



UNIVERSIDADE ESTADUAL PAULISTA
“JÚLIO DE MESQUITA FILHO”
Faculdade de Ciências e Tecnologia
Câmpus de Presidente Prudente

Um método numérico para o tratamento de mudanças topológicas em escoamentos viscoelásticos com superfície livre

Hugo Leonardo França

Orientador: Prof. Dr. Cassio Machiaveli Oishi

Programa: Matemática Aplicada e Computacional

Presidente Prudente, Setembro de 2018

UNIVERSIDADE ESTADUAL PAULISTA

Faculdade de Ciências e Tecnologia de Presidente Prudente

Programa de Pós-Graduação em Matemática Aplicada e Computacional

**Um método numérico para o tratamento de
mudanças topológicas em escoamentos
viscoelásticos com superfície livre**

Hugo Leonardo França

Orientador: Prof. Dr. Cassio Machiaveli Oishi

Dissertação apresentada ao Programa de Pós-Graduação em Matemática Aplicada e Computacional da Faculdade de Ciências e Tecnologia da UNESP para obtenção do título de Mestre em Matemática Aplicada e Computacional.

Presidente Prudente, Setembro de 2018

Ficha catalográfica elaborada pela Seção Técnica de Aquisição e Tratamento da Informação - Diretoria Técnica de Biblioteca e Documentação - UNESP, Campus de Presidente Prudente

França, Hugo Leonardo.
F881m Um método numérico para o tratamento de mudanças topológicas em escoamentos viscoelásticos com superfície livre / Hugo Leonardo França. - 2018
117 f. : il.

Orientador: Cassio Machiaveli Oishi
Dissertação (mestrado) - Universidade Estadual Paulista. Faculdade de Ciências e Tecnologia, Presidente Prudente, 2018
Inclui bibliografia

1. Equações de Navier-Stokes incompressíveis. 2. Mudanças topológicas. 3. Formulação tensão natural. I. França, Hugo Leonardo. II. Oishi, Cassio Machiaveli. III. Universidade Estadual Paulista. Faculdade de Ciências e Tecnologia. IV. Título.

Claudia Adriana Spindola
CRB-8º/5790

CERTIFICADO DE APROVAÇÃO

TÍTULO: *Um método numérico para o tratamento de mudanças topológicas em escoamentos viscoelásticos com superfície livre*

AUTOR: HUGO LEONARDO FRANÇA

ORIENTADOR: CÁSSIO MACHIAVELI OISHI

Aprovado como parte das exigências para obtenção do Título de Mestre em MATEMÁTICA APLICADA E COMPUTACIONAL, pela Comissão Examinadora:

Prof. Dr. CÁSSIO MACHIAVELI OISHI

Departamento de Matemática e Computação / Faculdade de Ciências e Tecnologia de Presidente Prudente

Prof. Dr. FABRÍCIO SIMEONI DE SOUSA

Instituto de Ciências Matemáticas e de Computação / UNIVERSIDADE DE SÃO PAULO

Prof. Dr. MESSIAS MENEGUETTE JUNIOR

Departamento de Matemática e Computação / Faculdade de Ciências e Tecnologia de Presidente Prudente

Presidente Prudente, 10 de setembro de 2018.

Agradecimentos

Agradeço ao meu orientador Cassio Machiaveli Oishi, por todo o conhecimento cedido e ajuda prestada durante este projeto.

Aos colegas de laboratório Fernando Pacanelli, Débora Medeiros, Irineu Palhares e Amauri G. Junior, pelas trocas de informações e contribuições.

Ao professor Jonathan Evans, pela orientação durante o período fora do Brasil.

Aos funcionários da Seção de Pós-Graduação e do Escritório de Pesquisa, pela ajuda com a parte burocrática do projeto.

À Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) pelo apoio financeiro dado através do processo 2016/00456-2. As opiniões, hipóteses e conclusões ou recomendações expressas neste material são de responsabilidade do autor e não necessariamente refletem a visão da FAPESP.

Resumo

Neste trabalho é apresentado o estudo de um método numérico para resolver as equações de Navier-Stokes incompressíveis em escoamentos que possuem superfícies livres e mudanças topológicas. As equações governantes são resolvidas por um método de projeção que desacopla as incógnitas velocidade e pressão. A discretização é feita através de aproximações por diferenças finitas aplicadas a uma malha computacional não-uniforme. O método numérico é aplicado para a solução de problemas envolvendo fluidos Newtonianos e não-Newtonianos. Em particular, os efeitos viscoelásticos são descritos pelo modelo Oldroyd-B, utilizando a formulação Cartesiana clássica e uma forma alternativa para a decomposição da parte polimérica do tensor tensão extra. Esta estratégia alternativa de decomposição, conhecida como Formulação Tensão Natural, é muito atual e resultados numéricos são originalmente discutidos neste trabalho. O novo código com malha não-uniforme é testado nos seguintes problemas: escoamento na cavidade (*lid-driven cavity*), escoamento no *cross-slot*, e escoamento no canal com contração.

A representação da superfície livre é feita através do método *Front-Tracking*, que descreve a interface de forma explícita através de partículas marcadoras. O algoritmo de mudanças topológicas é baseado em uma técnica que detecta e desfaz embaraçamentos presentes na interface. Este algoritmo é testado em simulações numéricas como: o impacto entre uma gota e uma camada de fluido, o impacto entre gotas e uma parede rígida, e o alongamento de um jato pela tensão superficial.

Palavras-Chave: *Equações de Navier-Stokes Incompressíveis, Mudanças Topológicas, Formulação Tensão Natural, Tensão Superficial, Fluidos Viscoelásticos.*

Abstract

This work presents the study of a numerical method for solving the incompressible Navier-Stokes equations for free-surface flows that undergo topological changes. The governing equations are solved through a projection method that decouples the velocity and pressure fields. The discretization is performed via finite differences approximations applied to a non-uniform mesh. The numerical scheme is applied for solving Newtonian and non-Newtonian fluid flows. In particular, the viscoelastic effects are described by the Oldroyd-B model, using the classic Cartesian formulation and also an alternative approach for the decomposition of the polymeric part of the extra stress tensor. This alternative decomposition strategy is known as Natural Stress Formulation, and numerical results are originally discussed in this work. The new code with a non-uniform mesh is tested in the following problems: the lid-driven cavity, the cross-slot problem, and the flow through a channel with contraction.

In order to represent the free-surface, a Front-Tracking method that describes the interface explicitly using marker particles is used. The algorithm for topological changes is based in a technique that detects when the interface is tangled and untangles it. This algorithm is tested in numerical simulations such as: the impact between a drop and a layer of fluid, the impact between drops and a solid wall, and the jetting break-up process under the effect of surface tension.

Keywords: *Navier-Stokes Equations, Topological Changes, Natural Stress Formulation, Surface Tension, Viscoelastic Fluid.*

Lista de Figuras

3.1	Célula computacional i, j com seus pontos de interesse nomeados.	27
3.2	Exemplo da classificação de células a partir de uma superfície livre.	28
3.3	Casos que geram todas as oito possibilidades de vetores normais utilizadas no método.	29
4.1	Exemplos de interfaces com diferentes formas e quantidades de curvas. . .	36
4.2	Representação Front-Tracking das interfaces apresentadas na Figura 4.1. .	37
4.3	Exemplo de uma curva Front-Tracking com sentido horário.	37
4.4	Exemplo de mudança topológica em um escoamento no qual duas bolhas de mesmo fluido se encontram. As bolhas são respectivamente advectadas com velocidades V_1 e V_2 que possuem sentidos opostos.	42
4.5	Exemplo de mudança topológica em um escoamento no qual uma bolha se desprende do restante do fluido.	43
4.6	Exemplos de interfaces com e sem embaraçamento. Os pontos com formato de quadrado indicam os nós das curvas. A cor dos segmentos de retas indicam curvas diferentes. O sentido das curvas não foi indicado nas imagens, pois não é relevante para o propósito destes exemplos.	44
4.7	Exemplo de advecção no qual pontos da interface cruzam múltiplas curvas em um único passo temporal. Isso não deve ocorrer para que o algoritmo de desembaraçamento possa ser utilizado. Esta figura é baseada em uma imagem dada como exemplo em [14].	45
4.8	Processo de divisão de curvas no ponto de intersecção. O painel 4.8a contém as curvas antes do processo de divisão. O painel 4.8b as apresenta após a divisão. Apenas os segmentos de reta que estão mais próximos do ponto de intersecção foram mostrados na figura.	48
4.9	Processo de divisão de curvas no problema da união de bolhas. 4.9a: interface embaraçada antes do processo de divisão. 4.9b: interface após o processo de divisão ser realizado no primeiro nó de cruzamento. 4.9c: interface após a divisão ser realizada no segundo nó de cruzamento.	49
4.10	Ângulo de cada curva ao redor de um nó de cruzamento. Estes ângulos são facilmente calculados aplicando a função arco-tangente a cada um dos quatro segmentos.	50
4.11	Interface do exemplo de união de bolhas (Figura 4.9c) após todas as etapas do algoritmo de embaraçamento.	56
4.12	Ilustração da superfície livre durante o processo que gera o embaraçamento da interface no caso da união de blocos de fluido.	58
4.13	Ilustração da superfície livre durante o processo que gera o embaraçamento da interface no caso da separação de um bloco de fluido, tornando-se dois.	59
4.14	Ilustração das etapas realizadas para aproximação da curvatura da superfície livre em uma célula.	62

4.15	Exemplo de caso no qual a superfície livre apresenta ondulação geradas pelo cálculo da curvatura.	63
4.16	Comparação entre uma superfície livre com ondulação (amarelo) e a nova superfície calculada através de sua curva de Bézier (verde).	65
5.1	Comparação entre solução numérica obtida pelo método implementado e solução de referência obtida de [27].	68
5.2	Linhas de corrente do escoamento na cavidade com diferentes números de Reynolds.	69
5.3	Geometria da contração com razão 4:1 usada nas simulações.	70
5.4	Estrutura das malhas não-uniformes usadas na geometria com contração.	71
5.5	Representação da quina de contração e da direção r na qual a análise assintótica será feita.	71
5.6	Valores de diversas propriedades de interesse em pontos próximos a uma quina da contração. O eixo horizontal representa a distância (r) de um dado ponto até a quina, e o eixo vertical é o valor da propriedade em questão neste ponto. As inclinações destacadas em cada imagem vêm da análise teórica realizada em [19, 13].	72
5.7	Malha não-uniforme utilizada na simulação <i>Cross-Slot</i> . A imagem mostra apenas a malha ao redor da região de maior refinamento. As células em azul são aquelas pelas quais há escoamento.	73
5.8	Visualização da superfície livre na simulação <i>die-swell</i> em diferentes instantes de tempo.	75
5.9	Instantes da simulação do impacto de uma gota. A coluna da esquerda representa a simulação realizada usando o algoritmo de mudança topológica. A coluna da direita não usa o algoritmo de mudança topológica.	77
5.10	Instantes da simulação do impacto de uma gota. A coluna da esquerda representa a simulação realizada usando o algoritmo de mudança topológica. A coluna da direita não usa o algoritmo de mudança topológica.	78
5.11	Visualização do momento de impacto da gota para diferentes valores de Reynolds (Re). A coluna da esquerda é uma simulação com $Re = 10$. A coluna central com $Re = 80$, e a da direita com $Re = 200$	79
5.12	Visualização da superfície livre na simulação <i>cross-slot</i> em diferentes instantes de tempo.	80
5.13	Visualização dos campos de velocidade e pressão para a geometria <i>cross-slot</i> no instante $t=7.5$, antes do preenchimento total do canal.	81
5.14	Estado inicial das simulações com gotas que impactam sobre uma superfície rígida. À esquerda, uma única gota é lançada. À direita, duas gotas são lançadas.	82
5.15	Largura das gotas calculadas com diferentes malhas e comparadas com a referência [21].	83
5.16	Interface da simulação não-Newtoniana em diferentes instantes de tempo.	84
5.17	Largura das gotas calculadas com diferentes Wi . São comparadas simulações com apenas uma gota e simulações com duas.	85
5.18	Interface da simulação de duas gotas com $Wi = 5$ em diferentes instantes de tempo.	86
5.19	Efeito da variação da distância entre gotas no valor da largura.	86
5.20	Esboço da geometria utilizada no problema do jato.	87

5.21	Comparação entre malhas para o caso B do jato com tensão superficial. As malhas em vermelho, verde e azul possuem $\Delta_x = 0.033$, $\Delta_x = 0.025$ e $\Delta_x = 0.02$ respectivamente. Todas elas são uniformes.	89
5.22	Caso A da simulação do jato com tensão superficial em diferentes instantes de tempo.	90
5.23	Caso B da simulação do jato com tensão superficial em diferentes instantes de tempo.	91
5.24	Caso C da simulação do jato com tensão superficial em diferentes instantes de tempo.	92
5.25	Caso D da simulação do jato com tensão superficial em diferentes instantes de tempo.	93
5.26	Comprimento do jato em função de $We^{0.28}Fr^{0.78}$	93
5.27	Comparação da espessura do jato antes da quebra para cada um dos quatro casos. Em vermelho e azul são apresentadas as soluções numérica e analítica, respectivamente.	94
B.1	Malha não-uniforme usada como exemplo na discretização.	103
B.2	Definição dos pontos x_D , x_U e x_R que serão usados no método CUBISTA. (a) caso com $u_f > 0$ e (b) caso com $u_f < 0$	104

Lista de Tabelas

4.1	Execução passo-a-passo do algoritmo 4.4 aplicado ao nó de cruzamento superior da figura 4.9c.	53
4.2	Tabela de regiões e curvas relacionada a Figura 4.9c após múltiplas aplicações do algoritmo 4.4 aos nós de cruzamento.	54
5.1	Detalhes das malhas uniforme e não-uniformes para a geometria com contração.	70
5.2	Comparação entre o comprimento do vórtice obtido neste trabalho com valores de referência obtidos em [38] para diferentes números de Reynolds.	74
5.3	Espessura do jato após o processo de inchamento pelos métodos CSF e NSF.	75
5.4	Quatro casos utilizados nas simulações desta seção. Cada caso possui uma velocidade de injeção (U), levando a parâmetros adimensionais diferentes.	88

Sumário

Resumo	5
Abstract	7
Lista de Figuras	8
Lista de Tabelas	11
Capítulos	
1 Introdução	17
2 Modelagem matemática para escoamentos incompressíveis	19
2.1 Equações governantes para fluidos Newtonianos e viscoelásticos	19
2.2 Formulação tensão natural	20
2.3 Condições iniciais e de contorno	22
2.3.1 Paredes rígidas	22
2.3.2 Entrada de fluido (<i>inflow</i>)	23
2.3.3 Saída de fluido (<i>outflow</i>)	23
2.3.4 Superfície livre	23
3 Método numérico	25
3.1 Método de projeção	25
3.2 Discretização do domínio	26
3.3 Malha computacional	28
3.4 Discretização temporal da equação de quantidade de movimento	29
3.5 Construção da equação de atualização da pressão	30
3.6 Condição de contorno para a equação de Poisson sobre a superfície livre . .	30
3.7 Atualização do tensor tensão	31
3.7.1 Atualização pela formulação CSF	31
3.7.2 Atualização pela formulação NSF	32
3.8 Resumo do método numérico	33
4 Tratamento numérico da superfície livre	35
4.1 Representação e advecção de interfaces pelo método Front-Tracking	35
4.1.1 Representação Front-Tracking	35
4.1.2 Advecção Front-Tracking	39
4.2 Descrição de um método para realização de mudanças topológicas	41
4.2.1 Mudanças topológicas: conceitos básicos	41
4.2.2 Algoritmo de desembaraçamento	42
4.3 Mudanças topológicas em escoamentos com o método MAC	55

4.3.1	Caso 1: União	56
4.3.2	Caso 2: Separação	57
4.4	Aproximação da curvatura da superfície livre	58
4.4.1	Algoritmo para o cálculo da curvatura	59
4.4.2	Dificuldades encontradas com o algoritmo	62
5	Resultados Numéricos	67
5.1	Escoamentos confinados	67
5.1.1	Escoamento na cavidade (<i>lid-driven cavity</i>)	67
5.1.2	Escoamento com contração	68
5.1.3	Escoamento no <i>cross-slot</i>	72
5.2	Escoamentos com superfície livre	74
5.2.1	Inchamento do extrudado (<i>die-swell</i>)	74
5.2.2	Impacto entre gota e uma camada de fluido	75
5.2.3	Escoamento no <i>cross-slot</i> com superfície livre	77
5.2.4	Impacto entre gotas e uma parede rígida	81
5.2.5	Jato com tensão superficial	87
6	Conclusão e etapas seguintes	95
	Referências	96
	Apêndices	
A	Criando fórmulas de diferenças sobre malhas não-uniformes	101
A.1	Fórmulas de diferenças finitas em uma malha não-uniforme	101
B	Aproximação dos termos convectivos pelo método CUBISTA	103
B.1	Exemplo 1: Equação de quantidade de movimento	104
B.2	Exemplo 2: Equações constitutivas	105
C	Discretização espacial das equações	107
C.1	Discretização da equação de quantidade de movimento	107
C.2	Discretização da equação de Poisson	109
C.3	Discretização das equações de atualização da velocidade	110
C.4	Discretização das equações constitutivas	110
C.4.1	Formulação CSF	110
C.4.2	Formulação NSF	111
C.5	Discretização das condições de contorno de superfície livre	111
C.5.1	Caso 1: $\mathbf{n} = (0, 1)$	112
C.5.2	Caso 2: $\mathbf{n} = (0, -1)$	113
C.5.3	Caso 3: $\mathbf{n} = (1, 0)$	114
C.5.4	Caso 4: $\mathbf{n} = (-1, 0)$	114
C.5.5	Caso 5: $\mathbf{n} = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$	115
C.5.6	Caso 6: $\mathbf{n} = (\frac{-\sqrt{2}}{2}, \frac{\sqrt{2}}{2})$	115
C.5.7	Caso 7: $\mathbf{n} = (\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})$	115
C.5.8	Caso 8: $\mathbf{n} = (-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2})$	115

Introdução

Existem inúmeros exemplos de escoamentos de fluidos que podem ser considerados complexos, isto é, aqueles nos quais poucas informações teóricas, experimentais e numéricas são apresentadas na literatura. Uma classe de escoamentos complexos envolve fenômenos encontrados em problemas onde o domínio pode sofrer deformações e mudanças topológicas. Nestes problemas, a interface ou superfície livre possui uma dinâmica que é acoplada ao comportamento do fluido, o que gera dificuldades tanto do ponto de vista teórico, como numérico. Do ponto de vista prático, escoamentos de fluidos com mudanças topológicas são encontrados em muitas aplicações, como no desenvolvimento de novas metodologias de impressoras jato de tinta [6, 46], no processo de criação de polímeros [35], no estudo de atomização e pulverização [4], entre outras. Portanto, é fundamental uma compreensão melhor dos fenômenos que ocorrem nestas aplicações para auxiliar no desenvolvimento de novas tecnologias e nas soluções de problemas complexos já existentes e identificados pela indústria.

