim uma resolução mais nítida da interface. As equações da continuidade para ambas as fases são:

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\boldsymbol{U_l} \alpha) = 0 \tag{3.11a}$$

$$\frac{\partial (1 - \alpha)}{\partial t} + \nabla \cdot [\boldsymbol{U_g} (1 - \alpha)] = 0 \tag{3.11b}$$

Uma vez que:

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\boldsymbol{U}\alpha) = 0 \tag{3.11c}$$

temos:

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot \{[\boldsymbol{U}_l \alpha + \boldsymbol{U}_g (1-\alpha)]\alpha\} = 0 \tag{3.11d}$$

Definindo-se aqui uma velocidade relativa entre as duas fases:

$$\boldsymbol{U}_r = \boldsymbol{U}_l - \boldsymbol{U}_g \tag{3.11e}$$

e substituindo $\boldsymbol{U}_g$ na equação anterior temos:

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot \{[\boldsymbol{U}_l - \boldsymbol{U}_r (1-\alpha)]\alpha\} = 0 \tag{3.11f}$$

Rearranjando os termos temos:

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\boldsymbol{U}_l \alpha) - \nabla \cdot [\boldsymbol{U}_r (1-\alpha)\alpha] = 0 \tag{3.11g}$$

Os dois primeiros termos a esquerda da equação acima são nulos, portanto ficamos apenas com:

$$\nabla \cdot [\boldsymbol{U}_r (1-\alpha)\alpha] = 0 \tag{3.11h}$$

Finalmente, adicionamos esse termo à equação do transporte para a função α e obtemos:

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\boldsymbol{U}\alpha) + \nabla \cdot [\boldsymbol{U}_r (1-\alpha)\alpha] = 0 \tag{3.11i}$$

O termo adicionado (3.11h) não contém qualquer significado na formulação analítica, uma vez que a superfície livre tem uma espessura finíssima, $\boldsymbol{U}_r$ pode ser

considerada nula e voltamos à equação (3.11c). Ainda assim a adição de (3.11h) é de grande utilidade na abordagem discreta da superfície livre.

A tensão superficial na interface líquido-gás gera um gradiente de pressão adicional dando originando uma força dada por:

$$\boldsymbol{F}_S = \rho\kappa\nabla\alpha \tag{3.12}$$

Onde, κ é a curvatura da superfície livre que é obtida se fazendo:

$$\kappa = \nabla \cdot \hat{\boldsymbol{n}} \tag{3.13}$$

Onde $\hat{\mathbf{n}}$ é vetor unitário normal à superfície livre dado por:

$$\hat{\boldsymbol{n}} = \frac{\nabla\alpha}{|\nabla\alpha|} \tag{3.14}$$

O OF realiza todo esses procedimentos através de sofisticadas rotinas de C/C++. Estas serão mostradas no próximo capítulo onde serão resolvidos casos práticos de escoamento multifásico através do solver *interFoam*.

4 Casos Práticos: ESCOAMENTO MULTIFÁSICO NO OPENFOAM®

O objetivo deste capítulo é mostrar a importância e utilidade que simulações de CFD podem ter para tratar de casos de aplicação prática. O software é capaz de resolver uma ampla de escoamentos envolvendo duas ou mais fases. Entre eles podemos destacar casos compressíveis, mudança de fase (cavitação), escoamento multifásico em meios porosos, com malha móvel dentre outros.

Baseado no que já foi exposto anteriormente analisaremos dois casos de escoamento laminar multifásico. Primeiramente nos debruçaremos sobre uma simulação bidimensional no qual um recipiente inicialmente repleto de ar passa a ser cheio de água. Este, que recebeu o nome de *Bottle*, trata-se de um tutorial que poder ser encontrado na internet, ver **[4]**. O código usado no presente trabalho não é exatamente igual ao produzido pelo autor em 2008, uma vez que foram necessários alguns ajustes devido à diferença de sintaxe entre a versão da época e a versão 2.2.2 aqui utilizada.

Em seguida será a vez de estudar um caso tridimensional, portanto de maior complexidade e demanda computacional, onde será simulada a ruptura parcial da parede de uma barragem, daí sendo batizado de *Dam 3D*. Este foi resultado de uma experiência junto ao Depto de Engenharia Civil da Universidade de Coimbra, enquanto aluno daquela instituição durante graduação-sanduíche**[5]**.

Na exposição dos casos a seguir serão omitidas algumas partes dos códigos referentes diretórios /system e */constant*, com exceção dos *blockMeshDict* utilizados e o setFieldsDict.

4.1 O caso Bottle.

4.1.1 Descrição, Configuração e Solução.

O problema aqui simulado consiste no preenchimento de um recipiente inicialmente com ar com água, a partir de um dado intervalo de tempo. Por razões de simplicidade iremos considerar que a geometria tem dimensão infinita na direção $\hat{\boldsymbol{z}}$. O domínio computacional é constituído por duas regiões diferentes: uma de onde a água será injetada chamada inlet, e a região do recipiente bottle. As dimensões geométricas são mostradas na figura 4.1. Num dado instante de tempo, considerado como zero, a água será injetada verticalmente para baixo e se acumulará dentro do recipiente. Foi atribuída uma velocidade inicial à água de 0.1m/s na direção $-\hat{\boldsymbol{y}}$.

Como dito em capítulos anteriores, o OpenFOAM® permite a resolução de escoamentos turbulentos através de abordagens já conhecidas como LES e RANS, porém aqui o escoamento será tratado como laminar a fim de simplificação.

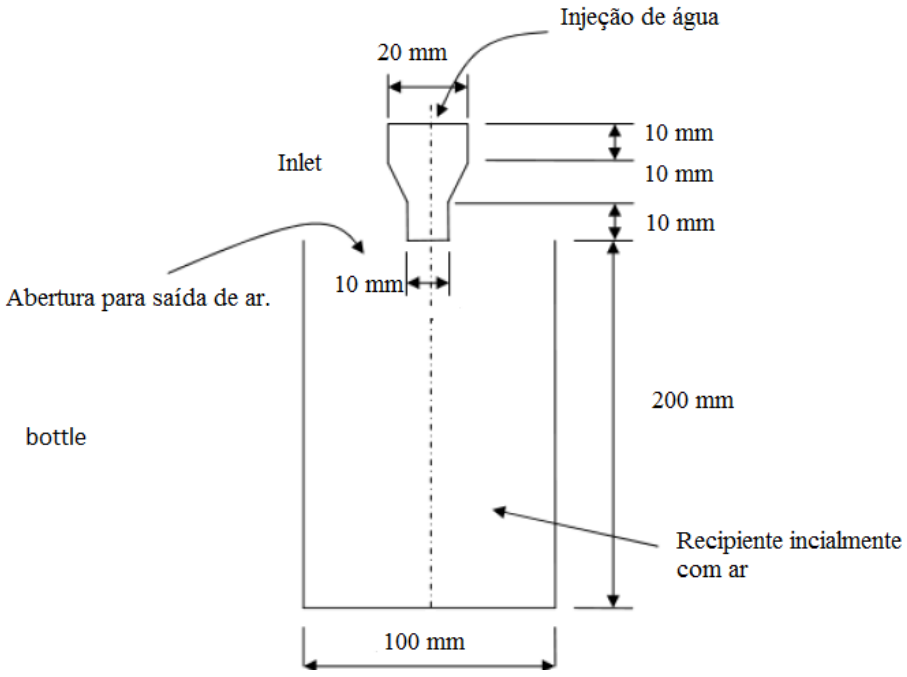


Figura 4.1 Geometria do problema (adaptado do original).

Um procedimento de como resolver o caso acima o tratando como turbulento pode ser encontrado no original **[4]**, no qual o autor deste trabalhou usou RANS para calcular os efeitos oriundos da turbulência.

Assim como exposto no capítulo anterior, temos que configurar o caso inserindo os dados relacionados com as condições iniciais e de contorno, propriedades físicas e de controle da solução numérica nos diretórios */0, /constant* e */system* respectivamente. Uma vez que os fluidos que compõe o escoamento são água e ar, temos no *transportProperties* dois fluidos considerados newtonianos. Com seus valores de viscosidade e massa específica sendo declarados logo em seguida. A entrada *CrossPowerLawCoeffs* somente será utilizada caso seja tomado outro modelo de transporte não newtoniano. Finalizando, é declarado o coeficiente de tensão superficial entre as fases. Todo o conjunto de configurações citados são mostrados no código a seguir.

```
 FoamFile
{
   version    2.0;
   format     ascii;
   class      dictionary;
   location   "constant";
   object     transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

phase1 // Água
{
   transportModel  Newtonian; // Fluido Newtoniano
   nu              nu [ 0 2 -1 0 0 0 0 ] 1e-06;
   rho             rho [ 1 -3 0 0 0 0 0 ] 1000;
   CrossPowerLawCoeffs
   {
      nu0           nu0 [ 0 2 -1 0 0 0 0 ] 1e-06;
      nuInf         nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
      m             m [ 0 0 1 0 0 0 0 ] 1;
      n             n [ 0 0 0 0 0 0 0 ] 0;
   }

   BirdCarreauCoeffs
   {
      nu0           nu0 [ 0 2 -1 0 0 0 0 ] 0.0142515;
      nuInf         nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
      k             k [ 0 0 1 0 0 0 0 ] 99.6;
      n             n [ 0 0 0 0 0 0 0 ] 0.1003;
   }
}

phase2 // Ar
{
   transportModel  Newtonian; // Fluido Newtoniano
   nu              nu [ 0 2 -1 0 0 0 0 ] 1.48e-05;
   rho             rho [ 1 -3 0 0 0 0 0 ] 1;
   CrossPowerLawCoeffs
   {
      nu0           nu0 [ 0 2 -1 0 0 0 0 ] 1e-06;
      nuInf         nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
      m             m [ 0 0 1 0 0 0 0 ] 1;
```

```
    n           n [ 0 0 0 0 0 0 0 ] 0;
  }

  BirdCarreauCoeffs
  {
    nu0          nu0 [ 0 2 -1 0 0 0 0 ] 0.0142515;
    nuInf        nuInf [ 0 2 -1 0 0 0 0 ] 1e-06;
    k            k [ 0 0 1 0 0 0 0 ] 99.6;
    n            n [ 0 0 0 0 0 0 0 ] 0.1003;
  }
}

sigma         sigma [ 1 0 -2 0 0 0 0 ] 0.07; // tensão superficial entre os 2 fluidos
```

