UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS
DEPARTAMENTO DE HIDRÁULICA E SANEAMENTO

MODELAGEM E SIMULAÇÃO DA HIDRODINÂMICA EM AERAÇÃO FORÇADA CONSIDERANDO ASPECTOS GLOBAIS DE ESCOAMENTO TURBULENTO

Andreza Bortoloti Franco de Oliveira
Orientador: Prof. Dr. Nivaldo Aparecido Corrêa

VERSÃO CORRIGIDA

SÃO CARLOS

2013

ANDREZA BORTOLOTI FRANCO DE OLIVEIRA

MODELAGEM E SIMULAÇÃO DA HIDRODINÂMICA EM AERAÇÃO FORÇADA CONSIDERANDO ASPECTOS GLOBAIS DE ESCOAMENTO TURBULENTO

Tese apresentada à Escola de Engenharia de São Carlos – como parte dos requisitos para obtenção do título de Doutora em Ciências, Programa de Engenharia Hidráulica e Saneamento.

Orientador: Prof. Dr. Nivaldo Aparecido Corrêa

VERSÃO CORRIGIDA

São Carlos

2013

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

O48m

Oliveira, Andreza Bortoloti Franco de
Modelagem e simulação da hidrodinâmica em aeração forçada considerando aspectos globais de escoamento turbulento / Andreza Bortoloti Franco de Oliveira; orientador Prof. Nivaldo Aparecido Corrêa . São Carlos, 2013.

Tese (Doutorado) - Programa de Pós-Graduação e Área de Concentração em Hidráulica e Saneamento -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2013.

1. Turbulência. 2. Escoamento bifásico. 3. Simulação numérica. I. Título.

Candidata: Engenheira **ANDREZA BORTOLOTI FRANCO DE OLIVEIRA.**

Título da tese: "Modelagem e simulação da hidrodinâmica em aeração forçada considerando aspectos globais de escoamento turbulento".

Data da defesa: 24/05/2013

Comissão Julgadora:	Resultado:
Prof. Dr. **Nivaldo Aparecido Corrêa** (Orientador) (Escola de Engenharia de São Carlos/EESC)	Aprovado
Prof. Titular **Woodrow Nelson Lopes Roma** (Escola de Engenharia de São Carlos/EESC)	APROVADO
Prof. Associado **Leandro Franco de Souza** (Instituto de Ciências Matemáticas e de Computação/USP)	Aprovado
Prof. Dr. **José Antonio Silveira Gonçalves** (Universidade Federal de São Carlos/UFSCar)	Aprovado
Prof. Dr. **Werner Siegfried Hanisch** (Instituto Federal de Educação, Ciência e Tecnologia de São Paulo/IFSP)	APROVADO

Coordenador do Programa de Pós-Graduação em Engenharia Hidráulica e Saneamento:
Prof. Titular **Edson Cezar Wendland**

Presidente da Comissão de Pós-Graduação:
Prof. Titular **Denis Vinicius Coury**

À minha família.

AGRADECIMENTOS

Agradeço ao Professor Nivaldo Aparecido Corrêa, meu orientador que esteve presente do início ao fim deste doutorado.

Aos Professores Harry Edemar Schulz e Fabio Freire pelo incentivo e contribuição nas críticas e sugestões dadas no decorrer deste trabalho.

Agradeço aos membros da Banca Examinadora, Prof. Leandro Franco de Souza, Prof. José Antonio Silveira Gonçalves, Prof. Werner Siegfried Hanisch, Prof. Woodrow Nelson Lopes Roma.

Aos meus amigos de laboratório Raquel Lobosco, Cristian Youlton, Antonio Meira, Elaine Schornobay, Felipe da Silva Nascimento, Nayara Falanca, Natalia Benatto, Amanara Potykytã, Guilherme Vuitik, pela amizade e colaboração.

Ao Paulo Rogério Martins da Silva, pelos conselhos e incentivos prestados, meu muito obrigada.

Meus agradecimentos á todos os funcionários do PPG-SHS.

Agradeço ao Dr. Diener Volpin Ribeiro Fontoura pelos conselhos e ajuda prestada no desenvolvimento do trabalho.

Finalmente, agradeço de forma especial aos meus pais, minha irmã, ao Antonio Carlos e meus filhos, por todo amor e carinho.

RESUMO

OLIVEIRA, A. B. F. DE. **Modelagem e simulação da transferência de oxigênio em aeração forçada considerando aspectos de escoamento turbulento.** 2013. 162p. Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2013.

O presente trabalho refere-se à modelagem fenomenológica da transferência de movimento das bolhas de ar para o meio líquido, simulação com integrações numéricas de Equações Diferenciais Parciais (EDP's) e comparação com resultados experimentais de viscosidades turbulentas. Esse tipo de modelo com transporte turbulento se faz necessário para avaliar a natureza dos escoamentos, seja em situação enquanto em monitoramento de recursos hídricos ou em operações nos tratamentos de efluentes líquidos. Foram obtidas soluções para o caso laminar e turbulento considerando modelo kappa-epsilon para um volume de controle com geometria cartesiana bidimensional. O método de resolução das equações parciais usado foi o dos volumes finitos. Os valores das viscosidades turbulentas calculadas convergiram com valores muito próximos dos encontrados experimentalmente num período de tempo de 5 (cinco) segundos, após esse período de tempo, os resultados divergiram muito dos experimentais, mesmo com grande refino de malha, o que sugere mudança de estratégia na proposição do modelo de turbulência.

Palavras-chaves: turbulência, escoamento bifásico e simulação numérica.

ABSTRACT

OLIVEIRA, A. B. F. DE. **Modelagem e simulação da transferência de oxigênio em aeração forçada considerando aspectos de escoamento turbulento.** 2013. 106p. Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2013.

The present work refers to the phenomenological modeling of the air bubbles transfer movementto the liquid phase, numerical simulation with integrations of Ordinary Differential Equations (ODEs) and Partial Differential Equations (PDE's), compared with experimental results. Such models contribute to the turbulent transport, which are needed to assess the nature of the flows, in situations while monitoring water resources or in operations of wastewater treatment. Solutions were obtained considering the case laminar or turbulent using kappa-epsilon model. The calculated kinematic viscosities values diverged greatly from experimental results, suggesting the necessity of strategy changes in the proposed turbulence model.

Key words: Turbulence, bi-phase flow and numerical simulation.

SUMÁRIO

1 INTRODUÇÃO E JUSTIFICATIVA 1

2 OBJETIVOS 7

3 FORMULAÇÃO MATEMÁTICA 8

3.1 EQUACIONAMENTO 8

3.2 EQUAÇÕES PARA SISTEMA BIFÁSICO LÍQUIDO-GÁS 12

3.3 CONDIÇÕES DE CONTORNO 13

4 ESTRATÉGIAS UPWIND 15

4.1 INTRODUÇÃO 15

5 ESTRUTURA DA METODOLOGIA NUMÉRICA 18

5.1 DISCRETIZAÇÃO EM VOLUMES FINITOS (MVF) 18

5.2 SISTEMA ALGÉBRICO DE EQUAÇÕES 23

5.3 NOTAÇÃO DA FORMULAÇÃO DISCRETA 24

6 MODELAGEM BUBBLEFOAM 27

6.1 EQUACIONAMENTO 27

6.2 MODELO DE TURBULÊNCIA 29

6.3 IMPLEMENTAÇÃO DO SOLVER BUBBLEFOAM 30

6.4 EQUAÇÃO DE MOVIMENTO DA FASE 31

6.5 EQUAÇÃO DA CONTINUIDADE DA FASE 33

6.6 EQUAÇÃO DA PRESSÃO 34

7 MÉTODO 37

7.1 MATERIAL E MÉTODOS 37

7.1.1 Método Numérico 39

7.2 GERAÇÃO DE MALHA 39

8 RESULTADOS E DISCUSSÕES DAS SIMULAÇÕES REALIZADAS 41

8.1 RESULTADOS EXPERIMENTAIS. 41

8.2 RESULTADOS PRELIMINARES DE VELOCIDADES DAS BOLHAS PARA O REGIME LAMINAR 42

8.3 RESULTADOS COM OS DIFUSORES NO FUNDO DO CANAL PARA O REGIME LAMINAR 44

8.4 RESULTADOS PARA O MODELO DE TURBULÊNCIA κ-ε 56

9 CONCLUSÃO E SUGESTÕES PARA TRABALHOS FUTUROS 65

9.1.1 Sugestões *65*

10 REFERÊNCIAS BIBLIOGRÁFICAS **67**

APENDICE A – CÓDIGO COMPUTACIONAL MODIFICADO. **73**

LISTA DE SÍMBOLOS

Letras Latinas:

C_D	coeficiente de arraste
C_l	coeficiente de sustentação
$C_t d$	coeficiente de dispersão turbulenta
C_{vm}	coeficiente de massa virtual
C_μ, C_t	constantes de modelo k-epsilon
d	diâmetro da partícula, *m*
f	fator de fricção
Fr	número de Froude
g	aceleração gravitacional, m s^{-2}
k	energia cinética turbulenta
p	pressão, *Pa*
Re	número de Reynolds
S_φ	termo fonte das equações de transporte
t	tempo, *s*
T	tensor de tensões, m s^{-1}
u	velocidade superficial, m s^{-1}
U	velocidade, m s^{-1}

Letras Gregas:

α	referente a fase , *g* ou *l*
ε	taxa de dissipação de k, m^2s^{-3}
μ	viscosidade dinâmica, kg m^{-1} s^{-1}
ν	viscosidade cinemática m^2s^{-1}
ø	variável genérica
ρ	massa especifica, kg m^{-3}
Γ	coeficiente de difusão

LISTA DE SIGLAS

CFD	Computacional Fluid Dynamics.
OF	OpenFoam.
OpenFOAM	Open Field Operation And Manipulation.
PISO	Pressure Implicit with Splitting of Operators.
PIV	Particle Image Velocimetry
RANS	Reynolds-averaged Navier-Stokes.
VOF	Volume of Fluid.

1 INTRODUÇÃO E JUSTIFICATIVA

A maior parte dos escoamentos encontrados na natureza e em aplicações industriais é turbulenta. Consequentemente, é muito importante compreender os mecanismos físicos que governam este tipo de fenômeno. Os escoamentos turbulentos apresentam instabilidade, velocidade e contém flutuações que são dependentes do tempo e da posição no espaço. Entre as características mais importantes dos escoamentos turbulentos, destaca-se multiplicidade de escalas que os caracterizam.

A turbulência e transição são campos da Mecânica de Fluidos cuja discussão científica se coloca entre os mais seriamente pesquisados no ultimo século. Tal fato fornece um testemunho das dificuldades e dos desafios científicos oferecidos por este tema, o qual está bem longe de ser esgotado, e, ao contrário, é ainda muito mal compreendido nas suas bases fundamentais.

Os primeiros estudos sobre a instabilidade e turbulência foram desenvolvidos por Osborne Reynolds e Lorde Rayleigh no século XIX (1883), que estabeleceu mais claramente a existência de dois regimes de escoamentos: laminar e turbulento. Reynolds também definiu a primeira caracterização da transição à turbulência através do parâmetro adimensional,

$$\mathrm{Re} = \frac{UL}{\upsilon} \tag{1}$$

No qual, U é a escala de velocidade, L é a escala de comprimento e υ é a viscosidade cinemática do fluido. Este parâmetro se tornou conhecido posteriormente como sendo o número de Reynolds. Outras contribuições importantes de Reynolds foram a verificação da existência de regiões turbulentas intermitentes (spots), uma propriedade posteriormente estabelecida como comum aos escoamentos próximos de paredes, e o desenvolvimento das equações médias de Reynolds para os escoamentos turbulentos quando ele introduziu as assim denominadas tensões de Reynolds (1884).

Em paralelo aos trabalhos experimentais de Reynolds, Lord Rayleigh desenvolvia suas investigações teóricas sobre instabilidades de escoamentos paralelos de fluidos invíscidos. Seus estudos, que deram origem a vários outros trabalhos, permitiram determinar quando uma pequena perturbação na forma de um trem de ondas infinito e de amplitude uniforme se amplifica ou se amortece com o tempo (Rayleigh, 1878). Entre seus resultados, destaca-se a demonstração de que a condição necessária para que um escoamento paralelo seja instável é a presença de uma região inflexional no campo de velocidade (Rayleigh, 1880). As primeiras investigações feitas por Rayleigh concernentes à estabilidade dos escoamentos cisalhantes

livres invíscidos foram sucedidas por investigações nas quais se leva em conta os efeitos viscosos.

A idéia de se estudar um escoamento turbulento como sendo laminar e modificar a viscosidade molecular, via conceito de viscosidade turbulenta, nasceu com Boussinesq (1877). Ele supôs que as tensões turbulentas de Reynolds são proporcionais às taxas de deformação, como foi feito por Stokes para o caso das tensões viscosas, mas com um coeficiente de proporcionalidade denominado viscosidade turbulenta, a qual é, normalmente, maior que a viscosidade molecular do fluido. A princípio, uma viscosidade turbulenta constante foi utilizada para escoamentos livres do tipo esteira, jatos e camadas de mistura. No entanto, para escoamentos sobre placas ou no interior de dutos, do tipo camada limite, não se consegue resultados coerentes sem que a viscosidade turbulenta varie com a distância à parede.

Na realidade, é difícil dar uma definição para turbulência, embora seja fácil dar exemplos específicos de como ela está presente em nosso cotidiano: no escoamento rápido ao entorno da asa do avião, na maioria dos ecossistemas terrestres, correntes atmosféricas e oceânicas, o movimento da atmosfera do sol e dos planetas, o escoamento no interior das instalações industriais (Mathieu & Scott, 2000). No interior de uma bolha em ascensão por força de empuxo pode-se encontrar transição à turbulência. A fumaça gerada por um cigarro dá origem a instabilidades do tipo toroides, as quais se desestabilizam e geram turbulência tridimensional. O processo de transporte de calor ou poluentes num dado ambiente pode ser extremamente acelerado pela presença de turbulência. Se considerarmos apenas o processo difusivo molecular, a difusão de um contaminante volátil depositado numa dada posição de uma sala poderia demorar alguns dias para se completar, enquanto que, na realidade, as forças, empuxo ou mesmo as correntes de escoamentos forçados aceleram este processo a alguns minutos, devido à ação da turbulência. O escoamento turbulento no interior de nossos pulmões acelera a difusão de oxigênio e facilita o processo de absorção. Graças à turbulência de pequenas escalas o processo de mistura de combustível e oxigênio no interior de uma câmara de combustão se torna eficiente o bastante para aumentar o rendimento do motor e reduzir os efeitos da poluição dos gases tóxicos liberados pelos automóveis e aviões. O movimento turbulento da atmosfera se encarrega de dispersar e transportar para outras regiões, os gases poluentes originários das cidades e das industrias que tendem a subir por forças de empuxo. No entanto pode acontecer que estes gases, normalmente mais leves que o ar, subam até encontrar uma camada de mesma densidade, onde eles estacionam. No estado atual sobre a natureza da turbulência, entre todas as suas características destacam-se aqui as

mais importantes: A turbulência é um fenômeno altamente difusivo, pois, em consequência, a mistura de massa, contaminantes, energia, quantidade de movimento é muito mais eficiente neste regime de escoamento. No regime laminar as partículas de fluido não têm a oportunidade de se deslocar rapidamente de uma posição para outra, enquanto que no regime turbulento partículas em altas temperaturas, ou portadoras de muita concentração de um dado contaminante, viajam rapidamente de uma posição a outra, entrando em contato com outras partículas que se encontram em baixa temperatura ou portadoras de uma concentração menor de contaminante. Isto implica em fortes gradientes dos potenciais associados, o que acelera o processo de difusão molecular.

A turbulência ocorre a altos números de Reynolds: a transição de um escoamento para o regime turbulento, bem como a sua manutenção dependem da importância relativa entre os efeitos inerciais e viscosos. Os efeitos inerciais, altamente não lineares, são efeitos amplificadores de perturbações e geradores de instabilidades. Por outro lado os efeitos viscosos são amortecedores ou inibidores da formação de instabilidades. O número de Reynolds (Re) é definido como a razão entre os efeitos inerciais e os efeitos viscosos. Desta forma um escoamento só terá a possibilidade de ser classificado como transitório ou tender para turbulento quando Re for maior que a unidade.

A turbulência é altamente dissipativa. O processo de dissipação viscosa de energia cinética turbulenta, gerando aumento de energia interna, acontece nas maiores frequências do fenômeno. Sabe-se ainda que, em regime turbulento completamente desenvolvido toda a energia injetada no escoamento deve decair em cascata ao longo do espectro de turbilhões até as frequências dissipativas (Silveira-Neto, 2003).

Para se manter um escoamento turbulento é necessário fornecer energia continuamente. Caso contrário, entra-se em regime de turbulência em decaimento.

A turbulência é um fenômeno contínuo (é contínuo enquanto houver fornecimento constante de energia) e qualquer escoamento de fluidos newtonianos pode ser modelado utilizando-se as equações de Navier-Stokes. Se o fluido for não-newtoniano estas equações devem ser modificadas no seu termo viscoso. É importante enfatizar que estas equações modelam qualquer escoamento, independente do regime ser ou não turbulento. Demonstra-se, inclusive que as menores escalas de comprimento da turbulência são ainda muito maiores que o livre caminho médio molecular do fluido. Este fato, no entanto, está limitado a escoamentos com número de Mach inferior a 15 (≈54000 Km/h) Acima deste patamar a aplicação das equações de Navier-Stokes torna-se questionável (Silveira-Neto et al, 2003).

Os métodos utilizados para análise dos escoamentos turbulentos podem ser classificados em dois grandes grupos: os experimentais e os teóricos. No primeiro grupo são usuais tipos de medidas com diferentes tipos de transdutores: anemômetros a fio quente, a filme quente, a laser, assim como, velocimetria por imagens de partículas (Particle Image Velocimetry- PIV). A visualização bi e tridimensional sempre se coloca como um dos recursos mais poderosos para se compreender fisicamente um escoamento. As vantagens e desvantagens são inerentes ao processo, tais como: alta confiabilidade e alteração da natureza do escoamento pela inserção intrusiva de sensores ou o uso de partículas de contraste.

Os métodos numéricos despontam com um potencial cada vez maior em função do desenvolvimento de modelos e métodos de solução mais avançados, assim como, devido ao desenvolvimento de máquinas de alto potencial de cálculo e de armazenamento de informações. No domínio de métodos numéricos, apesar de todo o avanço nas máquinas, elas ainda se apresentam insuficientes para a solução, a contento, das equações originais de Navier-Stokes, de forma a resolver escoamentos turbulentos com altos números de Reynolds.

Faz-se necessário o modelamento adicional dos efeitos físicos ligados à turbulência. Estes modelos vêm sendo desenvolvidos e utilizados ao longo dos tempos e dispõe-se, atualmente de diferentes famílias de modelagem, de acordo com suas origens e também de acordo ao potencial de cálculo disponível.

O sucesso de se obter soluções numéricas aceitáveis de escoamentos turbulentos realísticos por modelagem da turbulência depende fundamentalmente de dois fatores: modelos de turbulência e algoritmos numéricos. Ambos os fatores refletem na qualidade dos resultados obtidos. O propósito da modelagem é descrever de maneira adequada a física do problema, enquanto que o objetivo do método numérico é resolver as equações modeladas com eficiência e precisão. Geralmente, a construção de modelos de turbulência segue alguns princípios básicos como: a escolha de um conjunto de propriedades que seja suficiente para caracterizar o movimento turbulento; a derivação de um conjunto de equações algébricas ou diferenciais que supostamente governa a distribuição dessas grandezas por todo o campo de escoamento; e a dedução, a partir de dados experimentais, dos valores das constantes, ou funções, provenientes da modelagem matemática do problema.

Em décadas passadas muitos códigos em CFD incorporavam modelos de turbulência simples para demonstrar as suas capacidades em resolver escoamentos turbulentos. Alguns deles consideravam erroneamente um coeficiente de viscosidade aumentado constante. Outros incluíam expressões algébricas envolvendo um coeficiente de viscosidade turbulento, com o propósito de descrever as características da turbulência. Com o aprimoramento dos

computadores e os avanços dos algoritmos numéricos, tornou-se possível modelar os processos turbulentos por meio de equação de transporte. Desde que, em geral, as equações resultantes dessa modelagem contêm mais incógnitas do que equações, o problema de fechamento da turbulência pode ser abordado utilizando-se diferentes níveis de sofisticação. Uma classe de modelos bastante popular é a de duas equações, em que duas EDPs são acopladas às equações do movimento para descrever as evoluções das escalas de velocidade e de comprimento, representativas da turbulência.

Entre os mais diversos modelos de duas equações encontrados na literatura está o modelo κ-ε clássico (Launder & Spalding, 1974) *apud* FERREIRA, 2001, que constitui uma ferramenta poderosa na previsão de escoamentos turbulentos altamente complexos e com substanciais reduções no volume de dados armazenado e tempo de computação. Uma aplicação simples do modelo permite resolver as grandes escalas da turbulência e evita o problema de modelar a influência direta da viscosidade molecular quando próximo às fronteiras rígidas. Infelizmente, a validade dessa técnica está restrita às situações em que o número de Reynolds é suficientemente elevado para garantir que os efeitos do contorno rígido sejam desprezíveis. Por outro lado, para baixos valores do número de Reynolds, o modelo κ-ε deve incorporar relações empíricas bem estabelecidas (chamadas funções de parede) ou termos de amortecimento. Neste último caso, as equações para as variáveis turbulentas devem ser modificadas, no sentido de se levar em consideração o amortecimento da velocidade flutuante normal à parede rígida e os efeitos devidos à viscosidade molecular. Os modelos resultantes dessas modificações constituem a classe dos modelos κ-ε para números de Reynolds pequenos (Patel et al., 1985).

Na questão da modelagem da aeração por bolhas, o modelo é o trivial com maior ou menor grau de hipóteses quanto se possa fazer para se aproximar da realidade, no entanto a solução é o maior obstáculo e o emprego de métodos numéricos poderosos tem sido a saída para a maioria das investigações. Para tornar os modelos de aeração mais realísticos, podem ser incluídos na modelagem a questão do regime de escoamento; do perfil hidrodinâmico das plumas de bolhas; da forma e tamanho das bolhas; do perfil de pressão na coluna de água; do perfil de temperatura e da influência das impurezas.

Outra dificuldade na simulação dos modelos é o desconhecimento de parâmetros inerentes ao processo. Por exemplo, no caso da aeração da água líquida por bolhas ou pela superfície, que envolve transferência de massa, é necessário saber a respeito do coeficiente de transferência de oxigênio, k_l, principalmente. Muitos pesquisadores obtiveram esse parâmetro

na investigação de fenômenos de transferência entre fases para as mais diversas situações experimentais.

A fluidodinâmica computacional é adotada como estratégia numérica para a resolução das equações de modelos, devido à complexidade ou impossibilidade de se obter uma solução analítica das equações parciais que representam o modelo. Dentre os métodos empregados pela fluidodinâmica computacional, pode-se citar o método das diferenças finitas, o método dos elementos finitos e o método dos volumes finitos, sendo esse último o de maior empregabilidade.

Atualmente inúmeros programas para resolução de sistemas de fluidodinâmica estão disponíveis, como por exemplo, o FLUENT, o CFX comerciais e o OpenFOAM livre, que possibilitam os usuários criar rotinas para problemas de fluido dinâmica computacional.

Este trabalho utiliza a fluidodinâmica computacional com técnica numérica de solução e o modelo de hidrodinâmica de aeração por injeção de bolhas foi implementado no software gratuito OpenFOAM.

O pacote CFD OpenFOAM é uma ótima opção para a solução de uma grande variedade de problemas. Além disso, este pacote é distribuído gratuitamente e junto com seu código fonte e, portanto, permite uma interação mais profunda com o usuário. Este trabalho apresenta a metodologia numérica utilizada em sua implementação. Detalhes sobre a interpretação da linguagem de programação pelo pacote e seu uso para a solução de problemas CFD. Por fim, um estudo dos modelos multifásicos já implementados no OpenFOAM é realizado, com uma análise mais aprofundada da abordagem bifásica Euleriana (Silva 2008).

O trabalho aqui apresentado se iniciou com a versão 1.7.0 do OpenFOAM e acompanhou sua evolução na versão 2.1.1.