Técnicas numéricas capazes de resolver problemas com movimento de interfaces ou superfícies livres em escoamentos que geram mudanças topológicas são, em geral, classificadas em três grupos: métodos com representação explícita, implícita ou híbridos. A primeira classe, foco do estudo desse projeto, está relacionada com métodos de rastreamento de frente (*Front-tracking*), onde a interface é representada por partículas marcadoras ou pontos marcadores (*Marker-and-Cell*). Os trabalhos pioneiros dentro desta classe foram apresentados pelos grupos de Glimm e colaboradores [16, 17] e Tryggvason e colaboradores [50, 51]. A segunda classe, a dos métodos com representação implícita da interface, é a mais popular quando o domínio no qual o escoamento de fluido é analisado sofre deformações e mudanças topológicas. Isto se justifica pelo fato de que nesta classe de métodos, as mudanças topológicas são tratadas automaticamente pela formulação. Neste grupo, os trabalhos de Hirt e Nichols [20], relacionados ao método VOF (*Volume-of-Fluid*), e de Sussman et al. [45], que apresentaram o método Level-set em aplicações de escoamentos de fluidos, são os mais citados. A terceira classe, aquela dos métodos híbridos, surgiu como uma tentativa de combinar as vantagens de métodos com representação explícita e implícita de interface. Esses estudos são mais atuais, e as principais metodologias deste grupo foram propostas em [8, 32, 7, 28].

Devido a complexidade para sua implementação, métodos de rastreamento de frente combinados a representação via partículas marcadoras são mais usados em escoamentos com superfície livre sem mudanças topológicas significativas [31, 34, 33, 30] ou em escoamentos multifásicos [12, 37, 11]. Quando mudanças topológicas ou deformações precisam ser representadas com a aplicação do método *Front-tracking*, algumas dificuldades precisam ser superadas, como por exemplo o alto custo computacional envolvido na estrutura

de dados que lida com a geometria do domínio. Alternativas para superar essas dificuldades foram propostas por Torres e Brackbill [49], no qual não há exigência de conectividade entre os pontos que definem a interface, Shin e Juric [42], que descreveram um método com reconstrução (função de contorno de nível) que lida naturalmente com mudanças topológicas, e por Glimm e colaboradores [14, 15] onde um processo de reconhecimento de embaraçamento de interfaces é utilizado para lidar com a mudança topológica.

Neste trabalho, os dois principais objetivos alcançados foram:

- Implementação de um código que resolva as equações de Navier-Stokes para fluidos Newtonianos e viscoelásticos através do método de diferenças finitas aplicado a uma malha não-uniforme. Esta etapa foi iniciada com escoamentos confinados (sem superfície livre) e testes de validação foram realizados em geometrias que possuem singularidades geométricas, como a geometria com contração, estudada também em [10, 36], ou a *cross-slot*, estudada em [38]. Por possuírem quinas, estas geometrias destacam a importância de se utilizar uma malha não-uniforme nas simulações, já que a região ao redor das quinas exige uma resolução mais precisa do que as demais.
- Implementação de um método numérico que resolve problemas com superfície livres utilizando o método *Front-Tracking*. Este implementação foi capaz de lidar com mudanças topológicas e deformações em escoamentos complexos. O algoritmo de mudança topológica teve como base os trabalhos de Glimm e colaboradores [14, 15]. Testes numéricos foram realizados em problemas como o impacto de gotas em uma placa rígida, também realizado em [21, 39], e o jato com tensão superficial, simulado também em [43, 41].

Durante seis meses do desenvolvimento deste projeto, foi realizado um estágio na Universidade de Bath, na Inglaterra. Este estágio, supervisionado pelo professor Jonathan Evans, teve como principal objetivo a implementação do módulo que trata os efeitos viscoelásticos no código. Em especial, foi implementada uma formulação conhecida como *Natural Stress Formulation* que altera a decomposição usada para a parte polimérica do tensor tensão extra. Esta formulação, que visa melhorar a precisão do tensor obtido em geometrias com singularidades, foi proposta por [40], e sua aplicação a escoamentos transientes é muito atual. Resultados obtidos neste trabalho com esta formulação aplicada a uma geometria com contração foram sintetizados em um artigo intitulado “*Application of the natural stress formulation for solving transient sharp corner flows*” que será submetido em breve.

Modelagem matemática para escoamentos incompressíveis

2.1 Equações governantes para fluidos Newtonianos e viscoelásticos

Problemas que envolvem escoamento de fluidos podem ser modelados pelas equações de Navier-Stokes. Estas equações podem ser vistas de diversas formas diferentes, dependendo da natureza do problema, do tipo de fluido, do tipo do escoamento e da quantidade de dimensões.

As equações de Navier-Stokes dimensionais para um escoamento incompressível são as equações de quantidade de movimento e continuidade, dadas respectivamente por

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) \right) = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \rho \mathbf{g}, \quad (2.1)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (2.2)$$

onde \mathbf{u} é a velocidade do fluido, p é a pressão, ρ é a densidade do fluido e \mathbf{g} é um termo fonte (geralmente aceleração da gravidade). O tensor tensão extra $\boldsymbol{\tau}$ é decomposto em uma parcela de solvente $\boldsymbol{\tau}^s$ e uma polimérica $\boldsymbol{\tau}^p$ e será escrito como

$$\boldsymbol{\tau} = \boldsymbol{\tau}^s + \boldsymbol{\tau}^p. \quad (2.3)$$

A componente solvente representa o comportamento Newtoniano do fluido e é dada por

$$\boldsymbol{\tau}^s = 2\eta_s \mathbf{D}, \quad (2.4)$$

onde η_s é a viscosidade do solvente e $\mathbf{D} = \frac{1}{2} (\nabla \mathbf{u} + (\nabla \mathbf{u})^T)$ é o tensor taxa de deformação. A equação constitutiva para a parcela polimérica $\boldsymbol{\tau}^p$ do tensor depende do modelo de representação adotado. Neste trabalho usaremos o modelo Oldroyd-B, nos levando à equação constitutiva

$$\boldsymbol{\tau}^p + \lambda_p \overset{\nabla}{\boldsymbol{\tau}}^p = 2\eta_p \mathbf{D}, \quad (2.5)$$

onde λ_p é o tempo de relaxação do polímero e η_p é a viscosidade do polímero. A derivada $\overset{\nabla}{\boldsymbol{\tau}}^p$ é definida por

$$\overset{\nabla}{\boldsymbol{\tau}}^p = \frac{\partial \boldsymbol{\tau}^p}{\partial t} + (\mathbf{u} \cdot \nabla) \boldsymbol{\tau}^p - (\nabla \mathbf{u}) \boldsymbol{\tau}^p - \boldsymbol{\tau}^p (\nabla \mathbf{u})^T. \quad (2.6)$$

Neste momento, vamos adimensionalizar as equações da seguinte forma

$$\mathbf{x} = L\bar{\mathbf{x}}, \quad t = \frac{L}{U}\bar{t}, \quad \mathbf{u} = U\bar{\mathbf{u}}, \quad p = \rho U^2 \bar{p}, \quad \boldsymbol{\tau}^s = \frac{\eta_0 U}{L} \bar{\mathbf{T}}^s, \quad \boldsymbol{\tau}^p = \frac{\eta_0 U}{L} \bar{\mathbf{T}}, \quad \mathbf{g} = g_0 \bar{\mathbf{g}}, \quad (2.7)$$

usando U e L como comprimento e velocidade características, respectivamente, g_0 como constante de escala para o termo fonte, e a viscosidade total $\eta_0 = \eta_s + \eta_p$.

Removendo, por conveniência, as barras em (2.7), encontramos que as equações governantes adimensionais para o escoamento incompressível de um fluido viscoelástico do tipo Oldroyd-B são dadas por

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = -\nabla p + \frac{\beta}{Re} \nabla^2 \mathbf{u} + \frac{1}{Re} \nabla \cdot \mathbf{T} + \frac{1}{Fr^2} \mathbf{g}, \quad (2.8)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (2.9)$$

$$\mathbf{T} + Wi \left(\frac{\partial \mathbf{T}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{T} - (\nabla \mathbf{u}) \mathbf{T} - \mathbf{T} (\nabla \mathbf{u})^T \right) = 2(1 - \beta) \mathbf{D}, \quad (2.10)$$

com parâmetros adimensionais: número de Reynolds (Re), número de Weissenberg (Wi), número de Froude (Fr) e razão das viscosidades ($\beta \in (0, 1]$) definidos como

$$Re = \frac{\rho UL}{\eta_0}, \quad Wi = \frac{\lambda_p U}{L}, \quad Fr = \frac{U}{\sqrt{g_0 L}}, \quad \beta = \frac{\eta_s}{\eta_0}. \quad (2.11)$$

Escrevendo todos os operadores e o tensor das equações (2.8)-(2.10) em coordenadas cartesianas bidimensionais, as equações podem ser escritas como

$$\frac{\partial u}{\partial t} + \frac{\partial(uu)}{\partial x} + \frac{\partial(uv)}{\partial y} = -\frac{\partial p}{\partial x} + \frac{\beta}{Re} \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + \frac{1}{Re} \left(\frac{\partial T^{xx}}{\partial x} + \frac{\partial T^{xy}}{\partial y} \right) + \frac{1}{Fr^2} g_x, \quad (2.12)$$

$$\frac{\partial v}{\partial t} + \frac{\partial(uv)}{\partial x} + \frac{\partial(vv)}{\partial y} = -\frac{\partial p}{\partial y} + \frac{\beta}{Re} \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + \frac{1}{Re} \left(\frac{\partial T^{xy}}{\partial x} + \frac{\partial T^{yy}}{\partial y} \right) + \frac{1}{Fr^2} g_y, \quad (2.13)$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0. \quad (2.14)$$

$$T^{xx} + Wi \left(\frac{\partial T^{xx}}{\partial t} + \frac{\partial(uT^{xx})}{\partial x} + \frac{\partial(vT^{xx})}{\partial y} - 2\frac{\partial u}{\partial x} T^{xx} - 2\frac{\partial u}{\partial y} T^{xy} \right) = 2(1 - \beta) \frac{\partial u}{\partial x}, \quad (2.15)$$

$$T^{yy} + Wi \left(\frac{\partial T^{yy}}{\partial t} + \frac{\partial(uT^{yy})}{\partial x} + \frac{\partial(vT^{yy})}{\partial y} - 2\frac{\partial v}{\partial x} T^{xy} - 2\frac{\partial v}{\partial y} T^{yy} \right) = 2(1 - \beta) \frac{\partial v}{\partial y}, \quad (2.16)$$

$$T^{xy} + Wi \left(\frac{\partial T^{xy}}{\partial t} + \frac{\partial(uT^{xy})}{\partial x} + \frac{\partial(vT^{xy})}{\partial y} - \frac{\partial v}{\partial x} T^{xx} - \frac{\partial u}{\partial y} T^{yy} \right) = (1 - \beta) \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right), \quad (2.17)$$

onde $\mathbf{T} = T^{xx} \mathbf{i}\mathbf{i}^t + T^{xy} (\mathbf{i}\mathbf{j}^t + \mathbf{j}\mathbf{i}^t) + T^{yy} \mathbf{j}\mathbf{j}^t$, em que \mathbf{i} e \mathbf{j} são os vetores da base canônica do sistema de coordenadas cartesiano bidimensional. Quando o escoamento é Newtoniano ($\beta = 1$), o tensor \mathbf{T} é nulo, de modo que apenas as equações (2.12)-(2.14) precisam ser resolvidas.

2.2 Formulação tensão natural

Nas equações (2.12)-(2.17) o tensor tensão \mathbf{T} foi escrito em relação a base canônica do sistema de coordenadas cartesiano. Neste trabalho, iremos utilizar também uma formulação mais recente que muda esta base e, consequentemente, altera as equações constitutivas

para o tensor. Essa formulação é chamada de Tensão Natural, ou *Natural Stress Formulation* (NSF).

A NSF foi proposta em [40] com a intenção de melhorar o cálculo do tensor tensão em escoamentos cuja geometria possui quinas. A singularidade presente nestas quinas dificulta o cálculo preciso de \mathbf{T} e a estratégia NSF busca atacar este problema. Um escoamento bastante estudado na literatura que possui este tipo de singularidade é o escoamento em uma geometria com contração, que será apresentado no capítulo de resultados numéricos deste trabalho.

A proposta desta formulação é escrever o tensor em uma base alinhada às linhas de corrente do escoamento. Os passos a seguir são baseados na construção feita em [40].

Introduziremos inicialmente o tensor configuração \mathbf{A} , definido pela relação

$$\mathbf{T} = \frac{(1 - \beta)}{Wi}(\mathbf{A} - \mathbf{I}), \quad (2.18)$$

e, sendo assim, a equação constitutiva (2.10) se torna

$$Wi \overset{\nabla}{\mathbf{A}} + (\mathbf{A} - \mathbf{I}) = 0, \quad (2.19)$$

após usar que $\overset{\nabla}{\mathbf{I}} = -2\mathbf{D}$. Agora, seguindo o proposto em [40], iremos escrever \mathbf{A} em função do vetor velocidade \mathbf{u} e um vetor ortogonal \mathbf{w} da seguinte forma

$$\mathbf{A} = \lambda \mathbf{u}\mathbf{u}^t + \mu(\mathbf{u}\mathbf{w}^t + \mathbf{w}\mathbf{u}^t) + \nu \mathbf{w}\mathbf{w}^t, \quad (2.20)$$

onde $\mathbf{u} = (u, v)^t$ e $\mathbf{w} = \frac{1}{|\mathbf{u}|^2}(-v, u)^t$, com \mathbf{w} escolhido de forma que $|\mathbf{u} \times \mathbf{w}| = 1$. Substituindo (2.20) em (2.19) encontramos então que as variáveis NSF λ , μ e ν satisfazem as seguintes equações

$$Wi \left[\frac{\partial \lambda}{\partial t} + \frac{2\lambda}{|\mathbf{u}|} \frac{\partial |\mathbf{u}|}{\partial t} + \frac{2\mu}{|\mathbf{u}|^4} \left(v \frac{\partial u}{\partial t} - u \frac{\partial v}{\partial t} \right) + (\mathbf{u} \cdot \nabla) \lambda + 2\mu \nabla \cdot \mathbf{w} \right] + \lambda - \frac{1}{|\mathbf{u}|^2} = 0, \quad (2.21)$$

$$Wi \left[\frac{\partial \mu}{\partial t} + \left(\lambda - \frac{\nu}{|\mathbf{u}|^4} \right) \left(u \frac{\partial v}{\partial t} - v \frac{\partial u}{\partial t} \right) + (\mathbf{u} \cdot \nabla) \mu + \nu \nabla \cdot \mathbf{w} \right] + \mu = 0, \quad (2.22)$$

$$Wi \left[\frac{\partial \nu}{\partial t} + \frac{2\nu}{|\mathbf{u}|} \frac{\partial |\mathbf{u}|}{\partial t} + 2\mu \left(u \frac{\partial v}{\partial t} - v \frac{\partial u}{\partial t} \right) + (\mathbf{u} \cdot \nabla) \nu \right] + \nu - |\mathbf{u}|^2 = 0, \quad (2.23)$$

onde

$$\nabla \cdot \mathbf{w} = \frac{1}{|\mathbf{u}|^4} \left[(v^2 - u^2) \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right) + 4uv \frac{\partial u}{\partial x} \right]. \quad (2.24)$$

As equações (2.21)-(2.23) sugerem uma redefinição das variáveis NSF da seguinte forma,

$$\lambda = \frac{\hat{\lambda}}{|\mathbf{u}|^2}, \quad \mu = \hat{\mu}, \quad \nu = \hat{\nu} |\mathbf{u}|^2, \quad (2.25)$$

fazendo então com que as equações (2.21)-(2.23) para as novas variáveis se tornem

$$\left[\frac{\partial \hat{\lambda}}{\partial t} + \frac{2\hat{\mu}}{|\mathbf{u}|^2} \left(v \frac{\partial u}{\partial t} - u \frac{\partial v}{\partial t} \right) + |\mathbf{u}|^2 (\mathbf{u} \cdot \nabla) \left(\frac{\hat{\lambda}}{|\mathbf{u}|^2} \right) + 2\hat{\mu} |\mathbf{u}|^2 \nabla \cdot \mathbf{w} \right] + \frac{1}{Wi} (\hat{\lambda} - 1) = 0, \quad (2.26)$$

$$\left[\frac{\partial \hat{\mu}}{\partial t} + \left(\frac{\hat{\lambda} - \hat{\nu}}{|\mathbf{u}|^2} \right) \left(u \frac{\partial v}{\partial t} - v \frac{\partial u}{\partial t} \right) + (\mathbf{u} \cdot \nabla) \hat{\mu} + \hat{\nu} |\mathbf{u}|^2 \nabla \cdot \mathbf{w} \right] + \frac{\hat{\mu}}{Wi} = 0, \quad (2.27)$$

$$\left[\frac{\partial \hat{\nu}}{\partial t} + \frac{2\hat{\mu}}{|\mathbf{u}|^2} \left(u \frac{\partial v}{\partial t} - v \frac{\partial u}{\partial t} \right) + \frac{1}{|\mathbf{u}|^2} (\mathbf{u} \cdot \nabla) (\hat{\nu} |\mathbf{u}|^2) \right] + \frac{1}{Wi} (\hat{\nu} - 1) = 0, \quad (2.28)$$

em que

$$|\mathbf{u}|^2 \nabla \cdot \mathbf{w} = \frac{1}{|\mathbf{u}|^2} \left[(v^2 - u^2) \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right) + 4uv \frac{\partial u}{\partial x} \right]. \quad (2.29)$$