A seguir o que campo gravitacional deve ser configurado. Assim como no exemplo do que foi mostrado no capítulo 03, o arquivo ***g*** é configurado do seguinte modo.

```
FoamFile
{
  version    2.0;
  format     ascii;
  class      uniformDimensionedVectorField;
  location   "constant";
  object     g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions     [0 1 -2 0 0 0 0];


value         ( 0 -9.81 0 ); // definição da aceleração da gravidade no sentido negativo de 'y'
```

Notando que nos códigos exibidos, tudo que vem a direita das barras duplas (//) são somente comentários, uma vez que se usa a linguagem de programação C++. Logo não são interpretados em nenhum instante.

Sendo o escoamento laminar, essa informação precisa estar declarada em *turbulenceProperties*.

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
//

simulationType  laminar;
```

Caso se deseje usar algum modelo de turbulência deve-se usar *LESModel* ou *RASModel* aqui.

Dependendo da versão do OF utilizada ainda há no diretório */constant* outro arquivo chamado *dynamicMeshDict*, cuja função é especificar se a malha é móvel; isto é se ela se deformará durante a solução, ou se ela é fixa. No escopo do presente trabalho não há necessidade de se usar malha móveis, o que aumenta muito a complexidade do caso e o esforço computacional para sua solução. Logo a malha é declarada fixa através da expressão *staticFvMesh* como mostrado abaixo.

```
FoamFile
{
   version    2.0;
   format     ascii;
   class      dictionary;
   location   "constant";
   object     dynamicMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

dynamicFvMesh   staticFvMesh;
```

Ainda dentro de */constant* há o subdiretório */polyMesh* onde devemos configurar a geometria do caso e os parâmetros de sua malha. O arquivo criado pelo autor é mostrado a seguir.

```
FoamFile
{
   version    2.0;
   format     ascii;
   class      dictionary;
   object     blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 0.001;

vertices
(
   ( -10 230 0 )    //0
   (10 230 0  )     //1
   (10 220 0  )     //2
   (-10 220 0 )     //3
   ( 5  210  0 )    //4
   (-5 210 0 )      //5
   ( 5 200 0 )      //6
   (-5 200 0 )      //7
   ( 5 0 0 )        // 8
   ( -5 0 0 )       // 9
   ( 50 200 0 )     //10
   ( 50 0 0 )       //11
   ( -50 200 0 )    //12
```

```
    ( -50 0 0 )        //13
    ( -10 230 1)     //14
    ( 10 230 1 )      //15
    ( 10 220 1 )      //16
    (-10 220 1 )     //17
    ( 5 210 1 )       //18
    (-5 210 1 )       //19
    ( 5 200 1 )       //20
    (-5 200 1 )      //21
    ( 5 0 1 )          //22
    ( -5 0 1 )         //23
    (50 200 1 )     //24
    (50 0 1)          //25
    (-50 200 1)     //26
    (-50 0 1)        //27
);

blocks
(
    hex (3 2 1 0 17 16 15 14) (10 5 1) simpleGrading (1 1 1)          // 1
    hex (5 4 2 3 19 18 16 17) (10 5 1) simpleGrading (1 1 1)          //2
    hex (7 6 4 5 21 20 18 19) (10 5 1) simpleGrading (1 1 1)          //3
    hex (9 8 6 7 23 22 20 21) (10 100 1) simpleGrading (1 1 1)      //4
    hex (8 11 10 6 22 25 24 20) (40 100 1) simpleGrading (1 1 1)  //5
    hex (13 9 7 12 27 23 21 26) (40 100 1) simpleGrading (1 1 1)  //6
);
edges
(

boundary
(
    inlet
    {
        type wall;
        faces
        (
            (0 1 15 14)

        );
    }
    inletWall
    {
        type wall;
        faces
        (
            (0 3 17 14)
            (3 5 19 17)
            (5 7 21 19)
            (1 2 16 15)
            (2 4 18 16)
            (4 6 20 18)
        );
    }
    bottleWall
    {
        type wall;
        faces
        (
            (10 11 25 24)
            (11 8 22 25)
            (8 9 23 22)
            (9 13 27 23)
```

```
        (13 12 26 27)
    );

}
  atmosphere
  {
    type patch;
    faces
    (
      (6 10 24 20)
      (12 7 21 26)

    );
  }
    frontAndBack
  {
    type empty;
    faces
    (
      (0 1 2 3)
      (3 2 4 5)
      (4 6 7 5)
      (6 10 11 8)
      (6 8 9 7)
      (7 9 13 12)
      (14 15 16 17)
      (17 16 18 19)
      (19 18 20 21)
      (20 24 25 22)
      (20 22 23 21)
      (21 23 27 26)
    );
  }
);
mergePatchPairs
(
```

Os vértices da geometria são as primeiras entradas, em seguida são declarados os blocos do domínio, a geometria de suas células e a divisão nos três eixos. É notório também que as dimensões estão sendo declaradas em milímetros.

Neste caso, temos um domínio composto por seis blocos como mostrado na figura 4.2 a seguir.

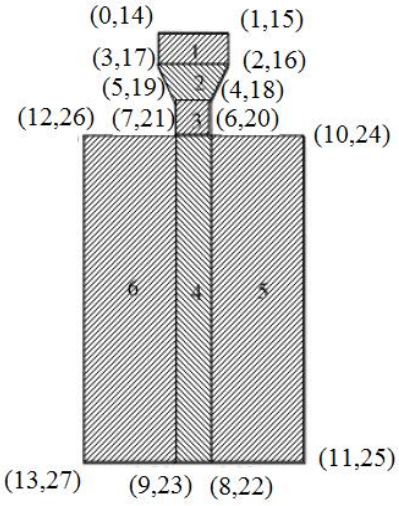


Figura 4.2 Diferentes blocos do domínio (adaptado do original)

A figura mostra a numeração do vértice posterior e anterior em casa extremidade do domínio.

Em seguida, são definidos os patches em função dos blocos. Estes são inlet, inletWall, bottleWall, atmosphere, e frontAndBack que definem a injeção de água, as paredes do recipiente, as paredes do bico injetor, a abertura do recipiente para a atmosfera e as faces frontais e traseiras do mesmo respectivamente. Vale notar que a sequencia dos vértices na declaração do blocos segue a regra da mão direita. Esta sendo adotada como padrão no OF

Agora em */system* definiremos os parâmetros de controle da solução como já mostrado na seção 3.2.1.1, porém aqui temos uma peculiaridade. Em casos de escoamentos multifásicos, temos que indicar inicialmente quais as regiões do domínio computacional estão preenchidas com as determinadas fases. Isto é, no instante inicial qual a posição da água e o ar. Isto se dá através do arquivo chamado *setFieldsDict* mostrado a seguir.

```
FoamFile
{
   version    2.0;
   format     ascii;
   class      dictionary;
   location   "system";
   object     setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
defaultFieldValues
(
   volScalarFieldValue alpha1 0
);

regions
(
   boxToCell
   {
      box (-0.01 0.2  0) (0.01 0.23 1);
      fieldValues
      (
         volScalarFieldValue alpha1 1
      );
   }
);


// ************************************************************************* //
```

O trecho definido por *defaultFieldValues* indica o valor padrão para todo o domínio computacional. Isto é, todo espaço estará preenchido por ar através da função *volScalarFieldValue alpha1* igual à zero. A região do espaço que inicialmente é ocupada por água é tratada como uma exceção da definição anterior. Em *regions* definiremos uma região delimitadora através das coordenadas dos dois vértices opostos de sua diagonal. Esses não necessariamente precisam ser os vértices antes definidos no *blockMeshDict*. No caso temos os vértices dados pelas coordenadas $(-0.01\ 0.2\ \ 0)\ (0.01\ 0.23\ 1)$ que corresponde aos vértices 3 e 5 respectivamente, já tendo sofrendo a conversão para milímetros estipulada no *blockMeshDict*. Mais um peculiaridade de casos multifásicos no OF é a declaração da função alpha em conjunto dos demais campos **U** e p em */0* através do arquivo *alpha1*. Esta, assim como as demais, precisa ter configuradas as condições de contorno em cada face da malha.