2 OBJETIVOS

O objetivo dessa pesquisa foi a modelagem do fenômeno de transferência de quantidade de movimento turbulento em um canal de aeração em operação contínua; simular o modelo com um método numérico de volumes finitos.

Para tanto os objetivos secundários foram:

- Estabelecer um modelo com um sistema composto pelas equações de Navier-Stokes (com turbulência, equações de médias de Reynolds) acopladas ao modelo kapa-epsilon, de distribuição energética turbulenta.

- Estabelecer um método numérico, a princípio, o método dos volumes finitos, usando programação de código aberto, Open-Foam.

- Comparação com perfis de velocidade e distribuição de viscosidade turbulenta com valores experimentais, obtidos em Oliveira (2008), por meio da metodologia PIV.

3 FORMULAÇÃO MATEMÁTICA

3.1 Equacionamento

As equações fundamentais que governam o escoamento dos fluidos são as equações de Navier-Stokes compostas pelas equações da continuidade, da quantidade de movimento e da energia.

Para os escoamentos turbulentos as equações básicas são expressões matemáticas dos princípios de conservação da massa (continuidade)

$$\nabla \cdot u = 0 \tag{3.1}$$

e conservação de quantidade de movimento

$$\frac{\partial u}{\partial t} + \nabla \cdot (uu) = -\frac{\nabla p}{\rho} + \upsilon \nabla^2 u + g \tag{3.2}$$

u é o vetor velocidade, t é o tempo, p é a pressão, ρ é a densidade, υ é a viscosidade cinemática do fluido e g é o vetor aceleração gravitacional. No caso bidimensional, e caso as variações da densidade sejam desprezíveis, a equação utiliza a convenção da somatória de Einstein, as equações de conservação (3.1) e (3.2) tornam-se respectivamente:

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{3.3}$$

$$\frac{\partial u_i}{\partial t} + \frac{\partial (u_i u_j)}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \upsilon \frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right) + g_i \qquad i = 1,2 \tag{3.4}$$

Em 1880, Osborne Reynolds, engenheiro britânico, estudou a transição entre regimes laminar e turbulento num tubo, e definiu o parâmetro adimensional

$$\text{Re} = \rho \frac{UL}{\mu} = \frac{UL}{\upsilon} \tag{3.5}$$

É um critério pelo qual o regime do escoamento pode ser determinado, onde L é um comprimento característico, descritivo da geometria do campo de escoamentos, ρ é a densidade, μ a viscosidade aparente, υ a viscosidade cinemática e U a velocidade. Esse parâmetro recebeu o nome de número de Reynolds, que é a razão entre forças de inércia e viscosas, que tem papel importante no escoamento do fluido.

Willian Froude foi um engenheiro naval britânico e, juntamente com seu filho, Robert Edmund Froude, descobriram que o parâmetro

$$Fr = \frac{U}{\sqrt{gL}} \tag{3.6}$$

O número de Froude relaciona as forças de inércia e a força graviacional na superfície da terra, e não deve ser desprezado em escoamentos com superfícies livres.

Para adimensionalização das equações da quantidade de movimento e continuidade definem-se as variáveis adimensionais como:

$$u = U\overline{u} \qquad x = L\overline{x} \qquad t = LU^{-1}\overline{t} \qquad p = U^2\overline{p} \qquad g = G\overline{g} \tag{3.7}$$

Substituindo-se as transformações (3.7) nas equações de conservação de massa e quantidade de movimento obtêm-se as formas bidimensional cartesiana e adimensional (as barras foram omitidas por simplicidade) das equações:

$$\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0 \tag{3.8}$$

$$\frac{\partial u}{\partial t} + \frac{\partial (uu)}{\partial x} + \frac{\partial (uv)}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \frac{1}{Fr^2} gx \tag{3.9}$$

$$\frac{\partial u}{\partial t} + \frac{\partial (uv)}{\partial x} + \frac{\partial (vv)}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + \frac{1}{Fr^2} gy \tag{3.10}$$

As equações instantâneas (3.8) – (3.10) podem modelar qualquer escoamento incompressível em qualquer regime do número de Reynolds, pois se aplica a menor escala

possível do regime laminar a qual compõe qualquer escoamento, não importando o grau de agitação. Entretanto, são equações de trabalho para simulação do regime laminar. Para seu uso em escoamentos turbulentos, haveria necessidade de métodos adequados, tais como o DNS, e uso de malhas altamente refinadas com custos computacionais elevados (Brandi, 2005).

Para a simulação de escoamentos turbulentos incompressíveis, a metodologia RANS (Reynolds- Averaged Navier-Stokes) é empregada, em que uma média temporal (um caso particular de filtro) é usada para simplificar a modelagem. Em particular, adota-se o modelo κ-ε padrão, o qual é baseado no conceito de viscosidade turbulenta de Boussinesq. (Wrobel et al, 1989).

As equações de conservação de massa e de quantidade de movimento (equações médias de Reynolds) podem ser escritas como:

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0 \tag{3.11}$$

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial \left(\overline{u}_i \overline{u}_j\right)}{\partial x_j} = -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\upsilon \frac{\partial \overline{u}_i}{\partial x_j} - \overline{u}_i' \overline{u}_j'\right) + g_i \qquad i = 1,2 \tag{3.12}$$

em que $\overline{u}_i' \overline{u}_j'$ é o tensor de tensões de Reynolds que representa os efeitos da turbulência sobre o escoamento médio. Comparando-se as equações médias e instantâneas do movimento, o tensor de tensões de Reynolds pode ser interpretado como tensões sobre as partículas materiais em adição às tensões determinadas pela pressão e tensões viscosas. Assim, o tensor médio de tensões em um escoamento turbulento pode ser aproximado por:

$$\overline{\sigma}_{ij} \approx -\overline{p}\delta_{ij} + \upsilon \overline{D}_{ij} - \overline{u}_i' \overline{u}_j' \tag{3.13}$$

em que $\overline{D}_{ij}$ é o tensor médio de deformações definido por:

$$\overline{D}_{ij} = \frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} \tag{3.14}$$

Boussinesq assumiu que o tensor médio de tensões num escoamento incompressível e em regime turbulento pode ser representado da seguinte forma:

$$\overline{\sigma}_{ij} = \left(-\overline{p}\delta_{ij} + \upsilon\overline{D}_{ij}\right) + \left(-\overline{p}_t\delta_{ij} + \upsilon_t\overline{D}_{ij}\right) \tag{3.15}$$

em que $\overline{p}_t$ é a pressão turbulenta e υ_t é a viscosidade turbulenta.

Comparando-se (3.13) e (3.15) identifica-se a aproximação de Boussinesq:

$$-\frac{2}{3}\overline{\kappa}\delta_{ij} + \upsilon_t\overline{D}_{ij} = -\overline{u}_i'\overline{u}_j' \tag{3.16}$$

em que κ é uma grandeza instantânea, relacionada às flutuações turbulentas e que mede a intensidade das flutuações de velocidade. A função escalar κ foi introduzida por Boussinesq em analogia com a pressão produzida por flutuações moleculares, e seu significado físico pode ser obtido pela contração de (3.16), de fato, desde que $\overline{D}_{ii} = 0$ obtém-se $\kappa = \frac{1}{2}\overline{u}_i'\overline{u}_j'$.

Utilizando-se as transformações (3.7) em conjunto com a viscosidade turbulenta adimensionalizada como:

$$\upsilon_t = \upsilon\overline{\upsilon}_t \tag{3.17}$$

e substituindo-se a aproximação de Boussinesq obtém-se as equações adimensionais de Navier-Stokes (N-S) com média de Reynolds (“RANS equations”):

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{3.18}$$

$$\frac{\partial u_i}{\partial t} + \frac{\partial\left(u_i u_j\right)}{\partial x_j} = -\frac{\partial p_e}{\partial x_i} + \frac{1}{\mathrm{Re}}\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j}\right) + \frac{1}{Fr^2}g_i + \frac{1}{\mathrm{Re}}\frac{\partial}{\partial x_j}\left(\upsilon_t D_{ij}\right) \quad i = 1,2 \tag{3.19}$$

em que p_e é a pressão efetiva média, definida por:

$$p_e = p + p_t = p + \frac{2}{3}\frac{1}{\mathrm{Re}}\kappa \tag{3.20}$$

É importante observar que as equações médias (3.18) e (3.19) constituem um sistema de EDPs que contém três equações e quatro incógnitas e, portanto, necessitam de mais uma hipótese para o fechamento. Assim, no contexto da hipótese de Boussinesq, o problema de fechamento da dinâmica média é encontrar uma equação para υ_t em função da posição e do tempo. Por razões dimensionais, a viscosidade turbulenta pode ser estimada utilizando-se a seguinte aproximação:

$$\upsilon_t \approx u_t l_t \tag{3.21}$$

em que υ_t *e* l_t são, respectivamente, escalas de velocidade e de comprimento da turbulência.

3.2 Equações para sistema bifásico líquido-gás

Para sistema liquido-gás as equações devem ser aplicadas para cada fase conforme modelo de Smith (1998).

$$\frac{\partial(\alpha_l \rho_l)}{\partial t} + u_l \nabla(\alpha_l \rho_l) - \nabla(\rho \rho_t \nabla \alpha_l) = 0 \tag{3.22}$$

$$\frac{\partial(\alpha_g \rho_g)}{\partial t} + u_g \nabla(\alpha_g \rho_g) - \nabla(\rho_g D_t \nabla \alpha_g) = 0 \tag{3.23}$$

$$\frac{D(\alpha_l \rho_l u_l)}{Dt} + \nabla\left\{\alpha_l [(\mu_l + \mu_t)(\nabla u_l + \nabla u_l^t) - \frac{2\rho_l k_l}{3}]\right\} + \alpha_l \nabla p - F_l = 0 \tag{3.24}$$

$$\frac{D(\alpha_g \rho_g u_g)}{Dt} + \nabla[\alpha_g \mu_g (\nabla u_g + \nabla u_g^t)] + \alpha_g \nabla p - F_g = 0 \tag{3.25}$$

Esse equacionamento será detalhado no capítulo 6.

3.3 Condições de Contorno

O domínio computacional a ser estudado é dado pela secção em destaque (Figura 2) no canal. Sendo que o regime de escoamento é considerado permanente.

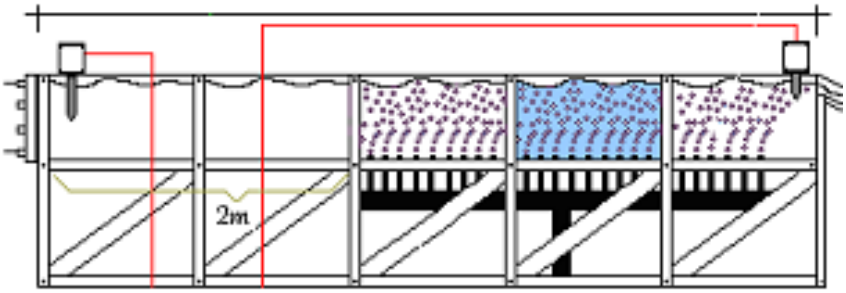


Figura 3.1- Domínio computacional.

V Max, τ =0

(fronteira livre, plana e sem quebras)

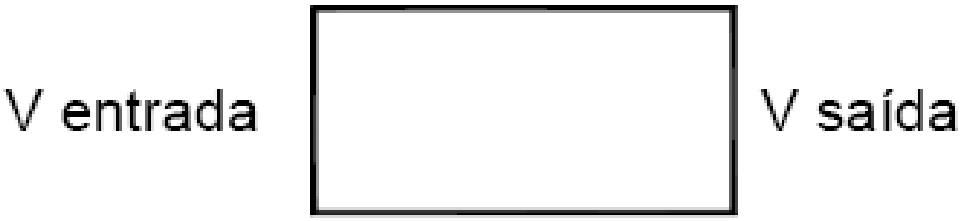


Figura 3.2 - Representação da geometria do canal.

v = 0 (nas paredes)

A entrada é a condição que representa a entrada do fluido no domínio computacional, em que pode ser aplicada uma condição de contorno de primeiro tipo, através da aplicação de um valor a velocidade e sua respectiva flutuação turbulenta da ordem de 5%. Conforme Figura 1.

As condições de saída são a representação do fluido que escoa para fora do domínio computacional do canal. A condição de simetria do escoamento mássico da entrada pode ser aplicada ou pode ser feita uma extrapolação da velocidade através do gradiente nulo.

O tratamento da interface água-ar é considerado condição de contorno denominada atmosfera. Ocorre transferência entre os meios, com escoamento de fluido para dentro e para fora do domínio. As tensões de cisalhamento são nulas e a velocidade é máxima.

Na condição de parede, escrita a princípio, de não aderência, sendo que a velocidade é considerada nula e no canal de acrílico a rugosidade é maior, para a simplicidade do modelo foi considerado como tubo liso. Na literatura existem tratamentos feitos à parede como superfície rugosa com a aplicação de leis de parede aos modelos de turbulência, mas esse não se aplica ao caso experimental em que o canal tinha paredes de acrílico.

O escoamento perpendicular à parede também é nulo, portanto a derivada tem valor zero.

As condições de simetria podem ser aplicadas para redução do domínio computacional e consequentemente do custo computacional. Neste caso, o escoamento é espelhado ao longo do canal através de um plano de simetria.

A entrada de ar comprimido no canal é representada por uma condição de contorno de primeiro tipo, sendo fornecida uma velocidade média que foi calculada através do orifício de entrada e da vazão de ar injetada em cada orifício.

4 ESTRATÉGIAS UPWIND

4.1 Introdução

A discretização dos termos convectivos das equações 3.9 e 3.10 são termos responsáveis por fenômenos complexos no escoamento e são os principais causadores de dificuldades numéricas nas simulações.

Nas ultimas décadas, o estado da arte na modelagem numérica dessas não-linearidades, utilizando-se o método das diferenças finitas ou volumes finitos, tem sido dominado principalmente pelas aproximações upwind de primeira ordem. Muito embora os esquemas upwind de primeira ordem imitem o método das características e assegurem a estabilidade do esquema numérico, alguns pesquisadores (Raithby, 1976), (deVahl e Malison, 1976) apud Ferreira 2001, alertam para o fato da introdução da viscosidade numérica, intrínseca a esses esquemas.

Para estimar a velocidade de convecção $\bar{u}$ no centro da ***i***-ésima célula, pode-se calcular:

$$\bar{u}_i^e = \frac{u_{i-1/2} - u_{i+1/2}}{2} \tag{4.1}$$

em que o índice e indica estimativa. Assim, por exemplo, quando $u_i^e \geq 0$ o escoamento é da esquerda para a direita. Em função da velocidade u_i^e pode-se definir funções de interpolação para os valores da propriedade u_i.

Utilizando o método upwind, faz-se uma extrapolação da velocidade u à montante do ponto i:

$$u_i = u_{i-1/2} \text{ , se } u_i^e \geq 0 \tag{4.2}$$

$$u_i = u_{i+1/2} \text{ , se } u_i^e < 0 \tag{4.3}$$

Utilizando a função sinal

$$S_i = 1, \text{ se } u_i^e \geq 0 \tag{4.4}$$

$$S_i = -1, \text{ se } u_i^e < 0 \tag{4.5}$$

Pode-se escrever a equação de forma compacta como:

$$u_i = \left[\frac{1+S_i}{2}\right] u_{i-1/2} + \left[\frac{1-S_i}{2}\right] u_{i+1/2} \tag{4.6}$$

A discretização upwind introduz em geral uma forte e indesejável difusão numérica na solução e seu emprego não é, em geral, recomendado. Apesar disso, por não produzir soluções oscilatórias, dispersivas, ela é ainda encontrada na literatura como um dos componentes do esquema híbrido.

O outro componente do esquema híbrido adotado neste trabalho é o das diferenças centrais para discretização dos termos convectivos, os quais podem gerar soluções oscilatórias, na presença de grandes gradientes de u, quando o número de Peclet (valor adimensional que mede a razão entre as intensidades dos processos de convecção e de difusão de quantidade de movimento, quanto maior "Pe", maior a intensidade de convecção), Pe, é maior que 2. Essa discretização, contudo, fornece bons resultados quando $Pe_i < 2$.

$$Pe_i = \frac{u_i^e(\Delta x)}{\upsilon} \tag{4.7}$$

Em que o esquema híbrido combina a precisão das diferenças finitas nas regiões do escoamento em que $Pe_i < 2$, com a upwind em zonas nas quais Pe_i 2.

Para evitar que a taxa de convergência do método numérico seja afetada pela mudança brusca do valor de u_i, quando Pe_i cruza o limite de 2, pode-se fazer essa mudança de forma mais gradual utilizando a função *fat*:

$$fat_i(Pe_i) = 0 \text{ , se } Pe_i < 1{,}9 \tag{4.8}$$

$$fat_i(Pe_i) = \frac{Pe_i - 1{,}9}{0{,}1} \text{ , se } 1{,}9 \leq Pe_i < 2 \tag{4.9}$$

$$fat_i(Pe_i) = 1 \text{ , se } Pe_i \geq 2 \tag{4.10}$$

A partir da definição de fat , a discretização híbrida pode ser escrita como:

$$u_i = (1 - fat_i) u_i^{DC} + (fat_i) u_i^{UP} \tag{4.11}$$

5 ESTRUTURA DA METODOLOGIA NUMÉRICA

O OpenFOAM possui implementações numéricas específicas para a solução de equações diferenciais parciais e entender seu funcionamento é vital para a implementação (ou extensão) dos códigos. Portanto, esta seção apresenta a metodologia da discretização e solução numérica das equações de transporte e sua interpretação pelo OpenFOAM.

5.1 Discretização em Volumes Finitos (MVF)

A metodologia numérica aplicada na discretização das equações no OpenFOAM é baseada primariamente no método dos volumes finitos (Patankar, Maliska, Pinto, Jasak, Versteeg). O termo “discretização” se refere à aproximação de um problema em um domínio contínuo por elementos discretos. O método dos volumes finitos, assim como o método das diferenças e elementos finitos, são aplicados na discretização do espaço físico e temporal. O primeiro tipo de discretização considera a divisão do domínio do espaço em um conjunto de volumes de controle interconectados entre si (contíguos) ou com o contorno do domínio. A discretização do espaço físico constrói a chamada malha computacional. De forma similar, a discretização temporal é aplicada a problemas transientes partindo de uma condição inicial e subdivide seu domínio em um número finito de intervalos no tempo. Uma representação de um domínio discretizado está colocada na Fig. 5.1.

Um volume de controle é composto por um conjunto de faces, que por sua vez é composta por um conjunto de arestas. As faces podem conectar dois volumes de controle ou um volume de controle ao contorno do domínio. A Fig. 5.2 apresenta as variáveis relevantes na interconexão de dois volumes de controle vizinhos, P e N.

Note que pela Fig. 5.2 o vetor S é normal à face f e sua magnitude é igual à área da face. Este vetor aponta do volume de interesse P para o vizinho N.

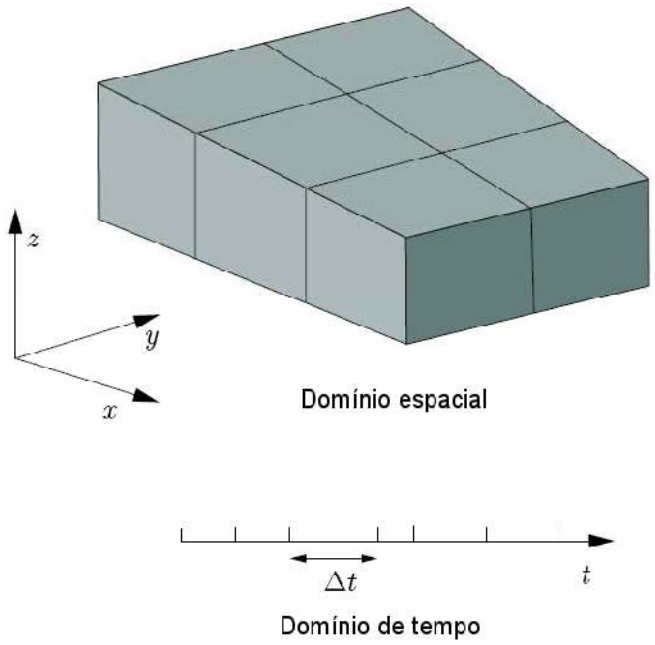


Figura 5.1 - Discretização nos domínios de espaço e tempo (retirado do OpenFOAM).

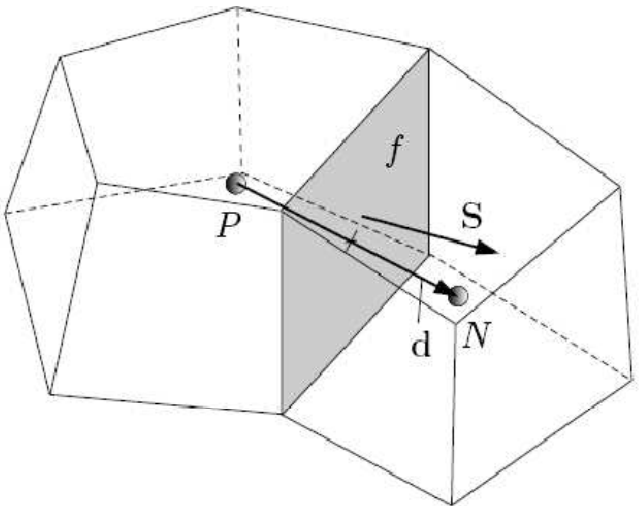


Figura 5.2 - Parâmetros na discretização por volumes finitos (retirado do OpenFOAM).

O vetor **S** é um vetor normal a face e sua magnitude é iagual a área da face. Este vetor aponta da célula de interesse *P* para acélula vizinha *N.* O vetor unitário normal à face **n** é definido como **n** = $\frac{S}{|S|}$ enquanto que a distância **d** é o vetor que aponta do centro da célula de interesse P ao centro da célula vizinha N é definida como **d** = $X_N - X_P$. Uma malha é considerada ortogonal quando **d** é paralelo ao vetor **S** em cada face da malha.

Um ponto importante na discretização é a alocação do campo de variáveis na malha. Usualmente, todas as variáveis dependentes (efetivamente solucionadas) são alocadas no centro dos volumes. Este tipo de alocação, denominado arranjo co-localizado, se torna a escolha mais óbvia pela sua simplicidade de controle dos índices da malha na implementação computacional. Contudo, problemas de oscilação e avaliação do gradiente de pressão podem aparecer no arranjo co-localizado (Maliska,2004).

Porém, com os avanços das técnicas numéricas (Rhie, 1983; Peric, 1988) estes problemas foram resolvidos e, desde então, o arranjo co-localizado é usado na maioria dos códigos CFD, inclusive no OpenFOAM.

O processo de discretização por volumes finitos transforma as equações diferenciais de variáveis contínuas em um sistema de equações algébricas correspondente.

Partindo de uma equação de transporte para uma variável φ tensorial genérica

$$\frac{\partial(\rho\varphi)}{\partial t} + \nabla.(\rho u\varphi) = \nabla.(\Gamma\nabla\varphi) + S_{\varphi}(\varphi) \tag{5.1}$$

onde ρ é a massa específica, u é o campo de velocidade, Γ é o coeficiente de difusão e $S_{\varphi}(\varphi)$é o termo fonte, sua discretização por volumes finitos é obtida integrando-a sobre um volume de controle V_P e em um intervalo de tempo.

$$\int_{t}^{t+\Delta t}\left[\int_{Vp}\frac{\partial(\rho\varphi)}{\partial t}dV + \int_{Vp}\nabla.(\rho u\varphi)dV\right]dt = \int_{t}^{t+\Delta t}\left[\int_{Vp}\nabla.(\Gamma\nabla\varphi)dV + \int_{Vp}S_{\varphi}(\varphi)dV\right]dt \tag{5.2}$$

O teorema de Gauss é usado para converter as integrais em volume dos termos derivativos no espaço em integrais sobre a superfície S no contorno dos volumes de controle.

$$\int_{V}(\nabla.\varphi)dV = \int_{\partial V}dS.\varphi \tag{5.3}$$

O teorema de Gauss é utilizado na Eq. 5.2 para aproximar seus termos pelo somatório dos fluxos das propriedades pelas faces do volume de controle. Este procedimento é detalhado na sequência.