Substituindo (2.20) e (2.25) em (2.18), encontramos as seguintes equações que relacionam as variáveis NSF $(\hat{\lambda}, \hat{\mu}, \hat{\nu})$ com as cartesianas (T^{xx}, T^{xy}, T^{yy}) :

$$T^{xx} = \frac{(1 - \beta)}{Wi} \left[-1 + \frac{1}{|\mathbf{u}|^2} \left(\hat{\lambda}u^2 - 2\hat{\mu}uv + \hat{\nu}v^2 \right) \right], \quad (2.30)$$

$$T^{xy} = \frac{(1 - \beta)}{Wi} \left[\hat{\lambda}uv + \hat{\mu}(u^2 - v^2) - \hat{\nu}uv \right], \quad (2.31)$$

$$T^{yy} = \frac{(1 - \beta)}{Wi} \left[-1 + \frac{1}{|\mathbf{u}|^2} \left(\hat{\lambda}v^2 + 2\hat{\mu}uv + \hat{\nu}u^2 \right) \right], \quad (2.32)$$

sendo que as relações inversas são dadas por

$$\hat{\lambda} - 1 = \frac{Wi}{(1 - \beta)|\mathbf{u}|^2} \left[u^2 T^{xx} + 2uv T^{xy} + v^2 T^{yy} \right], \quad (2.33)$$

$$\hat{\mu} = \frac{Wi}{(1 - \beta)|\mathbf{u}|^2} \left[-uv T^{xx} + (u^2 - v^2) T^{xy} + uv T^{yy} \right], \quad (2.34)$$

$$\hat{\nu} - 1 = \frac{Wi}{(1 - \beta)|\mathbf{u}|^2} \left[v^2 T^{xx} + 2uv T^{xy} + u^2 T^{yy} \right]. \quad (2.35)$$

Portanto, quando um escoamento viscoelástico é modelado pela formulação NSF, devemos combinar as equações de conservação (2.8)-(2.9) com as novas equações constitutivas (2.26)-(2.28) usando as relações dadas em (2.30)-(2.32).

2.3 Condições iniciais e de contorno

Neste trabalho, condições iniciais apropriadas são escolhidas de acordo com o problema a ser resolvido. Por exemplo, no caso de escoamentos confinados (sem superfícies livres), a condição inicial é dada como um campo de velocidade \mathbf{u} , pressão p e tensor \mathbf{T} , que devem ser conhecidos em todo o domínio no tempo $t = 0$.

As condições de contorno são equações fornecidas nas fronteiras que delimitam o domínio computacional e que devem ser escolhidas adequadamente de acordo com o fenômeno físico que se deseja reproduzir. São descritos abaixo os tipos de condições de contorno que são considerados neste trabalho.

2.3.1 Paredes rígidas

Para modelar paredes rígidas e impermeáveis presentes no domínio, utilizamos condições de contorno do tipo *no-slip*. Estas são dadas pelas equações

$$\begin{cases} u_n = 0, \\ u_m = u_f, \end{cases} \quad (2.36)$$

em que u_n é a componente da velocidade normal a parede, u_m é a componente tangencial e u_f é a velocidade com a qual a parede se move, caso haja movimento.

2.3.2 Entrada de fluido (*inflow*)

As fronteiras pelas quais o fluido é injetado no domínio recebem condições de contorno modeladas pelas seguintes equações

$$\begin{cases} u_n = u_i, \\ u_m = 0, \end{cases} \quad (2.37)$$

onde u_i é a velocidade prescrita com a qual o fluido é injetado.

As condições no *inflow* para \mathbf{T} são criadas considerando a solução completamente desenvolvida em um canal de placas paralelas. Em um escoamento que acontece na direção x , por exemplo, adotaremos

$$T^{xx} = 2(1 - \beta)Wi \left(\frac{\partial u}{\partial y} \right)^2, \quad T^{xy} = (1 - \beta) \left(\frac{\partial u}{\partial y} \right), \quad T^{yy} = 0. \quad (2.38)$$

Analogamente, as condições no *inflow* para as variáveis NSF λ , μ e ν , obtidas das equações (2.33)-(2.35), são dadas por

$$\hat{\lambda} = \frac{Wi}{(1 - \beta)} T^{xx} + 1, \quad \hat{\mu} = \frac{Wi}{(1 - \beta)} T^{xy}, \quad \hat{\nu} = \frac{Wi}{(1 - \beta)} T^{yy} + 1. \quad (2.39)$$

2.3.3 Saída de fluido (*outflow*)

Consideramos que nas fronteiras pelas quais o fluido sai do domínio, essa ejeção é feita de forma espontânea, isto é, não há nenhuma força que “puxa” o fluido para fora. Esse comportamento é modelado pela condição de Neumann homogênea, dada por

$$\begin{cases} \frac{\partial u_n}{\partial \mathbf{n}} = 0, \\ \frac{\partial u_m}{\partial \mathbf{n}} = 0, \\ \frac{\partial \mathbf{T}}{\partial \mathbf{n}} = \mathbf{0}, \end{cases} \quad (2.40)$$

em que \mathbf{n} é o vetor normal a superfície.

2.3.4 Superfície livre

Em escoamentos com superfície livre, isto é, nos quais existe contato entre o fluido e o ambiente externo, é aplicada uma condição de fronteira dinâmica sobre a superfície livre. Seguindo as ideias usadas também em [29], essa condição é modelada pelas condições de tensão normal e tangencial, dadas respectivamente por

$$\mathbf{n} \cdot (\boldsymbol{\sigma} \cdot \mathbf{n}) = \gamma \kappa, \quad (2.41)$$

$$\mathbf{m} \cdot (\boldsymbol{\sigma} \cdot \mathbf{n}) = 0, \quad (2.42)$$

em que \mathbf{n} é o vetor normal a superfície, \mathbf{m} o vetor tangencial, γ é o coeficiente de tensão superficial, κ é a curvatura da superfície livre, enquanto que o tensor total de tensões $\boldsymbol{\sigma}$ é dado por

$$\boldsymbol{\sigma} = \boldsymbol{\tau} - p\mathbf{I} = 2\eta_s \mathbf{D} + \boldsymbol{\tau}^p - p\mathbf{I}. \quad (2.43)$$

Iremos adimensionalizar as equações (2.41)-(2.42) usando os mesmos valores de escala dados na equação (2.7), além de

$$\kappa = \frac{1}{L} \bar{\kappa}. \quad (2.44)$$

Desta forma, as equações (2.41)-(2.42) se tornam

$$\mathbf{n} \cdot (\bar{\boldsymbol{\sigma}} \cdot \mathbf{n}) = \frac{Re}{We} \bar{\kappa}, \quad (2.45)$$

$$\mathbf{m} \cdot (\bar{\boldsymbol{\sigma}} \cdot \mathbf{n}) = 0, \quad (2.46)$$

em que

$$\bar{\boldsymbol{\sigma}} = 2\beta\bar{\mathbf{D}} + \bar{\mathbf{T}} - Re \cdot \bar{p}\mathbf{I}, \quad (2.47)$$

e We é o número de Weber dado por

$$We = \frac{\rho U^2 L}{\gamma}. \quad (2.48)$$

A partir deste momento, iremos abandonar as barras superiores usadas nas variáveis adimensionais destas equações.

Em coordenadas cartesianas, as equações (2.45) e (2.46) são respectivamente escritas como

$$p = \frac{2\beta}{Re} \left[n_x^2 \frac{\partial u}{\partial x} + n_x n_y \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) + n_y^2 \frac{\partial v}{\partial y} \right] + \frac{1}{Re} [n_x^2 T^{xx} + 2n_x n_y T^{xy} + n_y^2 T^{yy}] + \frac{1}{We} \kappa, \quad (2.49)$$

$$\beta \left[2n_x n_y \left(\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x} \right) + (n_x^2 - n_y^2) \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right) \right] = n_x n_y (T^{xx} - T^{yy}) + (n_y^2 - n_x^2) T^{xy}, \quad (2.50)$$

em que $\mathbf{n} = (n_x, n_y)$ é o vetor normal unitário e exterior a região de fluido.

Quando o escoamento considerado é puramente Newtoniano, as componentes do tensor \mathbf{T} são sempre nulas e podem ser eliminadas das equações (2.49)-(2.50).

Método numérico

Nesta seção detalharemos a construção do algoritmo baseado no método de projeção para resolver as equações de Navier-Stokes em escoamentos com superfície livre.

3.1 Método de projeção

Diversos tipos de métodos já foram propostos para solucionar as equações de conservação de massa e movimento. Um tipo de método popular é aquele capaz de desacoplar as incógnitas \mathbf{u} e p das equações (2.8) e (2.9), como por exemplo os métodos de projeção.

A motivação para se realizar um desacoplamento vem da dificuldade de se resolver um sistema envolvendo as equações (2.8)-(2.9) simultaneamente. Métodos desacoplados, em geral, apresentam equações mais simples e que podem ser resolvidas separadamente, uma após a outra.

Neste trabalho, as equações sempre serão resolvidas através do método de projeção. A formulação do método baseia-se no teorema de Helmholtz-Hodge, cuja demonstração pode ser vista em [9]. Este teorema garante que, sendo $\tilde{\mathbf{u}}$ um campo vetorial definido em uma região Ω com fronteira suave $\partial\Omega$, então a decomposição $\tilde{\mathbf{u}}$ na forma

$$\tilde{\mathbf{u}} = \mathbf{u} + \nabla\psi \quad (3.1)$$

existe e é única, com $\nabla \cdot \mathbf{u} = 0$ e ψ um campo escalar também definido em Ω .

A primeira etapa do método de projeção consiste em resolver a equação de quantidade de movimento (2.8) para um campo de velocidade intermediário $\tilde{\mathbf{u}}$. Para desacoplar os campos de velocidade e pressão, realiza-se a aproximação da incógnita p por uma pressão tentativa \tilde{p} que já deve ser conhecida. Sendo assim, queremos resolver a seguinte equação considerando um valor já conhecido de \mathbf{T} , que comentaremos oportunamente,

$$\frac{\partial \tilde{\mathbf{u}}}{\partial t} + \nabla \cdot (\tilde{\mathbf{u}}\tilde{\mathbf{u}}) = -\nabla\tilde{p} + \frac{\beta}{Re}\nabla^2\tilde{\mathbf{u}} + \frac{1}{Re}\nabla \cdot \mathbf{T} + \frac{1}{Fr^2}\mathbf{g}. \quad (3.2)$$

Resolvendo a equação (3.2), obtemos uma velocidade intermediária $\tilde{\mathbf{u}}$ para a qual, em geral, $\nabla \cdot \tilde{\mathbf{u}} \neq 0$ (pois desconsideramos a equação de incompressibilidade). Para conseguir então uma atualização para a velocidade \mathbf{u} que satisfaça a condição de incompressibilidade utiliza-se a decomposição de Helmholtz-Hodge. Note que, para calcular \mathbf{u} através da decomposição (3.1), precisamos ainda determinar o valor do campo escalar ψ .

Isolando a incógnita \mathbf{u} na equação (3.1) obtemos

$$\mathbf{u} = \tilde{\mathbf{u}} - \nabla\psi. \quad (3.3)$$

Aplicando o operador divergente em (3.3) e utilizando a restrição de incompressibilidade (2.9), encontramos a seguinte equação que será resolvida para calcular o valor de ψ :

$$\nabla^2 \psi = \nabla \cdot \tilde{\mathbf{u}}. \quad (3.4)$$

Sendo assim, o segundo passo do método de Projeção é a resolução da equação de Poisson (3.4).

Note que, para encontrar a solução das equações (3.2) e (3.4) são necessárias condições de contorno para as variáveis artificiais $\tilde{\mathbf{u}}$ e ψ .

Para a velocidade artificial $\tilde{\mathbf{u}}$, consideramos que as equações que eram válidas para \mathbf{u} no contorno também serão válidas para $\tilde{\mathbf{u}}$. Portanto, apenas aplicamos as condições que foram descritas na seção 2.3 na incógnita $\tilde{\mathbf{u}}$.

Para a incógnita ψ , segundo [47], as condições de contorno em fronteiras rígidas e injetores são

$$\frac{\partial \psi}{\partial \mathbf{n}} = 0, \quad (3.5)$$

e, em ejetores (*outflow*),

$$\psi = 0. \quad (3.6)$$

A condição de contorno para a equação de Poisson sobre a superfície livre será deduzida na seção 3.6.

Nas seções seguintes iremos descrever mais detalhadamente a construção do método numérico que implementa a resolução destas equações.

3.2 Discretização do domínio

O domínio será discretizado com uma malha computacional não-uniforme. Definiremos dois valores N_x e N_y , que representam:

- N_x : a quantidade de intervalos nos quais o domínio será dividido na direção x ;
- N_y : a quantidade de intervalos nos quais o domínio será dividido na direção y .

Sejam x_0, x_1, \dots, x_{N_x} os pontos discretizados na direção x e y_0, y_1, \dots, y_{N_y} os da direção y , definiremos também os seguintes espaçamentos:

$$\Delta_{x_i} = x_i - x_{i-1}, \quad i = 1, 2, \dots, N_x \quad \text{e} \quad (3.7)$$

$$\Delta_{y_j} = y_j - y_{j-1}, \quad j = 1, 2, \dots, N_y. \quad (3.8)$$

Com a malha criada, precisamos agora definir uma identificação para cada uma das células e, em seguida, para cada um dos pontos que iremos utilizar. Vamos identificar cada uma das células por dois índices i e j que representam, respectivamente, a posição horizontal e vertical da célula da malha começando por *zero*.

Com os índices de cada células definidos, podemos nomear alguns pontos que serão utilizados mais adiante. Chamaremos os pontos no centro de cada célula (i, j) da malha de $\mathbf{x}_{i,j}$, ou seja:

$$\mathbf{x}_{i,j} = \left(x_i + \frac{1}{2} \Delta_{x_i}, y_j + \frac{1}{2} \Delta_{y_j} \right). \quad (3.9)$$

Analogamente, vamos nomear os pontos que se localizam na face vertical na esquerda de cada célula de $\mathbf{x}_{i-\frac{1}{2},j}$, ou seja

$$\mathbf{x}_{i-\frac{1}{2},j} = \left(x_i, y_j + \frac{1}{2} \Delta_{y_j} \right). \quad (3.10)$$

Da mesma forma, os pontos das faces horizontais serão

$$\mathbf{x}_{i,j-\frac{1}{2}} = \left(x_i + \frac{1}{2}\Delta_{x_i}, y_j \right). \quad (3.11)$$

A Figura 3.1 mostra uma célula computacional (i, j) com os seus respectivos pontos de interesse como foram definidos acima.

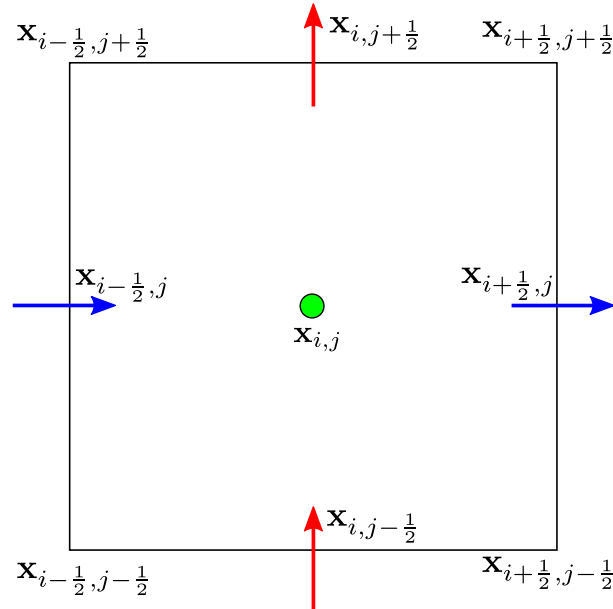


Figura 3.1: Célula computacional i, j com seus pontos de interesse nomeados.

Iremos agora decidir quais são os pontos discretizados nos quais cada uma das incógnitas será calculada. Teremos aqui cinco tipos de propriedades de interesse, sendo eles:

- u : componente da velocidade na direção x ;
- v : componente da velocidade na direção y ;
- p : pressão;
- T^{xx} , T^{xy} , T^{yy} , λ , μ , ν : componentes do tensor tensão (usando a formulação Cartesiana e a NSF);
- ψ : campo escalar usado na decomposição de Helmholtz-Hodge.

Neste caso, iremos utilizar uma malha deslocada para posicionar as 4 propriedades, isto é, elas não serão todas calculadas nos mesmos pontos. Os pontos nos quais cada propriedade será calculada são os seguintes:

- u : a velocidade na direção x será calculada nos pontos $\mathbf{x}_{i-\frac{1}{2},j}$;
- v : a componente na direção y é calculada nos pontos $\mathbf{x}_{i,j-\frac{1}{2}}$;
- p e ψ : ambas serão calculadas nos centros $\mathbf{x}_{i,j}$.
- T^{xx} , T^{xy} , T^{yy} , λ , μ , ν : todas também serão calculadas nos centros $\mathbf{x}_{i,j}$.