```
FoamFile
{
   version    2.0;
   format     ascii;
   class      volScalarField;
   object     alpha;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions     [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
   inlet
```

```
    {
        type            fixedValue;
        value           uniform 1;
    }

    inletWall
    {
        type            zeroGradient;
    }

    bottleWall
    {
        type            zeroGradient;
    }

    atmosphere
    {
        type            inletOutlet;
        inletValue      uniform 0;
        value           uniform 0;
    }

    frontAndBack
    {
        type            empty;
    }
}

// ************************************************************************* //
```

Uma vez que a função alpha só esta definida no interfalo de entre 0 e 1, esta se trata de um parâmetro adimensional. Isso é declarado através dos expoentes nulo na primeira linha após o cabeçalho do código.

A seguir, seguem as declarações das condições de contorno como visto na seção 3.2.1.3 em cada região da malha. A região de onde a água é injetada, inlet, tem o seu valor constante de alpha igual a um. Na abordagem desse caso surge uma nova condição que possibilita a saída de ar para a atmosfera através da abertura do recipiente. Essa condição é configurada através de expressão *inletOutlet* em *atmosphere.* Nas paredes do recipientes não temos variação da função alpha. Os demais campos em questão são os de pressão e de velocidade. Ambos também são configurados de forma semelhante.

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
   inlet
   {
      type          buoyantPressure;
      value         uniform 0;
   }

   inletWall
   {
      type          buoyantPressure;
      value         uniform 0;
   }

   bottleWall
   {
      type          buoyantPressure;
      value         uniform 0;
   }

   atmosphere
   {
      type          fixedValue;
      value         uniform 0;
   }

   frontAndBack
   {
      type          empty;
   }
}
```

O comando *buoyantPressure* indica que o ∇p é o atmosférico. A pressão atmosférica é arbitrariamente considerada nula. Ao campo de velocidade **U** foi atribuída uma pequena velocidade horizontal de 0.1m/s para a água na direção $-\hat{y}$ a fim de se obter um baixo número de Reynolds, garantindo assim um escoamento laminar.

```
FoamFile
{
   version     2.0;
   format      ascii;
   class       volVectorField;
   location    "0";
   object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);
```

```
boundaryField
{
   inlet
   {
      type          fixedValue;
      value         uniform (0 -0.1 0);
   }
   inletWall
   {
      type          fixedValue;
      value         uniform (0 0 0);
   }
   bottleWall
   {
      type          fixedValue;
      value         uniform (0 0 0);
   }
   atmosphere
   {
      type          pressureInletOutletVelocity;
      value         uniform (0 0 0);
   }
   frontAndBack
   {
      type          empty;
   }
}
```

Nas paredes foi adotada a condição de não deslizamento (*no-slip condition*) fazendo *fixedValue* identicamente a um vetor nulo. Mais uma vez temos *empty* em *frontAndBack* por se tratar de um caso com simetria em $\hat{\boldsymbol{z}}$.

Assim, como na função alpha, em *atmosphere* é necessário observar o fluxo das fases. Assim através do comando *pressureInletOutletVelocity* calcula-se **U** a partir do valor da pressão na face.

Uma vez configuradas a malha, as condições iniciais e de contorno, bem como os parâmetros físicos envolvidos é hora de dar inicio à solução do caso. Uma simulação no OF é iniciada através de comandos específicos no terminal a partir do diretório do caso. Primeiramente gera-se a malha do domínio computacional rodando-se o utilitário blockMesh. Depois será a vez de carregar os campos iniciais de cada uma das fases através do setFields. E finalmente digita-se o nome do solver a ser utilizado na simulação. Os comandos, a serem executados no terminal, para as respectivas ações citadas são :

```
blockMesh

setFields

interFoam
```

O tempo de resolução de um caso pode variar bastante. Numero de elementos na malha, tamanho do passo de tempo, critério de convergência adotado bem como o hardware disponível são somente algum dos itens que impactam sobre o tempo de solução.

O pós-processamento dos resultados aqui será feito através do Paraview®, que é o software mais utilizado pelos usuários de OpenFOAM®. Porém é possível realiza-lo em outras plataformas como ANSYS-Fluent® e o SALOME dentre outros.

4.1.2 Análise dos Resultados.

As figuras 4.3 mostram a malha gerada após a execução do blockMesh com os diferente patches indicados.

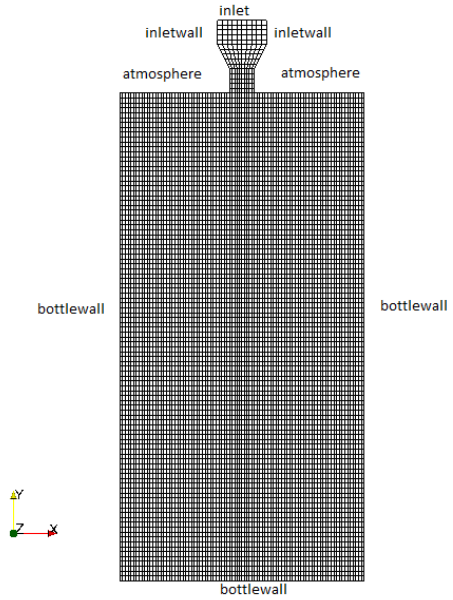


Figura 4.3(a) Malha obtida a partir do *blockMeshDict*

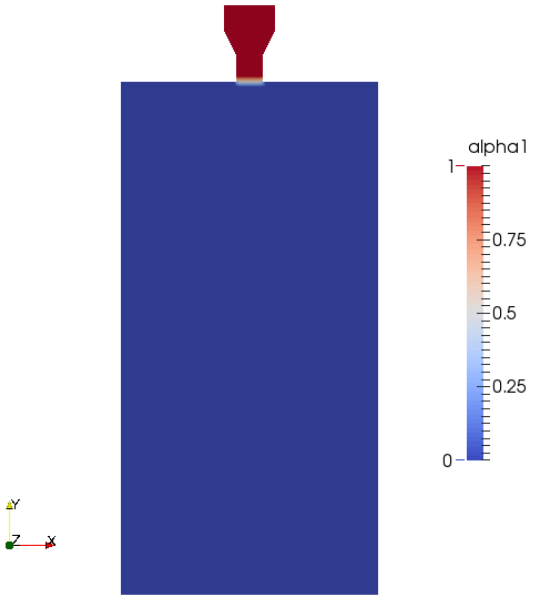
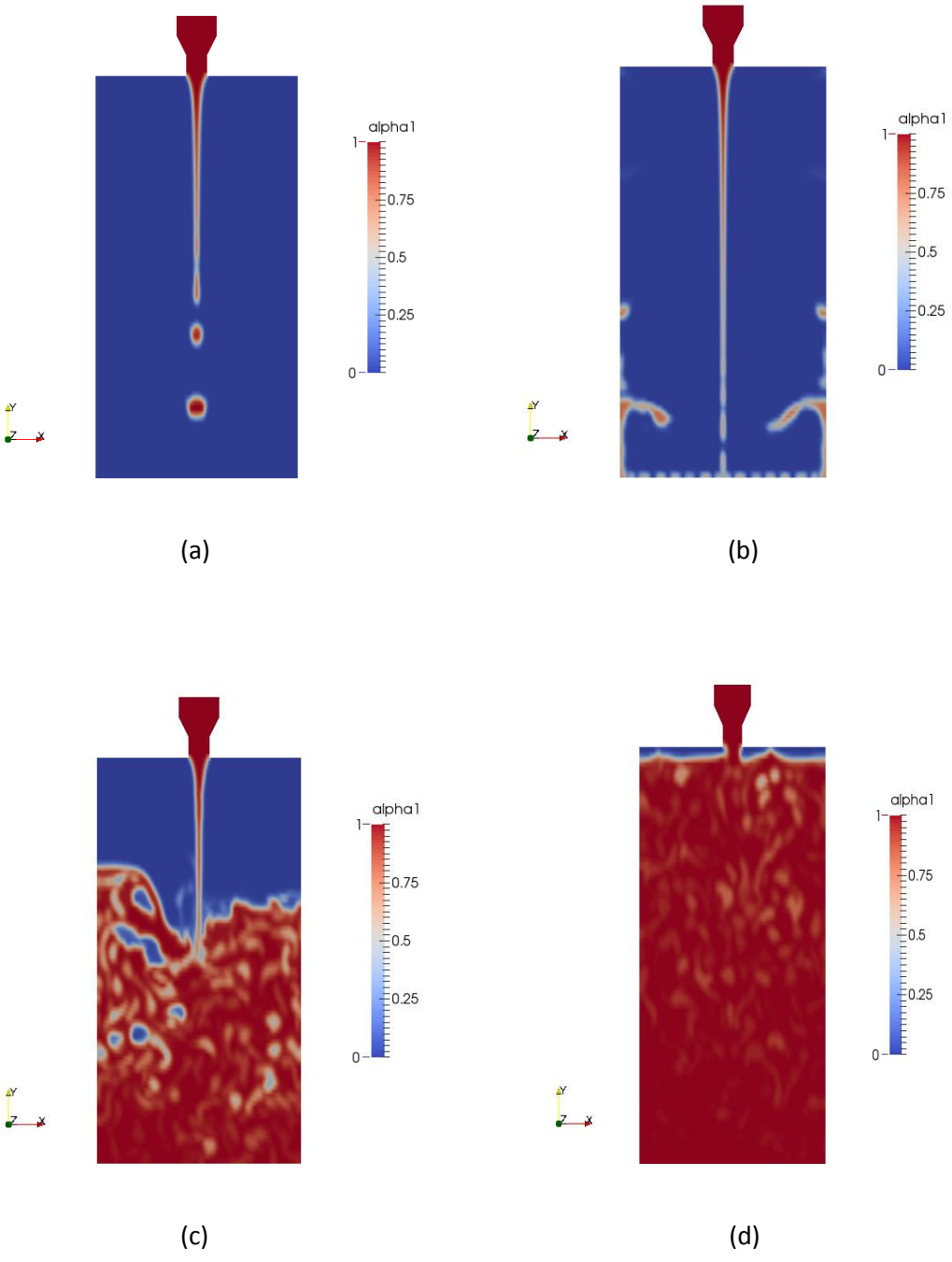


Figura 4.3(b) Configuração das fases em t=0.