O termo convectivo $\nabla.(\rho u \varphi)$ da Eq. 5.2 é aproximado convertendo as integrais em volume em superfície usando o teorema de Gauss e somando-as em todo o volume de controle:

$$\int_V \nabla.(\rho u \varphi) dV = \int_{\partial V} dS.(\rho u \varphi) \approx \sum_f S.(\rho u) F \varphi_f(F, M, \gamma) = \sum_f F \varphi_f(F, M, \gamma) \tag{5.4}$$

onde F é o fluxo mássico através da face f definido como $F = S.(\rho u)f$. O valor da propriedade na face $\varphi_f(F, M, \gamma)$ pode ser obtido usando uma função de interpolação M pré-definida. Esta última usualmente utiliza informações do próprio volume de controle e seus vizinhos, além de requerer o fluxo F sobre a face f e um ou mais parâmetros para efetuar a interpolação. Existem vários métodos de interpolação, entre os quais pode-se citar as abordagens de interpolação linear (diferenças centrais), upwind, QUICK, MUSCL, TVD (Total Variation Diminishing) e NVD (Normalised Variable Diagram). A formulação destes métodos encontram-se em vários trabalhos (Maliska, 2004; Patnakar, 1980; Pinto,2001) e suas formulações só serão abordadas quando for necessário.

Da mesma forma, o termo difusivo $\nabla.(\Gamma \nabla \varphi)$ da Eq. 5.2 é aproximado usando o teorema de Gauss:

$$\int_V \nabla.(\Gamma \nabla \varphi) dV = \int_{\partial V} dS.(\Gamma \nabla \varphi) \approx \sum_f \Gamma_f (S.\nabla_{f\varphi}) \tag{5.5}$$

considerando Γ como uma variável escalar. Em malhas ortogonais, o gradiente $\nabla_{f\varphi}$ pode ser obtido pela expressão

$$\nabla_{f\varphi} = \frac{\varphi N - \varphi P}{|d|} \tag{5.6}$$

Usando a Eq. 5.6, o gradiente de φ avaliado na face é calculado usando os dois valores centrais localizados entre a face f. Como alternativa, a discretização do gradiente pode ser calculada usando o teorema de Gauss para a integral no volume:

$$\int_V \nabla \varphi dV = \int_{\partial S} dS\varphi \approx \sum_f S\varphi_f \tag{5.7}$$

onde o valor de φ_f é obtido através de uma função de interpolação.

Termos fontes, como $S_\varphi(\varphi)$ mostrado na Eq. 5.2, podem ser funções genéricas de φ. Assim, estes termos são linearizados antes da discretização

$$S_\varphi(\varphi) = S_{I\varphi} + S_E \tag{5.8}$$

sendo que S_I e S_E podem ser dependentes de φ. Este termo integrado no volume de controle resulta em:

$$\int_V S_\varphi(\varphi) dV = S_{I\varphi P} V_P + S_E V_P \tag{5.9}$$

Ao assumir que os volumes de controle não variam com o tempo, a Eq. 5.2 pode ser reescrita usando as Eqs. 5.4, 5.5 e 5.9:

$$\int_t^{t+\Delta t} \left[\left(\frac{\partial(\rho\varphi)}{\partial t} \right)_P V_P + \sum_f F_{\varphi f(F,M,\gamma)} \right] dt = \int_t^{t+\Delta t} \left[\sum_f \Gamma_f \left(S.\Delta_{f\varphi} \right) + \left(S_{I\varphi P} + S_E \right) V_P \right] dt \tag{5.10}$$

A expressão acima usualmente é dita como a forma semi-discretizada da equação de transporte [Crowe et al, 1996].

Assumindo uma variação linear de $\varphi(t)$ no tempo, a discretização da derivada no tempo mostrada na Eq. 5.10 e a integral no tempo podem ser calculadas diretamente como:

$$\left(\frac{\partial(\rho\varphi)}{\partial t} \right)_P = \frac{\rho^n p \varphi^n p - \rho^0 p \varphi^0 p}{\Delta t} \tag{5.11}$$

$$\int_t^{t+\Delta t} \varphi(t) dt = \frac{1}{2} \left(\varphi^0 + \varphi^n \right) \Delta t \tag{5.12}$$

Em que $^{n} = (t + \Delta t)$ representa o novo valor de φ para passo de tempo a ser resolvido enquanto $\varphi^0 = \varphi(t)$ é o valor no passo anterior. Utilizando as Eqs. 5.10, 5.11 e 5.12, obtém-se a forma discretizada, temporal e espacial, da Eq.5.1.

$$\rho p \frac{\varphi^n p - \varphi^0 p}{\Delta t} Vp + \frac{1}{2}\sum_f F^n \varphi^n{}_{f(F,M,\gamma)} - \frac{1}{2}\sum_f \Gamma_f^n S.(\nabla_f \varphi\ )^n + \frac{1}{2}\sum_f F^0 \varphi_{f(F,M,\gamma)}{}^0 - \sum \Gamma_f^0 S.(\nabla_f \varphi)^0 =$$
$$\frac{1}{2}\left(S_E^n + S_E^0\right)Vp - \frac{1}{2}\left(S_I^n \varphi_P^n + S_I^0 \varphi_P^0\right)Vp \tag{5.13}$$

A formulação da discretização temporal apresentada na Eq. 5.13 é o método de segunda ordem de Crank-Nicholson. Este método requer os valores de φ e $\nabla\varphi$ no volume de controle e nas suas faces no instante de tempo a ser resolvido e passado.

Os valores nas faces são calculados a partir dos valores no centro dos volumes em cada lado da face e usando métodos adequados para a interpolação dos termos convectivos e difusivos.

5.2 Sistema Algébrico de Equações

De fato, o objetivo é obter o novo valor de φ_P. Como φ_f e $\nabla_f\varphi$ também dependem do valor de φ nos volumes vizinhos, a Eq. 5.13 pode ser representada como uma equação algébrica válida para o volume P.

$$a_P \varphi_P^n + \sum_N a_N \varphi_N^n = b_P \tag{5.14}$$

Ao considerar todos os volumes de controle da malha, a Eq. 5.14 pode ser escrita como um sistema algébrico de equações,

$$\mathrm{A}\varphi = b \tag{5.15}$$

onde A é uma matriz esparsa com coeficientes diagonais a_P e não-diagonais a_N, enquanto que φ e b são respectivamente os vetores das variáveis φ e os termos fontes referentes ao

centro dos volumes. A matriz A pode ser decomposta em duas matrizes contendo apenas os coeficientes diagonais D e os não-diagonais N, tal que:

$$A = D + N \tag{5.16}$$

Portanto, a estrutura da matriz de coeficientes A pode variar dependendo da dimensão do problema, da forma de discretização das equações e da ordenação dos volumes de controle na malha. A estrutura da matriz é um parâmetro importante na escolha do método mais apropriado para resolver o sistema linear. Em aplicações CFD, cerca de 20% do tempo de computação necessário para resolver um dado problema está associado à solução de sistema linear. Os outros 80% estão associados à obtenção das equações discretizadas. (Rusche, 2002).

Os métodos numéricos para solução de sistemas lineares podem ser divididos em duas classes: diretos e iterativos. Os métodos diretos determinam a solução exata, há menos de erros de truncamento, em um número finito de operações matriciais. Eles são indicados para sistemas lineares com um número não muito grande de equações (até algumas centenas) ou com uma matriz do sistema cuja estrutura permita um método especial de solução. A limitação destes métodos a sistemas pequenos ocorre por dois motivos. Primeiro, o número de operações (matemáticas) cresce muito rapidamente com o tamanho do sistema, o que aumenta o custo computacional superlinearmente. Além disso, o alto número de operações leva ao aumento dos erros de truncamento, o que degrada a exatidão da solução.

Os métodos iterativos foram desenvolvidos para superar as deficiências dos métodos diretos no tratamento de grandes sistemas. Os métodos iterativos também ganharam espaço devido a sua facilidade de implementação em computadores de alto desempenho com memória compartilhada ou distribuída (supercomputadores e clusters), quando comparado aos métodos diretos. Alguns dos métodos iterativos, como o Gauss-Seidel, o Gradiente Conjugado (CG) e suas variantes e o Algebraic Multigrid (AMG) podem ser citados (Saad,2003) apud Silva, (2008) e estão implementados no OpenFOAM.

5.3 Notação da Formulação Discreta

O processo de discretização pode ser efetuado de forma implícita ou explícita, onde a primeira é utilizada para construir o sistema linear através da discretização. Já a segunda forma realiza os cálculos da operação de forma explícita, usando os valores de φ no instante

atual. (RUSCHE, 2002) *apud* WELLER, (2002) introduziu uma notação específica para volumes finitos que permite especificar o tratamento aplicado no processo de discretização.

Com isso, a discretização de um operador $\mathcal{L}$ é representada por $\lfloor \mathcal{L}[\varphi] \rfloor$. O operador $\mathcal{L}$ pode ser derivativo no espaço ou no tempo, ou mesmo um termo fonte que possui tratamento implícito. A variável dependente φ colocada entre colchetes deve ter tratamento implícito e ser idêntica em todos os termos da equação. Por outro lado, os termos discretizados de forma explícita não são representados entre colchetes.

Assim, as operações implícitas e explícitas podem ser diferenciadas usando esta notação. Um resumo é apresentado na figura 5.1.

Termo	Notação por volumes finitos	Termo discretizado
Derivada no tempo	$\left\lfloor \frac{\partial \rho[\varphi]}{\partial t} \right\rfloor$	$\frac{\rho_P^n \varphi_P^n - \rho_P^0 \varphi_P^0}{\Delta t} V_P$
Convectivo	$\lfloor \nabla \cdot (F[\varphi]_{f(F,M,\gamma)}) \rfloor$	$\sum_f F \varphi_{f(F,M,\gamma)}$
Difusivo	$\lfloor \nabla \cdot (\Gamma \nabla[\varphi]) \rfloor$	$\sum_f F \varphi_{f(F,M,\gamma)}$
Divergente	$\nabla \cdot \varphi$	$\sum_f \mathbf{S} \cdot \varphi_f^0$
Gradiente no volume	$\nabla \varphi$	$\sum_f \mathbf{S} \varphi_f^0$
Fonte Implícito	$\lfloor S_I[\varphi] \rfloor$	$S_I V_P \varphi^n$
Fonte Explícito	S_E	$S_E V_P$

Figura 5.3- tabela de notação de operações implícitas e explicitas

Seguindo a notação mostrada na figura 5.1, o sistema linear obtido pela discretização da equação de transporte (Eq. 5.1) pode ser escrita como colocado na equação 5.17.

$$\mathrm{A} := \left[\frac{\partial \rho[\varphi]}{\partial t} \right] + \left[\nabla . \left(F[\varphi]_{f(F,M,\gamma)} \right) \right] - \left[\nabla . (\Gamma \nabla[\varphi]) \right] - \left[S_I[\varphi] \right] - S_E \tag{5.17}$$

Note que a equação discretizada para cada volume de controle é idêntica àquela mostrada na Eq. 5.2.

Uma vez que o sistema linear é montado, pode-se definir operadores especiais para extrair os coeficientes da matriz e seus termos fonte usando respectivamente $A_A \equiv A$ e $A_S \equiv b$. Da mesma forma, matrizes contendo os coeficientes diagonais e não diagonais também podem ser obtidas a partir de $A_D \equiv D$ e $A_N \equiv N$ respectivamente.

O operador H ", AH $_H$, definido na sequência, é usado exaustivamente nos algoritmos numéricos do OpenFOAM. Sua definição parte do método iterativo de Jacobi para obtenção de uma solução aproximada das equações discretizadas como mostrado na Eq. 5.18.

$$\varphi \approx A_D^{-1} A_H \tag{5.18}$$

Logo, o operador _H_ pode ser obtido utilizando as Eqs. 5.16 e 5.18.

$$A_H = A_S - A_N \varphi \tag{5.19}$$

Definitivamente, o conhecimento das equações, modelos, métodos de discretização, solução e algoritmos é muito mais importante que os conceitos de programação.

Com esse ponto de vista, recomenda-se fortemente que o leitor interessado em desenvolver códigos no OpenFOAM estude a fundo o trabalho de JASAK, 1996. Este último apresenta detalhadamente vários aspectos sobre a formulação numérica, incluindo a metodologia de discretização e condições de contorno, e a teoria dos algoritmos implementados, como o acoplamento pressão-velocidade, correção dos fluxos em malhas não estruturadas, implementados no OpenFOAM. Toda a implementação do código é baseada na teoria apresentada em JASAK (1996).

6 MODELAGEM BUBBLEFOAM

6.1 Equacionamento

Na abordagem de dois fluidos, uma equação da continuidade e uma equação de movimento são resolvidas para cada uma das fases presentes no sistema. Estas equações podem ser obtidas por cálculo da média das equações de fluxo de fase única. As equações de continuidade para cada fase φ tem a forma:

$$\frac{\partial}{\partial t}\left(\alpha_\varphi \rho_\varphi\right) + \nabla.\left(\alpha_\varphi \rho_\varphi U_\varphi\right) = 0 \tag{6.1}$$

Onde α_φ representa a fração da fase φ, ρ_φ é a densidade do material constituída pela mesma fase, e $\mathbf{U}_\varphi$ é da velocidade da fase.

O movimento da fase é dado pela equação:

$$\frac{\partial}{\partial t}\left(\alpha_\varphi \rho_\varphi U_\varphi\right) + \nabla.\left(\alpha_\varphi \rho_\varphi U_\varphi U_\varphi\right) + \nabla.\alpha_\varphi \tau_\varphi + \nabla.\left(\alpha_\varphi R_\varphi\right) = -\alpha_\varphi \nabla p + \alpha_\varphi \rho_\varphi g + M_\varphi, \tag{6.2}$$

Com τ_φ é a fase do tensor laminar, assumindo ser Newtoniana, $\mathbf{R}_\varphi$ é a fase do tensor de Reynolds, p é a pressão, $\mathbf{g}$ é o vetor da aceleração gravitacional, e $\mathbf{M}_\varphi$ é o termo da troca de movimento.

O tensor da fase laminar é definido como:

$$\tau_\varphi = -\rho_\varphi \nu_\varphi \left[\nabla U_\varphi + \nabla^{\mathrm{T}} U_\varphi\right] + \frac{2}{3}\rho_\varphi \nu_\varphi \left(\nabla.U_\varphi\right) I \tag{6.3}$$

onde ν_φ é a viscosidade cinemática molecular do fluido que constitui a fase φ, e $\mathbf{I}$ é a matriz identidade.

O tensor da fase de Reynolds é dado por:

$$R_\varphi = -\rho_\varphi \nu_{\varphi,t} \left[\nabla U_\varphi + \nabla^{\mathrm{T}} U_\varphi\right] + \frac{2}{3}\rho_\varphi \nu_{\varphi,t} \left(\nabla.U_\varphi\right) I + \frac{2}{3}\rho_\varphi k_\varphi I \tag{6.4}$$

κ_φ é a energia cinética turbulenta da fase (A implementação do bubbleFoam, assume que $\kappa_a = \kappa_b$. Em outras palavras, a energia cinética turbulenta é identica para ambas as fases presentes no sistema), e $\nu_{\varphi,t}$ é a viscosidade cinemática turbulenta, definida como:

$$\nu_{\varphi,t} = C_\mu \frac{k_\varphi^2}{\varepsilon_\varphi}, \tag{6.5}$$

C_μ é uma constante, e ε_φ é a fase da taxa de dissipação turbulenta. A fase da viscosidade efetiva é calculada. A fase da viscosidade efetiva é calculada com a soma da fase da viscosidade molecular e da viscosidade turbulenta. Como:

$$\nu_{\varphi,eff} = \nu_\varphi + \nu_{\varphi,t}. \tag{6.6}$$

O termo de troca dinâmica pode ser decomposto com uma contribuição de arrasto, uma força de sustentação e uma contribuição de massa virtual.

$$M_\varphi = M_{\varphi,drag} + M_{\varphi,lift} + M_{\varphi,vm}. \tag{6.7}$$

$M_{\varphi,drag}$ refere-se a força de arrasto, em que os termos são modelados em termos de mistura. Em particular as fases a e b consideradas no modelo de dois fluidos, em que a representa a fase dispersa o termo de resistência é descrito pela equação:

$$M_{a,drag} = \frac{3}{4}\alpha_a \alpha_b \left(\frac{C_{D,a}\rho_b}{d_a} + \alpha_a \frac{C_{Db}\rho_a}{d_b} \right) |U_r| U_r, \tag{6.8}$$

Em que d_a e d_b são diâmetros das partículas de fase, $U_r = U_a - U_b$ é relativa ao vetor velocidade, e $C_{D,a}$ e $C_{D,b}$ são os coeficientes de arrasto calculados em relação a cada uma das fases de acordo com:

$$C_{D,\varphi} = \frac{24}{\mathrm{Re}_\varphi}\left(1 + 0.15\,\mathrm{Re}_\varphi^{0.687}\right), \tag{6.9}$$

Em que:

$$\mathrm{Re}_{\varphi} = \left|U_{\varphi}\right| d_{\varphi} / \nu_{\varphi} \tag{6.10}$$

O elevador para o termo modelado (Note que U realmente deve ser U_b. Relatamos U de coerência para implementação do código.)

$$M_{\varphi,lift} = \alpha_a \alpha_b \left(\alpha_b C_{a,lift} \rho_b + \alpha_a C_{b,lift} \rho_a\right) U_r . \nabla . U, \tag{6.11}$$

Enquanto que o termo de força de massa virtual é avaliado como:

$$M_{\varphi,vm} = \alpha_a \alpha_b C_{vm} \rho_b \left(\left.\frac{dU_b}{dt}\right|_b - \left.\frac{dU_a}{dt}\right|_a \right), \tag{6.12}$$

Em que:

$$\left.\frac{dU_a}{dt}\right|_a = \frac{\partial U_a}{\partial t} + U_a . \nabla U_a \tag{6.13}$$

e

$$\left.\frac{dU_b}{dt}\right|_b = \frac{\partial U_b}{\partial t} + U_b . \nabla U_b \tag{6.14}$$

6.2 Modelo de Turbulência

O solver do bubbleFoam usa o modelo de turbulência de duas equações $\kappa - \varepsilon$ para fase contínua, e representa a influência da turbulência na fase dispersa. A equação da energia cinética da fase turbulenta contínua (b) lê-se:

$$\frac{\partial}{\partial t}\left(\alpha_b k_b\right) + \nabla . \left(\alpha_b U_b k_b\right) - \nabla^2 \left(\sigma_k \nu_{b,eff} k_b\right) = \alpha_b G - \alpha_b \varepsilon_b \tag{6.15}$$

Onde G é a produção da energia cinética turbulenta e é dada por:

$$G = 2\nu_{b,t}\left[\nabla U_b . dev\left(\nabla U_b + \nabla^{\mathrm{T}} U_b\right)\right] \tag{6.16}$$

e σ_κ é o número de Schmidt turbulento. A taxa de dissipação turbulenta é determinada pela resolução da equação de transporte.

$$\frac{\partial}{\partial t}\left(\alpha_b \varepsilon_b\right) + \nabla.\left(\alpha_b U \varepsilon_b\right) - \nabla^2\left(\sigma_\varepsilon \nu_{b,eff} \varepsilon_b\right) = C_1 \alpha_b G \frac{\varepsilon_b}{h_b} - C_2 \alpha_b \frac{\varepsilon_b^2}{k_b} \tag{6.17}$$

onde C_1 e C_2 são constantes do modelo de turbulência.

A viscosidade turbulenta da fase continua é calculada através da definição:

$$\nu_{b,t} = C_\mu \frac{k^2}{\varepsilon}, \tag{6.18}$$

Enquanto que a viscosidade turbulenta da fase dispersa é avaliada como:

$$\nu_{a,t} = C_t^2 \nu_{b,t}, \tag{6.19}$$

Sendo o coeficiente C_μ e o coeficiente de resposta a turbulência C_t constantes do modelo. Funções padrão de parede são adotadas para tratar região próxima à parede.

6.3 Implementação do solver BubbleFoam

A solução numérica das equações de duas fases depende de um algoritimo segregada a um procedimento PISO estendido de fluxo para duas fases (Oliveira e Issa,2003). As equações de movimento são manipuladas para estabilizar o sistema de equação no limite da gama de frações de volume a fim de evitar singularidades como sugerido em (Oliveira e Issa,2003; Weller,2002). Os detalhes da metodologia numérica serão resumidos a seguir:

6.4 Equação de movimento da fase

O método numérico utilizado no solver do bubbleFoam basea-se na formulação da fase intensiva da fase dinâmica propostos em (Weller,2002), para ultrapassar a instabilidade resultante quando a fração de volume de fase tende a tornar-se zero, o que é uma situação frequente e completamente separada dos fluxos. De acordo com esta abordagem, assumindo que a densidade ρ_φ da fase seja constante, a equação de movimento é reescrita de forma não conservadora para extrair a fração de volume das condições de transporte, levando a:

$$\frac{\partial U_\varphi}{\partial t} + U_\varphi.\nabla U_\varphi + \nabla.\left(\frac{\tau_\varphi}{\rho_\varphi} + R_\varphi\right) + \frac{\nabla \alpha_\varphi}{\alpha_\varphi}.\left(\frac{\tau_\varphi}{\rho_\varphi} + R_\varphi\right) = -\frac{\nabla p}{\rho_\varphi} + g + \frac{M_\varphi}{\alpha_\varphi \rho_\varphi}. \tag{6.20}$$

Renomeando o tensor da tensão total como:

$$R_{\varphi,eff} = \frac{\tau_\varphi}{\rho_\varphi} + R_\varphi, \tag{6.21}$$

E decompondo-o num componente difusivo e um termo de correção:

$$R_{\varphi,eff} = R^D_{\varphi,eff} + R^C_{\varphi,eff} \tag{6.22}$$

Com

$$R^D_{\varphi,eff} = -\nu_{\varphi,eff} \nabla U_{\varphi,} \tag{6.23}$$

$$R^D_{\varphi,eff} = R_{\varphi,eff} + \nu_{\varphi,eff} + \nabla U_\varphi \tag{6.24}$$

Obtém-se

$$\frac{\partial U_\varphi}{\partial t} + U_\varphi.\nabla U_\varphi - \nabla.\left(\nu_{\varphi,eff} \nabla U_\varphi\right) + \nabla.R^C_{\varphi,eff} - \nu_{\varphi,eff} \frac{\nabla \alpha_\varphi}{\alpha_\varphi} \nabla U_\varphi + \frac{\nabla \alpha_\varphi}{\alpha_\varphi}.R^C_{\varphi,eff} = -\frac{\nabla p}{\rho_\varphi} + g + \frac{M_\varphi}{\alpha_\varphi \rho_\varphi} \tag{6.25}$$

Finalmente a introdução da velocidade de fase total

$$U_{\varphi}^{\mathrm{T}} = U_{\varphi} - \nu_{\varphi,eff} \frac{\nabla \alpha_{\varphi}}{\alpha_{\varphi}}, \tag{6.26}$$

Introduzindo

$$A_d = \frac{3}{4}\left(\alpha_b \frac{C_{D,a}\rho_b}{d_a} + \alpha_a \frac{C_{Db}\rho_a}{d_b} \right)\left|U_r\right|, \tag{6.27}$$

$$A_1 = \left(\alpha_b C_{a,lift}\rho_b + \alpha_a C_{b,lift}\rho_a\right)U_r.\nabla.U, \tag{6.28}$$

$$A_{vm} = C_{vm}\rho_b \tag{6.29}$$

E considerando, por exemplo, fase a, a equação de movimento se torna

$$\frac{\partial U_a}{\partial t} + U_a^{\mathrm{T}}.\nabla U_a - \nabla.\left(\nu_{a,eff}\nabla U_{\varphi}\right) + \nabla.R_{\varphi,eff}^{C} + \frac{\nabla \alpha_a}{\alpha_a}.R_{a,eff}^{C} = -\frac{\nabla p}{\rho_a} + g + \frac{a_b}{\rho_a}\left[A_d\left(U_b - U_a\right) + A_1 + A_{vm}\left(\frac{dU_b}{dt}\Big|_b - \frac{dU_a}{dt}\Big|_a \right)\right] \tag{6.30}$$

A equação do movimento pode ser reescrita introduzindo os fluxos totais de movimentos convectivos

$$\phi_a^{\mathrm{T}} = \phi_a - \nu_{a,eff,f} \frac{S_f \nabla_f^{\perp}}{\alpha_{a,f} + \delta}, \tag{6.31}$$

e

$$\phi_b^{\mathrm{T}} = \phi_b - \nu_{b,b,f} \frac{S_f \nabla_f^{\perp} \alpha_b}{\alpha_{b,f} + \delta}, \tag{6.32}$$

onde $\nabla_{\mathrm{f}}^{\perp}\alpha_{\varphi}$ é o gradiente de superfície normal a superfície S_{f} de fração da fase, os valores subscritos f indicam valores calculados nas faces das células, e δ é o menor número. Uma vez que essa substituição é executada, a equação de movimento pode ser discretizada. Em especial a correção do termo de estresse é lida explicitamente, bem como os termos de elevação.