Nas seções seguintes, os pontos desta malha computacional serão usados para aproximar diversas derivadas por fórmulas de diferenças finitas. A criação destas fórmulas para uma malha não-uniforme é explicada no apêndice A.

3.3 Malha computacional

Vamos agora descrever como a superfície livre é utilizada em conjunto com a malha que foi criada anteriormente na discretização do domínio. A estratégia utilizada neste trabalho é baseada no método GENSMAC, que é descrito com detalhes em [29].

Na representação *Front-Tracking*, a superfície livre é representada por um conjunto de partículas conectadas localizadas no plano Cartesiano (ver Capítulo 4 para mais detalhes). A partir da posição destas partículas iremos classificar cada célula da malha computacional da seguinte forma:

- EMPTY: uma célula é classificada como EMPTY se ela não possui nenhuma partícula em seu interior e se está localizada fora da região que possui fluido;
- SURFACE: são classificadas como SURFACE todas as células que possuem uma partícula marcadora em seu interior e possuem ao menos uma célula EMPTY ao seu redor;
- FULL: são células que não possuem nenhuma célula EMPTY ao lado de suas quatro faces, ou seja, estão dentro da região que possui fluido.

A figura 3.2 mostra um exemplo de interface e a classificação das células da malha computacional de acordo com as regras descritas acima.

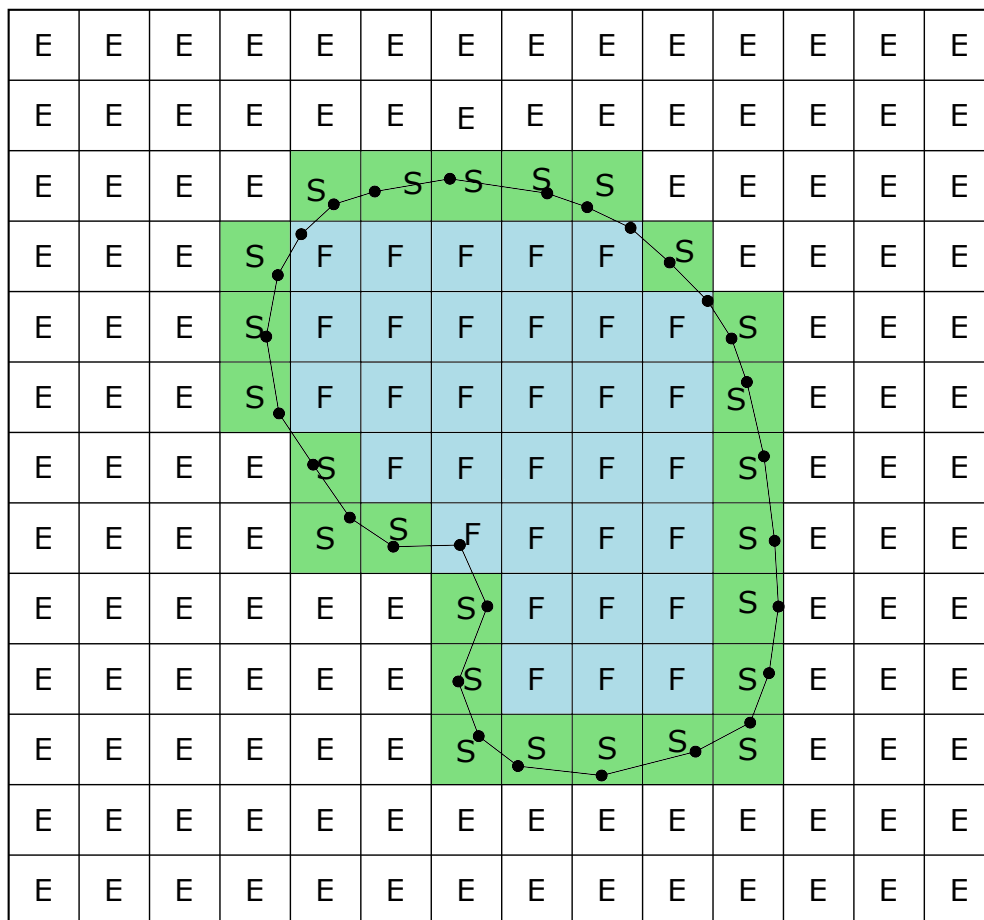


Figura 3.2: Exemplo da classificação de células a partir de uma superfície livre.

Esta classificação de células irá determinar que tipo de equação será utilizado em cada ponto da malha. Em pontos localizados dentro de regiões FULL, por exemplo, iremos

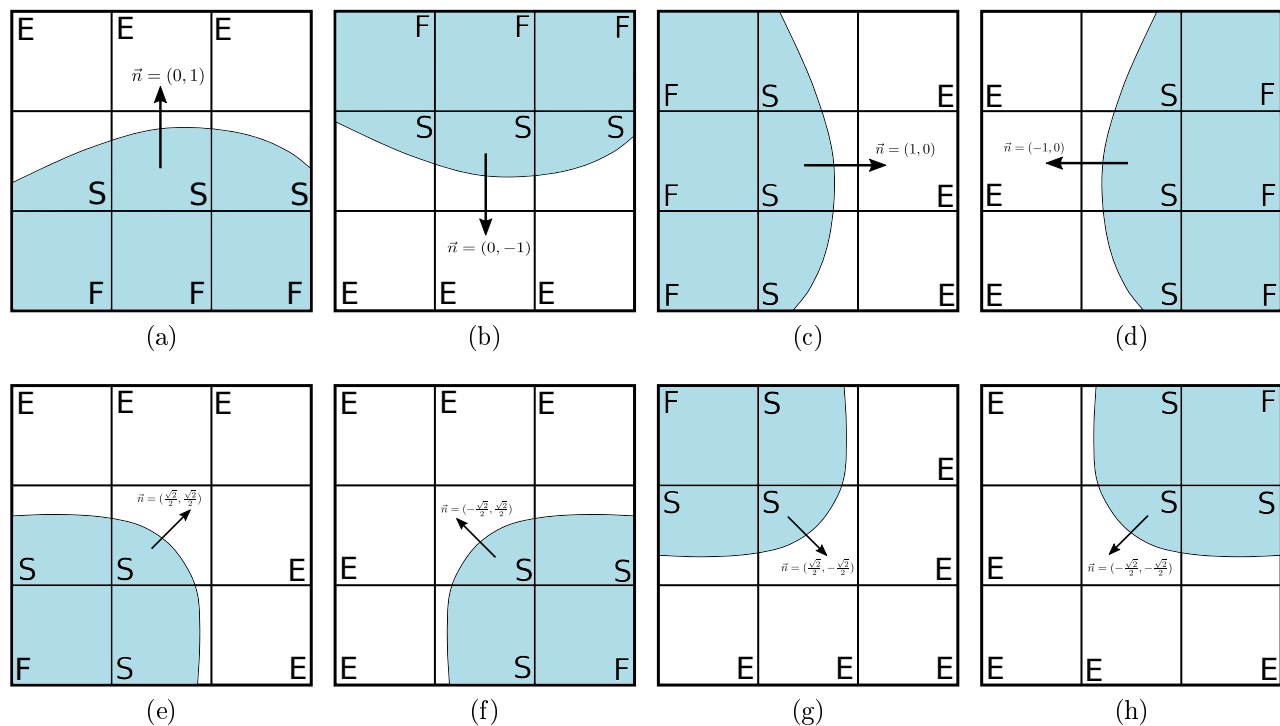


Figura 3.3: Casos que geram todas as oito possibilidades de vetores normais utilizadas no método.

apenas aplicar as equações de momento e Poisson normalmente. Em pontos de regiões EMPTY, nada será feito, já que não há escoamento sobre os mesmos. Já nas células SURFACE, as condições de contorno de superfície livre serão utilizadas.

A aplicação das condições de contorno nas células SURFACE não é uma tarefa trivial e exige cuidado. Até o momento, uma versão inicial deste processo foi implementada com base na proposta utilizada também em [29]. Esta versão simplifica consideravelmente as equações de contorno (2.41)-(2.42) através de uma aproximação grosseira do vetor normal à interface.

A aproximação do vetor normal em uma célula SURFACE é feita a partir da classificação de suas células vizinhas. Isso é feito de modo que apenas oito valores para \mathbf{n} sejam possíveis. A figura 3.3 mostra os casos que geram as oito possibilidades para \mathbf{n} .

As oito opções de vetores normais mostradas na figura 3.3 simplificam drasticamente as equações (2.41)-(2.42), pois vários termos poderão ser ignorados. Contudo, é claro que a precisão do método será um pouco prejudicada, já que vetores normais muito grosseiros serão usados.

No código implementado aqui, não há necessidade de definir células fantasmas que representam as regiões de contorno, como *inflow* e *outflow*. Apenas os pontos que estão exatamente sobre cada contorno serão sinalizados apropriadamente, de modo que não há necessidade de criar células fantasmas em nossa estrutura.

3.4 Discretização temporal da equação de quantidade de movimento

Vamos agora realizar a discretização temporal da equação de transporte (3.2). Neste trabalho, a discretização no tempo foi feita através de um método semi-implícito, isto é, a parte viscosa foi tratada implicitamente via Euler implícito, enquanto que os termos con-

vectivos são tratados explicitamente. Com este método, a versão discretizada da equação (3.2) será

$$\frac{\tilde{\mathbf{u}}^{n+1} - \tilde{\mathbf{u}}^n}{\Delta_t} + \nabla \cdot (\tilde{\mathbf{u}}\tilde{\mathbf{u}})^n = -\nabla \tilde{p}^{n+1} + \frac{\beta}{Re} \nabla^2 \tilde{\mathbf{u}}^{n+1} + \frac{1}{Re} \nabla \cdot \mathbf{T}^n + \frac{1}{Fr^2} \mathbf{g}^{n+1}. \quad (3.12)$$

Para que a única incógnita da equação (3.12) seja $\tilde{\mathbf{u}}^{n+1}$, precisamos aproximar $\tilde{\mathbf{u}}^n$ e \tilde{p}^{n+1} por valores já conhecidos. Iremos tomar $\tilde{\mathbf{u}}^n = \mathbf{u}^n$ e $\tilde{p}^{n+1} = p^n$. Logo, a equação discretizada no tempo que iremos resolver é

$$\frac{\tilde{\mathbf{u}}^{n+1} - \mathbf{u}^n}{\Delta_t} + \nabla \cdot (\mathbf{u}\mathbf{u})^n = -\nabla p^n + \frac{\beta}{Re} \nabla^2 \tilde{\mathbf{u}}^{n+1} + \frac{1}{Re} \nabla \cdot \mathbf{T}^n + \frac{1}{Fr^2} \mathbf{g}^{n+1}. \quad (3.13)$$

Observe que o termo convectivo $\nabla \cdot (\mathbf{u}\mathbf{u})^n$ de (3.13) foi aproximado de forma explícita (ele está no tempo n). Essa aproximação foi adotada apenas para eliminar a necessidade de resolver um sistema de equações não-lineares. Além disso, note que o tensor \mathbf{T} também foi discretizado explicitamente. Isto evita que a equação de momento seja acoplada às equações constitutivas que atualizam o tensor.

3.5 Construção da equação de atualização da pressão

Após a atualização da velocidade é necessário calcular também o novo valor para a pressão, isto é, p^{n+1} . Para encontrar uma fórmula de atualização, considere a equação de momento (2.8) discretizada no tempo, dada por

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta_t} + \nabla \cdot (\mathbf{u}\mathbf{u})^n = -\nabla p^{n+1} + \frac{\beta}{Re} \nabla^2 \mathbf{u}^{n+1} + \frac{1}{Re} \nabla \cdot \mathbf{T}^n + \frac{1}{Fr^2} \mathbf{g}^{n+1}. \quad (3.14)$$

Subtraindo a equação (3.13) de (3.14) temos

$$\frac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}^{n+1}}{\Delta_t} = -\nabla p^{n+1} + \nabla p^n + \frac{\beta}{Re} [\nabla^2 \mathbf{u}^{n+1} - \nabla^2 \tilde{\mathbf{u}}^{n+1}]. \quad (3.15)$$

Finalmente, substituindo $\tilde{\mathbf{u}}$ em (3.15) pela decomposição (3.1), obtemos a seguinte fórmula para atualização da pressão

$$p^{n+1} = p^n + \frac{\psi}{\Delta_t} - \frac{\beta}{Re} \nabla^2 \psi^{n+1}. \quad (3.16)$$

Para evitar a necessidade de discretizar o operador Laplaciano de (3.16), iremos desconsiderar a presença deste termo, como feito em [44]. Portanto, a pressão será atualizada pela seguinte equação aplicada ao ponto $X_{i,j}$

$$p^{n+1} = p^n + \frac{\psi}{\Delta_t}. \quad (3.17)$$

3.6 Condição de contorno para a equação de Poisson sobre a superfície livre

A condição de contorno para ψ na superfície livre será criada com auxílio das equações (3.3), (3.17) e (2.49), de acordo com a metodologia proposta em [34]. Escrevendo a

condição normal (2.49) em coordenadas cartesianas e no tempo t_{n+1} , temos

$$p^{n+1} = \frac{2\beta}{Re} \left[n_x^2 \frac{\partial u^{n+1}}{\partial x} + n_x n_y \left(\frac{\partial u^{n+1}}{\partial y} + \frac{\partial v^{n+1}}{\partial x} \right) + n_y^2 \frac{\partial v^{n+1}}{\partial y} \right] + \frac{1}{Re} \left[n_x^2 T^{xx,n} + 2n_x n_y T^{xy,n} + n_y^2 T^{yy,n} \right] + \frac{1}{We} \kappa, \quad (3.18)$$

sendo que as componentes do tensor tensão foram aproximadas explicitamente no tempo n .

Substituindo (3.17) em (3.18) obtemos

$$p^n + \frac{\psi^{n+1}}{\Delta_t} = \frac{2\beta}{Re} \left[n_x^2 \frac{\partial u^{n+1}}{\partial x} + n_x n_y \left(\frac{\partial u^{n+1}}{\partial y} + \frac{\partial v^{n+1}}{\partial x} \right) + n_y^2 \frac{\partial v^{n+1}}{\partial y} \right] + \frac{1}{Re} \left[n_x^2 T^{xx,n} + 2n_x n_y T^{xy,n} + n_y^2 T^{yy,n} \right] + \frac{1}{We} \kappa, \quad (3.19)$$

e, por fim, as velocidades são substituídas pela decomposição (3.3), gerando a equação

$$p^n + \frac{\psi^{n+1}}{\Delta_t} = \frac{2\beta}{Re} \left[n_x^2 \frac{\partial}{\partial x} \left(\tilde{u}^{n+1} - \frac{\partial \psi^{n+1}}{\partial x} \right) + n_x n_y \left(\frac{\partial}{\partial y} \left(\tilde{u}^{n+1} - \frac{\partial \psi^{n+1}}{\partial x} \right) + \frac{\partial}{\partial x} \left(\tilde{v}^{n+1} - \frac{\partial \psi^{n+1}}{\partial y} \right) \right) + n_y^2 \frac{\partial}{\partial y} \left(\tilde{v}^{n+1} - \frac{\partial \psi^{n+1}}{\partial y} \right) \right] + \frac{1}{Re} \left[n_x^2 T^{xx,n} + 2n_x n_y T^{xy,n} + n_y^2 T^{yy,n} \right] + \frac{1}{We} \kappa. \quad (3.20)$$

A equação (3.20) é utilizada como condição de contorno para a equação de Poisson e, em alguns casos nos quais valores de ψ^{n+1} forem necessários fora do domínio, utiliza-se $\psi^{n+1} = 0$.

3.7 Atualização do tensor tensão

O último passo do método implementado consiste em utilizar as equações constitutivas do modelo Oldroyd-B para calcular o valor de \mathbf{T}^{n+1} . Esse cálculo poderá ser feito de duas formas: utilizando a formulação NSF, ou utilizando a formulação Cartesiana tradicional (CSF).

3.7.1 Atualização pela formulação CSF

Quando uma simulação for feita usando a formulação CSF, o valor de \mathbf{T}^{n+1} será calculado através das equações (2.15)-(2.17). É preciso, é claro, realizar a discretização das mesmas. Como as três componentes (T^{xx} , T^{xy} , T^{yy}) são calculadas nos mesmos pontos, em alguns momentos desta seção elas serão nomeadas apenas como T . Quando isso for feito, estamos indicando que a fórmula de diferenças serve para qualquer uma das três componentes.

A discretização temporal das equações (2.15)-(2.17) será feita pelo método Euler Explícito. Sendo assim, as equações discretizadas no tempo se tornam

$$T^{xx,n+1} = T^{xx,n} + \Delta_t \cdot F_{CSF}^{xx}, \quad (3.21)$$

$$T^{yy,n+1} = T^{yy,n} + \Delta_t \cdot F_{CSF}^{yy}, \quad (3.22)$$

$$T^{xy,n+1} = T^{xy,n} + \Delta_t \cdot F_{CSF}^{xy}, \quad (3.23)$$

onde

$$F_{CSF}^{xx} = -\frac{\partial(u^{n+1}T^{xx,n})}{\partial x} - \frac{\partial(v^{n+1}T^{xx,n})}{\partial y} + 2\frac{\partial u^{n+1}}{\partial x}T^{xx,n} + 2\frac{\partial u^{n+1}}{\partial y}T^{xy,n} + 2\frac{(1-\beta)}{Wi}\frac{\partial u^{n+1}}{\partial x} - \frac{1}{Wi}T^{xx,n}, \quad (3.24)$$

$$F_{CSF}^{yy} = -\frac{\partial(u^{n+1}T^{yy,n})}{\partial x} - \frac{\partial(v^{n+1}T^{yy,n})}{\partial y} + 2\frac{\partial v^{n+1}}{\partial x}T^{xy,n} + 2\frac{\partial v^{n+1}}{\partial y}T^{yy,n} + 2\frac{(1-\beta)}{Wi}\frac{\partial v^{n+1}}{\partial y} - \frac{1}{Wi}T^{yy,n}, \quad (3.25)$$

$$F_{CSF}^{xy} = -\frac{\partial(u^{n+1}T^{xy,n})}{\partial x} - \frac{\partial(v^{n+1}T^{xy,n})}{\partial y} + \frac{\partial v^{n+1}}{\partial x}T^{xx,n} + \frac{\partial u^{n+1}}{\partial y}T^{yy,n} + \frac{(1-\beta)}{Wi}\left(\frac{\partial u^{n+1}}{\partial y} + \frac{\partial v^{n+1}}{\partial x}\right) - \frac{1}{Wi}T^{xy,n}. \quad (3.26)$$

3.7.2 Atualização pela formulação NSF

Se a formulação NSF for escolhida na simulação, precisamos então utilizar as equações (2.26)-(2.28) para atualizar as variáveis λ, μ, ν . Em seguida, o valor de \mathbf{T}^{n+1} é calculado através das relações (2.30)-(2.32).