Vale notar que na figura 4.3(a) foram omitidas as faces frontais e traseira, frontAndBack, da malha. Já em 4.3(b) temos a distribuição da função α conforme

indicado no setFiedsDict inicialmente. Isto é, o recipiente repleto de ar e a água contida apenas no bico injetor.

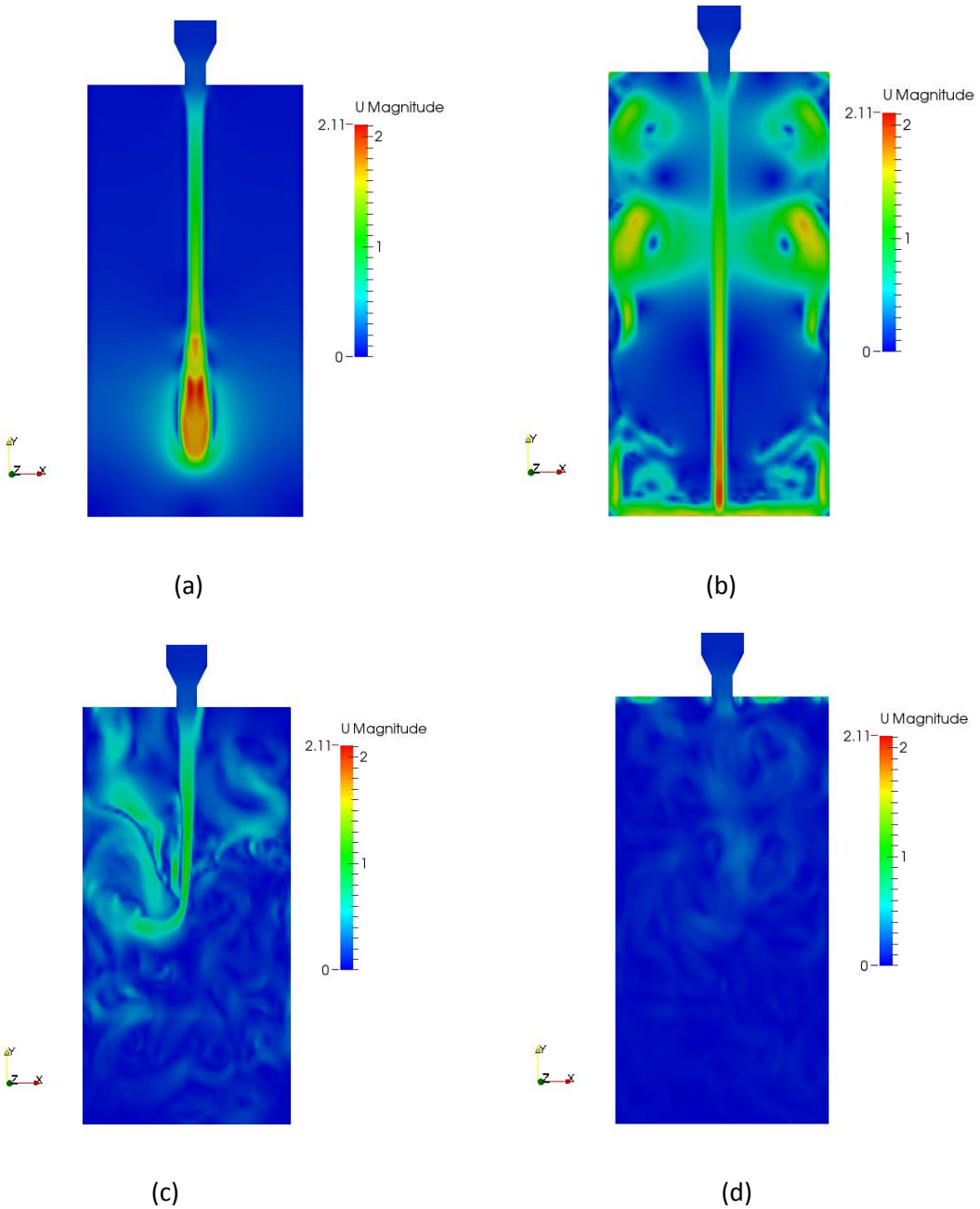


Figuras 4.4 Distribuição de α para t=0.2(a), t=0.4(b), t=6.0(c), t=12.0(d)

O conjunto de figuras 4.4 mostra a evolução temporal do escoamento. Em t=0,2s observamos o filamento de água se precipitando a partir do bico injetor, bem como a formação de algumas gotas. O estreitamento do fio de água é consequência do principio de Bernoulli (2.17), uma vez que este está sendo acelerado para baixo há uma diminuição do seu diâmetro. Já a formação das gotas é consequência da tensão superficial relativamente alta da água que tende a encapsular pequenas porções do líquido. Com 0,4 segundos de escoamento, figura 4.4(b), tem-se um movimento ascendente do líquido através das paredes do recipiente. Em 4.4(c) o líquido já ocupa praticamente a metade do volume do recipiente, porém nota-se ainda uma considerável quantidade de ar diluído na fase aquosa. Já em 4.4(d) a água preenche o recipiente em quase sua totalidade, observando-se apenas uma pequena quantidade de ar ainda retido.

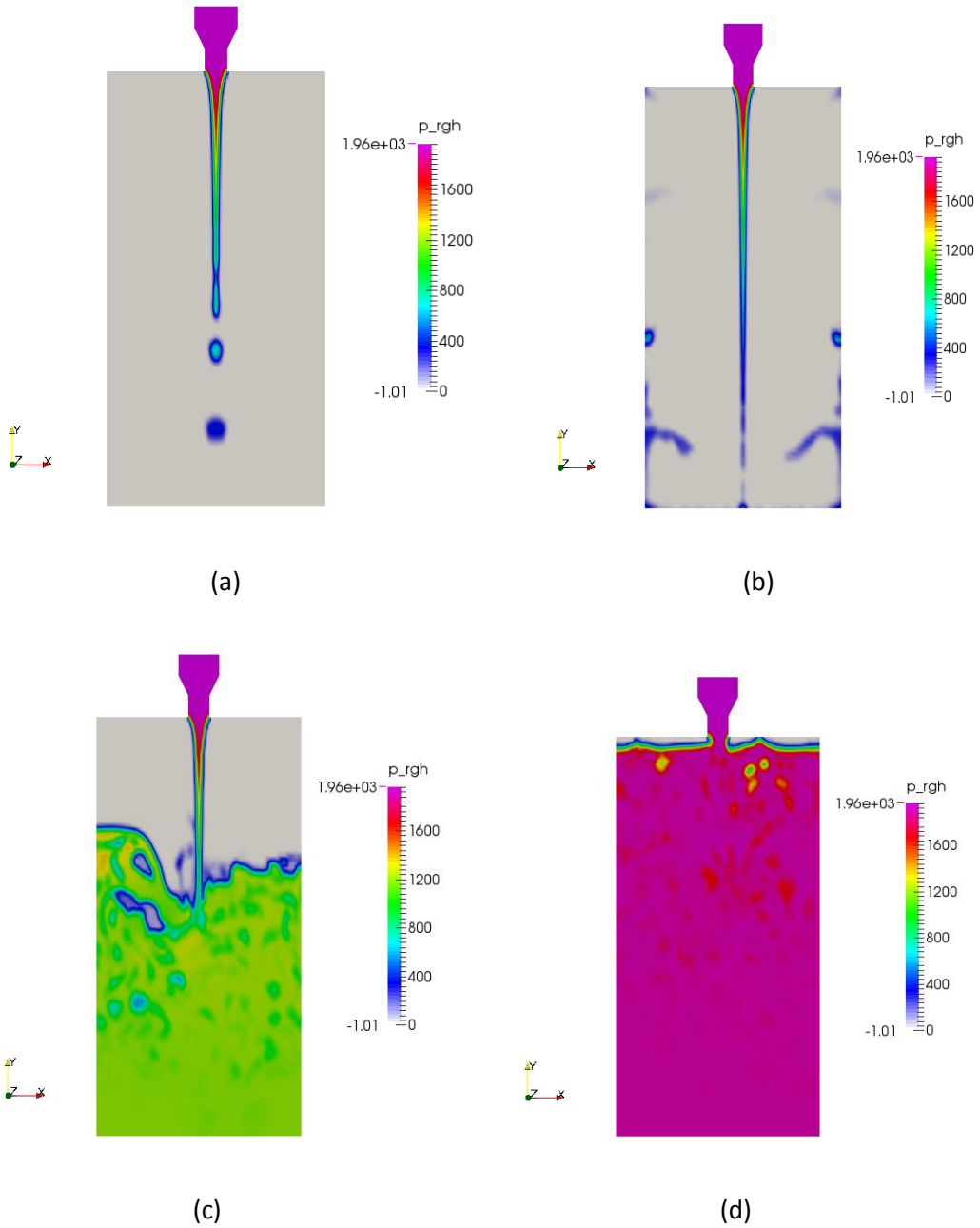
As figuras 4.5 a seguir mostram os campos de velocidade **U** de ambas as fases nos mesmos intervalos de tempo das imagens anteriores.

Na primeira destas, 4.5(a), nota-se o ar inicialmente em repouso sendo deslocado à medida que as gostas de água ganha velocidade verticalmente pra baixo. Em 4.5(b) evidencia-se uma circulação de **U** (vórtices não turbulentos), induzidos pelo movimento ascendente da água nas paredes do recipiente. Em 4.5(c) temos a fase líquida se assentando na parte inferior do domínio e já possuindo uma velocidade muito menor, sendo mais evidente apenas o fluxo vertical ainda existente. E finalmente em 4.5(d) com o recipiente praticamente cheio temos apenas alguns sutis movimentos residuais, praticamente todo o sistema esta em repouso novamente.