- os termos convectivos e difusivos são tratados totalmente implicitamente;
- o gradiente de pressão não é incluído na equação de momento diretamente, mas o seu efeito é contabilizado quando a velocidade de fase são corrigidos;
- o mesmo procedimento adotado para o gradiente de pressão é usada para a gravidade, que não aparece na equação de momentum no código, mas seu efeito é contabilizado na equação da pressão.

As discretizações das equações podem ser representadas como:

$$U_a = \frac{H_a}{A_a} - \frac{\nabla p}{\rho_a A_a}, \tag{6.33}$$

$$U_b = \frac{H_b}{A_b} - \frac{\nabla p}{\rho_a A_b} \tag{6.34}$$

Onde A_φ é a parte diagonal da matriz proveniente da discretização da fase da equação de movimento e H_a é a parte remanescente. Estas equações foram utilizadas para corrigir a velocidade após atualizadoo campo de pressão.

6.5 Equação da Continuidade da Fase

A equação da continuidade da fase deve ser resolvida garantindo que a fração de cada uma das fases seja mantida entre zero e um. Para obter este resultado, a equação de continuidade da fase dispersa é reescrita como uma função da velocidade média e relativa. Reescrevendo a velocidade de fase em função da velocidade relativa.

$$U_r = U_a - U_b, \tag{6.35}$$

E a velocidade média ponderada da fase das frações:

$$U = \alpha_a U_a + \alpha_b U_b, \tag{6.36}$$

Encontra-se:

$$U_a = U + \alpha_b U_r. \tag{6.37}$$

Substituindo na fase da equação da continuidade, obtém-se uma nova expressão:

$$\frac{\partial \alpha_a}{\partial t} + \nabla.(\alpha_a U) + \nabla.(\alpha_a (1-\alpha_a) U_r) = 0 \tag{6.38}$$

Que, é iterativamente (nota-se que a equação não é linear) resolvida de uma maneira totalmente implícita e proporciona uma solução limitada para o campo de fração da fase.

A fração de volume da fase contínua pode ser calculada como $\alpha_b = 1 - \alpha_a$. Entretanto, em alguns casos, esta abordagem não poderia ser satisfatória do ponto de vista numérico, e a solução de uma equação da fase da continuidade também é recomendada para a fase contínua, a fim de aumentar a taxa de convergência. Em tal caso, depois da resolução de uma equação da continuidade para a fase b. A fase de volume dispersa é assegurada por avaliar o novo valor da fração da fase dispersa como:

$$\alpha_{a,bounded} = \frac{1}{2}\left[1 - (1-\alpha_a)^2 + (1-\alpha_b)^2\right], \tag{6.39}$$

E recompondo a fase da fração continua como:

$\alpha_{b,bounded} = 1 - \alpha_{a,bounded}$.

6.6 Equação da Pressão

A equação da pressão é obtida impondo que o divergente do fluxo de mistura é zero.

$$\nabla.\phi = \nabla(\alpha_{a,f}\phi_a + \alpha_{b,f}\phi_b) = 0, \tag{6.40}$$

A fase do fluxo é obtida interpolando as equações

$$U_a = \frac{H_a}{A_a} - \frac{\nabla p}{\rho_a A_a} \tag{6.41}$$

$$U_b = \frac{H_b}{A_b} - \frac{\nabla p}{\rho_b A_b} \tag{6.42}$$

Nas células de faces, lê-se:

$$\phi_a = \phi_a^* - \frac{1}{\rho_a A_a}\Big|_f S_f \nabla_f^{\perp} p + \phi_{dg,a} \tag{6.43}$$

$$\phi_b = \phi_b^* - \frac{1}{\rho_b A_b}\Big|_f S_f \nabla_f^{\perp} p + \phi_{dg,b} \tag{6.44}$$

$$\phi_a^* = \frac{H_a}{A_a}\Big|S_f \tag{6.45}$$

$$\phi_b^* = \frac{H_b}{A_b}\Big|S_f \tag{6.44}$$

Formula da pressão:

$$\nabla.\left(D_p \nabla_p\right) = \nabla.\left(\varepsilon_g \phi_g + \varepsilon_l \phi_l\right) \tag{6.46}$$

O efeito da gravidade e da parte explícita do arrasto estão incluídos nos fluxos como:

$$\phi_{dg,a} = \frac{\alpha_b C_D}{\rho_a A_a}\Big|_f + \phi_b^* + \frac{g.S_f}{A_a} \tag{6.47}$$

$$\phi_{dg,b} = \frac{\alpha_a C_D}{\rho_b A_b}\Big|_f + \phi_a^* + \frac{g.S_f}{A_b} \tag{6.48}$$

7 MÉTODO

Iniciou a modelagem considerando-se hipóteses simplificadoras e foi aumentando o grau de complexidade de acordo com os sucessos obtidos. Foi considerado o regime laminar, onde aplicou a equação de Navier-Stokes em duas dimensões para o líquido simultaneamente com a equação do transporte de massa. Até chegar em um modelo de transporte turbulento composto por EDP's semelhantes às equações de 1 a 8 e simular com um método numérico apropriado de volumes finitos (Fortuna, 2000; Rice e Do, 1995; Carnahan et al., 1969; Lesieur, 1997; Mathieu e Scott, 2000; Rodi, 2000; Pope, 2003; Bech, 2005; Smith, 1998; Zboray e Cachard, 2005; Bernard et al, 2000; Rabha e Buwa, 2010; Dhotre e Smith, 2007; Maliska, 2004). Uma vez obtido um modelo satisfatório, passa-se a comparação com dados da planta experimental a ser construída.

7.1 Material e métodos

Este projeto de doutorado foi vinculado a um auxílio de pesquisa FAPESP, pelo processo de nº. 0302768-1.

Para a modelagem, foi usado o trabalho de mestrado de Oliveira (2008), o qual usou a técnica PIV ("particle image velocimetry").

Na técnica PIV, as imagens foram tratadas em um programa específico que forneceu valores de velocidade em diversos pontos (campo de velocidades) de acordo com a relação entre o deslocamento de partículas (traçadoras fluorescentes) e a frequência de captura. O uso do laser foi essencial para obter dados experimentais de viscosidade turbulenta e validar os pretendidos modelos. Foi usado simulador programado e Paraview.

A figura 7.1 mostra a planta experimental utilizada no trabalho de Oliveira (2008).

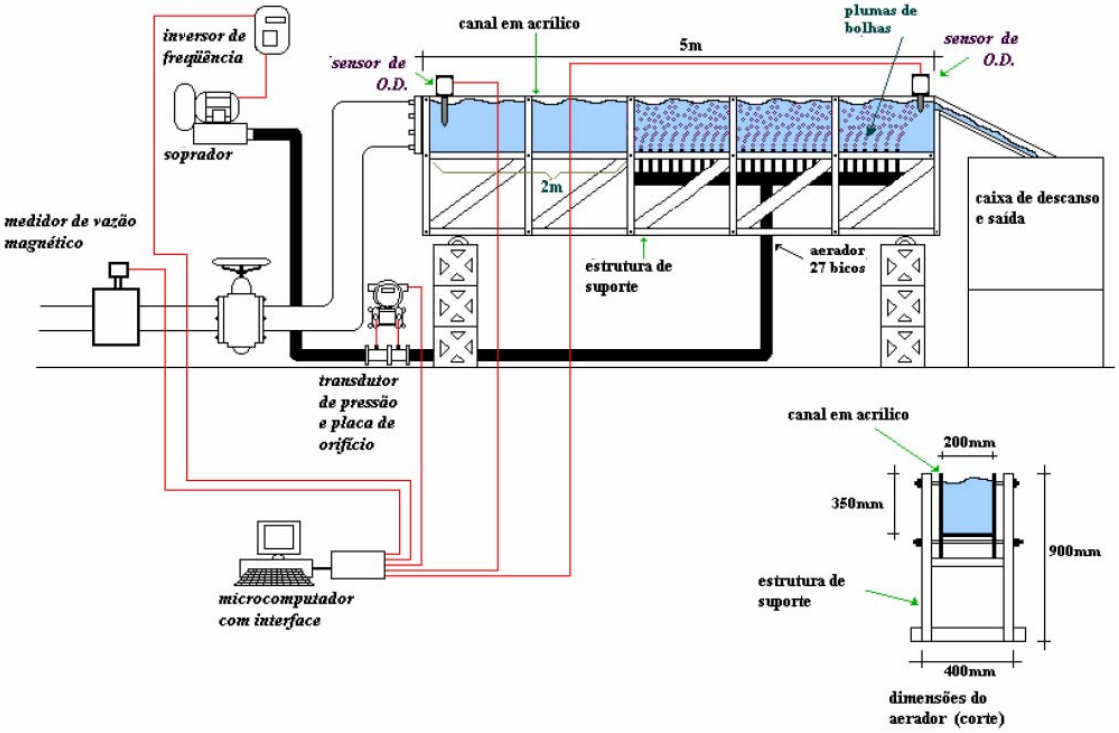


Figura 7.1 -Planta Experimental.

A parte experimental deste estudo foi realizada no laboratório de Hidráulica Ambiental, o qual se encontra localizado no Centro de Recursos Hídricos e Ecologia Aplicada (CRHEA) da Escolade Engenharia de São Carlos, Universidade de São Paulo.

A planta experimental constitui-se de um aerador de fluxos cruzados sem agitação mecânica (conforme observado na figura 7.1).

A vazão de entrada de água no canal foi de 10L/min medida por um rotâmetro, fornece uma velocidade de entrada de 0,00238m/s para a secção transversal do canal de área = 0,35m x 0,2m. Os aeradores foram distanciados em 10 cm e a vazão média dos difusores é de 0,037m^3/s. Por fim, compara-se os resultados experimentais com os resultados numéricos.

A figura 7.2 mostra o domínio computacional.

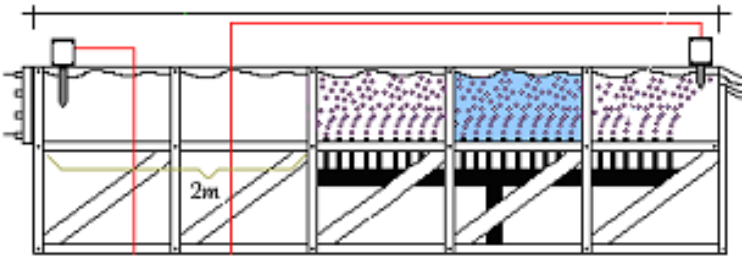


Figura 7.2- Domínio computacional.

Por fim, compara-se os resultados numéricos com os dados experimentais.

7.1.1 Método Numérico

O escoamento utilizado neste estudo foi bidimensional, optou-se pelo pacote CFD não comercial OpenFOAM usando-se o solver padrão BubbleFoam.

Desta forma, para a resolução do modelo, o solver BubbleFoam adota as seguintes simplificações:

- as fases são tratadas como fluidos incompressíveis;
- apenas duas fases são contabilizadas;
- não há transferência de massa na interface nem reações químicas;
- a modelagem da turbulência é feita somente para a fase líquida.

7.2 Geração de malha

Uma malha computacional é constituída por linhas e pontos, os pontos são considerados onde essas linhas se interceptam e servem de orientação para o cálculo de propriedades físicas baseado num modelo matemático. Uma malha computacional nada mais é que uma representação ou a "discretização" do plano físico utilizado na simulação numérica. A solução de um sistema de equações diferenciais (modelo matemático) pode ser geralmente simples quando empregada uma malha bem construída.

O método mais simples para se gerar uma malha computacional é fazê-la manualmente, desenhando a geometria que se deseja discretizar numa folha de papel milimetrado, identificando as coordenadas de cada ponto formado pela intersecção de várias linhas que representam toda região da geometria desejada. Essas coordenadas então são informadas ao computador, que automaticamente são lidas formando a malha computacional

da geometria. Há outros métodos classificados como automáticos para se gerar malhas computacionais: os algébricos e os diferenciais.

Os algébricos empregam diferentes tipos de interpolações e são bastante versáteis e rápidos. Os diferenciais, assim chamados por empregarem equações diferenciais, são mais gerais, mas, em contrapartida, apresentam tempo de computação sensivelmente maior e uma maior elaboração matemática. (MALISKA, 1995, p. 253).

Para Bortoli, (2000), uma malha dita estruturada é quando cada volume interno tem sempre o mesmo número de vizinhos e a sua numeração tem uma sequência natural. E quando se diz que uma malha é não-estruturada, tem-se o número de vizinhos variando de volume para volume, ficando difícil estabelecer uma regra de ordenação.

Neste contexto, as condições de contorno e malha numérica são apresentadas.

A malha foi construída pelo construtor de malhas nativo do OpenFOAM chamado blockmesh, esse gerador de malhas é fornecido como uma utilidade do software.

De acordo com o OpenFOAM o blockMesh pode criar malhas paramétricas com taxa de expansão das células e contornos curvos. A malha é gerada a partir de um arquivo dicionário, chamado blockMeshDict, localizado no diretório do caso em estudo (constante/polyMesh).

O blockMesh lê o arquivo BlockMeshDict, o interpreta e gera a malha. Amalha gerada é representada por um conjunto de arquivos de dados com informações dos pontos, faces, células e dos contornos da geometria. Este conjunto encontra-se nos arquivos points, faces, cellZones e boundary respectivamente, no mesmo diretório do blockMeshDict. (Fontoura, 2013).

Para a discretização física, utilizou-se uma geometria com iguais dimensões das encontradas no problema padrão, dotada de malha com características estruturadas totalizando 14.000 elementos discretos. A Figura 5.4 apresenta a malha numérica utilizada e as condições de contorno.

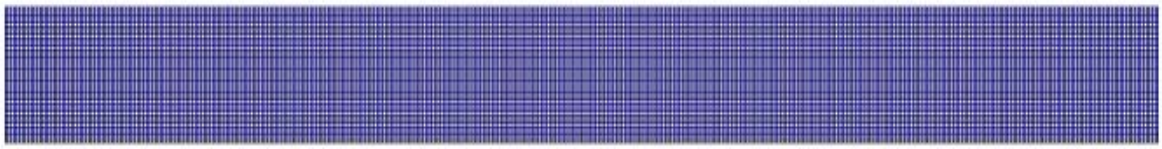


Figura 7.3 - malha adotada para simulação

8 RESULTADOS E DISCUSSÕES DAS SIMULAÇÕES REALIZADAS

8.1 Resultados experimentais.

A seguir são apresentados os dados experimentais obtidos em Oliveira, 2008.

A Figura 8.1 apresenta os valores de velocidade do gás através da representação de contornos, onde a cor do contorno relaciona-se a escala de cor para a velocidade.

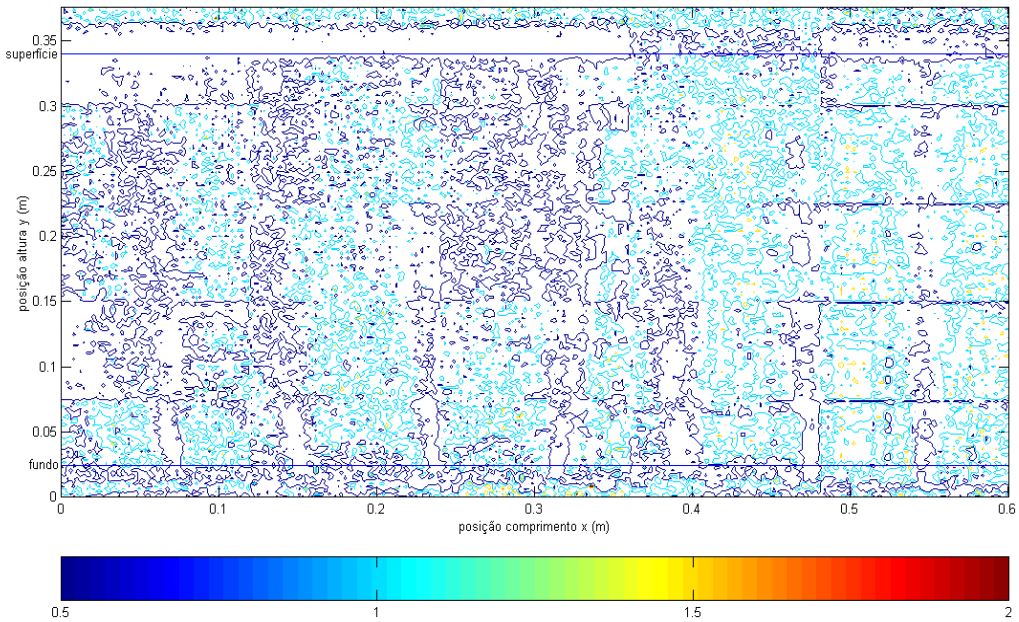


Figura 8.1 - Disposição do campo de velocidades em contornos obtidos na fase experimental. A cor do contorno relaciona-se à escala de cor para velocidades em m/s.

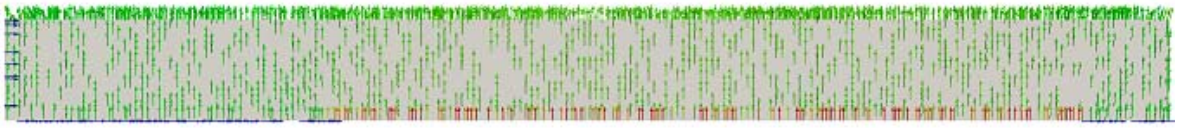
Analisando os resultados de velocidades na figura 8.2 pode-se verificar um comportamento assimétrico do perfil em relação à posição vertical (movimento característico de escoamento bifásico).

O movimento turbulento do líquido mostrando a interação com a fase gasosa que dispersa no ar ambiente. Verifica-se uma região com perfil de velocidades mais elevadas na parte central do canal, corresponde à região onde se encontra o acúmulo de gás. Nesta região a pequena fração do líquido sofre os efeitos da transferência de quantidade de movimento entre as fases ao ser arrastada pelo grande volume do gás e assume um perfil de velocidade maior devido a sua menor viscosidade molecular.

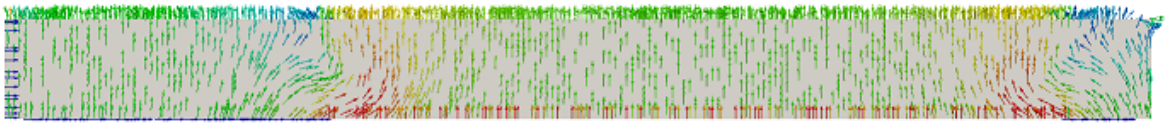
A figura 8.1 mostra disposição de velocidade obtido na fase experimental e para efeito de comparação entre os dados experimentais e os dados obtidos na simulação. Nesta fase considera-se a fase laminar, onde os valores de velocidade obtidos foram constantes e os resultados obtidos na simulação ficaram próximos aos valores obtidos na fase experimental que foram entre 0 e 1 (de acordo com a Figura 8.2).

8.2 Resultados preliminares de velocidades das bolhas para o regime laminar

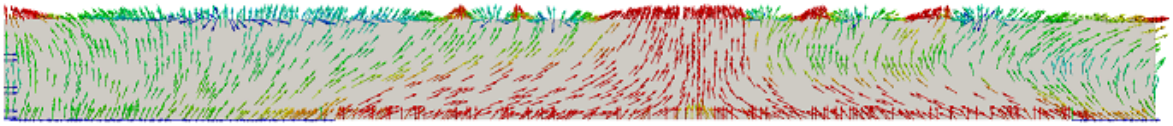
Em uma etapa preliminar de testes, a estratégia foi colocar a entrada de ar linearmente espalhada na parede inferior, não em pontos de bicos de injeção, ou seja, em um trecho do contorno inferior a condição era somente de entrada de gás. A simulação foi realizada para regime laminar com as condições já apresentadas. Os perfis de velocidade do gás no canal são representados por vetores e os gráficos correspondentes a uma simulação transiente em que se pode observar o desenvolvimento do perfil de velocidade para os tempos 0,1 s, 1,0 s, 10 s, 15 s, 20 s e 23 s (Figura 8.2). Iniciou-se a simulação em regime transiente com uma velocidade constante, dada pelos vetores verdes com 0,2 m/s, em todo o meio, partindo-se da perturbação na região central, impondo-se aumento na velocidade, representada pelos vetores vermelhos. Aqui não são mostradas as velocidades do líquido, entretanto, a sua entrada é de 0,0023m/s, que é condição experimental. Pode-se observar a influência de uma fase na outra. O gás tem um movimento na direção ascendente enquanto sofre influência do líquido e é arrastado na direção X.



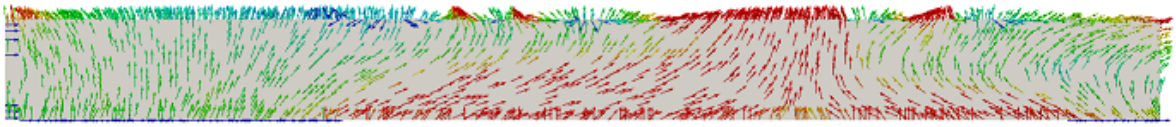
Tempo 0,1s



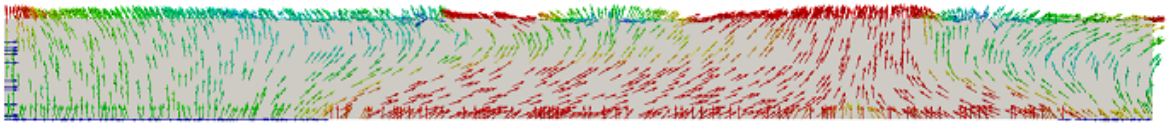
Tempo 1,0s



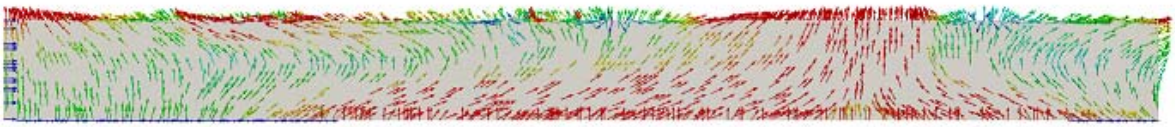
Tempo 10,0s



Tempo 15,0s



Tempo 20,0s



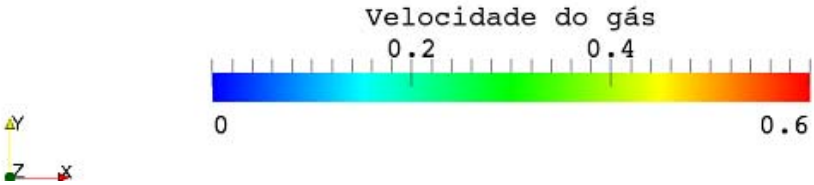


Figura 8.2 - Vetor velocidade em m/s variando no tempo de 0,1 a 23,0s

Nessa primeira simulação, embora não tão representativa devido à grande simplificação do modelo, pôde-se observar valores de velocidade máxima do gás de 0,6 m/s muito próximos aos valores encontrados através dos dados coletados experimentalmente com PIV (OLIVEIRA, 2008).

8.3 Resultados com os difusores no fundo do canal para o regime laminar

Para essa simulação a estratégia foi definir os bicos de aeração exclusivamente em pontos específicos na parede do fundo. Construiu-se um campo de fração de gás com valor fixo de 0.5 na linha central de cada difusor, somente com o intuito de evidenciar a posição de cada bico aerador (Figura 8.3).