Realizando a discretização temporal de (2.26)-(2.28) pelo método Euler Explícito, obtemos:

$$\hat{\lambda}^{n+1} = \hat{\lambda}^n + \Delta t \cdot F_{NSF}^{\hat{\lambda}}, \quad (3.27)$$

$$\hat{\mu}^{n+1} = \hat{\mu}^n + \Delta t \cdot F_{NSF}^{\hat{\mu}}, \quad (3.28)$$

$$\hat{\nu}^{n+1} = \hat{\nu}^n + \Delta t \cdot F_{NSF}^{\hat{\nu}}, \quad (3.29)$$

onde

$$F_{NSF}^{\hat{\lambda}} = -\frac{2\hat{\mu}^n}{(|\mathbf{u}^{n+1}|^2 + tol)} \left(v^{n+1} \frac{\partial u^{n+1}}{\partial t} - u^{n+1} \frac{\partial v^{n+1}}{\partial t} \right) - |\mathbf{u}^{n+1}|^2 \left(\frac{\partial(u^{n+1}\lambda^n)}{\partial x} + \frac{\partial(v^{n+1}\lambda^n)}{\partial y} \right) - 2\hat{\mu}^n |\mathbf{u}^{n+1}|^2 \nabla \cdot \mathbf{w}^{n+1} - \frac{1}{Wi} (\hat{\lambda}^n - 1), \quad (3.30)$$

onde $\lambda^n = \frac{\hat{\lambda}^n}{(|\mathbf{u}^{n+1}|^2 + tol)}$ é usado no termo convectivo;

$$F_{NSF}^{\hat{\mu}} = -\left(\frac{\hat{\lambda}^n - \hat{\nu}^n}{|\mathbf{u}^{n+1}|^2 + tol} \right) \left(u^{n+1} \frac{\partial v^{n+1}}{\partial t} - v^{n+1} \frac{\partial u^{n+1}}{\partial t} \right) - \left(\frac{\partial(u^{n+1}\hat{\mu}^n)}{\partial x} + \frac{\partial(v^{n+1}\hat{\mu}^n)}{\partial y} \right) - \hat{\nu}^n |\mathbf{u}^{n+1}|^2 \nabla \cdot \mathbf{w}^{n+1} - \frac{\hat{\mu}^n}{Wi}, \quad (3.31)$$

$$F_{NSF}^{\hat{\nu}} = -\frac{2\hat{\mu}^n}{(|\mathbf{u}^{n+1}|^2 + tol)} \left(u^{n+1} \frac{\partial v^{n+1}}{\partial t} - v^{n+1} \frac{\partial u^{n+1}}{\partial t} \right) - \frac{1}{(|\mathbf{u}^{n+1}|^2 + tol)} \left(\frac{\partial(u^{n+1}\nu^n)}{\partial x} + \frac{\partial(v^{n+1}\nu^n)}{\partial y} \right) - \frac{1}{Wi} (\hat{\nu}^n - 1), \quad (3.32)$$

sendo que

$$|\mathbf{u}^{n+1}|^2 \nabla \cdot \mathbf{w}^{n+1} = \frac{1}{(|\mathbf{u}^{n+1}|^2 + tol)} \left[((v^{n+1})^2 - (u^{n+1})^2) \left(\frac{\partial v^{n+1}}{\partial x} + \frac{\partial u^{n+1}}{\partial y} \right) + 4u^{n+1}v^{n+1} \frac{\partial u^{n+1}}{\partial x} \right]. \quad (3.33)$$

Observe que, nas equações (3.27)-(3.29) foram adicionados termos tol em alguns dos denominadores. Isto foi feito pois os vetores da base (2.20) se degeneram em regiões onde

o campo de velocidade se torna nulo, tipicamente perto de paredes rígidas e pontos de estagnação. Para regularizar alguns termos, adicionamos uma constante tol em locais onde uma divisão por zero pode ocorrer.

3.8 Resumo do método numérico

Utilizando todas as equações deduzidas até aqui, podemos então resumir o método de projeção utilizado neste trabalho com as seguintes etapas

1. Calcular a velocidade em pontos ao redor das células SURFACE com as equações de superfície livre (2.41)-(2.42);
2. Calcular a velocidade intermediária $\tilde{\mathbf{u}}$ resolvendo a equação (3.13) com suas derivadas aproximadas. Os valores vindos da etapa 1 são usados como condição de contorno aqui;
3. Calcular o campo escalar ψ resolvendo a equação de Poisson (3.4) com condições de contornos apropriadas ao problema, isto é, seguindo as equações (3.5), (3.6) ou (3.20);
4. Calcular \mathbf{u}^{n+1} pela fórmula de atualização (3.3);
5. Calcular p^{n+1} pela fórmula de atualização (3.17);
6. Atualizar o tensor \mathbf{T} utilizando a formulação escolhida (CSF ou NSF) e suas respectivas equações ((3.21)-(3.23) ou (3.27)-(3.29));
7. Advectar as partículas da interface com a nova velocidade através da equação

$$\dot{\mathbf{x}} = \mathbf{u}. \quad (3.34)$$

As etapas acima devem ser executadas para cada passo temporal, de modo que a velocidade e pressão seguem sendo atualizadas de forma iterativa.

Tratamento numérico da superfície livre

Neste capítulo será descrito o estudo e implementação de um método numérico para representação explícita da interface (Front-Tracking) que seja capaz de lidar com mudanças topológicas e deformações em escoamentos complexos. O estudo toma como base os trabalhos de Glimm e colaboradores [14, 15]. Além disso, vamos apresentar a estratégia utilizada para aproximar a curvatura da interface, necessária para a aplicação da tensão superficial.

4.1 Representação e advecção de interfaces pelo método Front-Tracking

Nesta seção descreveremos como é feita a representação de uma interface pelo método Front-Tracking. A estrutura de dados apresentada aqui busca facilitar a implementação do método de mudança topológica que será apresentado na seção 4.2.

4.1.1 Representação Front-Tracking

O método Front-Tracking tem como sua principal característica a representação de interfaces de forma explícita, isto é, o método armazena diretamente as coordenadas no plano (ou espaço) nas quais a interface está localizada.

Este estudo trata apenas da representação de interfaces bidimensionais e, sendo assim, todas as construções feitas a partir de agora serão realizadas sobre o plano cartesiano.

Definição 1 *Uma interface é um conjunto de uma ou mais curvas que divide o plano cartesiano em um número finito de regiões.*

A Figura 4.1 apresenta exemplos de interfaces, note que:

- 4.1a contém uma única curva que divide o plano em duas regiões r_1 e r_2 . A curva é a circunferência que realiza esta divisão.
- 4.1b contém duas curvas que dividem o domínio em três regiões.
- 4.1c mostra uma interface com uma única curva, mas com uma forma mais elaborada;
- 4.1d contém 3 curvas e 4 regiões. As curvas de uma interface podem ter diferentes formas.

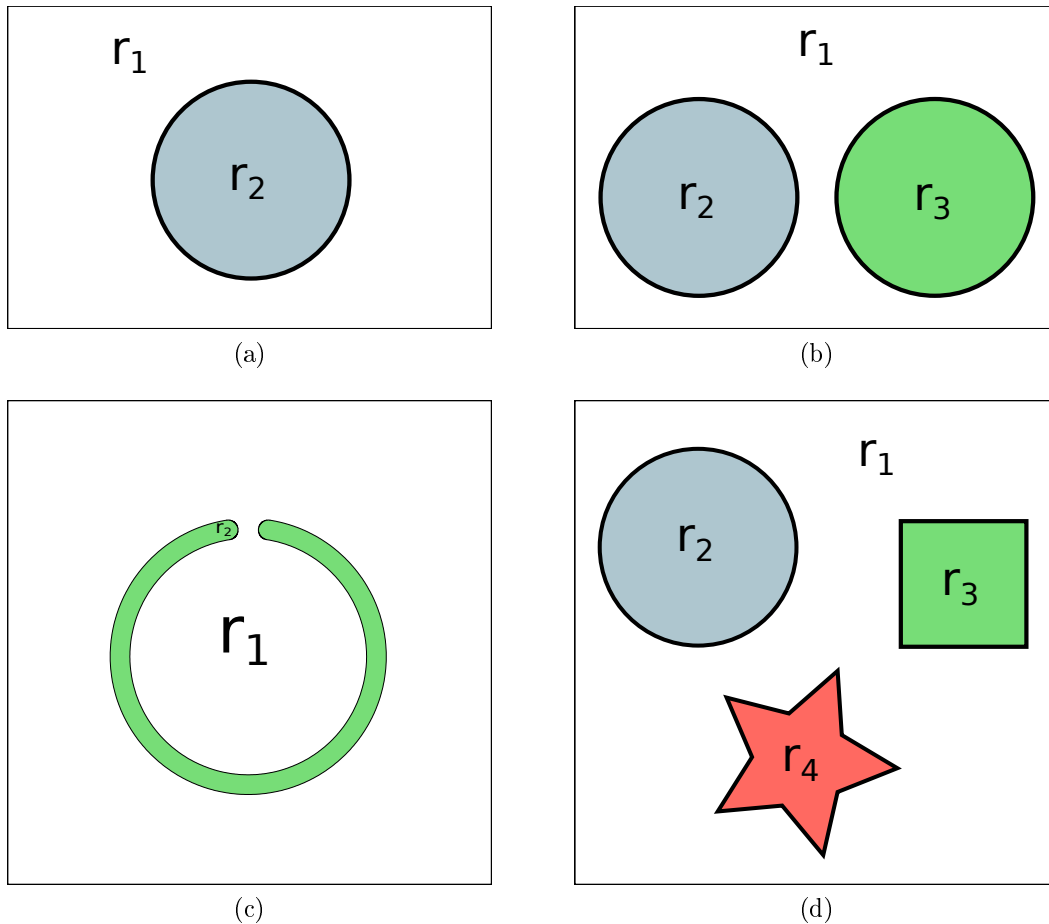


Figura 4.1: Exemplos de interfaces com diferentes formas e quantidades de curvas.

Sabemos que uma curva no plano cartesiano é definida de forma contínua, isto é, é formada por uma quantidade infinita de pontos. Como o computador não é capaz de armazenar quantidades infinitas, o método Front-Tracking realiza uma discretização das curvas e armazena apenas um número finito de pontos. Desta forma uma curva será vista como um conjunto finito de pontos conectados por segmentos de retas.

A Figura 4.2 mostra uma possível representação Front-Tracking das mesmas interfaces utilizadas como exemplo na Figura 4.1. Observe que, devido ao método utilizar uma quantidade finita de pontos e segmentos, a interface Front-Tracking acaba perdendo o formato da interface original. Utilizar uma quantidade maior de pontos torna a discretização mais precisa, porém, aumenta o custo computacional e o espaço de armazenamento exigido pelo método.

Devido ao algoritmo de mudanças topológicas que será visto na seção 4.2, vamos também considerar neste trabalho que as curvas possuem um sentido. Ou seja, será importante definir qual é o ponto inicial da curva Front-Tracking e qual é a ordenação dos pontos seguintes. A Figura 4.3, por exemplo, mostra uma curva com o formato de circunferência e que possui sentido horário. Observe que, o que define o sentido da curva é a ordem com a qual os pontos serão armazenados; se a ordem fosse invertida, o sentido seria anti-horário.

Observe que, cada curva fechada, sempre divide uma região do plano em duas sub-regiões. Como acabamos de definir que a curva possui um sentido, podemos agora dizer que uma destas duas sub-regiões se localiza no lado esquerdo da curva, e a outra se localiza no lado direito. Olhando novamente a Figura 4.3, note que a região r_1 está do lado esquerdo da curva e r_2 está do lado direito.

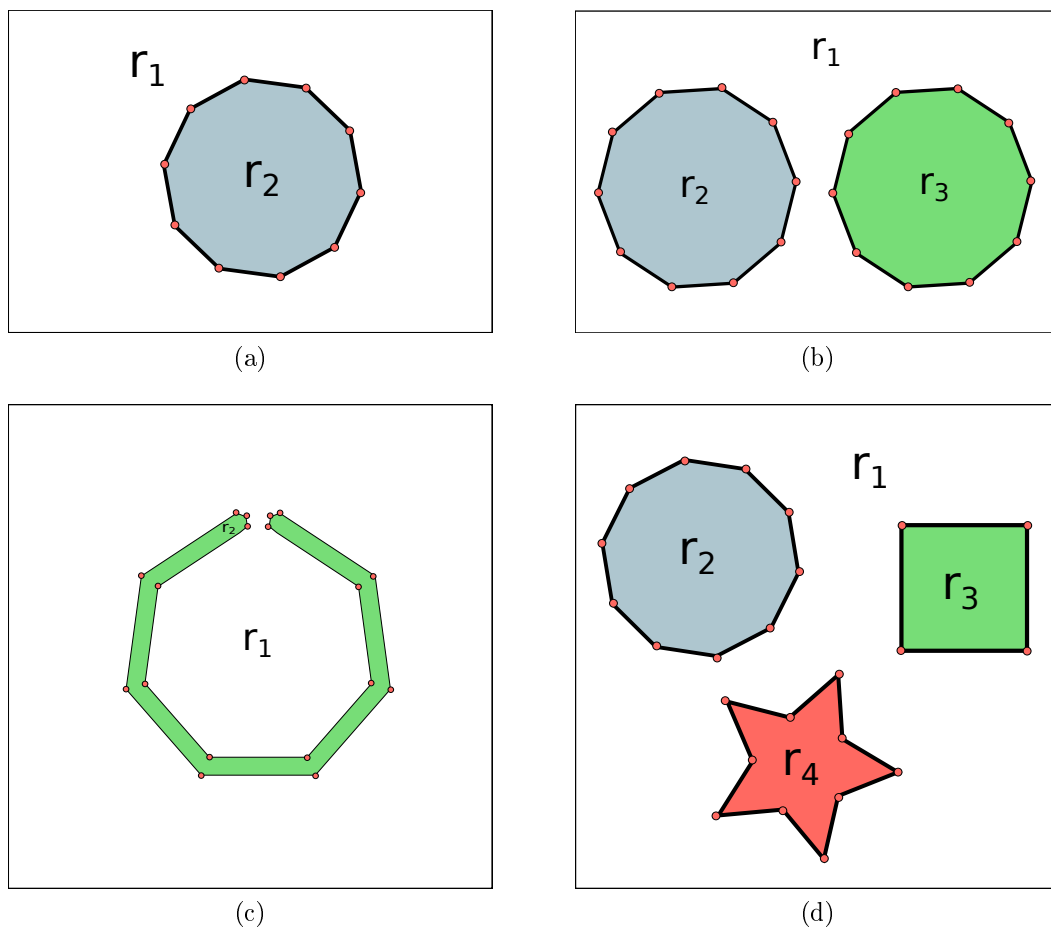


Figura 4.2: Representação Front-Tracking das interfaces apresentadas na Figura 4.1.

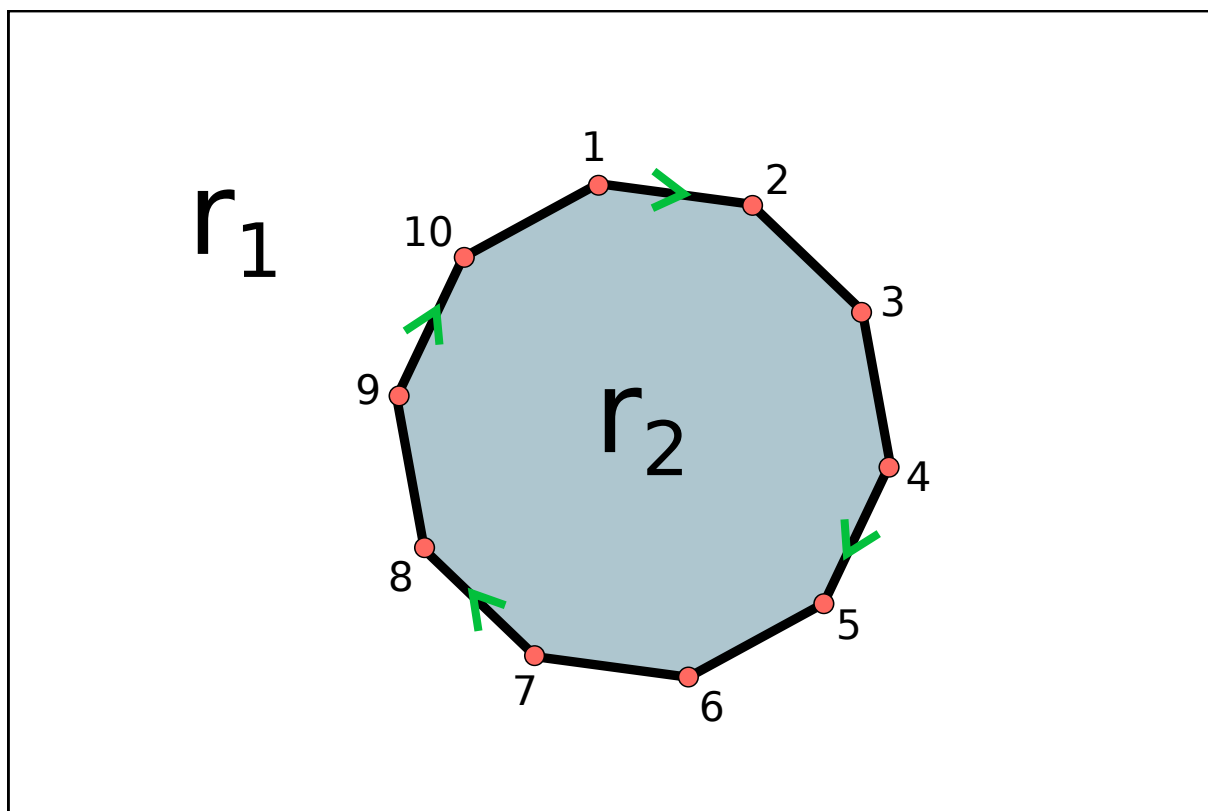


Figura 4.3: Exemplo de uma curva Front-Tracking com sentido horário.