Figuras 4.5 Magnitude de U, em m/s, de ambas as fases para t=0.2(a), t=0.4(b), t=6(c), t=12(d)

A pressão atmosférica foi convenientemente considerada nula, uma vez que o recipiente é aberto. Portanto o único campo de pressão a ser mapeado é da pressão hidrostática que e é isso que mostrado nas figuras 4.6 a seguir.



Figuras 4.6 Distribuição temporal da pressão p em Pa.

Em t=0,2s e t=0.4s é notória a maior magnitude de p no bico injetor, uma vez que essa é a fonte do fluxo. Nota-se ainda a existência de pressão hidrostática no interior das gotas e do filamento, porém com uma magnitude muito inferior. Em 4.6(c) tem se um aumento significativo de pressão devido ao acumulo de líquido no recipiente, e com este já cheio, 4.6(d), temos uma distribuição de pressão praticamente homogênea em todo domínio computacional. Nessas duas últimas figuras, é interessante ressaltar que fica ainda mais evidente o aprisionamento de bolhas de ar dentro da fase líquida, denunciadas por regiões de baixa pressão.

4.2 O caso DAM 3D.

Nesta seção estudaremos um caso um pouco mais complexo do que o anterior. Aqui o nosso objetivo principal será o mapeamento da superfície livre num escoamento multifásico. Isto é, o formato da interface entre o ar é a água. O caso recebe o nome de DAM 3D por se tratar da simulação da ruptura da parede de uma represa. O ponto de partida foi o tutorial denominado *dambreak*, disponível no User Guide no OpenFOAM® **[6]**. Apesar de manter algumas similaridades com a simulação anterior, aqui trabalharemos com uma malha tridimensional composta por mais de 500.000 elementos. O que implica num esforço computacional muito maior do que o demandado pelo caso bottle.

4.2.1 Descrição, Configuração e Solução.

A geometria é composta por dois reservatórios separados por uma parede. Um contendo apenas água represada e outro quase totalmente preenchido por ar, porém com certa quantidade de água. Ambos são abertos à atmosfera. A ruptura da parede será simulada pela existência de uma fenda, mostrada no esquema da figura 4.7, no qual a fluirá de um reservatório a outro devido à pressão hidrostática.

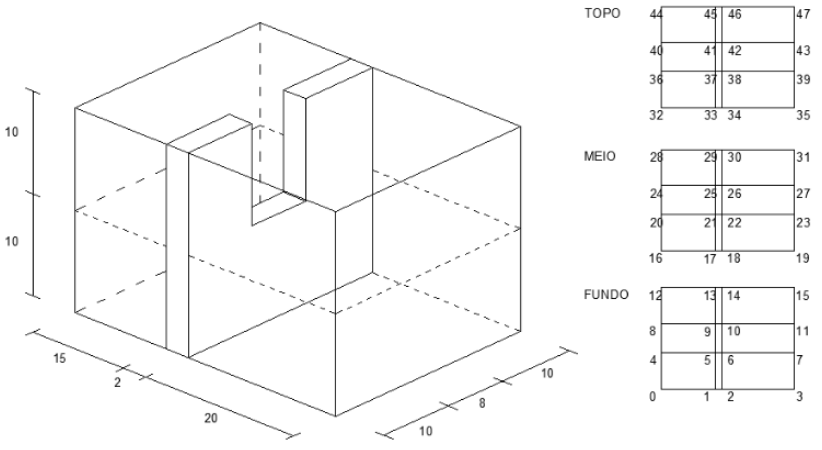


Figura 4.7 Dimensões e vértices da geometria do caso DAM 3D

O *blockMeshDict* que gerará a malha nesse caso é dado a seguir.

```
FoamFile
{
   version    2.0;
   format     ascii;
```

```
    class       dictionary;
    object      blockMeshDict;
}
convertToMeters 1;
vertices
(
            // PLANO DO FUNDO (Z=0)
            ( 0  0  0)   //00
            (15  0  0)  //01
            (17  0  0)  //02
            (37  0  0)  //03
            ( 0 10  0)  //04
            (15 10  0) //05
            (17 10  0) //06
            (37 10  0) //07
            ( 0 18  0)  //08
            (15 18  0) //09
            (17 18  0) //10
            (37 18  0) //11
            ( 0 28  0)  //12
            (15 28  0) //13
            (17 28  0) //14
            (37 28  0) //15
            // PLANO DO MEIO (Z=10)
            ( 0  0  10) //16
            (15  0  10)         //17
            (17  0  10)         //18
            (37  0  10)         //19
            ( 0 10  10)         //20
            (15 10  10)         //21
            (17 10  10)         //22
            (37 10  10)         //23
            ( 0 18  10)         //24
            (15 18  10)         //25
            (17 18  10)         //26
            (37 18  10)         //27
            ( 0 28  10)         //28
            (15 28  10)         //29
            (17 28  10)         //30
            (37 28  10)         //31
            // PLANO DE TOPO (Z=20)
            ( 0  0  20) //32
            (15  0  20)         //33
            (17  0  20)         //34
            (37  0  20)         //35
            ( 0 10  20)         //36
            (15 10  20)         //37
            (17 10  20)         //38
            (37 10  20)         //39
            ( 0 18  20)         //40
            (15 18  20)         //41
            (17 18  20)         //42
            (37 18  20)         //43
            ( 0 28  20)         //44
            (15 28  20)         //45
            (17 28  20)         //46
            (37 28  20)         //47
);

blocks
(
                        // BLOCOS INFERIORES
```

```
        hex ( 0  1  5  4 16 17 21 20) (15 10 10) simpleGrading (1 1 1)      //0
        hex ( 4  5  9  8 20 21 25 24) (15  8 10) simpleGrading (1 1 1)      //1
        hex ( 8  9 13 12 24 25 29 28) (15 10 10) simpleGrading (1 1 1)      //2
        hex ( 2  3  7  6 18 19 23 22) (20 10 10) simpleGrading (1 1 1)      //3
        hex ( 6  7 11 10 22 23 27 26) (20  8 10) simpleGrading (1 1 1)      //4
        hex (10 11 15 14 26 27 31 30) (20 10 10) simpleGrading (1 1 1)      //5
                // BLOCOS SUPERIORES
        hex (16 17 21 20 32 33 37 36) (15 10 10) simpleGrading (1 1 1)      //6
        hex (20 21 25 24 36 37 41 40) (15  8 10) simpleGrading (1 1 1)      //7
        hex (24 25 29 28 40 41 45 44) (15 10 10) simpleGrading (1 1 1)      //8
        hex (18 19 23 22 34 35 39 38) (20 10 10) simpleGrading (1 1 1)      //9
        hex (22 23 27 26 38 39 43 42) (20  8 10) simpleGrading (1 1 1)      //10
        hex (26 27 31 30 42 43 47 46) (20 10 10) simpleGrading (1 1 1)      //11
                // BLOCO FENDA
        hex (21 22 26 25 37 38 42 41) ( 2  8 10) simpleGrading (1 1 1)      //12
);

edges
(
);

patches
(
  patch atmosphere
  (
        (32 33 37 36)
        (36 37 41 40)
        (40 41 45 44)
        (34 35 39 38)
        (38 39 43 42)
        (42 43 47 46)
        (37 38 42 41)
  )
  wall Wall //
  (
        //Parede Frente
        ( 0  1 17 16)
        ( 2  3 19 18)
        (16 17 33 32)
        (18 19 35 34)
        //Parede Trás
        (13 12 28 29)
        (15 14 30 31)
        (29 28 44 45)
        (31 30 46 47)
        //Parede Lateral Esquerda
        (4 0 16 20)
        (8 4 20 24)
        (12 8 24 28)
        (20 16 32 36)
        (24 20 36 40)
        (28 24 40 44)
        //Parede Lateral Direita
        (3 7 23 19)
        (7 11 27 23)
        (11 15 31 27)
        (19 23 39 35)
        (23 27 43 39)
        (27 31 47 43)
        // Parede Interior Esquerda
        (1 5 21 17)
        (5 9 25 21)
```

```
            (9 13 29 25)
            (17 21 37 33)
            (25 29 45 41)
            // Parede Interior Direita
            (6 2 18 22)
            (10 6 22 26)
            (14 10 26 30)
            (22 18 34 38)
            (30 26 42 46)
            // Parede Fenda
            (21 25 26 22)
            (26 25 41 42)
            (21 22 38 37)
            // Fundo
            (0 4 5 1)
            (4 8 9 5)
            (8 12 13 9)
            (2 6 7 3)
            (6 10 11 7)
            (10 14 15 11)
    )
);


mergePatchPairs
(
);
```

A primeira parte se refere às dimensões, que aqui serão todas medidas em metros. Em seguida, vem à declaração das coordenadas dos vértices da geometria e logo após a nomeação das faces (patches). Aqui só teremos do tipo *wall* e *atmosphere* que se referem às paredes e abertura para atmosfera respectivamente. Aqui temos um maior numero de divisões nas três direções $\hat{x}, \hat{y}$ e $\hat{z}$ o que permite ter uma malha de melhor resolução, o que torna assim a determinação da superfície livre mais precisa.

Novamente usaremos o solver interFoam e trataremos o escoamento laminar. O passo seguinte é a determinação da configuração inicial dos campos preenchidos por água e ar, por meio do setFieldsDict. A superfície livre, ou seja, a interface entre as duas fases é mapeada fazendo $alpha1 = 0.5$. Isto é, as células do domínio computacional estão igualmente preenchidas por ar e água.