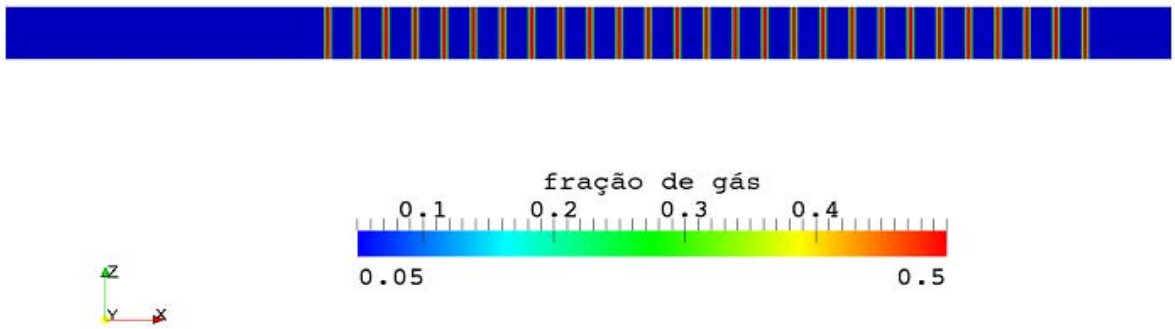


Figura 8.3 - Posição dos difusores

O tempo total de simulação em regime laminar no BubbleFoam foi de 29 segundos considerando os dados já comentados. Os resultados para os perfis de fração volumétrica da fase gasosa são apresentados de 0 a 3,5s a cada 0,1s e depois a 17s e 29s. Após 17 segundos foi verificado que o sistema atingiu o regime permanente. Nessas simulações observa-se o transiente da partida da operação. Evidencia-se a entrada de gás no meio líquido no fundo do canal, após 0, 0,1 e 0,2 segundos, o líquido começa a ser aerado com ar proveniente dos difusores formando-se assim um sistema bifásico líquido-gás. A evolução da pluma de bolhas (fase gasosa) apresenta-se coerente desde o seu início e converge para um estado estacionário após 17 s (Figura 8.13). A natureza de interação entre os movimentos das fases também é indicada (Figura 8.13), desde que a corrente líquida deforma a pluma até arrastá-la para o contorno de saída do líquido na parede lateral direita do domínio. Observa-se que há uma dinâmica rápida, pois as primeiras partes superiores da pluma (primeiras bolhas) atingem a superfície em muito menos de 1s. No experimento real, embora não medidos esse tempo, mas

por sentimento físico, percebia-se que as bolhas possuíam um movimento muito rápido na linha central da bolha, possivelmente na ordem de 1 a 2 segundos. Essa diferença perceptível, possivelmente, pode ser atribuída, entre outros fatores, à turbulência, cuja modelagem aqui não foi levada em conta. Entretanto, mesmo considerando a modelagem laminar, os resultados para as revoluções da pluma (Figuras 8.7 em diante) mostram claramente o aspecto tendencioso de um movimento turbilhonar, o que leva a considerar a aplicação de uma modelagem de simulação direta das Equaçãoes de Navier-Stokes como sugestão para o futuro.

A dispersão da fase gasosa no meio líquido fica muito evidente a partir das Figuras (8.12). A fase gasosa se dilui em todo o domínio após a região de aeração, como era de se esperar.

A Figura 8.13 mostra um setor de velocidades da fase gasosa após o regime permanente estabelecido, onde evidencia-se a tendência de movimentação das bolhas. O movimento sofre um desvio curvado com a advecção do líquido, como se espera em um sentimento físico, entretanto, no experimento, devido à velocidade do líquido ser pequena comparada com a das bolhas, essa curvatura é insignificante.

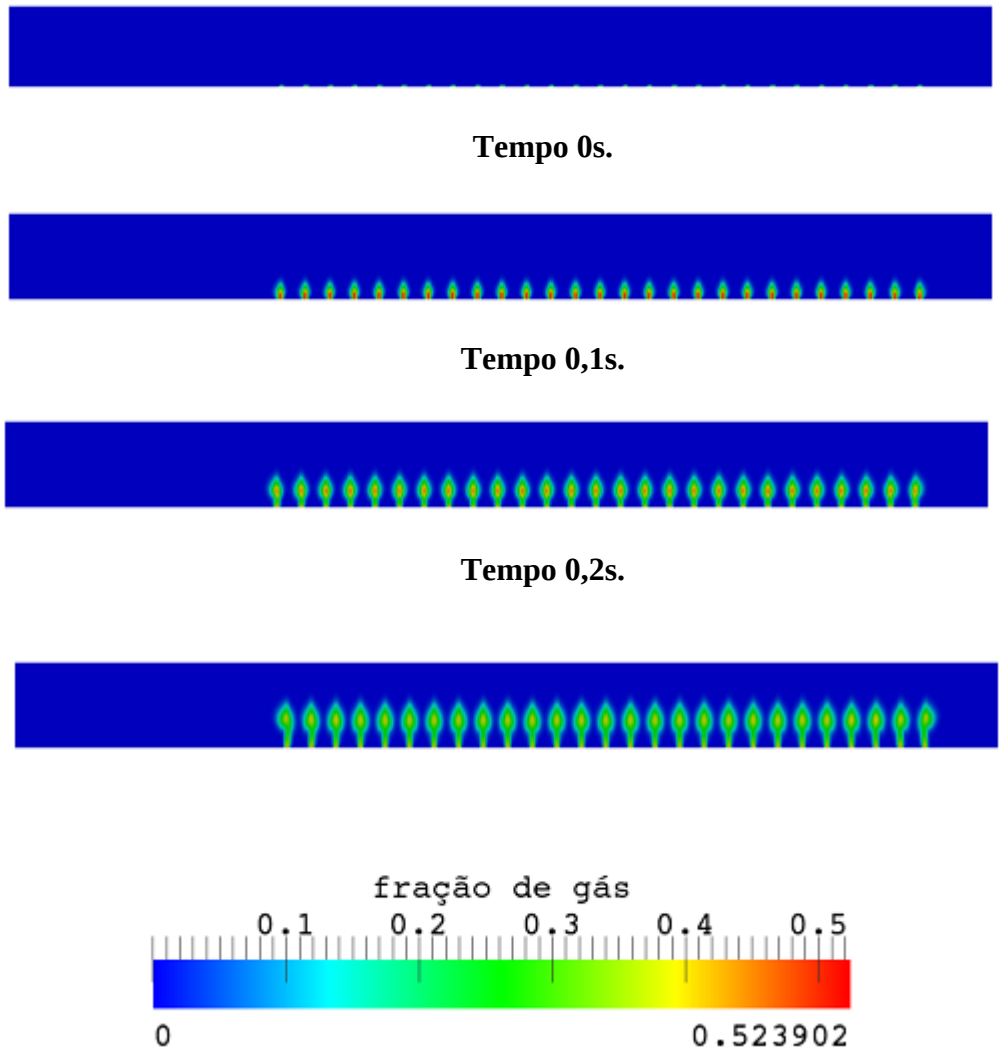
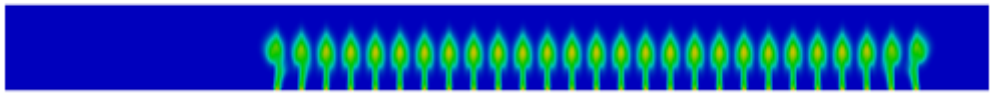
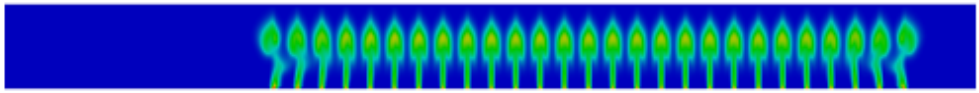


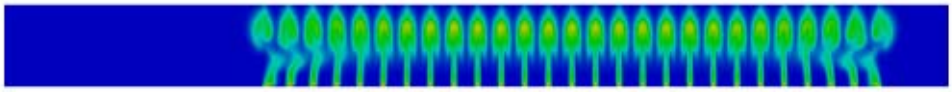
Figura 8.4 - Fração de gás, considerando difusores. Regime laminar. Variando no Tempo de 0 a 0,3s.



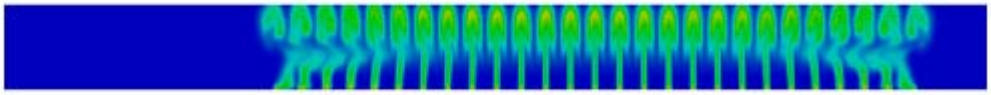
Tempo 0,4s.



Tempo 0,5s.



Tempo 0,6s.



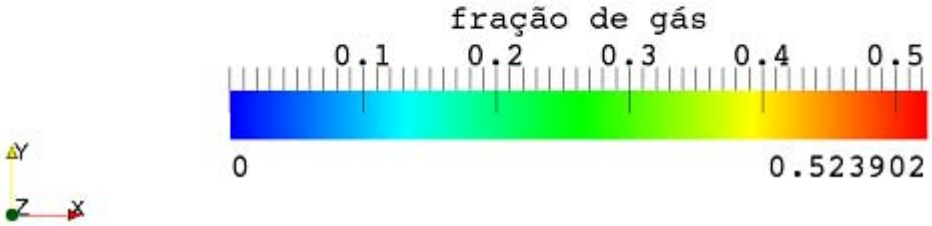
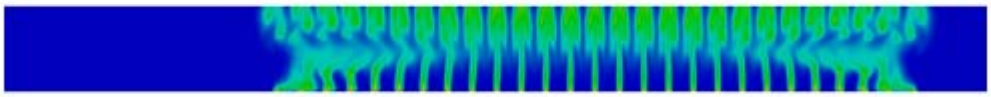
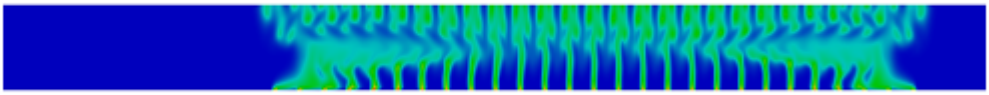


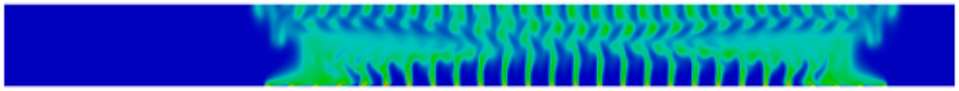
Figura 8.5 - Fração de gás, considerando difusores. Regime laminar. Variando no tempo de 0,4 a 0,7s.



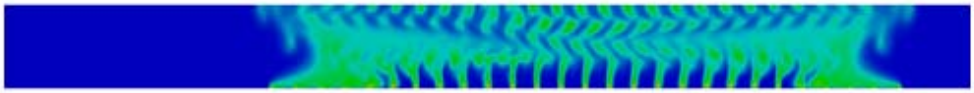
Tempo 0,8s.



Tempo 0,9s.



Tempo 1,0s.



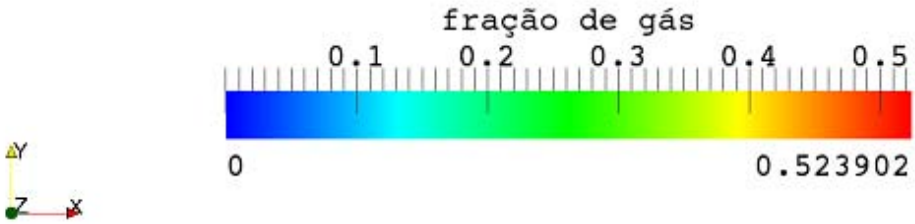
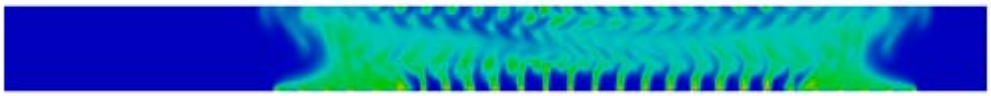
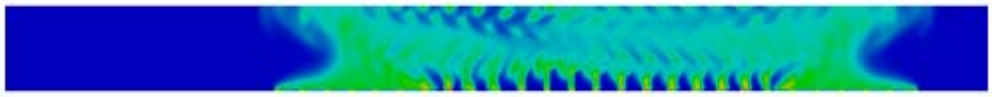


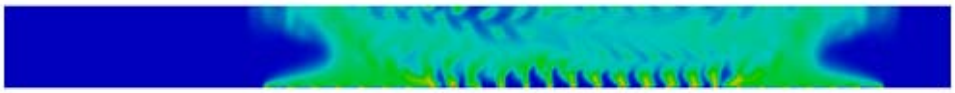
Figura 8.6 - Fração de gás, considerando difusores. Regime laminar. Variando no tempo de 0,8 a 1,1s.



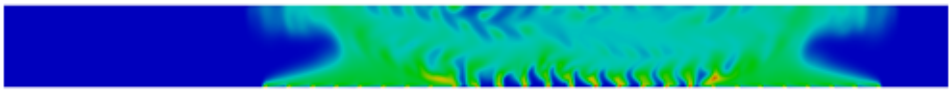
Tempo 1,2s.



Tempo 1,3s.



Tempo 1,4s.



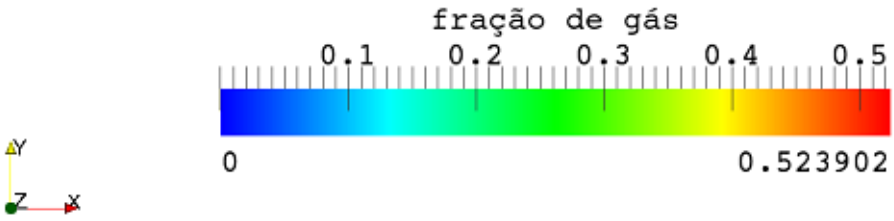
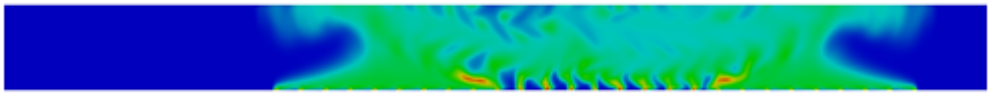
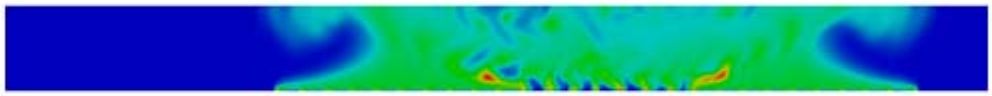


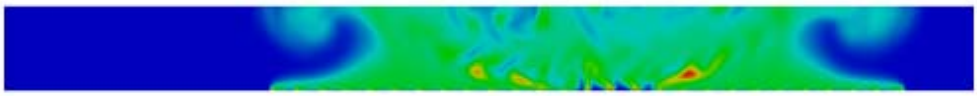
Figura 8.7 - Fração de gás, considerando difusores. Regime laminar. Variando no tempo de 1,2 a 1,5s.



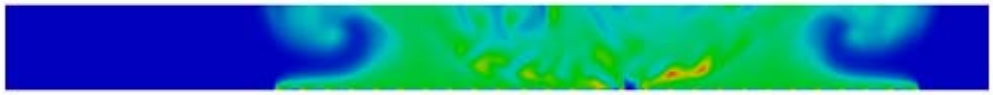
Tempo 1,6s.



Tempo 1,7s.



Tempo 1,8s.



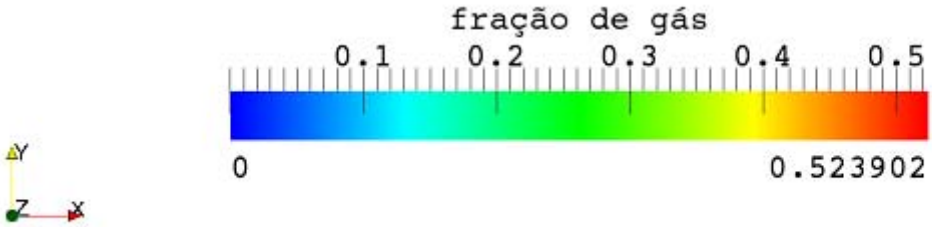


Figura 8.8 - Fração de gás, considerando difusores. Regime laminar. Variando no tempo de 1,6 a 1,9s.

Tempo 2,0s.

Tempo 2,1s.

Tempo 2,2s.

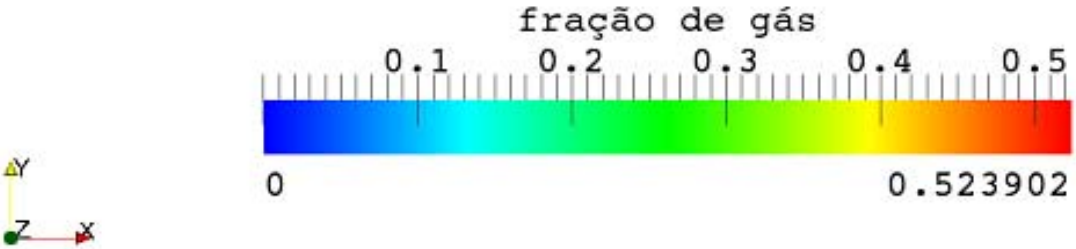


Figura 8.9 - Fração de gás, considerando difusores. Regime laminar. Variando no tempo de 2,0 a 2,3s.

Tempo 2,4s.

Tempo 2,5s.

Tempo 2,6s.

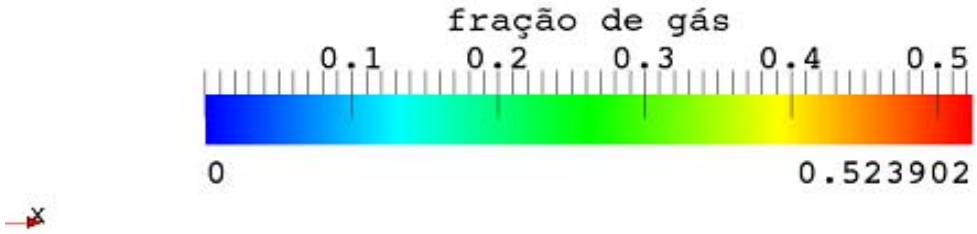
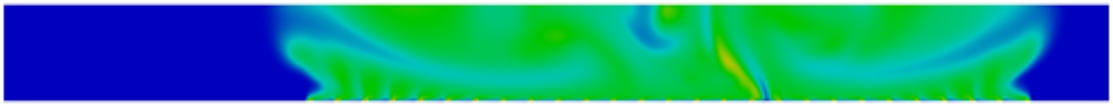
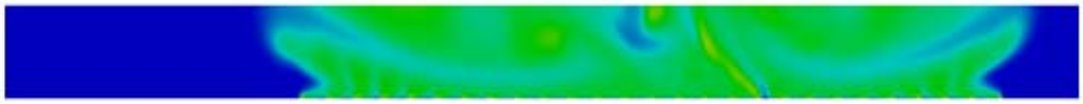


Figura 8.10 - Fração de gás, considerando difusores. Regime laminar. Variando no tempo de 2,4 a 2,7s.

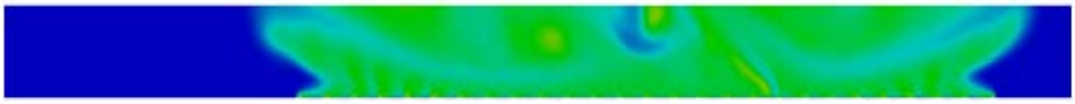
Tempo 2,8s.



Tempo 2,9s.



Tempo 3,0s.



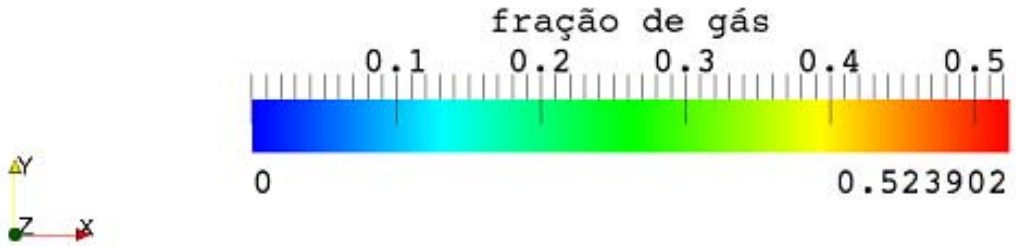
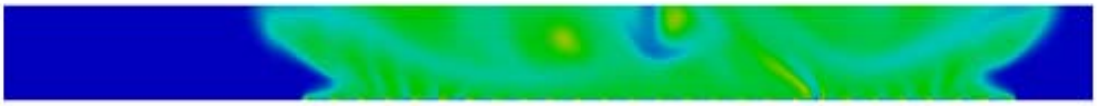
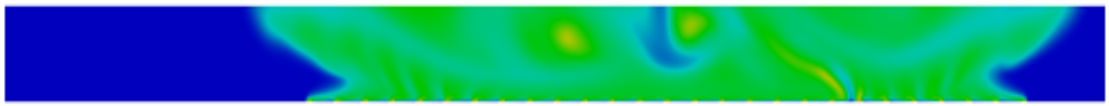


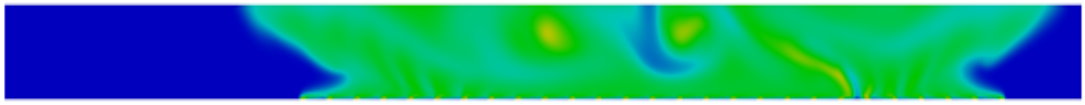
Figura 8.11 - Fração de gás, considerando difusores. Regime laminar. Variando no tempo de 2,8 a 3,1s.



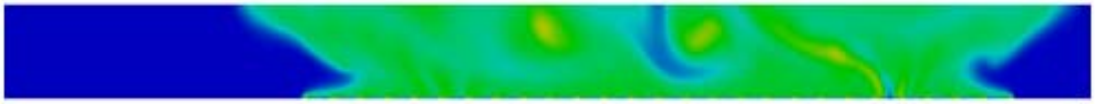
Tempo 3,2s.



Tempo 3,3s.



Tempo 3,4s.



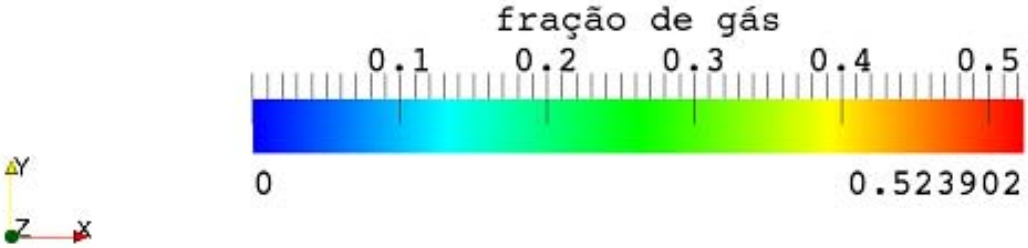


Figura 8.12 - Fração de gás, considerando difusores. Regime laminar. Tempo 3,5s.

Tempo 17s.

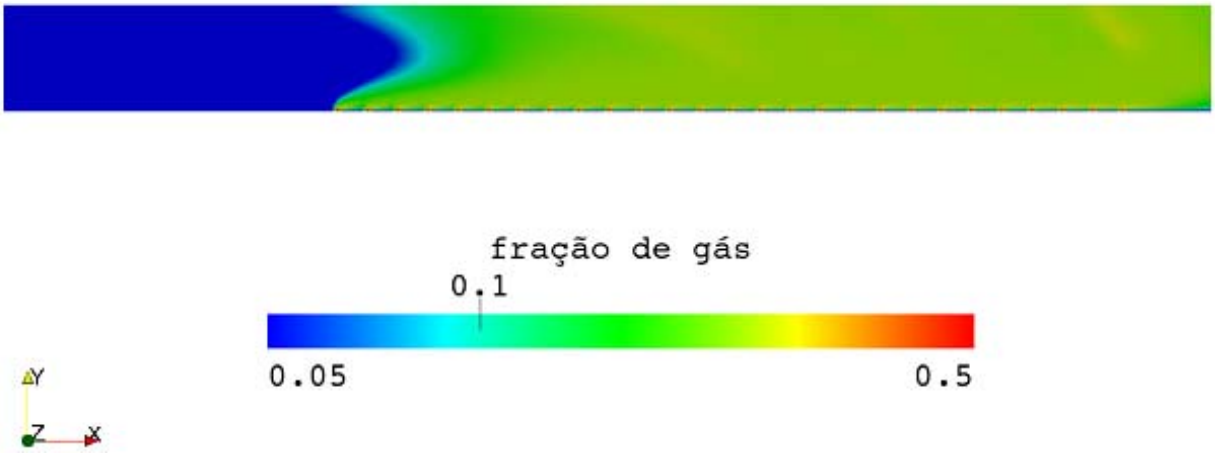


Figura 8.13 - Fração de gás, considerando difusores. Regime laminar. Variando no tempo de 17 a 29s.