Com todas estas observações, podemos resumir o conceito de curva Front-Tracking da seguinte forma:

Definição 2 *Uma curva Front-Tracking é um conjunto ordenado de pontos conectados por segmentos de retas. Toda curva possui:*

- Um ponto inicial e um ponto final que são chamados de nós;
- Um conjunto ordenado e finito de pontos entre o inicial e o final;
- Um sentido, que é definido pela ordenação dos pontos;
- Uma região localizada a sua esquerda e uma localizada a sua direita.

Vamos agora entender como foi feita a implementação da representação Front-tracking de uma interface. O código em linguagem C que implementa cada uma das estruturas descritas a seguir é apresentado no Código 4.1.

A primeira estrutura criada em nosso código tem o propósito de armazenar informações sobre uma região do plano. Esta estrutura, chamada de REGIAO, encapsula as seguintes informações:

- numero: é um valor inteiro utilizado como identificador para esta região;
- status: é uma *flag* que será utilizada futuramente no algoritmo de desembarçamento. Ela pode possuir os valores: INDEFINIDA, FISICA ou NAO_FISICA. O significado destes valores não é relevante agora, mas ficará claro na seção 4.2.
- localRegiao: é uma *flag* opcional que indica se uma região representa a parte interna de um material (fluido) ou a parte externa (sem fluido). Esta *flag* pode receber os seguintes valores: LOCAL_INDEFINIDO, EXTERNA ou INTERNA.

A segunda estrutura será usada para representar um ponto no plano cartesiano. Como precisamos armazenar um conjunto ordenado de pontos, optamos por fazer isto utilizando uma lista duplamente encadeada. Ou seja, cada ponto aponta diretamente para um ponto anterior e um ponto seguinte. Os seguintes valores são armazenados pela estrutura PONTO:

- x: é um número real que armazena a coordenada x deste ponto no plano cartesiano;
- y: armazena a coordenada y deste ponto no plano cartesiano;
- prox: é um ponteiro para uma estrutura PONTO que armazena o próximo ponto nesta lista encadeada;
- ant: é um ponteiro para uma estrutura PONTO que contém o ponto anterior nesta lista encadeada.

A terceira estrutura é utilizada para armazenar uma curva. Seguindo a definição dada anteriormente, esta estrutura armazena as seguintes informações:

- nodeInicial: é um ponteiro para um PONTO que contém o primeiro ponto desta curva;
- regioEsq: é um ponteiro para uma variável do tipo REGIAO que contém a região localizada no lado esquerdo desta curva;

- `regiaoDir`: ponteiro para a região do lado direito desta curva;
- `proxCurva`: lembrando que uma interface pode possuir mais de uma curva, optamos por criar uma lista encadeada de curvas. Essa variável é um ponteiro para a próxima curva da lista;
- `curvaAnt`: ponteiro para a curva anterior na lista.

A quarta e última estrutura utilizada em nosso código representa uma interface. A estrutura `INTERFACE` possui apenas dois valores:

- `curvaInicial`: ponteiro para a primeira curva na lista desta interface;
- `qtdCurvas`: quantidade total de curvas existentes nesta interface.

O código 4.1 traz a implementação em linguagem C das quatro estruturas detalhadas acima para armazenar uma interface Front-Tracking.

```

1 typedef enum StatusRegiao {INDEFINIDA, FISICA, NAO_FISICA} STATUS_REGIAO;
2
3 typedef enum TipoLocalRegiao {LOCAL_INDEFINIDO, EXTERNA, INTERNA}
4   TIPO_LOCAL_REGIAO;
5 typedef struct Regiao {
6     int numero;
7     STATUS_REGIAO status;
8     TIPO_LOCAL_REGIAO localRegiao;
9 } REGIAO;
10
11 typedef struct Ponto {
12     double x, y;
13     struct Ponto *prox;
14     struct Ponto *ant;
15 } PONTO;
16
17 typedef struct Curva {
18     PONTO *nodeInicial;
19
20     REGIAO *regiaoEsq;
21     REGIAO *regiaoDir;
22
23     struct Curva *proxCurva;
24     struct Curva *curvaAnt;
25 } CURVA;
26
27 typedef struct Interface {
28     CURVA *curvaInicial;
29     int qtdCurvas;
30 } INTERFACE;

```

Código 4.1: Código fonte em linguagem C das estruturas utilizadas para armazenar uma interface Front-Tracking.

4.1.2 Advecção Front-Tracking

Na seção anterior, vimos como definir e armazenar computacionalmente uma interface pelo método Front-Tracking. Em escoamentos de fluidos, uma interface se move constantemente de acordo com uma velocidade, sendo assim, é importante também estudar

como é feito este transporte da interface ao longo do tempo. Esse processo de transporte é também chamado de advecção.

Suponha que seja conhecida uma interface em um instante de tempo $t = t_n$, ou seja, sabemos qual é a posição (x, y) de todas as N partículas p_0, p_1, \dots, p_{N-1} da interface neste instante. Considere também que seja dada uma velocidade de advecção para cada ponto da interface em um instante de tempo futuro $t_{n+1} = t_n + \Delta t$.

Sabemos que a taxa de variação da posição de um objeto em função do tempo é a velocidade. Sendo assim, a advecção da interface é modelada pela equação diferencial

$$\dot{\mathbf{x}} = \mathbf{u}, \quad (4.1)$$

em que $\mathbf{x} = (x, y)$ é a posição de um ponto da interface e $\mathbf{u} = (u, v)$ é a velocidade de advecção neste ponto.

Considerando uma representação discreta, a equação (4.1) ao ser aplicada a cada ponto p_i da interface será reescrita como

$$\dot{\mathbf{x}}_i|_{t=t_{n+1}} = \mathbf{u}_i(t_{n+1}), \quad (4.2)$$

em que $\mathbf{u}_i(t_{n+1}) = (u(t_{n+1}), v(t_{n+1}))$ é a velocidade no ponto $\mathbf{x}_i = (x_i, y_i)$ no tempo t_{n+1} .

Observe que a equação vetorial (4.2) corresponde a duas equações escalares, dadas por

$$\begin{cases} \left. \frac{dx_i}{dt} \right|_{t=t_{n+1}} = u(t_{n+1}), \\ \left. \frac{dy_i}{dt} \right|_{t=t_{n+1}} = v(t_{n+1}). \end{cases} \quad (4.3)$$

É importante lembrar que conhecemos a posição da partícula no instante t_n , ou seja, $\mathbf{x}_i(t_n)$. Queremos agora utilizar as equações (4.3) para determinar a posição no tempo seguinte, isto é, $\mathbf{x}_i(t_{n+1})$. A estratégia que vamos adotar, é resolver (4.3) numericamente através de uma discretização pelo método de diferenças finitas.

Aproximando as derivadas de (4.3) de forma semi-implícita, obtemos as seguintes equações de diferenças

$$\begin{cases} x_i^{n+1} = x_i^n + \Delta t u(t_{n+1}), \\ y_i^{n+1} = y_i^n + \Delta t v(t_{n+1}), \end{cases} \quad (4.4)$$

em que x_i^{n+1} e y_i^{n+1} representam aproximações para $x_i(t_{n+1})$ e $y_i(t_{n+1})$, respectivamente. Observe que a advecção das partículas na equação (4.4) é feita de forma desacoplada à solução das equações de Navier-Stokes.

As equações (4.4) são aplicadas para cada um dos pontos $i = 1, 2, \dots$. Observe que a solução é obtida de forma explícita e, portanto, não apresenta nenhuma dificuldade computacional. Em nossa implementação, o processo de advecção é feito por uma função bastante simples nomeada *AdvectaParticula*. Esta função, que é apresentada no Código 4.2, é executada em cada ponto da interface separadamente.

```

1 void AdvectaParticula(PONTO *Ponto, double U, double V, double dt, double *NovoX, double
2   *NovoY)
3 {
4   *NovoX = Ponto->x + dt*U;
5   *NovoY = Ponto->y + dt*V;
6   return;
7 }
```

Código 4.2: Código fonte da função que realiza a advecção de uma partícula.

4.2 Descrição de um método para realização de mudanças topológicas

O principal objetivo deste estudo é a implementação de um método que realize mudanças topológicas em uma interface Front-Tracking. Nesta seção, iremos descrever toda a construção, funcionamento e implementação do algoritmo utilizado. Iremos inicialmente explicar o que chamamos de mudanças topológicas de uma interface e, em seguida, partiremos para a descrição do algoritmo que as realiza.

4.2.1 Mudanças topológicas: conceitos básicos

Vimos que uma interface representada pelo método Front-Tracking é definida por um conjunto de curvas que movem ao longo do tempo sobre um plano. Conforme a movimentação das curvas é feita, elas podem, em algum momento, se interceptar umas com as outras, o que gera uma mudança na topologia da interface definida por estas curvas. Este processo é também chamado em [14] de bifurcação.

De acordo com Glimm e colaboradores ([14]), as mudanças topológicas que ocorrem durante uma simulação podem ser classificadas em duas categorias iniciais:

- bifurcações artificialmente geradas por características numéricas: são mudanças topológicas que não existem na solução exata do problema matemático, porém, ocorrem na simulação devido a características do método numérico usado (erros de arredondamento e instabilidade, por exemplo).
- bifurcações que existem na solução matemática: são bifurcações que ocorrem na simulação e que também existem na solução exata do problema.

A primeira categoria ocorre com mais frequência em simulações que envolvem o estudo de filamentos muito finos de fluidos. As extremidades destes filamentos podem, fisicamente, se tornar extremamente próximas. Numericamente, devido a erros de arredondamento ou de aproximação do método numérico, as curvas do filamento que estão muito próximas podem acabar se interceptando, gerando uma mudança de topologia que não deveria ocorrer matematicamente.

No segundo caso, a mudança topológica faz parte da solução exata do problema e realmente deve ocorrer na simulação. Um exemplo bastante trivial é a movimentação de duas bolhas de um fluido que se movem uma em direção a outra, com velocidades opostas. Neste exemplo, ilustrado na Figura 4.4, é intuitivo perceber que as bolhas irão se encontrar e uma união acontecerá, de forma que somente uma bolha de fluido passará a existir. Note em 4.4c que, movimentando as curvas Front-Tracking em direções opostas, parte delas irá se interceptar, criando um embaraçamento de curvas. Algum algoritmo precisa ser executado para remover este embaraçamento e gerar a nova topologia da interface mostrada em 4.4d.

Portanto, modificações na topologia da interface são necessárias tanto por razões físicas, quanto numéricas. Algoritmos que realizam esta operação são chamados em [14] de algoritmos de desembaraçamento, pois eles removem o embaraçamento entre as curvas da interface.

As mudanças topológicas podem também ser classificadas em duas outras categorias, de acordo com o tipo de interação que irá ocorrer entre as regiões da interface. Vamos nomear essas categorias de: mudanças topológicas de **união** e mudanças topológicas de **separação**.

Mudanças topológicas de união são todas aquelas nas quais a nova interface possui uma região formada pela união de duas ou mais regiões da interface antiga (antes da

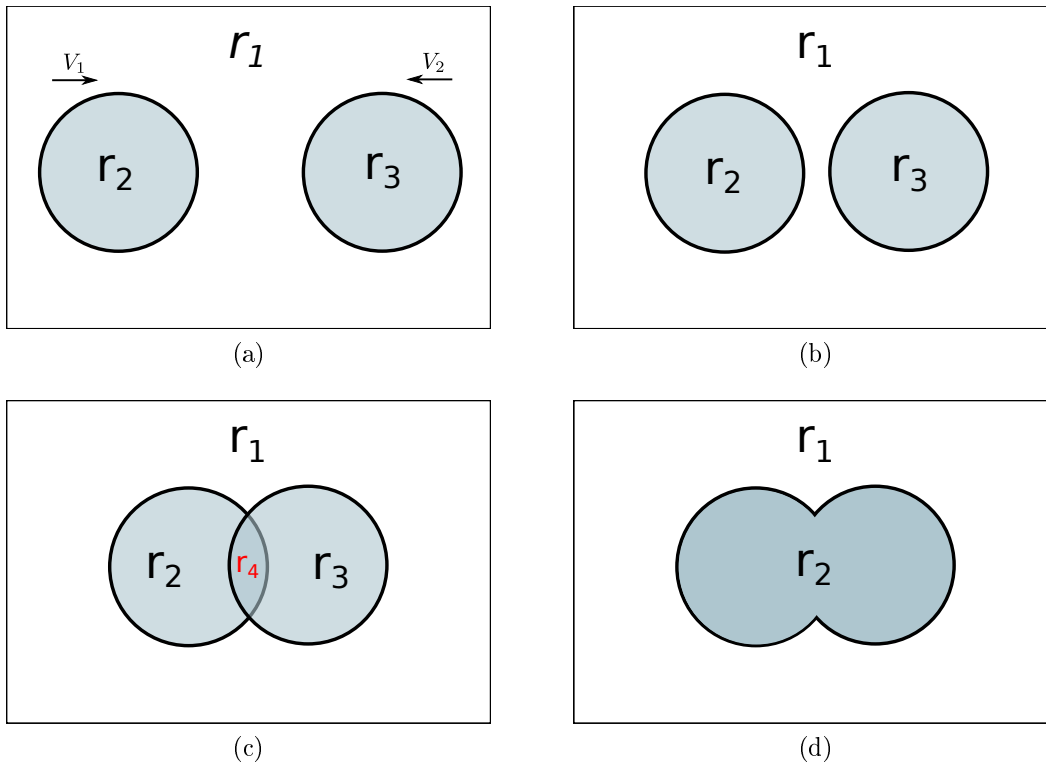


Figura 4.4: Exemplo de mudança topológica em um escoamento no qual duas bolhas de mesmo fluido se encontram. As bolhas são respectivamente advectadas com velocidades V_1 e V_2 que possuem sentidos opostos.

mudança). Um exemplo deste tipo de mudança topológica foi dada na Figura 4.4, na qual duas bolhas se unem, formando uma única região de fluido.

As mudanças topológicas de separação são aquelas nas quais uma região da interface se torna muito fina, de forma que uma separação ocorre, possivelmente gerando múltiplas novas regiões. Um exemplo de mudança topológica de separação é apresentado na Figura 4.5, na qual uma bolha que, pela ação da gravidade, é puxada para baixo e se desprende do restante do fluido.

Mais exemplos de escoamentos com mudanças topológicas serão apresentados nos testes numéricos da seção 4.

4.2.2 Algoritmo de desembaraçamento

Vimos até aqui o que são mudanças topológicas de interface e quais são seus tipos mais básicos. Iremos agora detalhar o algoritmo apresentado em [14] para realizar essa mudança de topologia em uma interface representada pelo método Front-Tracking.

Este algoritmo parte do princípio de que uma mudança topológica será realizada sempre que há um cruzamento entre curvas da interface, ou seja, sempre que houver um embaraçamento. O algoritmo tem como objetivo remover este embaraçamento e, por consequência, realizar uma mudança de topologia.

Usando o conceito de curva Front-Tracking visto na seção 4.1, vamos agora definir o que é uma interface Front-Tracking embaraçada.