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      setFieldsDict;
```

```
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

defaultFieldValues
(
    volScalarFieldValue alpha1 0
    volVectorFieldValue U (0 0 0)
);

regions
(

   boxToCell
   {
      box (0 0 0) (17 28 18);
      fieldValues
      (
         volScalarFieldValue alpha1 1
      );
   }

   boxToCell
   {
      box (17 0 0) (37 28 1);
      fieldValues
      (
         volScalarFieldValue alpha1 1
      );
   }

);
```

Aqui, teremos duas regiões do tipo *boxToCell,* que se referem às duas regiões inicialmente com água já mencionadas. Para o campo **U** tem-se uma diferença fundamental em relação ao caso bottle que é a ausência de uma velocidade inicial. A massa de água está inicialmente em repouso, sendo o ∇p hidrostático a origem do escoamento.

```
FoamFile
{
   version     2.0;
   format      ascii;
   class       volVectorField;
   object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);
boundaryField
{

   Wall
   {
      type          fixedValue;
```

```
    value         uniform (0 0 0);
  }

  atmosphere
  {
    type          pressureInletOutletVelocity;
    value         uniform (0 0 0);
  }

}
```

Novamente temos *pressureInletOutletVelocity* em atmosphere para que seja tomado ao fluxo das fases em função de p. A função *totalPressure* em *atmosphere* indica o cálculo do acoplamento pressão-velocidade em função do princípio de Bernoulli (2.17).

```
FoamFile
{
  version    2.0;
  format     ascii;
  class      volScalarField;
  object     p;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions     [1 -1 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
  Wall
  {
    type          buoyantPressure;
    value         uniform 0;
  }