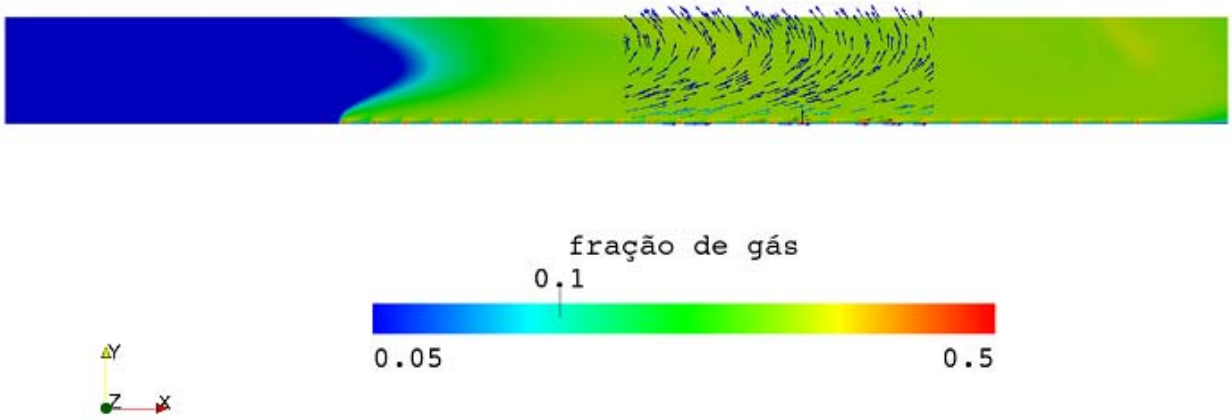
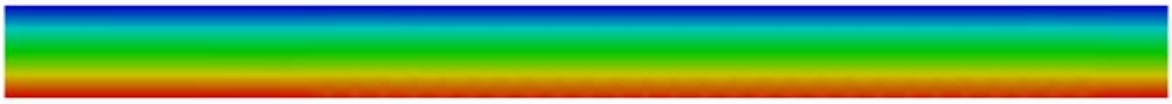
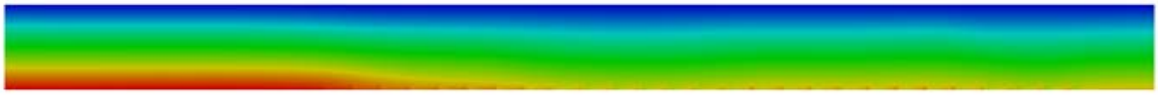


Figura 8.14 - Fração de gás, considerando difusores. Regime laminar. Tempo 29s. Vetores de velocidades da fase gasosa.

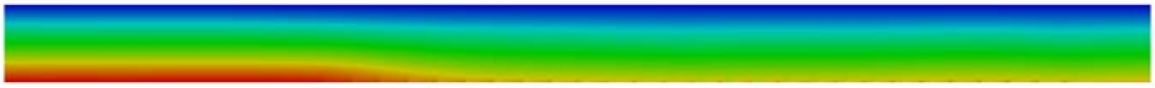
Quanto ao perfil de pressão média do domínio (relativa à atmosférica), observa-se pela Figura 8.15, que a pressão decaiu junto à parede inferior na região de injeção de gás, resultado esperado devido a grande movimentação do líquido nessa vizinhança.



Tempo 0,1s.



Tempo 17s.



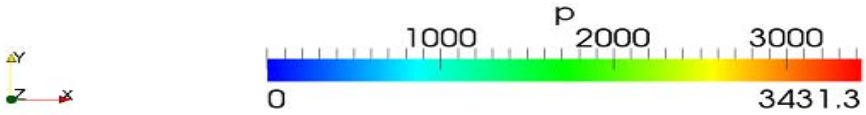


Figura 8.15 - Pressão relativa média em Pa. Regime laminar. Variando no tempo de 0,1 a 29s.

8.4 Resultados para o modelo de turbulência κ-ε

Como dito anteriormente, faz-se uso do pacote CFD não comercial Open-FOAM com o "solver" BubbleFoam, próprio para escoamentos bifásicos líquido-gás com formação de estruturas particuladas (bolhas) na fase mais diluída ou descontínua. O modelo aplicado é baseado no enfoque Euleriano-Euleriano, bifásico, bidimensional e transiente. A discretização temporal é realizada de maneira totalmente implícita de primeira ordem. Os termos convectivos são discretizados utilizando um esquema de interpolação de segunda ordem do tipo diferenças centrais.

Para o acoplamento pressão-velocidade, utiliza-se o esquema PISO. As equações referentes aos termos turbulentos são tratadas por interpolação de primeira ordem ditas ”upwind”, sendo esses efeitos também modelados pelo modelo $k-\varepsilon$.

O tempo real gasto foi de aproximadamente 78 horas para 5 segundos de tempo simulado da dinâmica na coluna com passo de tempo escolhido por análises e testes preliminares de 0,1 s. As variáveis calculadas são feitas desde o início da solução do caso, feitas a cada passo de tempo. E os resultados capturados são referentes às mesmas posições dos dados experimentais.

Utilizou-se uma geometria com iguais dimensões das encontradas no problema padrão, dotada de malha com características estruturadas e volumes discretos totalizando o tamanho da malha refinada com 800 divisões em X e 70 divisões em Y e o número de células da malha foi, então, de 56 mil retângulos iguais (ou hexaedros iguais, se considerar a estrutura tridimensional de célula para volumes finitos).

A seguir apresenta-se, para a simulação do escoamento bifásico água-ar com o modelo de turbulência κ-ε no OpenFOAM, os perfis foram obtidos para o tempo simulado de 5 segundos.

Iniciando por uma apresentação do perfil de pressão médio obtido após 5s simulados (Figura 8.16), observa-se que houve um maior grau de detalhamento de valores nas camadas, comparada ao caso laminar. A pressão relativa decaiu na região próxima aos pontos de aeração, como esperado.

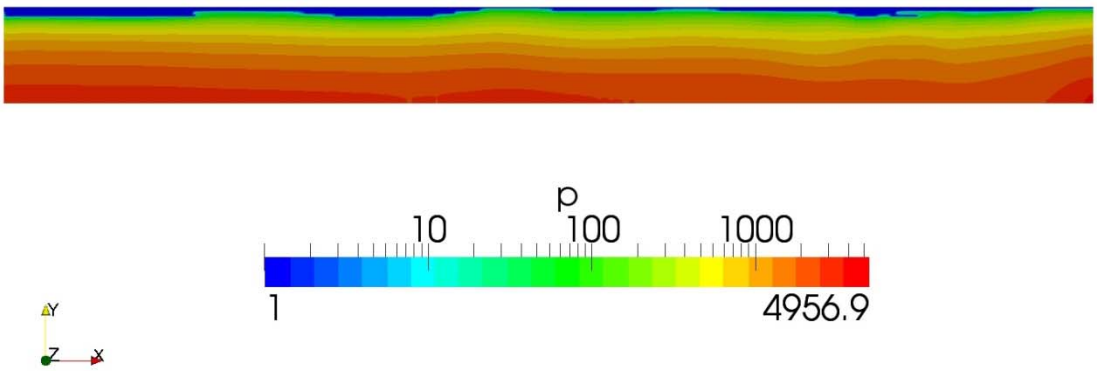
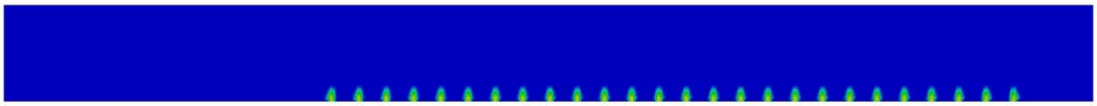


Figura 8.16 - Pressão relativa média em Pa. Regime turbulento. Tempo 5s.

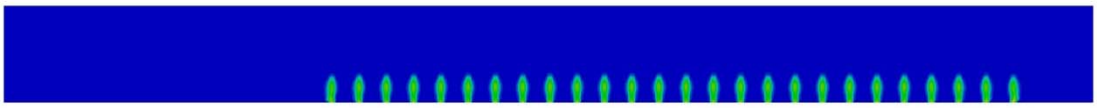
A evolução das plumas de frações volumétricas é mostrada para os tempos simulados de 0,1 s a 0,9 s, 1,0 s, 2,0 s, 3,0 s, 4,0 s e 5,0s, respectivamente nas Figuras 8.18, os gráficos são correspondentes a uma simulação transiente de partida em que se pode observar o desenvolvimento da dispersão da fase gasosa na fase líquida.

A Figura 8.19, momento quando o tempo simulado atinge 5,0 segundos, é possível observarmos o perfil da fase gasosa completamente dispersado sendo intensamente influenciado pelo escoamento da fase líquida. É notável o efeito turbilhonar nesses períodos. Entretanto, o líquido em baixa velocidade arrasta demasiadamente a pluma de bolhas para a saída na direita, o que se observarmos no experimento, isso não se verifica. Essa situação de escoamento talvez ficasse mais apropriada para plumas de bolhas apropriadamente minúsculas, as quais teriam um movimento com uma fraca contribuição do empuxo perante o arrasto turbilhonar do meio líquido. Para o experimento que aqui se deseja representar, com formação de plumas de bolhas, em cujo tamanho o empuxo é mais atuante resultando em plumas mais verticalizadas, esse escoamento simulado não é apropriado, entretanto, mesmo assim, um estudo comparativo foi realizado pelo confronto entre viscosidades turbulentas simuladas e experimentais para se afirmar o grau de validade da modelagem. A energia cinética κ e a taxa de seu consumo ε são apresentadas nas Figuras 8.21 e 8.22, respectivamente, relativas ao tempo simulado de 5s, nas quais observou-se que existem valores médios constantes dispersos em quase toda a região de aeração (tonalidade verde), o que era de se esperar para o local de aeração intensamente turbilhonar, onde a frequência de flutuações é maior e a dissipação é aproximadamente homogênea. Entretanto, energias turbulentas e seus consumos se revelaram maiores na região de entrada do canal, onde a calmaria era esperada ser maior, ou seja, o vigor turbilhonar deveria ser menos intenso acarretando em menores κ e ε. Tal contrariedade pode ser mais bem entendida ao se examinar o perfil de velocidades vetorizadas do líquido na Figura 8.23. Percebe-se, então, que o líquido ganhando um movimento ascendente, não podendo avançar além da superfície livre, é forçado a escoar para as laterais em um movimento cíclico, como um jorro. Na região de entrada, o líquido acelerado choca-se com a porção que avança à montante criando uma região de grande disputa pela conservação de energia e massa, ocasionando no maior grau de conversões em energias turbilhonares (efeitos semelhantes ao do ressalto hidráulico). Na região de jusante a direita, observa-se que esse fato não é presente, simplesmente, porque o líquido carrega, advectivamente, a parte reciclada para a saída com todo o seu potencial turbilhonar não convertido, uma vez que não há resistência ao escoamento (seção transversal de saída é totalmente aberta) que seja suficiente para intensificar o seu retorno ao domínio.

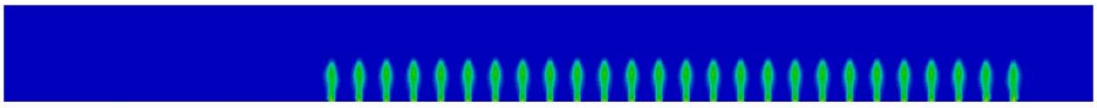
Tempo 0s.



Tempo 0,1s.



Tempo 0,2s



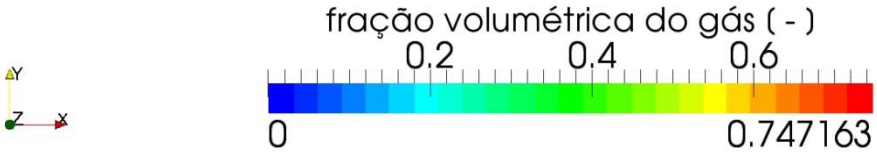
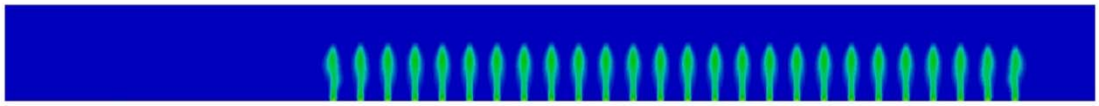
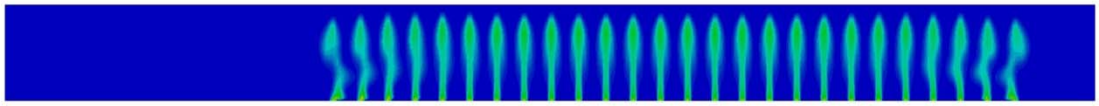


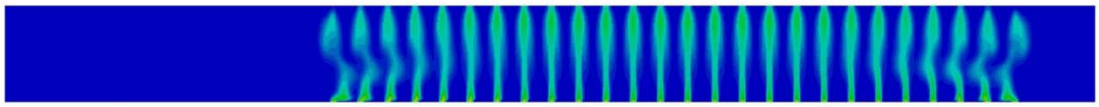
Figura 8.17 - Fração de gás, considerando difusores. Regime turbulento. Variando no tempo de 0 a 0,3s.



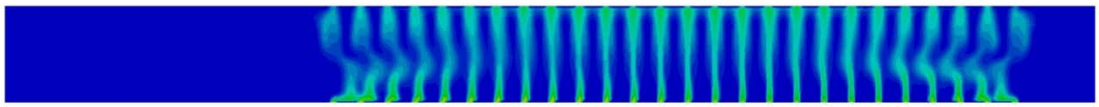
Tempo 0,5s.



Tempo 0,6s.



Tempo 0,7s.



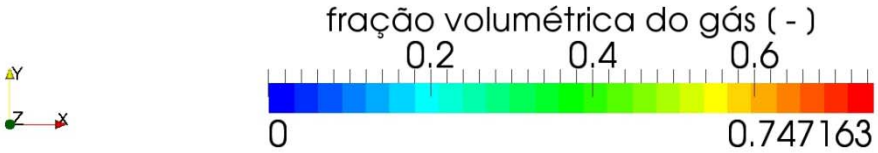
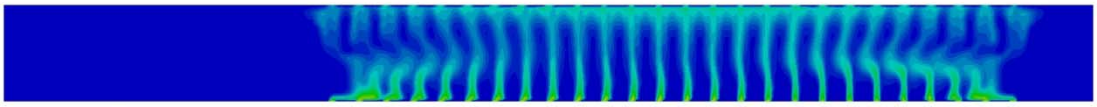
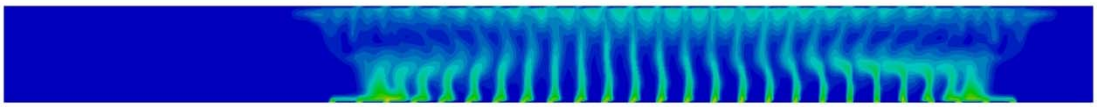


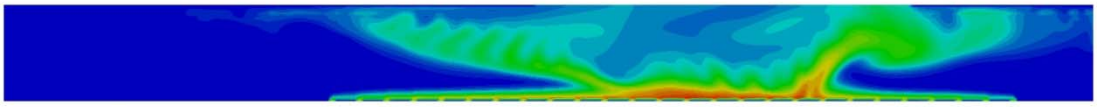
Figura 8.18 - Fração de gás, considerando difusores. Regime turbulento. Variando no tempo de 0,5 a 0,8s.



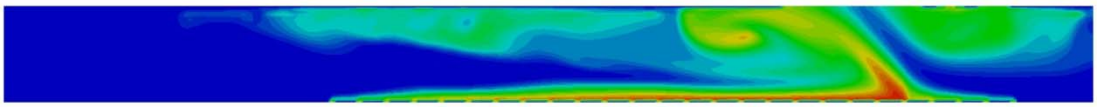
Tempo 0,9s.



Tempo 1,0s.



Tempo 2,0s.



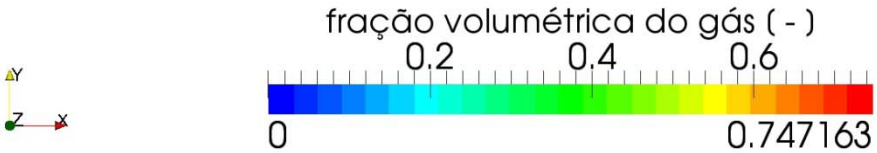
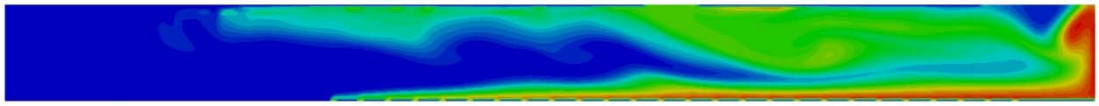
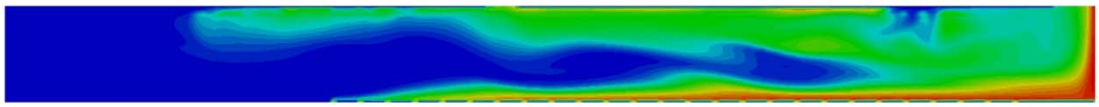


Figura 8.19 - Fração de gás, considerando difusores. Regime turbulento. Variando no tempo de 0,9 a 3,0s.



Tempo 4,0s.



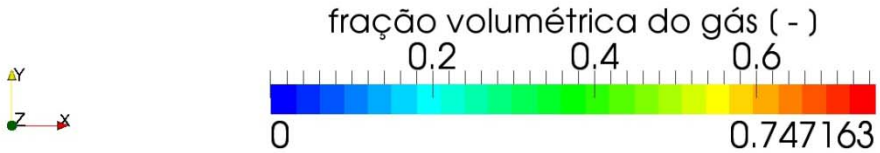
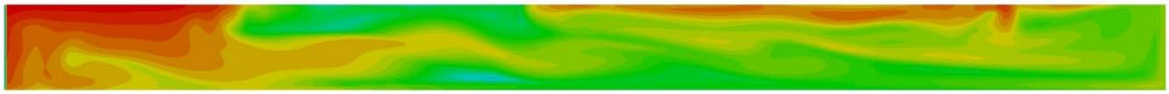


Figura 8.20 - Fração de gás, considerando difusores. Regime turbulento. Nos tempos 4,0 e 5,0s.



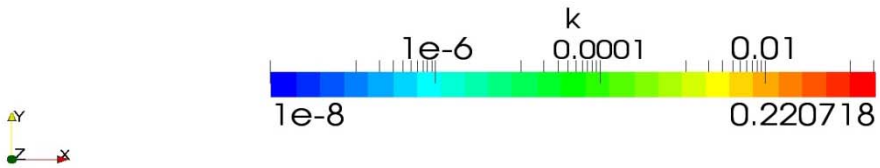


Figura 8.21 - Perfil da energia cinética turbulenta κ (em J). Tempo 5,0s.

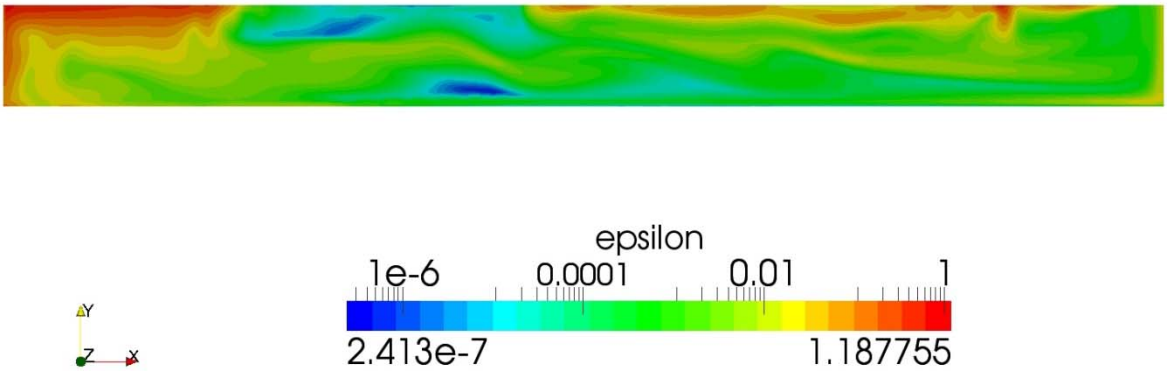


Figura 8.22 - Perfil da dissipação de energia cinética turbulenta ε (em J/s). Tempo 5,0s.

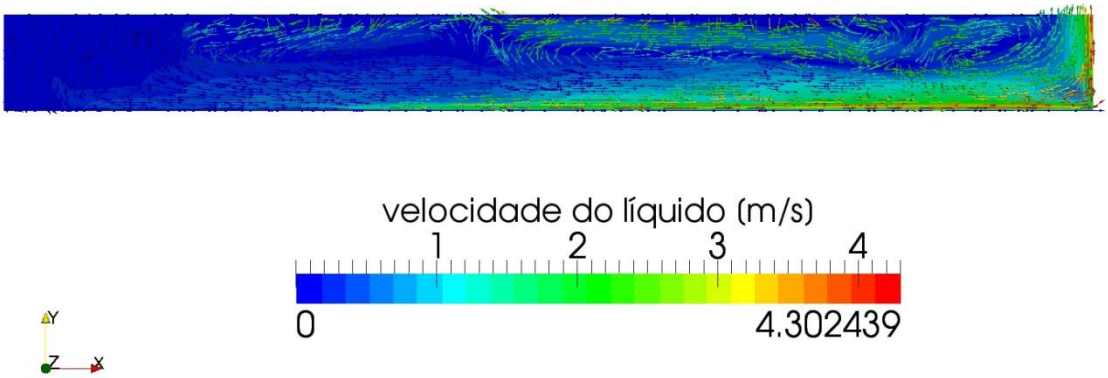


Figura 8.23 - Perfil da velocidade do líquido (vetores e contornos, em m/s). Tempo 5,0s.

No código computacional BubbleFoam a viscosidade turbulenta é disponibilizada como um único escalar variante ao longo da direção advectiva da fase líquida, ao passo que no experimento (Oliveira, 2008) com o qual a simulação foi comparada neste trabalho a viscosidade turbulenta foi considerada como propriedades componentes que acompanham a orientação do tensor tensão de Reynolds, ou seja, possui 4 magnitudes no domínio bidimensional.

Para ter um parâmetro comum de comparação, então realizou-se a média de 4 viscosidades turbulentas experimentais para cada ponto de medição no comprimento X (comprimento longitudinal do canal no plano central que corta os pontos aeradores). Esses valores médios foram confrontados com os simulados, coincidentes com os pontos de medição em X, a cada 10cm aproximadamente, somente na região de borbulhamento

enquadrada no experimento para a técnica de PIV, ou seja um campo de 70cm de comprimento longitudinal para o plano central (Figura 8.24).

Nota-se pelos dados experimentais que os perfis (Figura 8.24), tanto experimental como simulado, apresentam as viscosidades médias maiores à da montante do canal pela esquerda, vindo a decair, conforme já delineado anteriormente. Observa-se, então, que qualitativamente essa tendência foi representada na comparação entre ambos. Porém, quantitativamente, os resultados numéricos das viscosidades turbulentas dinâmicas se revelaram maiores que experimental em cerca de 90%, entretanto, é mais viável dizer que todos os valores ficaram na mesma ordem de grandeza de 10^{-1}, o que torna o modelo válido do ponto de vista físico, por essa abordagem.