Definição 3 *Uma interface Front-Tracking é dita embaraçada se as suas curvas se cruzam em algum ponto que não é um nó. Se uma interface não é embaraçada, dizemos que ela está em um estado correto.*

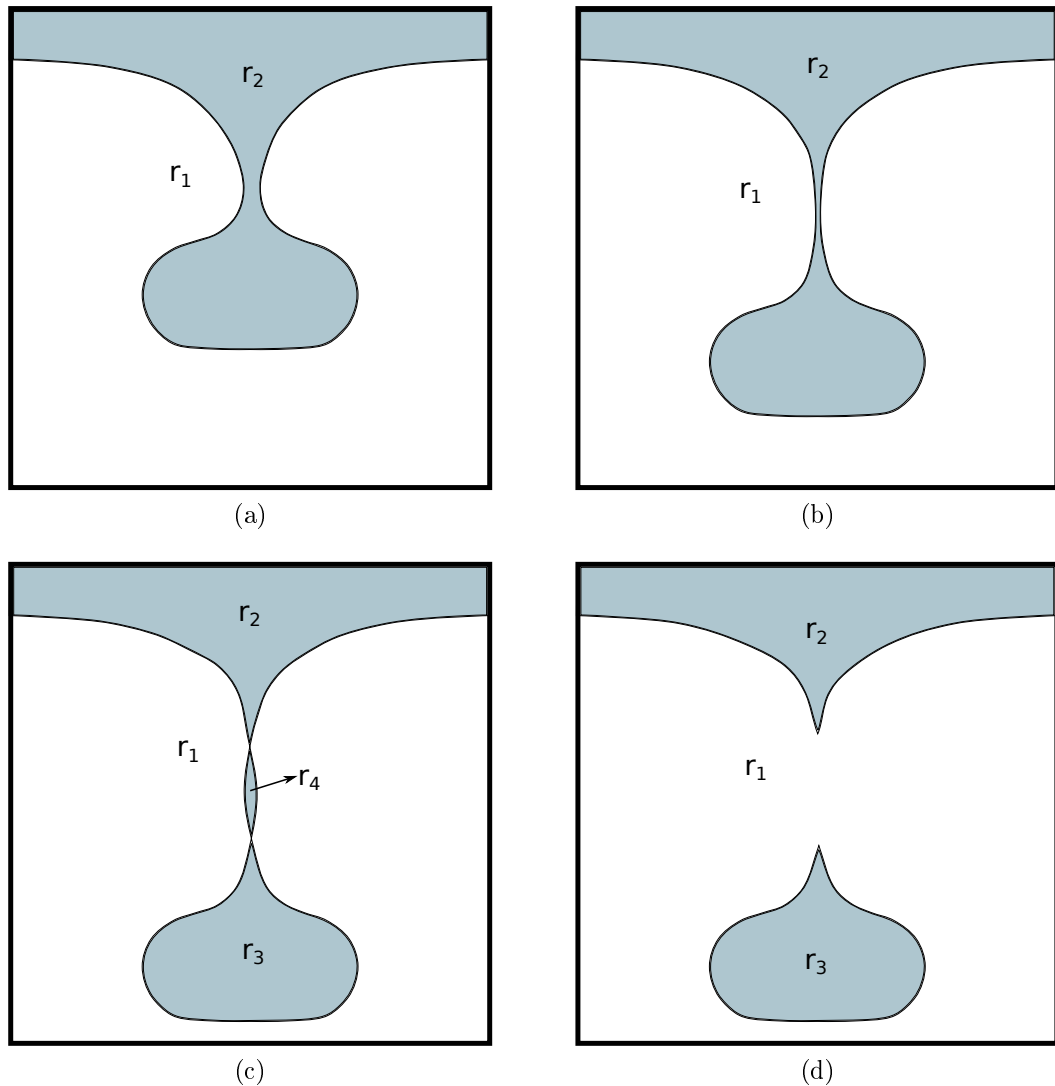


Figura 4.5: Exemplo de mudança topológica em um escoamento no qual uma bolha se desprende do restante do fluido.

Portanto, uma interface estará em estado correto se suas curvas se tocam apenas em seus pontos iniciais ou finais (nós). Se essa propriedade não for satisfeita, é necessário desembaraçar a interface antes de continuar a simulação do problema.

A Figura 4.6 mostra exemplos de interfaces embaraçadas e corretas, note que:

- 4.6a mostra um exemplo de uma interface com formato de circunferência formada por uma única curva. O nó inicial e final desta curva são o mesmo ponto, ou seja, a curva se toca somente em seus nós. A interface não está embaraçada.
- 4.6b representa exatamente a mesma interface do exemplo anterior, porém ela foi dividida em duas curvas. Note que as curvas possuem os mesmos pontos como nós inicial e final, ou seja, só há cruzamento nos nós. A interface não está embaraçada.
- 4.6c é um exemplo já visto anteriormente com duas bolhas no plano. A interface não está embaraçada.
- 4.6d apresenta um instante no qual as bolhas se tornaram tão próximas que as curvas se cruzaram em pontos que não são nós. A interface está embaraçada.

- 4.6e é um exemplo também já visto anteriormente de uma interface não-embaraçada com formato de anel.
- 4.6f mostra que uma única curva também pode cruzar com ela mesma. Este exemplo é idêntico ao caso 4.6e, mas com um aumento na espessura do anel, de forma que há intersecções na curva. Neste caso a interface está embaraçada.

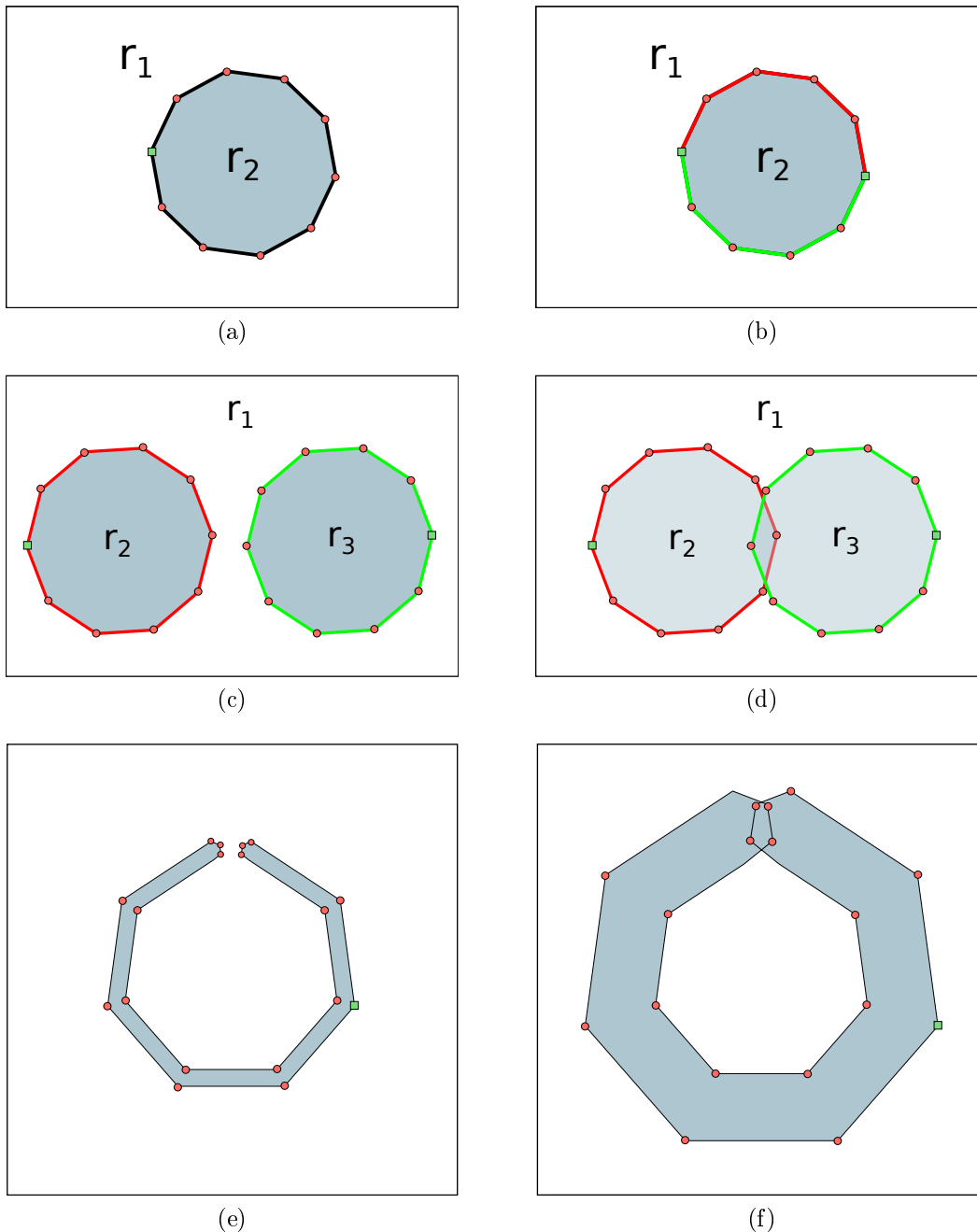


Figura 4.6: Exemplos de interfaces com e sem embaraçamento. Os pontos com formato de quadrado indicam os nós das curvas. A cor dos segmentos de retas indicam curvas diferentes. O sentido das curvas não foi indicado nas imagens, pois não é relevante para o propósito destes exemplos.

Com o conceito de embaraçamento bem definido, podemos agora começar a introduzir o algoritmo que realiza o desembaraçamento da interface e, conseqüentemente, a mudança topológica da mesma.

Vimos anteriormente que uma interface divide o plano cartesiano em um conjunto finito de regiões. Quando um embaraçamento ocorre, as curvas da interface embaraçada formam um número de regiões diferente da quantidade que existia anteriormente.

Observando a Figuras 4.4b e 4.4c, por exemplo, veja que em 4.4b o plano estava dividido em apenas três regiões. Após o embaraçamento, na Figura 4.4c, uma região artificial aparece entre as curvas, de forma que a interface embaraçada divide o plano em 4 regiões. Uma destas regiões foi criada artificialmente pelo embaraçamento e, por isso, é chamada de região não-física. O objetivo básico do algoritmo de desembaraçamento é identificar qual das regiões é a não-física para, em seguida, removê-la. Na Figura 4.4c, podemos intuitivamente perceber que a região não-física é a r_4 , porém, em um caso genérico, esta identificação não é tão simples. Vamos agora detalhar o algoritmo que identifica as regiões não-físicas.

Para construir o algoritmo em [14], considera-se que quatro hipóteses são satisfeitas sobre a interface embaraçada. Se essas hipóteses não forem satisfeitas, o algoritmo não pode ser usado.

Seja I^0 uma interface em um determinado passo de tempo e I^1 a interface obtida após o processo de advecção em I^0 . A nova interface I^1 pode, ou não, estar embaraçada. As quatro propriedades a seguir precisam ser satisfeitas para que o algoritmo de desembaraçamento possa ser utilizado:

A_1 . Cada ponto de uma curva em I^1 cruzou alguma curva no máximo uma vez durante a última advecção que levou I^0 a I^1 .

A Figura 4.7 ilustra o enunciado desta condição. Em 4.7a, a elipse cruzou duas curvas em um único passo, o que não pode acontecer. Em 4.7b apenas uma curva foi cruzada, portanto a interface advectada satisfaz a condição A_1 .

Observe que esta condição nada mais é do que uma restrição no tamanho do passo temporal, pois, reduzindo o passo, a distância percorrida pela interface será menor e os pontos deixarão de cruzar múltiplas curvas.

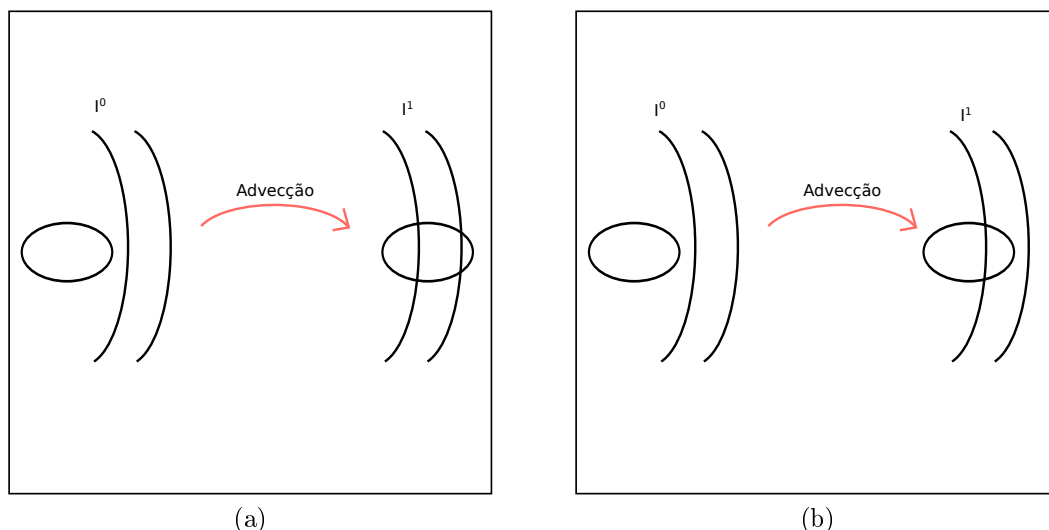


Figura 4.7: Exemplo de advecção no qual pontos da interface cruzam múltiplas curvas em um único passo temporal. Isso não deve ocorrer para que o algoritmo de desembaraçamento possa ser utilizado. Esta figura é baseada em uma imagem dada como exemplo em [14].

A_2 . Cada região não-física possui no seu contorno ao menos um ponto de cruzamento no qual curvas de I^1 se interceptam.

Observando novamente o exemplo da Figura 4.4c, é fácil notar que o contorno da região não-física r_4 possui os dois pontos nos quais as curvas se cruzam. Ou seja, a interface satisfaz a condição A_2 .

Essa condição, em geral, não exige cuidados especiais na implementação. Em todos os testes realizados ela sempre foi satisfeita naturalmente.

A_3 . *Nos pontos de cruzamento, as curvas realmente se cruzam, e não apenas se tocam.*

É importante que o algoritmo seja executado apenas quando as curvas realmente se cruzam. Se as bolhas da Figura 4.4c estivessem apenas se tocando, por exemplo, seria necessário realizar mais uma advecção antes de aplicar o algoritmo de embarçamento.

A_4 . *Nenhum nó pode ter cruzado uma curva durante a última advecção.*

Esta é, possivelmente, a condição que exige o maior cuidado no momento da utilização deste algoritmo. Ela diz que somente pontos comuns podem cruzar curvas, isto é, os nós (pontos iniciais e finais) não podem.

Quando curvas cíclicas são utilizadas (como no caso das bolhas), qualquer ponto da curva pode ser escolhido como nó. Contudo, para respeitar a condição A_4 , é importante que seja escolhido algum ponto que esteja distante da região onde irá acontecer a mudança topológica, garantindo que ele não irá cruzar outras curvas.

Esta escolha de nós é particular a cada problema que estiver sendo simulado e, até o momento, foi implementada separadamente para cada teste numérico feito. Na Figura 4.5, por exemplo, uma boa opção seria escolher como nó um ponto que está na parte inferior da bolha. Como a separação ocorre na parte superior da mesma, o nó estaria sempre distante do local de mudança topológica.

Supondo que as quatro propriedades acima sejam válidas, vamos agora detalhar as etapas do algoritmo de desembarçamento para identificar e remover regiões não-físicas.

O algoritmo de desembarçamento será dividido em 5 etapas, sendo elas:

- E_1 . Detectar embarçamento;
- E_2 . Dividir segmentos embarçados;
- E_3 . Definir regiões das curvas;
- E_4 . Classificar regiões como físicas e não-físicas;
- E_5 . Remover regiões não-físicas.

Vamos agora estudar cada etapa separadamente.

4.2.2.1 Etapa E_1 : Detectar embarçamento

Esta etapa consiste em verificar se a interface está em um estado de embarçamento ou não. Utilizando a definição de embarçamento, precisamos determinar se existe intersecção entre as curvas da interface.

Uma curva Front-Tracking é formada por um conjunto de segmentos de reta e, sendo assim, verificar se há intersecção entre as curvas equivale a verificar se há intersecção entre os segmentos de reta.

Primeiramente vamos estudar o processo escolhido para detectar se existe intersecção entre dois segmentos de reta dados.

Sejam:

- S_1 : um segmento de reta no plano que contém em cada uma de suas extremidades os pontos $P_1 = (x_1, y_1)$ e $\tilde{P}_1 = (\tilde{x}_1, \tilde{y}_1)$, respectivamente;
- S_2 : um segmento de reta no plano que contém em cada uma de suas extremidades os pontos $P_2 = (x_2, y_2)$ e $\tilde{P}_2 = (\tilde{x}_2, \tilde{y}_2)$, respectivamente.

Conhecendo os extremos dos segmentos S_1 e S_2 , podemos facilmente escrevê-los em sua forma paramétrica como

$$\begin{aligned} S_1 : (x, y) &= (x_1, y_1) + \lambda_1(\tilde{x}_1 - x_1, \tilde{y}_1 - y_1), \quad \lambda_1 \in [0, 1], \\ S_2 : (x, y) &= (x_2, y_2) + \lambda_2(\tilde{x}_2 - x_2, \tilde{y}_2 - y_2), \quad \lambda_2 \in [0, 1]. \end{aligned} \quad (4.5)$$

Note que as equações (4.5) representariam equações de retas, caso os parâmetros λ_1 e λ_2 não estivessem restritos ao intervalo $[0, 1]$.

Para encontrar o possível ponto de intersecção entre os dois segmentos, vamos então igualar as equações paramétricas de S_1 e S_2 e, se o ponto de intersecção encontrado satisfizer $\lambda_1 \in (0, 1)$ e $\lambda_2 \in (0, 1)$, então os segmentos se cruzam.

Igualando as equações de S_1 e S_2 em (4.5), temos

$$\begin{aligned} (x_1, y_1) + \lambda_1(\tilde{x}_1 - x_1, \tilde{y}_1 - y_1) &= (x_2, y_2) + \lambda_2(\tilde{x}_2 - x_2, \tilde{y}_2 - y_2) \Leftrightarrow \\ \begin{bmatrix} \tilde{x}_1 - x_1 & -(\tilde{x}_2 - x_2) \\ \tilde{y}_1 - y_1 & -(\tilde{y}_2 - y_2) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} &= \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}. \end{aligned} \quad (4.6)$$

Resolvendo o sistema linear (4.6) encontramos os valores de λ_1 e λ_2 do ponto de intersecção. Se ambos estiverem no intervalo $(0, 1)$, então os segmentos se cruzam. Este é o teste utilizado para verificar se há intersecção entre dois segmentos de retas.

Portanto, a ideia mais simples para verificar se existe embaraçamento na interface Front-Tracking é aplicar o teste acima em todos os segmentos de reta da interface dois a dois. O algoritmo 4.3 traz um pseudocódigo da rotina que realiza a detecção do embaraçamento.

```

1 Para cada curva  $C_1$  da interface, faça
2   Para cada segmento de reta  $S_1$  da curva  $C_1$ , faça
3      $(x_1, y_1) :=$  ponto inicial de  $S_1$ ;
4      $(\tilde{x}_1, \tilde{y}_1) :=$  ponto final de  $S_1$ ;
5
6   Para cada curva  $C_2$  da interface, faça
7     Para cada segmento de reta  $S_2$  da curva  $C_2$ , faça
8        $(x_2, y_2) :=$  ponto inicial de  $S_2$ ;
9        $(\tilde{x}_2, \tilde{y}_2) :=$  ponto final de  $S_2$ ;
10       $A = \begin{bmatrix} \tilde{x}_1 - x_1 & -(\tilde{x}_2 - x_2) \\ \tilde{y}_1 - y_1 & -(\tilde{y}_2 - y_2) \end{bmatrix}$ ;
11       $b = \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}$ ;
12      Resolver o sistema  $A \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = b$ ;
13
14      Se  $(\lambda_1 > 0$  e  $\lambda_1 < 1$  e  $\lambda_2 > 0$  e  $\lambda_2 < 1)$ 
15        Existe embaraçamento na interface;
16      Fim Se
17    Fim Para
18  Fim Para
19
20 Fim Para
21 Fim Para

```

Código 4.3: Algoritmo que detecta se existe embaraçamento em uma interface Front-Tracking.