  atmosphere
  {
    type          totalPressure;
    p0            uniform 0;
  }
```

4.2.2 Análise dos Resultados.

As figuras 4.8 e 4.9 a seguir mostram a malha gerada a partir do *blockMeshDict* distribuição de $alpha$ para $t = 0$.

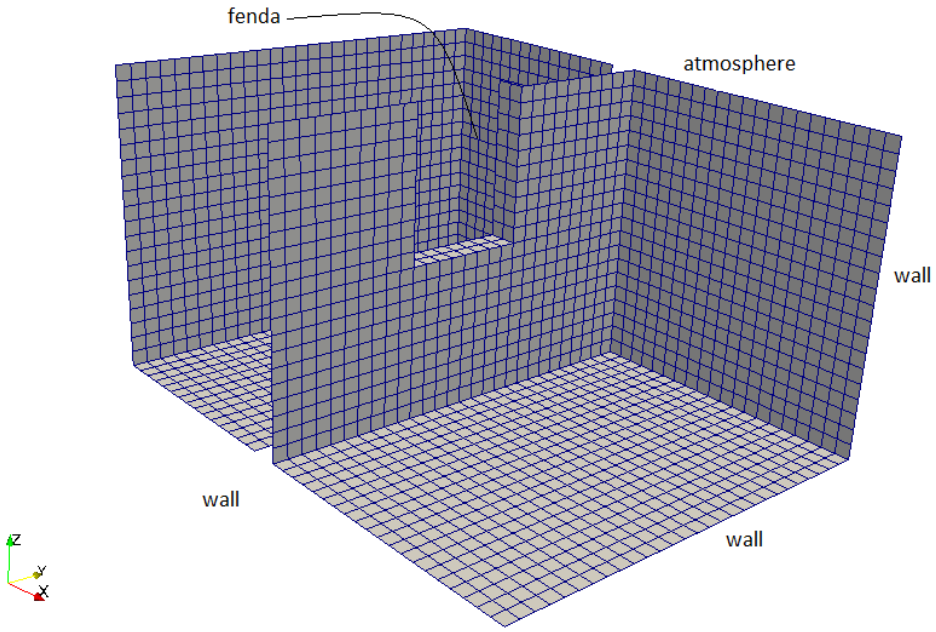
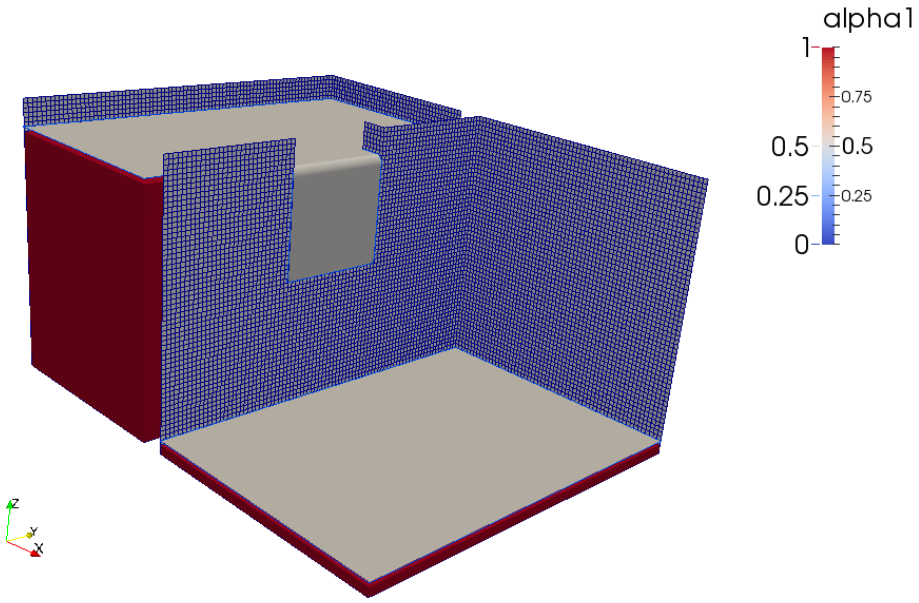
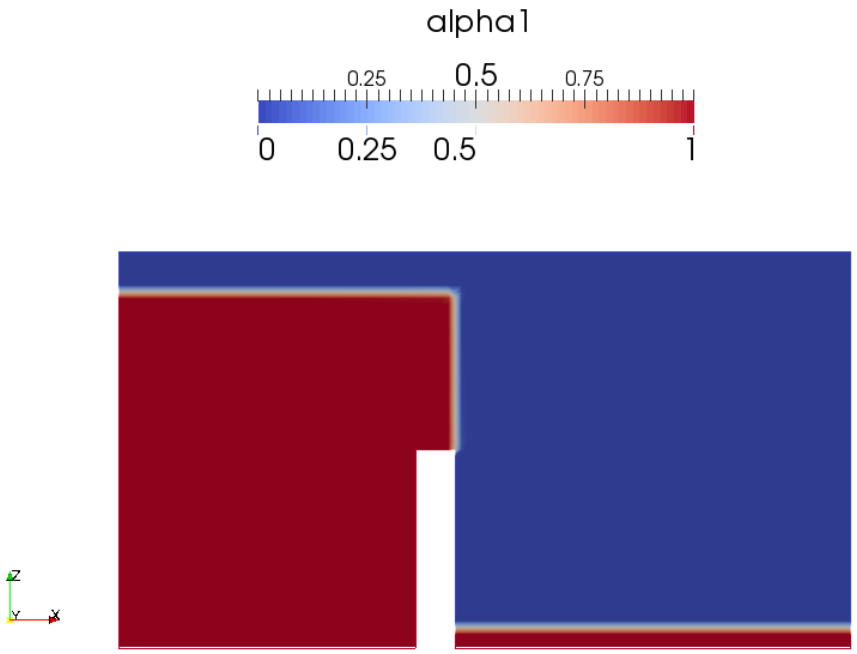


Figura 4.8 Malha e patches de DAM 3D



Figuras 4.9 (a) - Distribuição inicial da função de fase alpha1, vista em perspectiva.



Figuras 4.9(b) - Distribuição inicial da função de fase alpha1, vista em perfil.

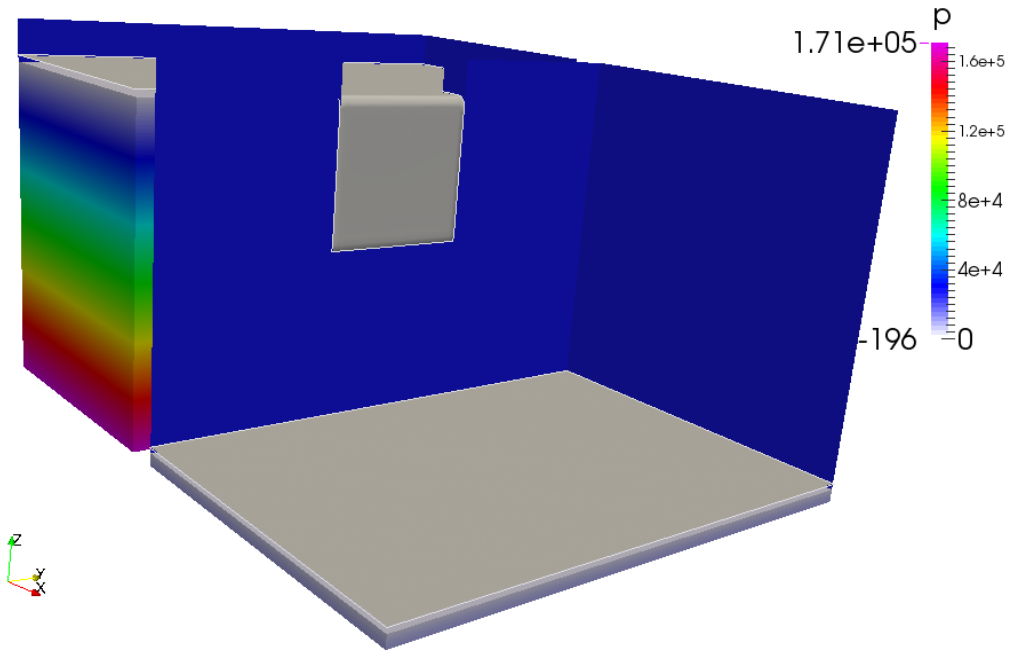


Figura 4.10(a) Distribuição de p em perspectiva

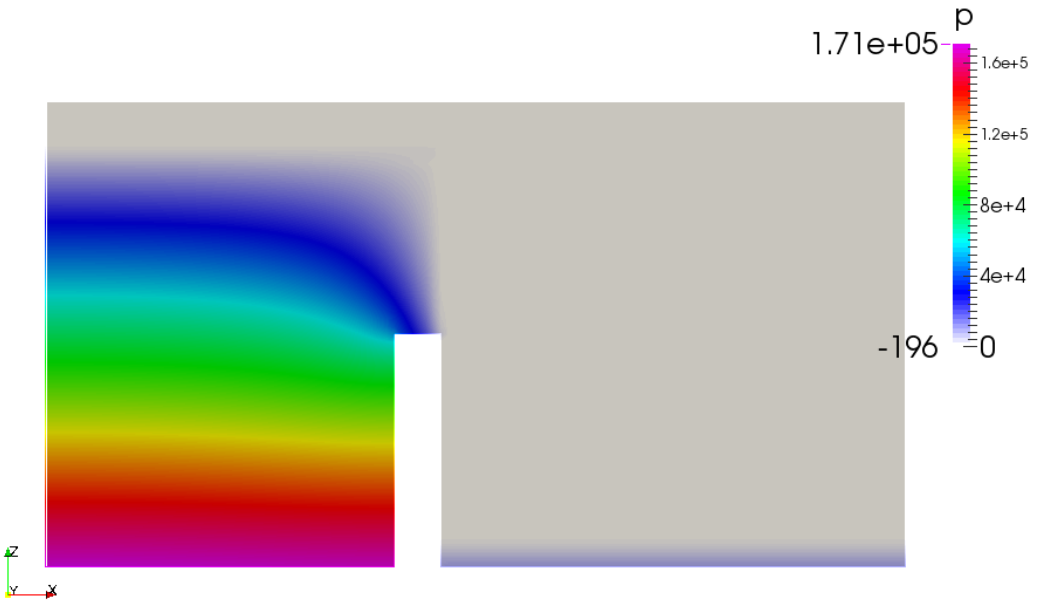


Figura 4.10(b) - Distribuição de p, em Pa, vista em perfil.

A figura 4.9(a) mostra apenas a água, foi omitida a representação do ar para visualização da interface entre ambos. Já na figura 4.9(b) temos a representação, em perfil, de ambas as fases bem como a interface entre as mesmas. A figuras 4.10 mostram a distribuição da pressão hidrostática, reservatório em perspectiva e em perfil, tanto no reservatório como na lamina d'água a direita na figura 4.10(b).

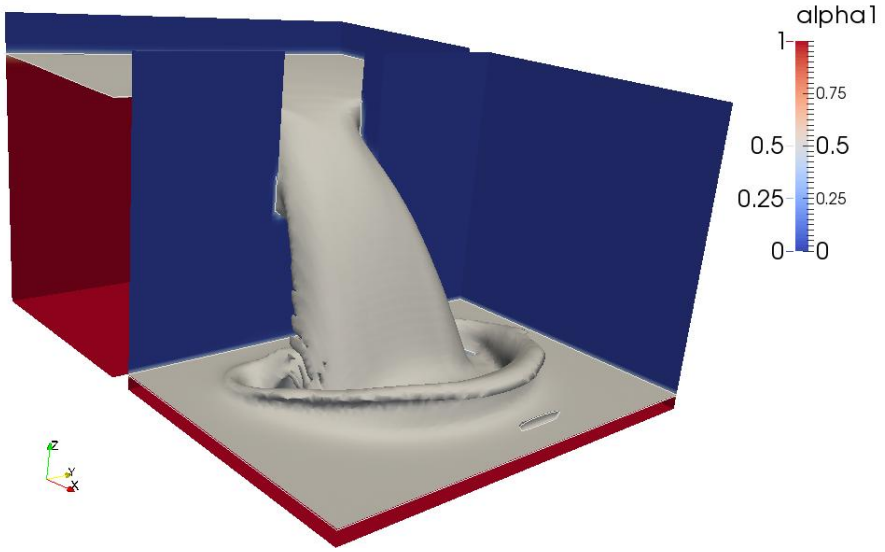


Figura 4.11 Escoamento em $t = 2s$

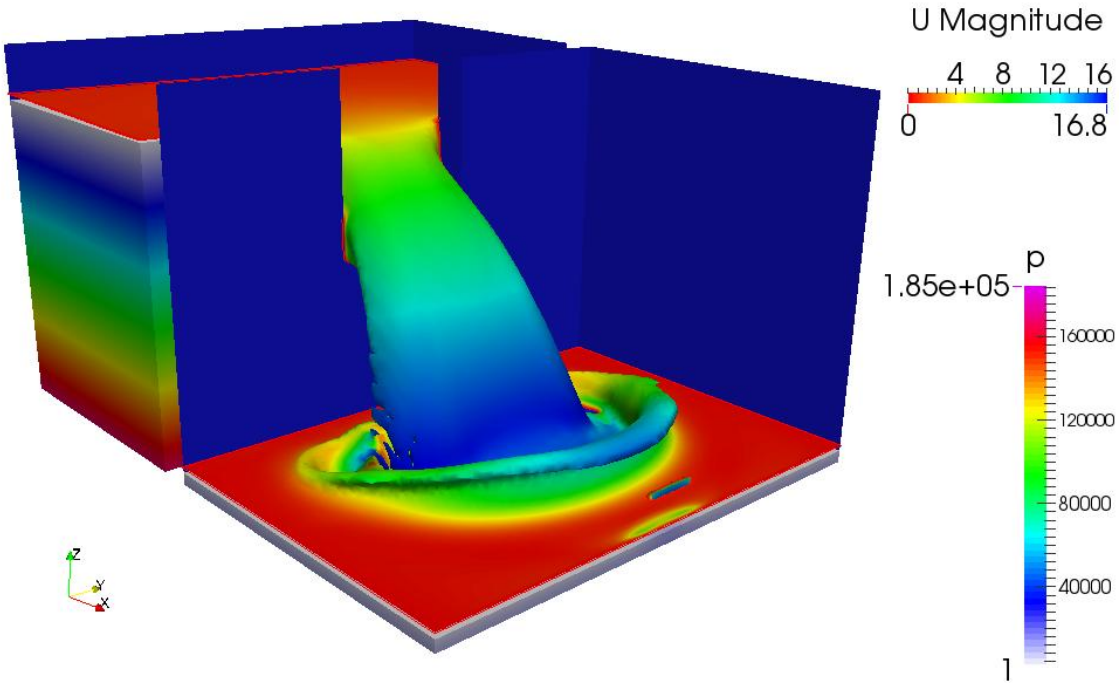


Figura 4.12 Distribuição de U[m/s] e p [Pa] em $t = 2s$.

As figuras 4.11 e 4.12 mostram a superfície livre entre as fases após dois segundos de escoamento, porém esta última evidencia como é o campo **U** do escoamento e a pressão hidrostática no reservatório. Na figura, é notório que nas regiões mais elevadas o líquido ainda está quase estático, sob baixa pressão, porém sendo rapidamente acelerado à medida que se aproxima da fenda; sendo projetado para baixo. Nota-se também que a lamina d'água inicialmente em repouso na parte de baixo anterior à parede.

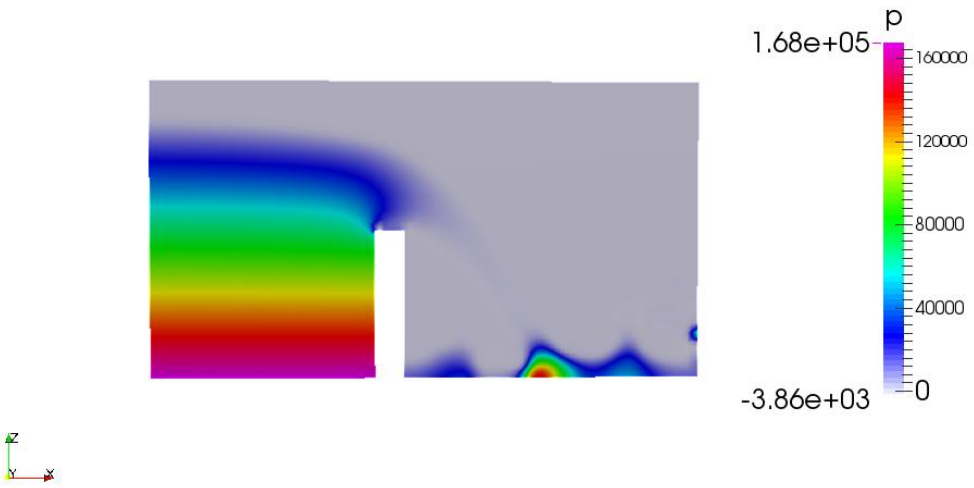


Figura 4.13 Pressão p, em Pa, em todo o domínio.

Outra representação de p é mostrada na figura 4.13, mostrando a pressão na parte do domínio repleta de ar. Sendo que nesta há apenas a pressão atmosférica, convenientemente considerada nula. A pressão menor na região onde o escoamento possui a maior velocidade, o que está de acordo com o principio de Bernoulli (2.17). A figura 4.12 acima mostra a distribuição dos campos **U** e p após 17 segundos de escoamento.

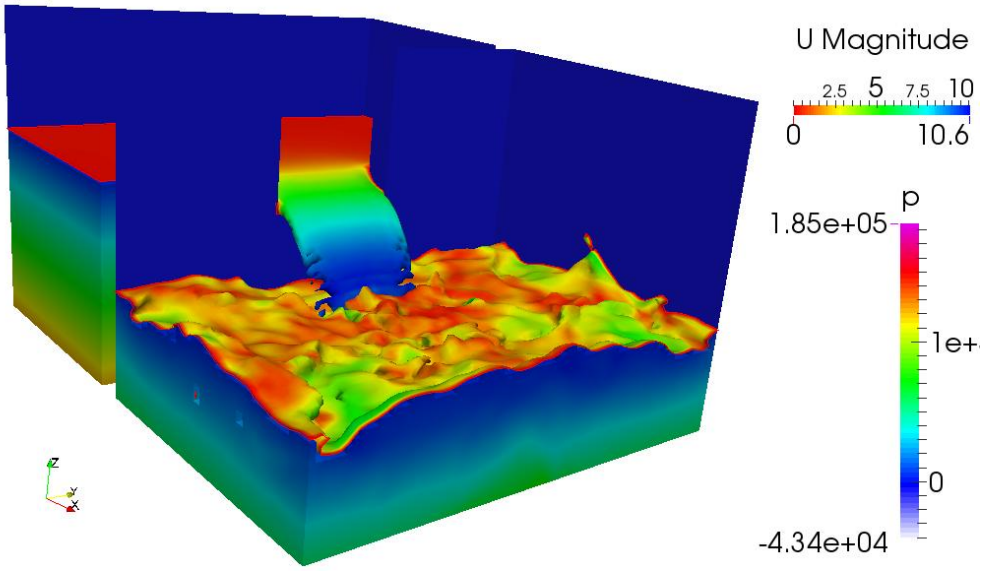


Figura 4.13 Campos de velocidade U (m/s) e pressão p (Pa) em $t = 17s$

Percebe-se que a superfície apresenta uma distribuição de velocidade bem menor, assim como uma um aumento significativo de p no reservatório que esta recebendo o fluido. Porém o que é mais notável na figura é a degeneração da superfície livre que ainda apresenta maior velocidade de escoamento. Acredita-se que isso se deve ao fator do escoamento ser considerado como laminar. É possível reparar que algo semelhante já acontece em t = 2s (figura 4.12), porém em menor grau.

Essa "corrosão" acontece nas regiões onde a superfície possui maior velocidade, consequentemente possuindo um maior Re. A falta de se considerar efeitos turbulentos gera uma propagação de erro numérico na determinação da superfície livre. A figura 4.13 a seguir destaca essa região.

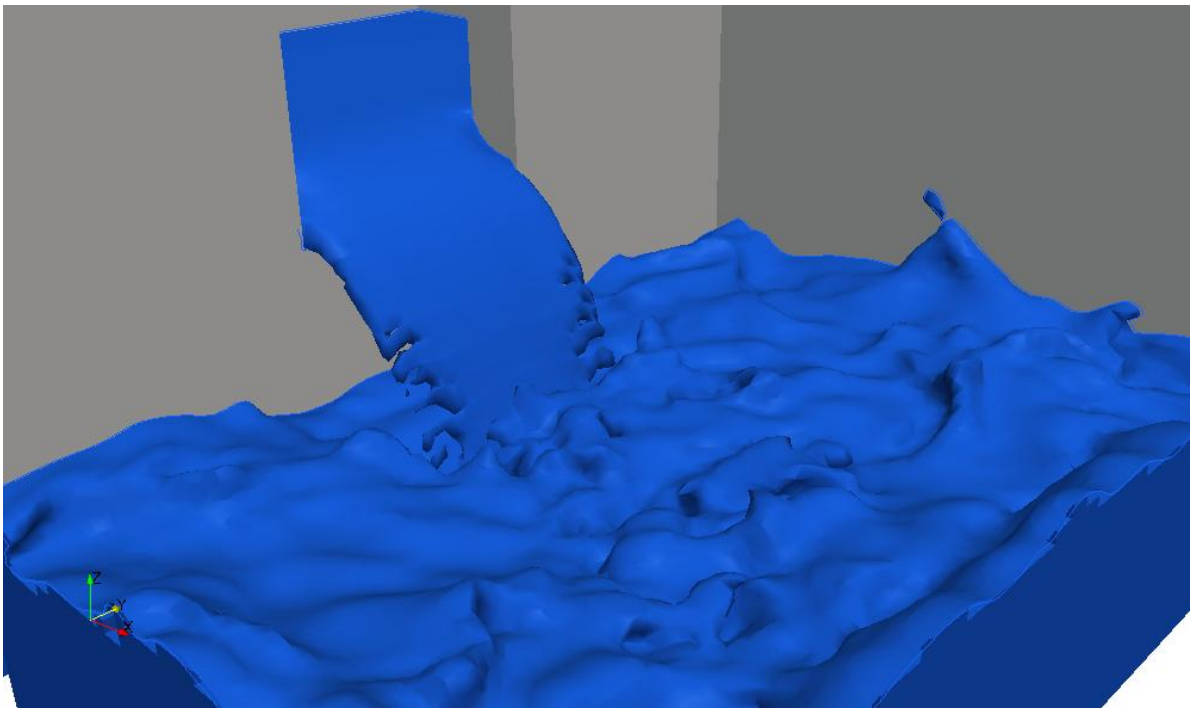


Figura 4.13 Degeneração da superfície livre em t =17s.

A utilização de abordagem *RANS* incrementaria significativamente a precisão numérica, minimizando efeitos de acumulo de resíduos durante as soluções e propiciando assim uma determinação mais crível da superfície livre.