Os erros residuais encontrados foram:

Alpha (fração volumétrica) = 3.98172e-11

p (pressão) = 1.52177e-06

k = 0.000174511

epsilon = 0.00012115

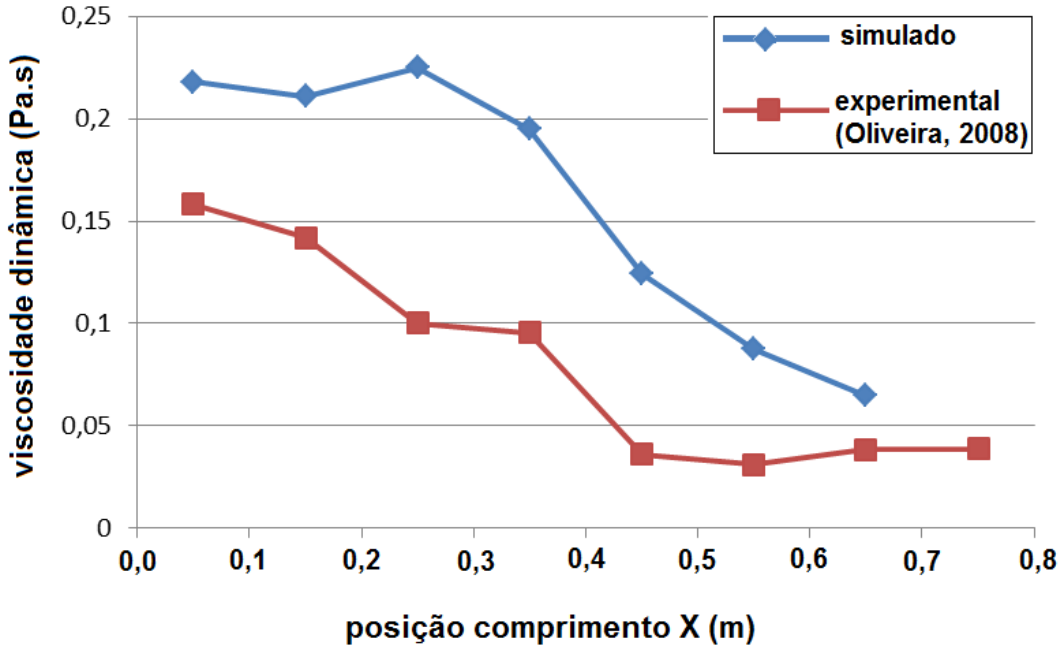


Figura 8.24 - Comparação entre os perfis de viscosidades médias para a região central de aeração Tempo simulado de 5s.

9 CONCLUSÃO E SUGESTÕES PARA TRABALHOS FUTUROS

Entre os efeitos observados na utilização do código BubbleFoam para os dados numéricos destacam-se: uma maior periodicidade da pluma de bolhas, perfis de velocidade e fração volumétrica da fase dispersa mais acentuados, aumentando a fase líquida e gasosa e diminuindo a velocidade lateral;

O uso da força de ascensão de bolhas promove um parâmetro ajustável dependendo da situação experimental, característica indesejável no que se refere às premissas da modelagem generalizada desejável;

Em relação à força de massa virtual, o seu uso afeta as características instantâneas de ambas as fases; o uso da força de massa virtual afeta positivamente os resultados obtidos em porções próximas ao distribuidor de gás, e afeta negativamente os resultados na parte superior do canal.

A comparação dos resultados experimentais (Oliveira, 2008) com o numérico do modelo κ-ε apresentou, com respeito às viscosidades turbulentas, uma tendência qualitativa concordante e, quantitativamente, houve um desvio que não altera significativamente a validação do modelo. Entretanto, a influência advectiva do líquido na deformação da pluma se mostrou muito intensa, o que não se verifica na realidade do experimento a que se faz referência. Talvez, isso seria válido para plumas de bolhas muito finas.

O software OpenFOAM apresentou-se como uma ferramenta eficiente nas simulações realizadas, porém com limitações quando utilizado o modelo de turbulência k-ε, pois quando o modelo de turbulência foi inserido a simulação com todos os difusores divergia após 1 (um) segundo e depois de alguns testes com varias versões do OpenFoam e refinamento da malha e diminuição da quantidade de difusores foi possível observar que: a simulação que até então convergia 5 (cinco) segundos com apenas 7 difusores nas versões do OpenFoam 1.7.0, 1.7.1 e 2.1.0, na versão do OpenFoam 2.1.1 e com um refinamento significativo da malha obteve-se uma simulação que convergiu em 5 (cinco) segundos com todos os difusores e a partir desse passo de tempo divergia com valores de velocidade muito elevados. Depois de vários testes pode-se concluir que é mesmo uma limitação no código computacional BubbleFoam do software OpenFOAM,

9.1.1 Sugestões

Procurando o aperfeiçoamento desse estudo no escoamento gás-líquido, sugere-se os seguintes temas para o prosseguimento:

- avaliação dos modelos de quebra, coalescência e deformação de bolhas;
- incorporação de modelos de quebra de superfície;
- avaliar os efeitos da adição de modelos que contemplem os efeitos turbulentos dentro da fase diluída;
- estudo dos diferentes modelos de turbulência para fase líquida como o κ-ω e simulação de grandes escalas de turbilhões (LES), por exemplo.

10 REFERÊNCIAS BIBLIOGRÁFICAS

BECH, K. (2005) Dynamic simulation of a 2Dbubble column, Chemical Engineering Science, V.60, 5294 – 5304.

BERNARD R. S.; MAIER, R.S.; FALVEY, H. T. (2000) A simple computational model for bubble plumes, Applied Mathematical Modelling, v.24. 215-233.

BORTOLI, Á.L. (2000) Introdução à Dinâmica de Fluidos Computacional**.** Porto Alegre: Ed. Universidade/UFRGS.

BRANDI, A. C. (2005) Estratégias Upwind e Modelagem κ-ε para Simulação Numérica de Escoamentos com Superfícies Livres em Altos Números de Reynolds, São Carlos, dissertação (mestrado), Instituto de Ciências Matemáticas e de Computação. Universidade d eSão Paulo.75p.

CARNAHAN, B.; LUTHER, H. A.; WILKES, J. O. (1969) Applied Numerical Methods. New York, London, Sydney, Toronto, John Wiley & Sons, Inc. 604p

CROWE, C.T.; TROUTT, T.R.; CHUNG, J. N. (1996) Numerical Models for Two-phase Turbulence. Ann. Rev. Fluid Mech.

COCKX, A; DO-QUANG, Z.; AUDIC, J. M.; LINÉ, A.; ROUSTAN, M. (2001). Global and Local Mass Transfer Coefficients in Waste Water Treatment Process by Computational Fluid Dynamics, Chem. Eng. Proc., vol. 40, p.187-194.

CORRÊA, N. (2003) Análise da dispersão turbulenta em aeração de corpos hídricos usando técnica PIV. Proposta de pesquisa Fapesp, 2003. 20p.

DEL FRARI, R. V.; PEDROSO, C. A. (2009) Malhas Computacionais para Simulação Numérica de Escoamentos de Fluidos entre Cilindros com Excentricidade**.** X Encontro Gaúcho de Educação Matemática 02 a 05 de junho de 2009, Ijuí/RS.

DEVAHL, g.; MALLISON, D. D. (1976) An Evaluation of Upwind and Central Difference Aproximations by a Study of Recirculating Flow. Comp. fluids, 4, 29-32.

DHOTRE M. T.; SMITH B. L. (2007) CFD simulation of large-scale bubble plumes: Comparisons against experiments, Chemical Engineering Science, v.62, 6615 – 6630.

FERREIRA V. G. (2001), Análise e Implementação de Esquemas de Convecção e Modelos de Turbulência para Simulação de Escoamentos Incompressíveis Envolvendo Superfícies Livres (tese) – USP – São Carlos;

FONTOURA, D. V. R. (2013) Desenvolvimento de um Modelo Tridimensional Bifásico para a Predição da Formação de Coque em Tubos de Fornos de Petróleo. Tese- Faculdade de Engenharia Química, Universidade Estadual de Campinas.

FORTUNA, A.O. (2000) Técnicas Computacionais para Dinâmica dos Fluidos – Conceitos Básicos e Aplicações. São Carlos, EDUSP. 426p.

FOX, R.W.; McDONALD, A.T. (1995) Introdução à Mecânica dos Fluidos**.** 4 ed. Rio de Janeiro: Ed. LTC.

JASAK, H. (1996) Error analysis and estimation for the finite volume method with applications to fluid-flows, Ph.D. Thesis, Imperial College of Science, Technology and Medicine, Londres, Reino Unido.

LAUNDER, B.E.; SPALDING, D.B. (1974) The numerical Computation of Turbulent Flows. Int. J. Numer. Meths. Fluids, 15, 127-146.

LESIEUR, M. (1997) Turbulence in Fluids. Third Revised and Enlarged Edition, Boston. 515p.

MALISKA, C.R., (1995), Transferência de Calor e Mecânica dos Fluidos Computacional, LTC, Rio de Janeiro.

MALISKA, C. R., (2004) Transferência de Calor e Mecânica dos Fluidos Computacional 2nd ed. LTC: Rio de Janeiro.

MARRUCI,G.;. APUZZO, G.; ASTARITA, G. (1970) Motion of Liquid Drops in Non-newtonian Systems. AIChE J., 16-538.

MATHIEU, J.; SCOTT, J. (2000) An Introduction to Turbulent Flow, New York. 374p.

POPE, S. B. (2003) Turbulent Flows, New York. 771p.

OLIVEIRA, A. B. F. (2008) Análise da Dispersão Turbulenta em Aeração de Corpos Hídricos Usando a Técnica PIV (Velocimetria por Imagens de Partículas) - Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo.

OLIVEIRA, J. (1999) Desenvolvimento de um Sistema de Simulação de Escoamentos de Fluidos com Superfícies Livres Bidimensionais – Dissertação-Universidade de São Paulo.

OLIVEIRA, J. P.; ISSA,R. I.(2003) Numerical aspects of an algorithm for Eulerian simulation of two-phase flows. International Journal for Numerical Methods in Fluids, 43:1177 – 1198.

PATANKAR, S. V. (1980) Numerical Heat Transfer and Fluid Flow . Taylor & Francis Group: Nova York.

PATEL, V. C.; WOLFGANG, R.; SCHEUERER, J. (1985). Turbulence Models for Near-Wall and Low-Reynolds-Number Flows: a review. AIAA, J., 23, 1308-1319.

PINTO, J. C., LAGE, P. L. C. (2001) Métodos Numéricos em Problemas de Engenharia Química. E-papers: Brasil, RJ.

RABHA, S. S.; BUWA V.V. (2010) Volume-of-fluid (VOF) simulations of rise of single/multiple bubbles in sheared liquids, Chemical Engineering Science, v. 65, 527 – 537.

RAITHBY G. D. (1976) A Critical Evaluation of Upstrean Differencing Aplied to Problems Involving Fluid Flow. Comp. Meths. Appl. Mech. Engrg, 9, 75-86.

RICE, R. G. & DO, D. D.(1995). Applied Mathematics and Modeling for Chemical Engineers. New York, Chichester, Brisbane, Toronto, Singapore, John Wiley & Sons. 706p.

RODI, W. (2000) Turbulence Mpdels and Their Aplication in Hydrulics. A state-of-the-art review. 104p.

RUSCHE, H. (2002) Computational Fluid Dynamics of Dispersed Two-Phase Flow at Righ Phase Fractions. Ph.D.. Thesis, Imperial College of Science, Technology and Medicine, Londres, Reino Unido.

SAAD, Y. (2003) Iterative Methods for Sparse Linear Systems . 2nd ed. SIAM, 2003.

SILVA (2008) Desenvolvimento de Metodologias para Simulação de Escoamentos Polidispersos usando Código Livre. Universidade Federal do Rio de Janeiro, COPPE, tese, 262p.

SILVEIRA-NETO A. Fundamentos da Turbulência nos Fluidos, Universidade Federal de Uberlândia, Faculdade de Engenharia Mecânica- Campus Santa Mônica, 52p. (Apostila).

SILVEIRA-NETO A. Turbulência nos Fluidos Aplicada (2003), Universidade Federal de Uberlândia, Faculdade de Engenharia Mecânica – Campus Santa Mônica,152p.(Apostila).

SCHULZ, E. H. (1985) Investigação do Mecanismo de Reoxigenação da Água em Escoamento e sua Correlação com o Nível de Turbulência junto à Superfície. São Carlos, SP, EESC-USP. 299p. (Dissertação).

SCHULZ, E. H. (1989) Investigação do Mecanismo de Reoxigenação da Água em Escoamento e sua Correlação com o Nível de Turbulência junto à Superfície. São Carlos, SP, EESC-USP. 348p. (Tese).

SMITH, B. L. (1998) On the Modelling of Bubbles Plumes in a Liquid Pool, App. Mathematical Modelling, vol. 22. p.773-797.

SOCCOL- JUNIOR, R. (2011) Modelagem e Simulação do Escoamento Gás-Líquido em Leitos Borbulhantes nos Reatores de Colunas de Bolha. Universidade Regional de Blumenau. Dissertação.190p.

VANINI, L.; TEIXEIRA, P. R. F. (2006) Adaptação de um código computacional de escoamentos incompressíveis 3-D para problemas com superfície livre. In: II Seminário e Workshop em Engenharia Oceânica, Rio Grande. Anais do II Seminário e Workshop em Engenharia Oceânica. Rio Grande: FURG, . v. 1. p. 12-12.

VERSTEEG, H. K., MALALASEKERA, W. (1995) An Introduction to Computational Fluid Dynamics - The finite volume method . Longman Scientific and Technical: Reino Unido.

WELLER, H. G. (2002) Derivastion, Modelling and solution of the Conditionally Averages Two-Phase Flow Equations. Technical Report TR/HGW/02, Nabla Ltd.

WHITE, F.M. (1991), Viscous Fluid Flow, McGraw-Hill, New York.

ZBORAY, R.; CACHARD F.(2005) Simulating Large-Scale Bubble Plumes using Various closure and two-phase Turbulence Models, Nuclear Engineering and Design, v.235, 867–884.

APENDICE A – Código computacional modificado.

SOLVER

BubbleFoam.C

```
#include "fvCFD.H"
#include "nearWallDist.H"
#include "wallFvPatch.H"
#include "Switch.H"

#include "pimpleControl.H"

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

int main(int argc, char *argv[])
{
    #include "setRootCase.H"

    #include "createTime.H"
    #include "createMesh.H"
    #include "readGravitationalAcceleration.H"
    #include "createFields.H"
    #include "initContinuityErrs.H"
    #include "readTimeControls.H"
    #include "CourantNo.H"
    #include "setInitialDeltaT.H"

    pimpleControl pimple(mesh);
```

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

Info<< "\nStarting time loop\n" << endl;

while (runTime.run())
{
    #include "readBubbleFoamControls.H"
    #include "CourantNo.H"
    #include "setDeltaT.H"

    runTime++;
    Info<< "Time = " << runTime.timeName() << nl << endl;

    // --- Pressure-velocity PIMPLE corrector loop
    while (pimple.loop())
    {
        #include "alphaEqn.H"
        #include "liftDragCoeffs.H"
        #include "UEqns.H"

        // --- Pressure corrector loop
        while (pimple.correct())
        {
            #include "pEqn.H"

            if (correctAlpha && !pimple.finalIter())
            {
                #include "alphaEqn.H"
```

```
            }
        }

        #include "DDtU.H"

        if (pimple.turbCorr())
        {
            #include "kEpsilon.H"
        }
    }

    #include "write.H"

    Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
        << "  ClockTime = " << runTime.elapsedClockTime() << " s"
        << nl << endl;
  }


  Info<< "End\n" << endl;

  return 0;
}
```

AlphaEqn.H

```
word scheme("div(phi,alpha)");

  surfaceScalarField phir(phia - phib);

  Info<< "Max Ur Courant Number = "
```

```
    << (
        max
        (
            mesh.surfaceInterpolation::deltaCoeffs()*mag(phir)
            /mesh.magSf()
        )*runTime.deltaT()
    ).value()
    << endl;

for (int acorr=0; acorr<nAlphaCorr; acorr++)
{
    fvScalarMatrix alphaEqn
    (
        fvm::ddt(alpha)
      + fvm::div(phi, alpha, scheme)
      + fvm::div(-fvc::flux(-phir, beta, scheme), alpha, scheme)
    );
    alphaEqn.relax();
    alphaEqn.solve();

    /*
    fvScalarMatrix betaEqn
    (
        fvm::ddt(beta)
      + fvm::div(phi, beta, scheme)
      + fvm::div(-fvc::flux(phir, scalar(1) - beta, scheme), beta, scheme)
```

```
        );
        betaEqn.relax();
        betaEqn.solve();

        alpha =
            0.5
           *(
                scalar(1)
              + sqr(scalar(1) - beta)
              - sqr(scalar(1) - alpha)
            );
        */

        beta = scalar(1) - alpha;
    }

    Info<< "Dispersed phase volume fraction = "
        << alpha.weightedAverage(mesh.V()).value()
        << "  Min(alpha) = " << min(alpha).value()
        << "  Max(alpha) = " << max(alpha).value()
        << endl;
}

rho = alpha*rhoa + beta*rhob;
createFields.H
Info<< "Reading field alpha\n" << endl;
    volScalarField alpha
```

```
(
    IOobject
    (
        "alpha",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

volScalarField beta
(
    IOobject
    (
        "beta",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::NO_WRITE
    ),
    scalar(1) - alpha
    //,alpha.boundaryField().types()
);
```

```
Info<< "Reading field p\n" << endl;
volScalarField p
(
    IOobject
    (
        "p",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

Info<< "Reading field Ua\n" << endl;
volVectorField Ua
(
    IOobject
    (
        "Ua",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
```

```
);

Info<< "Reading field Ub\n" << endl;
volVectorField Ub
(
    IOobject
    (
        "Ub",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

volVectorField U
(
    IOobject
    (
        "U",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
```

```
    alpha*Ua + beta*Ub
);

Info<< "Reading transportProperties\n" << endl;

IOdictionary transportProperties
(
    IOobject
    (
        "transportProperties",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);

dimensionedScalar rhoa
(
    transportProperties.lookup("rhoa")
);

dimensionedScalar rhob
(
    transportProperties.lookup("rhob")
);
```

```
dimensionedScalar nua
(
    transportProperties.lookup("nua")
);

dimensionedScalar nub
(
    transportProperties.lookup("nub")
);

dimensionedScalar da
(
    transportProperties.lookup("da")
);

dimensionedScalar db
(
    transportProperties.lookup("db")
);

dimensionedScalar Cvm
(
    transportProperties.lookup("Cvm")
);

dimensionedScalar Cl
(
    transportProperties.lookup("Cl")
```

```
);

dimensionedScalar Ct
(
    transportProperties.lookup("Ct")
);

#include "createPhia.H"
#include "createPhib.H"

surfaceScalarField phi
(
    IOobject
    (
        "phi",
        runTime.timeName(),
        mesh
    ),
    fvc::interpolate(alpha)*phia
  + fvc::interpolate(beta)*phib
);

volScalarField rho
(
    IOobject
    (
        "rho",
```

```
        runTime.timeName(),
        mesh
    ),
    alpha*rhoa + beta*rhob
);

#include "createRASTurbulence.H"

Info<< "Calculating field DDtUa and DDtUb\n" << endl;

volVectorField DDtUa
(
    fvc::ddt(Ua)
  + fvc::div(phia, Ua)
  - fvc::div(phia)*Ua
);

volVectorField DDtUb
(
    fvc::ddt(Ub)
  + fvc::div(phib, Ub)
  - fvc::div(phib)*Ub
);

Info<< "Calculating field g.h\n" << endl;
volScalarField gh("gh", g & mesh.C());

label pRefCell = 0;
```

```
    scalar pRefValue = 0.0;
    setRefCell(p, mesh.solutionDict().subDict("PIMPLE"), pRefCell, pRefValue);
```

createPhia.H

```
IOobject phiaHeader
    (
        "phia",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ
    );

    autoPtr<surfaceScalarField> phiaPtr(NULL);

    if (phiaHeader.headerOk())
    {
        Info<< "Reading face flux field phia\n" << endl;

        phiaPtr.reset
        (
            new surfaceScalarField
            (
                IOobject
                (
                    "phia",
                    runTime.timeName(),
                    mesh,
```

```
                IOobject::MUST_READ,
                IOobject::AUTO_WRITE
            ),
            mesh
        )
    );
}
else
{
    Info<< "Calculating face flux field phia\n" << endl;

    wordList phiTypes
    (
        Ua.boundaryField().size(),
        calculatedFvPatchScalarField::typeName
    );

    forAll(Ua.boundaryField(), i)
    {
        if (isA<fixedValueFvPatchVectorField>(Ua.boundaryField()[i]))
        {
            phiTypes[i] = fixedValueFvPatchScalarField::typeName;
        }
    }

    phiaPtr.reset
    (
```

```
        new surfaceScalarField
        (
            IOobject
            (
                "phia",
                runTime.timeName(),
                mesh,
                IOobject::NO_READ,
                IOobject::AUTO_WRITE
            ),
            fvc::interpolate(Ua) & mesh.Sf(),
            phiTypes
        )
    );
}

surfaceScalarField& phia = phiaPtr();
```

createPhib.H

```
IOobjephia.Hct phibHeader
    (
        "phib",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ
    );
```

```
autoPtr<surfaceScalarField> phibPtr(NULL);

if (phibHeader.headerOk())
{
    Info<< "Reading face flux field phib\n" << endl;

    phibPtr.reset
    (
        new surfaceScalarField
        (
            IOobject
            (
                "phib",
                runTime.timeName(),
                mesh,
                IOobject::MUST_READ,
                IOobject::AUTO_WRITE
            ),
            mesh
        )
    );
}
else
{
    Info<< "Calculating face flux field phib\n" << endl;
```

```
wordList phiTypes
(
    Ub.boundaryField().size(),
    calculatedFvPatchScalarField::typeName
);

forAll(Ub.boundaryField(), i)
{
    if (isA<fixedValueFvPatchVectorField>(Ub.boundaryField()[i]))
    {
        phiTypes[i] = fixedValueFvPatchScalarField::typeName;
    }
}

phibPtr.reset
(
    new surfaceScalarField
    (
        IOobject
        (
            "phib",
            runTime.timeName(),
            mesh,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
```

```
            fvc::interpolate(Ub) & mesh.Sf(),
            phiTypes
        )
    );
}

surfaceScalarField& phib = phibPtr();
```

createRASTurbulence.H

```
IOdictionary RASProperties
(
    IOobject
    (
        "RASProperties",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);

Switch turbulence
(
    RASProperties.lookup("turbulence")
);

dictionary kEpsilonDict
```

```
(
    RASProperties.subDictPtr("kEpsilonCoeffs")
);

dimensionedScalar Cmu
(
    dimensionedScalar::lookupOrAddToDict
    (
        "Cmu",
        kEpsilonDict,
        0.09
    )
);

dimensionedScalar C1
(
    dimensionedScalar::lookupOrAddToDict
    (
        "C1",
        kEpsilonDict,
        1.44
    )
);

dimensionedScalar C2
(
    dimensionedScalar::lookupOrAddToDict
```

```
    (
        "C2",
        kEpsilonDict,
        1.92
    )
);

dimensionedScalar alphak
(
    dimensionedScalar::lookupOrAddToDict
    (
        "alphak",
        kEpsilonDict,
        1.0
    )
);

dimensionedScalar alphaEps
(
    dimensionedScalar::lookupOrAddToDict
    (
        "alphaEps",
        kEpsilonDict,
        0.76923
    )
);
```

```
dictionary wallFunctionDict
(
    RASProperties.subDictPtr("wallFunctionCoeffs")
);

dimensionedScalar kappa
(
    dimensionedScalar::lookupOrAddToDict
    (
        "kappa",
        wallFunctionDict,
        0.41
    )
);

dimensionedScalar E
(
    dimensionedScalar::lookupOrAddToDict
    (
        "E",
        wallFunctionDict,
        9.8
    )
);

if (RASProperties.lookupOrDefault("printCoeffs", false))
```

```
{
    Info<< "kEpsilonCoeffs" << kEpsilonDict << nl
        << "wallFunctionCoeffs" << wallFunctionDict << endl;
}

nearWallDist y(mesh);

Info<< "Reading field k\n" << endl;
volScalarField k
(
    IOobject
    (
        "k",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

Info<< "Reading field epsilon\n" << endl;
volScalarField epsilon
(
    IOobject
    (
```

```
        "epsilon",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

Info<< "Calculating field nutb\n" << endl;
volScalarField nutb
(
    IOobject
    (
        "nutb",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    Cmu*sqr(k)/epsilon
);

Info<< "Calculating field nuEffa\n" << endl;
volScalarField nuEffa
(
```

```
        IOobject
        (
            "nuEffa",
            runTime.timeName(),
            mesh,
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        sqr(Ct)*nutb + nua
    );

    Info<< "Calculating field nuEffb\n" << endl;
    volScalarField nuEffb
    (
        IOobject
        (
            "nuEffb",
            runTime.timeName(),
            mesh,
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        nutb + nub
    );
```

DDtU.H

```
{
```

```
    DDtUa =
        fvc::ddt(Ua)
      + fvc::div(phia, Ua)
      - fvc::div(phia)*Ua;

    DDtUb =
        fvc::ddt(Ub)
      + fvc::div(phib, Ub)
      - fvc::div(phib)*Ub;
}
```

kEpsilon.H

```
if (turbulence)
{
    if (mesh.changing())
    {
        y.correct();
    }

    tmp<volTensorField> tgradUb = fvc::grad(Ub);
    volScalarField G(2*nutb*(tgradUb() && dev(symm(tgradUb()))));
    tgradUb.clear();

    #include "wallFunctions.H"

    // Dissipation equation
    fvScalarMatrix epsEqn
```

```
(
    fvm::ddt(beta, epsilon)
  + fvm::div(phib, epsilon)
  - fvm::laplacian
    (
        alphaEps*nuEffb, epsilon,
        "laplacian(DepsilonEff,epsilon)"
    )
  ==
     C1*beta*G*epsilon/k
   - fvm::Sp(C2*beta*epsilon/k, epsilon)
);

#include "wallDissipation.H"

epsEqn.relax();
epsEqn.solve();
epsilon.max(dimensionedScalar("zero", epsilon.dimensions(), 1.0e-15));

// Turbulent kinetic energy equation
fvScalarMatrix kEqn
(
    fvm::ddt(beta, k)
  + fvm::div(phib, k)
  - fvm::laplacian
    (
        alphak*nuEffb, k,
```

```
            "laplacian(DkEff,k)"
        )
      ==
        beta*G
      - fvm::Sp(beta*epsilon/k, k)
    );
    kEqn.relax();
    kEqn.solve();

    k.max(dimensionedScalar("zero", k.dimensions(), 1.0e-8));

    //- Re-calculate turbulence viscosity
    nutb = Cmu*sqr(k)/epsilon;

    #include "wallViscosity.H"
}

nuEffa = sqr(Ct)*nutb + nua;
nuEffb = nutb + nub;
```

liftDragCoeffs.H

```
volVectorField Ur(Ua - Ub);
    volScalarField magUr(mag(Ur));

    volScalarField CdaMagUr
    (
        (24.0*nub/da)*(scalar(1) + 0.15*pow(da*magUr/nub, 0.687))
    );
```

```
volScalarField CdbMagUr
(
    (24.0*nua/db)*(scalar(1) + 0.15*pow(db*magUr/nua, 0.687))
);

volScalarField dragCoef
(
    "Cd",
    0.75*(beta*rhob*CdaMagUr/da + alpha*rhoa*CdbMagUr/db)
);

volVectorField liftCoeff
(
    Cl*(beta*rhob + alpha*rhoa)*(Ur ^ fvc::curl(U))
);
```

pEqn.H

```
{
    surfaceScalarField alphaf(fvc::interpolate(alpha));
    surfaceScalarField betaf(scalar(1) - alphaf);

    volScalarField rUaA(1.0/UaEqn.A());
    volScalarField rUbA(1.0/UbEqn.A());

    surfaceScalarField rUaAf(fvc::interpolate(rUaA));
    surfaceScalarField rUbAf(fvc::interpolate(rUbA));

    Ua = rUaA*UaEqn.H();
```

```
Ub = rUbA*UbEqn.H();

surfaceScalarField phiDraga
(
    fvc::interpolate(beta/rhoa*dragCoef*rUaA)*phib + rUaAf*(g & mesh.Sf())
);
surfaceScalarField phiDragb
(
    fvc::interpolate(alpha/rhob*dragCoef*rUbA)*phia + rUbAf*(g & mesh.Sf())
);

forAll(p.boundaryField(), patchi)
{
    if (isA<zeroGradientFvPatchScalarField>(p.boundaryField()[patchi]))
    {
        phiDraga.boundaryField()[patchi] = 0.0;
        phiDragb.boundaryField()[patchi] = 0.0;
    }
}

phia = (fvc::interpolate(Ua) & mesh.Sf()) + fvc::ddtPhiCorr(rUaA, Ua, phia)
    + phiDraga;
phib = (fvc::interpolate(Ub) & mesh.Sf()) + fvc::ddtPhiCorr(rUbA, Ub, phib)
    + phiDragb;

phi = alphaf*phia + betaf*phib;

surfaceScalarField Dp
```

```
(
    "(rho*(1|A(U)))",
    alphaf*rUaAf/rhoa + betaf*rUbAf/rhob
);
while (pimple.correctNonOrthogonal())
{
    fvScalarMatrix pEqn
    (
        fvm::laplacian(Dp, p) == fvc::div(phi)
    );

    pEqn.setReference(pRefCell, pRefValue);


    pEqn.solve(mesh.solver(p.select(pimple.finalInnerIter())));

    if (pimple.finalNonOrthogonalIter())
    {
        surfaceScalarField SfGradp(pEqn.flux()/Dp);

        phia -= rUaAf*SfGradp/rhoa;
        phib -= rUbAf*SfGradp/rhob;
        phi = alphaf*phia + betaf*phib;

        p.relax();
        SfGradp = pEqn.flux()/Dp;

        Ua += (fvc::reconstruct(phiDraga - rUaAf*SfGradp/rhoa));
```

```
        //Ua += rUaA*(fvc::reconstruct(phiDraga/rUaAf - SfGradp/rhoa));
        Ua.correctBoundaryConditions();

        Ub += (fvc::reconstruct(phiDragb - rUbAf*SfGradp/rhob));
        //Ub += rUbA*(fvc::reconstruct(phiDragb/rUbAf - SfGradp/rhob));
        Ub.correctBoundaryConditions();

        U = alpha*Ua + beta*Ub;
      }
    }
}

#include "continuityErrs.H"
```

readBubbleFoamControls.H

```
 int nAlphaCorr(readInt(pimple.dict().lookup("nAlphaCorr")));

 Switch correctAlpha(pimple.dict().lookup("correctAlpha"));
```

UEqns.H

```
fvVectorMatrix UaEqn(Ua, Ua.dimensions()*dimVol/dimTime);
fvVectorMatrix UbEqn(Ub, Ub.dimensions()*dimVol/dimTime);

{
   volTensorField Rca(-nuEffa*(T(fvc::grad(Ua))));
   Rca = Rca + (2.0/3.0)*sqr(Ct)*I*k - (2.0/3.0)*I*tr(Rca);

   surfaceScalarField phiRa
   (
     - fvc::interpolate(nuEffa)
```

```
   *mesh.magSf()*fvc::snGrad(alpha)/fvc::interpolate(alpha + scalar(0.001))
);

UaEqn =
(
    (scalar(1) + Cvm*rhob*beta/rhoa)*
    (
        fvm::ddt(Ua)
      + fvm::div(phia, Ua, "div(phia,Ua)")
      - fvm::Sp(fvc::div(phia), Ua)
    )

  - fvm::laplacian(nuEffa, Ua)
  + fvc::div(Rca)

  + fvm::div(phiRa, Ua, "div(phia,Ua)")
  - fvm::Sp(fvc::div(phiRa), Ua)
  + (fvc::grad(alpha)/(fvc::average(alpha) + scalar(0.001)) & Rca)
 ==
//  g                    // Buoyancy term transfered to p-equation
  - fvm::Sp(beta/rhoa*dragCoef, Ua)
//+ beta/rhoa*dragCoef*Ub    // Explicit drag transfered to p-equation
  - beta/rhoa*(liftCoeff - Cvm*rhob*DDtUb)
);

UaEqn.relax();

volTensorField Rcb(-nuEffb*T(fvc::grad(Ub)));
```

```
Rcb = Rcb + (2.0/3.0)*I*k - (2.0/3.0)*I*tr(Rcb);

surfaceScalarField phiRb
(
   - fvc::interpolate(nuEffb)
    *mesh.magSf()*fvc::snGrad(beta)/fvc::interpolate(beta + scalar(0.001))
);

UbEqn =
(
    (scalar(1) + Cvm*rhob*alpha/rhob)*
    (
        fvm::ddt(Ub)
      + fvm::div(phib, Ub, "div(phib,Ub)")
      - fvm::Sp(fvc::div(phib), Ub)
    )

  - fvm::laplacian(nuEffb, Ub)
  + fvc::div(Rcb)

  + fvm::div(phiRb, Ub, "div(phib,Ub)")
  - fvm::Sp(fvc::div(phiRb), Ub)

  + (fvc::grad(beta)/(fvc::average(beta) + scalar(0.001)) & Rcb)
 ==
//  g                    // Buoyancy term transfered to p-equation
  - fvm::Sp(alpha/rhob*dragCoef, Ub)
```

```
    //+ alpha/rhob*dragCoef*Ua   // Explicit drag transfered to p-equation
      + alpha/rhob*(liftCoeff + Cvm*rhob*DDtUa)
    );

    UbEqn.relax();
}
```

wallDissipation.H

```
{
    const fvPatchList& patches = mesh.boundary();

    forAll(patches, patchi)
    {
        const fvPatch& p = patches[patchi];

        if (isA<wallFvPatch>(p))
        {
            epsEqn.setValues
            (
                p.faceCells(),
                epsilon.boundaryField()[patchi].patchInternalField()
            );
        }
    }
}
```

wallFunctions.H

```
{
```

```
labelList cellBoundaryFaceCount(epsilon.size(), 0);

scalar Cmu25 = ::pow(Cmu.value(), 0.25);
scalar Cmu75 = ::pow(Cmu.value(), 0.75);
scalar kappa_ = kappa.value();
scalar nub_ = nub.value();

const fvPatchList& patches = mesh.boundary();

//- Initialise the near-wall P field to zero
forAll(patches, patchi)
{
    const fvPatch& currPatch = patches[patchi];

    if (isA<wallFvPatch>(currPatch))
    {
        forAll(currPatch, facei)
        {
            label faceCelli = currPatch.faceCells()[facei];

            epsilon[faceCelli] = 0.0;
            G[faceCelli] = 0.0;
        }
    }
}

//- Accumulate the wall face contributions to epsilon and G
//  Increment cellBoundaryFaceCount for each face for averaging
```

```
forAll(patches, patchi)
{
    const fvPatch& currPatch = patches[patchi];

    if (isA<wallFvPatch>(currPatch))
    {
        const scalarField& nutbw = nutb.boundaryField()[patchi];

        scalarField magFaceGradU(mag(Ub.boundaryField()[patchi].snGrad()));

        forAll(currPatch, facei)
        {
            label faceCelli = currPatch.faceCells()[facei];

            scalar yPlus =
                Cmu25*y[patchi][facei]
                *::sqrt(k[faceCelli])
                /nub_;

            // For corner cells (with two boundary or more faces),
            // epsilon and G in the near-wall cell are calculated
            // as an average

            cellBoundaryFaceCount[faceCelli]++;

            epsilon[faceCelli] +=
                 Cmu75*::pow(k[faceCelli], 1.5)
                /(kappa_*y[patchi][facei]);
```

```
            if (yPlus > 11.6)
            {
                G[faceCelli] +=
                    (nutbw[facei] + nub_)*magFaceGradU[facei]
                    *Cmu25*::sqrt(k[faceCelli])
                    /(kappa_*y[patchi][facei]);
            }
        }
    }
}

// perform the averaging

forAll(patches, patchi)
{
    const fvPatch& curPatch = patches[patchi];

    if (isA<wallFvPatch>(curPatch))
    {
        forAll(curPatch, facei)
        {
            label faceCelli = curPatch.faceCells()[facei];

            epsilon[faceCelli] /= cellBoundaryFaceCount[faceCelli];
            G[faceCelli] /= cellBoundaryFaceCount[faceCelli];
        }
    }
```

```
    }
}
```

wallViscosity.H

```
{
    scalar Cmu25 = ::pow(Cmu.value(), 0.25);
    scalar kappa_ = kappa.value();
    scalar E_ = E.value();
    scalar nub_ = nub.value();

    const fvPatchList& patches = mesh.boundary();

    forAll(patches, patchi)
    {
        const fvPatch& currPatch = patches[patchi];

        if (isA<wallFvPatch>(currPatch))
        {
            scalarField& nutw = nutb.boundaryField()[patchi];

            forAll(currPatch, facei)
            {
                label faceCelli = currPatch.faceCells()[facei];

                // calculate yPlus
                scalar yPlus =
                    Cmu25*y[patchi][facei]
                   *::sqrt(k[faceCelli])
```

```
            /nub_;

          if (yPlus > 11.6)
          {
             nutw[facei] =
                yPlus*nub_*kappa_
               /::log(E_*yPlus)
              - nub_;
          }
          else
          {
             nutw[facei] = 0.0;
          }
       }
    }
  }
}
```

write.H

```
if (runTime.outputTime())
  {
    volVectorField Ur
    (
      IOobject
      (
        "Ur",
        runTime.timeName(),
```

```
            mesh,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        Ua - Ub
    );

    runTime.write();
}
```

Geometria e Malha

blockMeshDict

```
convertToMeters 1;

vertices
(
    (0 0 0)
    (4 0 0)
    (4 0.35 0)
    (0 0.35 0)
    (0 0 0.2)
    (4 0 0.2)
    (4 0.35 0.2)
    (0 0.35 0.2)
);

blocks
```

```
(
    hex (0 1 2 3 4 5 6 7) (400 35 1) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    downwalls
    {
        type wall;
        faces
        (
            (1 5 4 0)
        );
    }

    upwalls
    {
        type patch;
        faces
        (
            (3 7 6 2)
        );
```

```
    }

    inlet
    {
        type patch;
        faces
        (
            (0 4 7 3)


        );
    }

    outlet
    {
        type patch;
        faces
        (


            (2 6 5 1)
        );
    }
);

mergePatchPairs
(
);
```

Propriedades

g

```
dimensions      [0 1 -2 0 0 0 0];
value           ( 0 -9.81 0 );
```

RASProperties

```
RASModel        kEpsilon;

turbulence      on;

printCoeffs     on;
```

transporteProperties

```
rhoa            rhoa [ 1 -3 0 0 0 0 0 ] 1;
rhob            rhob [ 1 -3 0 0 0 0 0 ] 1000;

nua             nua [ 0 2 -1 0 0 0 0 ] 1.6e-05;

nub             nub [ 0 2 -1 0 0 0 0 ] 1e-06;

da              da [ 0 1 0 0 0 0 0 ] 0.0003;

db              db [ 0 1 0 0 0 0 0 ] 0.0001;

Cvm             Cvm [ 0 0 0 0 0 0 0 ] 0.1;

Cl              Cl [ 0 0 0 0 0 0 0 ] -0.1;
```

```
Ct          Ct [ 0 0 0 0 0 0 0 ] 0.1;
```

Sistema e esquemas.

controlDict

```
application     bubbleFoam;

startFrom       startTime;

startTime       0;
stopAt          endTime;

endTime         100;

deltaT          0.0001;

writeControl    adjustableRunTime;

writeInterval   0.1;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression uncompressed;

timeFormat      general;

timePrecision   6;

runTimeModifiable yes;
```

```
adjustTimeStep yes;

maxCo          0.5;

maxDeltaT      0.0005;
```

decomposeParDict

```
numberOfSubdomains 4;

method         scotch;

simpleCoeffs
{
   n             ( 1 1 4);
   delta         0.001;
}

hierarchicalCoeffs
{
   n             ( 2 2 12 );
   delta         0.001;
   order         xyz;
}

/*
metisCoeffs
{
```

```
    processorWeights (1 1 1 1 1 1 1 1 1 1 1 1);
}
*/

scotchCoeffs
{
    processorWeights (1 1 1 1 1 1 1 1 1 1 1 1);
}

manualCoeffs
{
    dataFile        "";
}

distributed     no;

roots           ( );
```

fvSchemes

```
ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         Gauss linear;
}
```

```
divSchemes
{
    default         none;
    div(phia,Ua)    Gauss upwind;
    div(phib,Ub)    Gauss upwind;
    div(phib,k)     Gauss upwind;
    div(phib,epsilon) Gauss upwind;
    div(phi,alpha)  Gauss upwind;
    div((-nuEffa*T(grad(Ua)))) Gauss linear;
    div((-nuEffb*T(grad(Ub)))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
    laplacian(nuEffa,Ua) Gauss linear corrected;
    laplacian(nuEffb,Ub) Gauss linear corrected;
    laplacian((rho*(1|A(U))),p) Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
```

```
{
    default        corrected;
}

fluxRequired
{
    default        no;
    p              ;
}
```

fvSolution

```
solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-07;
        relTol          0.01;
        smoother        DIC;
        nPreSweeps      0;
        nPostSweeps     2;
        nFinestSweeps   2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 10;
        agglomerator    faceAreaPair;
        mergeLevels     2;
```

```
    }

    pFinal
    {
        $p;
        tolerance       1e-07;
        relTol          0.01;
    }

    "(Ua|UaFinal)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-10;
        relTol          0.01;
    }

    "(Ub|UbFinal)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-10;
        relTol          0.01;
    }

    alpha
    {
```

```
    solver          PBiCG;
    preconditioner  DILU;
    tolerance       1e-10;
    relTol          0.1;
}

alphaFinal
{
    $alpha;
    tolerance       1e-10;
    relTol          0;
}

epsilon
{
    solver          PBiCG;
    preconditioner  DILU;
    tolerance       1e-6;
    relTol          0.1;
}

epsilonFinal
{
    $alpha;
    tolerance       1e-6;
    relTol          0.1;
}
```

```
    k
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-6;
        relTol          0.1;
    }

    kFinal
    {
        $alpha;
        tolerance       1e-6;
        relTol          0.1;
    }

}

PIMPLE
{
    nCorrectors     3;
    nNonOrthogonalCorrectors 1;
    nAlphaCorr      2;
    correctAlpha    2;
    pRefCell        0;
    pRefValue       0;
}
```

```
relaxationFactors
{
  // p                    2e-1;
   Ua                          2e-1;
   Ub                   2e-1;
   alpha                       2e-1;
   k                    2e-1;
   epsilon                     2e-1;

}
```

setFieldsDict (definição dos aeradores)

```
defaultFieldValues
(
    volScalarFieldValue alpha 0
            volVectorFieldValue Ua (0 0 0)
);

regions
(
/*
            boxToFace
    {
        box (1.1 0 0) (3.71 0 0.2);

                    fieldValues
```

```
        (
            volScalarFieldValue alpha 0.1
                                volVectorFieldValue Ua (0 0.6 0)
                                ////volVectorFieldValue Ub (0 0.5 0)
        );
    }

    boxToFace
    {
        box (1.2 0 0) (1.21 0 0.2);

                fieldValues
        (
            volScalarFieldValue alpha 1
                                volVectorFieldValue Ua (0 0.5 0)
                                ////volVectorFieldValue Ub (0 0.5 0)
        );
    }

*/

            boxToFace
            {
                box (1.2 0 0) (1.21 0 0.2);

                fieldValues
        (
```

```
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
                    box (1.3 0 0) (1.31 0 0.2);

            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
            box (1.4 0 0) (1.41 0 0.2);

            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
```

```
);
    }

    boxToFace
    {

            box (1.5 0 0) (1.51 0 0.2);

            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
            box (1.6 0 0) (1.61 0 0.2);

            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }
```

```
    boxToFace
    {
        box (1.7 0 0) (1.71 0 0.2);

        fieldValues
(
    volScalarFieldValue alpha 1
            volVectorFieldValue Ua (0 0.5 0)
            ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
        box (1.8 0 0) (1.81 0 0.2);

        fieldValues
(
    volScalarFieldValue alpha 1
            volVectorFieldValue Ua (0 0.5 0)
            ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
        box (1.9 0 0) (1.91 0 0.2);
```

```
            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
            box (2.0 0 0) (2.01 0 0.2);

            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
            box (2.1 0 0) (2.11 0 0.2);

            fieldValues
(
```

```
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
            box (2.2 0 0) (2.21 0 0.2);

            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
            box (2.3 0 0) (2.31 0 0.2);

            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
```

```
);
    }

    boxToFace
    {
            box (2.4 0 0) (2.41 0 0.2);


            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
            box (2.5 0 0) (2.51 0 0.2);

            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }
```

```
    boxToFace
    {
            box (2.6 0 0) (2.61 0 0.2);

            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
            box (2.7 0 0) (2.71 0 0.2);

            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
            box (2.8 0 0) (2.81 0 0.2);
```

```
            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
            box (2.9 0 0) (2.91 0 0.2);

            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
            box (3.0 0 0) (3.01 0 0.2);

            fieldValues
(
    volScalarFieldValue alpha 1
```

```
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
            box (3.1 0 0) (3.11 0 0.2);

            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
            box (3.2 0 0) (3.21 0 0.2);

    fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
```

```
);
    }

    boxToFace
    {
            box (3.3 0 0) (3.31 0 0.2);

            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
            box (3.4 0 0) (3.41 0 0.2);

            fieldValues
(
    volScalarFieldValue alpha 1
                    volVectorFieldValue Ua (0 0.5 0)
                    ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
```

```
    {
        box (3.5 0 0) (3.51 0 0.2);

        fieldValues
(
    volScalarFieldValue alpha 1
            volVectorFieldValue Ua (0 0.5 0)
            ////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {

        box (3.6 0 0) (3.61 0 0.2);

        fieldValues
(
    volScalarFieldValue alpha 1
            volVectorFieldValue Ua (0 0.5 0)
            //////volVectorFieldValue Ub (0 0.5 0)
);
    }

    boxToFace
    {
        box (3.7 0 0) (3.71 0 0.2);
```

```
        fieldValues
        (
            volScalarFieldValue alpha 1
                                volVectorFieldValue Ua (0 0.5 0)
                                ////volVectorFieldValue Ub (0 0.5 0)
        );

        }

);
```

Condições de contorno

alpha

```
dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    downwalls
    {
        type            fixedValue;
        value           uniform 0;
    }
    upwalls
    {
```

```
        type            inletOutlet;
        inletValue      uniform 0;
        value           uniform 0;
    }
    inlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    outlet
    {
        type            zeroGradient;
    }
    defaultFaces
    {
        type            empty;
    }
}
```

p

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

internalField   uniform 0;

boundaryField
{
```

```
inlet
{
    type            zeroGradient;
}

outlet
{
    //type            fixedValue;
    //value           uniform 0;

        type            zeroGradient;
}

upwalls
{
        type            fixedValue;
    value           uniform 0;

    //type            buoyantPressure;
    //value           uniform 0;
}

downwalls
{
    type            zeroGradient;
}

defaultFaces
```

```
    {
        type            empty;
    }
}
```

epsilon

```
dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 0.1;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 0.1;
    }

    outlet
    {
        type            inletOutlet;
        inletValue      uniform 0.1;
        value           uniform 0.1;
    }

    upwalls
    {
        type            zeroGradient;
```

```
    }

    downwalls
    {
        type            zeroGradient;
    }

    defaultFaces
    {
        type            empty;
    }
}
```

k

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

internalField   uniform 1e-8;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 1e-8;
    }

    outlet
    {
```

```
        type            inletOutlet;
        inletValue      uniform 1e-8;
        value           uniform 1e-8;
    }

    upwalls
    {
        type            zeroGradient;
    }
    downwalls
    {
        type            zeroGradient;
    }

    defaultFaces
    {
        type            empty;
    }
}
```

Ua

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

internalField   uniform (0 0 0);

boundaryField
{
    downwalls
```

```
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    upwalls
    {
        type            inletOutlet;
        inletValue      uniform (0 0 0);
        value           uniform (0 0 0);
    }
    inlet
    {
        type            fixedValue;
        value           uniform (0.00238 0 0);
    }
    outlet
    {
        type            zeroGradient;
    }
    defaultFaces
    {
        type            empty;
    }
}
```

Ub

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


internalField   uniform (0.00238 0 0);


boundaryField
{
    downwalls
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    upwalls
    {
        type            inletOutlet;
        inletValue      uniform (0 0 0);
        value           uniform (0 0 0);
    }
    inlet
    {
        type            fixedValue;
        value           uniform (0.00238 0 0);
    }
    outlet
    {
        type            zeroGradient;
```

```
    }
    defaultFaces
    {
        type            empty;
    }
}
```