Outro recurso que traria benefícios à solução seria o uso de malha mais refinada, isto é, um aumento do número de elementos que a constitui. Porém esse recurso traria uma maior demanda computacional, tanto de capacidade de processamento quanto de memória RAM disponível.

5. Conclusões e Perspectivas Futuras.

Os casos aqui mostrados evidenciam o quão poderoso é o OF na solução de problemas de CFD. Abordamos casos que continham algumas similaridades, porém com níveis diferentes de complexidade.

O *bottle,* apesar de sua simplicidade, pode ser adaptado para ser usado na modelagem de problemas para indústria de bebidas, enlatado etc. Já o caso *DAM3D* faz parte de uma solução para represas, tanques de armazenamentos, reservatórios e afins. O entendimento das forças envolvidas num caso como esse é fundamental para um adequado projeto de design e construção das estruturas envolvidas.

No presente trabalho abdicou-se de se fazer uso de modelos de turbulência na solução dos casos. Tal simplificação proporciona um resultado mais acessível, porém cobra um preço. É notório que a ausência dos parâmetros de turbulência forneceu soluções demasiadamente ilustrativas, faltando em muito à realidade física dos fenômenos. Em *bottle* há uma forte tendência de aprisionamento de porções de ar pela água, o que levaria a um mapeamento das fases distinto do que exibido nas figuras 4.4. Os campos **U** de ambas as fases também apresentaria diferenças significativas em relação às figuras 4.5, especialmente na indução de vórtices de maior intensidade para o ar. Em **[4]** é possível fazer esse comparativo. A determinação da superfície livre em *DAM 3D* seria muito aprimorada com o com o uso de modelos de turbulência. A anomalia apresentada em 4.13 poderia ser contornada, uma vez que os erros numéricos acumulados seriam menores. Outra melhoria significativa seria uma malha com uma quantidade ainda maior de elementos.

Malhas mais refinadas e uso de modelos de turbulência para aprimoramento dos resultados são os próximos passos a serem implementados pelo autor nos casos apresentados. Bem como o estudo de fluidos não-newtonianos e introdução de uma terceira fase em escoamento.

REFERENCIAS

[1] VERSTEEG, H.K., MALALASEKERA, W. .*An introduction to Computational Fluid Dynamics – The finite Volume Method*. 2º edição. Essex: Pearson Ed. Limited 2007.

[2] OLIVEIRA, Luis Adriano, LOPES, Antonio Gameiro. *Mecânica dos Fluidos*. 2º edição. Lisboa: Lidel. 2010.

[3] NUSSENZVEIG, H. Moysés. ***Curso Básico de Física Volume 2*:** Fluidos, Oscilações e Ondas, Calor. 4. ed. São Paulo: Blucher, 2002.

[4] HEMIDA, Hassan. ***OpenFOAM Tutorial*:** Free Surface tutorial using InterFoam e rasInterFoam. 2008. Disponível em: <http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2007/HassanHemida/Hassan_Hemida_VOF.pdf>.

[5] CARVALHO, Rita F. et al. ***Curso Teórico-Pratico OpenFOAM*.** 2012. IMAR - Depto de Eng. Civil. Faculdade de Ciências e Tecnologia da Universidade de Coimbra. Disponível em: <http://www1.ci.uc.pt/imar/unit/>

[6] *The OpenFOAM® **User Guide***, disponível online no site: http://www.openfoam.org/docs/, acessado em Julho 2014.

.

[7] *The OpenFOAM® **Programmer's Guide***, disponível online no site: http://www.openfoam.org/docs/, acessado em Julho 2014.

[8] SILVA, Luiz Fernando Lopes Rodrigues et al. ***Notas em CFD*.** 2007. Disponível em: <http://www.notasemcfd.com/>