As próximas etapas do algoritmo de desembaraçamento serão executadas apenas se for detectado embaraçamento na etapa E_1 .

4.2.2.2 Etapa E_2 : Dividir segmentos embaraçados

Após detectada a existência do embaraçamento e também os pontos nos quais as curvas se interceptam, o próximo passo é dividir os segmentos embaraçados no ponto de intersecção.

Essa divisão é feita da seguinte forma:

1. Cada ponto de intersecção entre segmentos embaraçados é adicionado na interface como um novo nó. Vamos chamar esses nós de nós de cruzamento.
2. Cada curva embaraçada será dividida em duas curvas. A divisão é feita exatamente neste novo nó.

A Figura 4.8 ilustra este processo de divisão. Observe que, após o processo de divisão, sempre haverá quatro curvas utilizando o nó de cruzamento: duas curvas que partem deste nó e duas curvas que chegam neste nó.

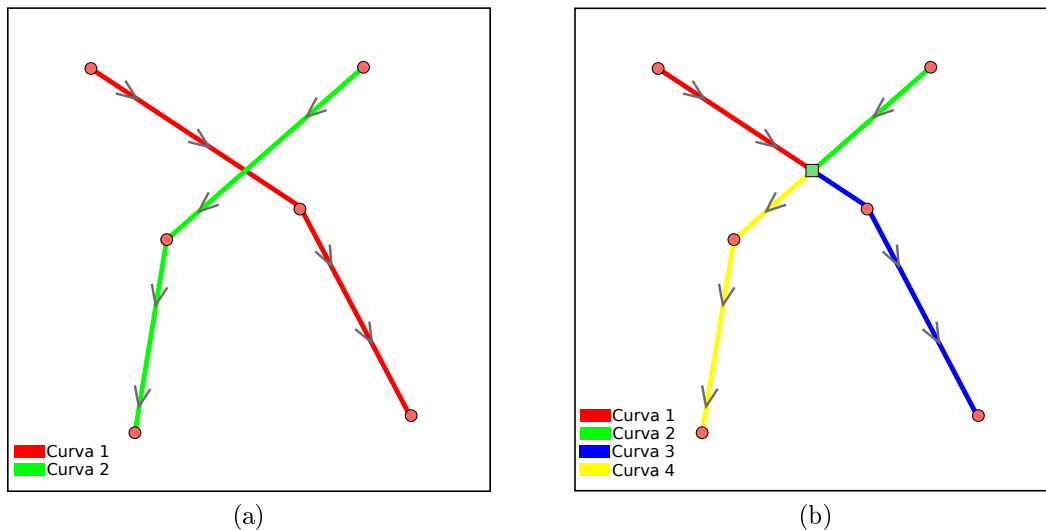


Figura 4.8: Processo de divisão de curvas no ponto de intersecção. O painel 4.8a contém as curvas antes do processo de divisão. O painel 4.8b as apresenta após a divisão. Apenas os segmentos de reta que estão mais próximos do ponto de intersecção foram mostrados na figura.

Para visualizar este processo de divisão em uma interface completa, a Figura 4.9 mostra-o no problema da união de duas bolhas. Note que, neste caso, existem dois nós de cruzamento e o processo de divisão precisa ser executado em cada um deles sequencialmente.

Após realizar o processo de divisão de curvas, é preciso armazenar em cada nó de cruzamento um vetor contendo as quatro curvas que utilizam este nó. Por exemplo, no nó de cruzamento 1 (superior) da Figura 4.9c, devemos armazenar um vetor contendo as curvas 2, 4, 5 e 3. No nó de cruzamento 2 (inferior) deve-se armazenar um vetor contendo as curvas 4, 6, 1 e 5.

Para finalizar a etapa E_2 do algoritmo, precisamos ordenar o vetor que contém as quatro curvas de cada nó de cruzamento. Esta ordenação é feita de modo que as curvas fiquem armazenadas sequencialmente no sentido anti-horário. Por exemplo, na Figura 4.9c, os vetores de cada nó de cruzamento devem possuir seus valores na seguinte ordem:

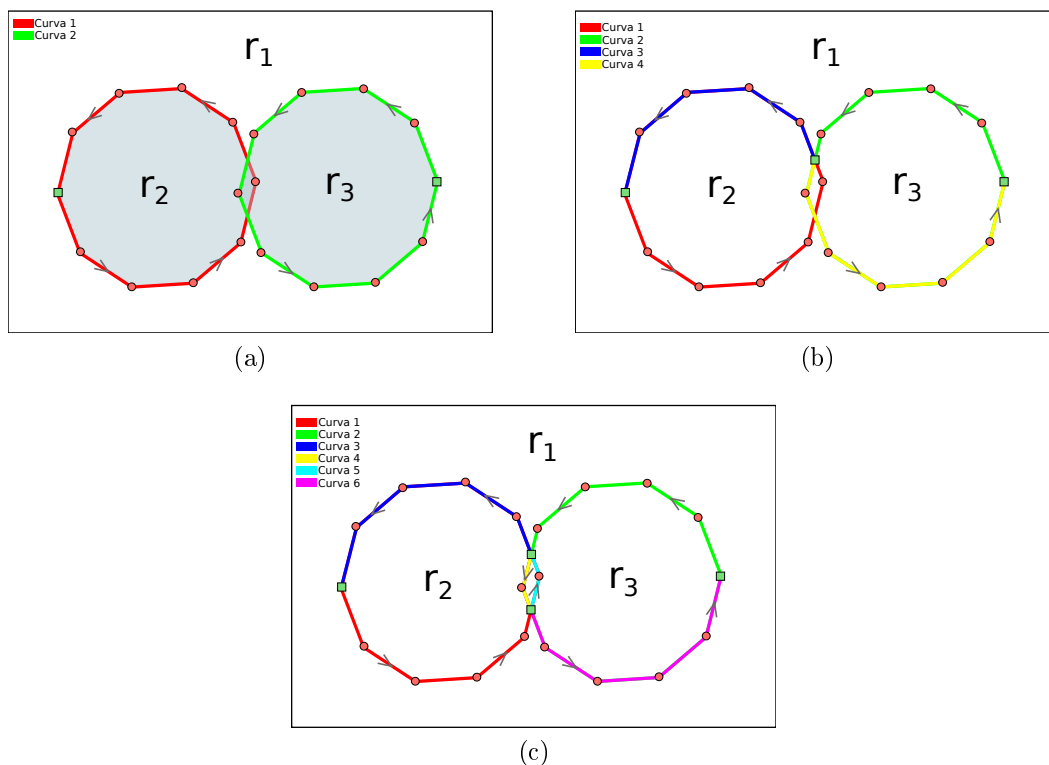


Figura 4.9: Processo de divisão de curvas no problema da união de bolhas. 4.9a: interface embaraçada antes do processo de divisão. 4.9b: interface após o processo de divisão ser realizado no primeiro nó de cruzamento. 4.9c: interface após a divisão ser realizada no segundo nó de cruzamento.

- Nó de cruzamento 1: $[2 \ 3 \ 4 \ 5]$;
- Nó de cruzamento 2: $[5 \ 4 \ 1 \ 6]$.

Para ordenar as curvas de um nó no sentido anti-horário utilizamos comparações baseadas no ângulo de cada segmento de reta. Por exemplo, na Figura 4.10, utilizamos a função arcotangente para calcular facilmente o valor dos ângulo α , β , γ e δ , cada um associado a uma curva. Em seguida, ordenamos as quatro curvas no vetor a partir do menor ângulo até o maior.

Portanto, no final da etapa E_2 do algoritmo de desembaraçamento, teremos:

- novos nós na interface, estes são chamados nós de cruzamento;
- novas curvas adicionadas na interface, todas elas saindo ou chegando a um nó de cruzamento;
- um vetor de 4 elementos armazenado em cada nó de cruzamento. Este vetor ordenado contém as quatro curvas ao redor do nó e elas estão armazenadas no sentido anti-horário.

4.2.2.3 Etapa E_3 : Definir regiões das curvas

Neste momento, possuímos uma interface com novas curvas que foram recentemente criadas pela etapa E_2 . Vamos agora lembrar que uma curva possui sentido e, sendo assim, é possível definir qual região do plano está localizada a sua esquerda e qual está a sua direita.

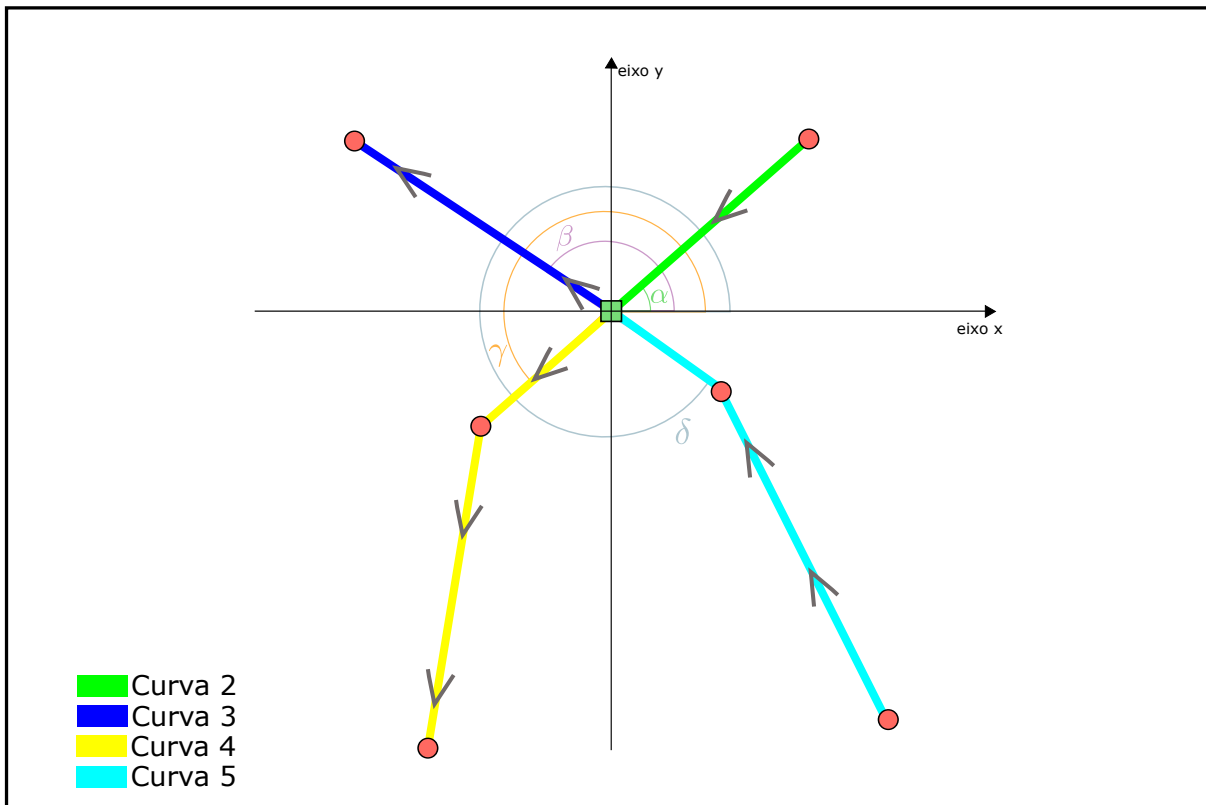


Figura 4.10: Ângulo de cada curva ao redor de um nó de cruzamento. Estes ângulos são facilmente calculados aplicando a função arco-tangente a cada um dos quatro segmentos.

Como acabamos de criar novas curvas, precisamos também definir qual é a região que está localizada de cada um dos lados das mesmas, pois isto não foi feito automaticamente pela etapa anterior.

Usando novamente o exemplo da união entre bolhas (Figura 4.9), note que, a interface antes da etapa E_2 possuía apenas duas curvas. De acordo com o sentido dado na Figura 4.9a, estas curvas possuíam as seguintes regiões em cada um de seus lados:

- Curva 1:
 - Região a esquerda: r_2
 - Região a direita: r_1
- Curva 2:
 - Região a esquerda: r_3
 - Região a direita: r_1

No final da etapa E_2 (Figura 4.9c), a interface passa a possuir 6 curvas, porém somente são conhecidas as regiões esquerda e direita das curvas 1 e 2, que já existiam anteriormente. É preciso agora executar um passo para definir quais regiões estão do lado esquerdo e direito das outras quatro novas curvas.

Este processo se torna bastante simples se lembrarmos que, na etapa anterior, as 4 curvas de cada nó de cruzamento foram ordenadas no sentido anti-horário. Usando novamente a Figura 4.9c como exemplo, observe que:

1. as quatro curvas do nó de cruzamento superior possuem a seguinte ordem (anti-horária): curva 2, curva 3, curva 4, curva 5;

2. já sabemos que a região esquerda da curva 2 é r_3 , pois esta curva já existia inicialmente, antes do processo de divisão;
3. a curva que vier imediatamente antes da curva 2 na ordenação anti-horária também possuirá a região r_3 de um dos seus lados;
4. como a curva 5 vem imediatamente antes da curva 2 na ordenação então ela também possui a região r_3 de um dos seus lados;
5. a curva 5 está “entrando” no nó de cruzamento e, portanto, sua região direita é que será r_3 ;
6. com estas observações, acabamos de determinar a região direita da curva 5.

Analogamente, observe que as curvas 2 e 3 são consecutivas na ordenação anti-horária e, conseqüentemente, compartilham a região r_1 . Desta forma podemos determinar que a região direita da curva 3 é r_1 .

De modo geral, este processo é representado pelo algoritmo 4.4.

```

1 Seja n um no de cruzamento da interface;
2 curvas := as quatro curvas de n ordenadas no sentido anti-horario;
3
4 Para i=1 até 4, faça
5
6   curva := curva[i];
7
8   %Obtendo as curvas anterior e proxima na ordenação anti-horaria
9   proxCurva := curvas[i+1]; %Se i=4, considere i+1=1
10  antCurva := curvas[i-1]; %Se i=1, considere i-1=4
11
12  %Vamos determinar uma das regioes de antCurva usando sua regio em comum com Curva
13  Se( curva está entrando em n ) faça
14    regio = curva.regiaoEsq;
15  Senao se( curva está saindo de n ) faça
16    regio = curva.regiaoDir;
17  Fim Se
18
19  %Colocando o valor em antCurva
20  Se( regio nao for indeterminada ) faça
21    Se( antCurva está entrando em n ) faça
22      antCurva.regiaoDir := regio;
23    Senao se( antCurva está saindo de n ) faça
24      antCurva.regiaoEsq := regio;
25    Fim Se
26  Fim Se
27
28
29  %Agora vamos determinar uma das regioes de proxCurva usando sua regio em comum com Curva
30  Se( curva está entrando em n ) faça
31    regio = curva.regiaoDir;
32  Senao se( curva está saindo de n ) faça
33    regio = curva.regiaoEsq;
34  Fim Se
35
36  %Colocando o valor em proxCurva
37  Se( regio nao for indeterminada ) faça
38    Se( proxCurva está entrando em n ) faça
39      proxCurva.regiaoEsq := regio;
40    Senao se( proxCurva está saindo de n ) faça
41      proxCurva.regiaoDir := regio;

```

```

42   Fim Se
43   Fim Se
44
45 Fim Para

```

Código 4.4: Algoritmo que determina novas regiões .

Para entender melhor o funcionamento do algoritmo, vamos realizar uma execução passo-a-passo do mesmo sendo aplicado ao nó de cruzamento superior da Figura 4.9c. Esta execução é apresentada na Tabela 4.1. Observe que, ao aplicar o algoritmo no nó superior, foi possível determinar duas entradas da tabela de curvas e regiões que previamente estavam indefinidas. O algoritmo precisa então ser aplicado também ao nó de cruzamento inferior, o que fará com que mais entradas da tabela sejam preenchidas. Reaplicando o algoritmo aos dois nós múltiplas vezes, chegará um instante em que nenhuma entrada da tabela será modificada; é nesse instante que o processo pode ser parado.

O processo, quando aplicado ao exemplo da Figura 4.9c, para de ser executado com a tabela de regiões e curvas apresentada na Tabela 4.2.

Observe na Tabela 4.2 que, utilizando as regiões já conhecidas, não foi possível determinar duas entradas da tabela. Isto indica que estas entradas representam novas regiões na interface. Isto fica muito claro comparando a Tabela 4.2 com a Figura 4.9c, pois as entradas da tabela que ficaram indefinidas são justamente aquelas que correspondem à região central da figura, que é uma região que apareceu após o embaraçamento. Esta região não existia na interface original.

Sendo assim, precisamos criar novas regiões para que a tabela de regiões e curvas possa ser preenchida por completo. O processo utilizado aqui será o seguinte:

1. Escolha aleatoriamente uma das entradas indefinidas da tabela de regiões e curvas;
2. Preencha esta entrada da tabela com uma nova região;
3. Execute novamente o processo do algoritmo 4.4 para, possivelmente, preencher as outras entradas indefinidas com esta nova região;
4. Se alguma entrada da tabela continuar indefinida, volte a etapa 1.

Na tabela 4.2, por exemplo, possuímos duas regiões indefinidas. Vamos preencher qualquer uma delas com uma nova região r_4 . Aplicando novamente o processo do algoritmo 4.4, a outra entrada indefinida da tabela automaticamente também será preenchida com r_4 , fazendo com que todas as curvas tenham suas regiões bem definidas.

Isto finaliza a etapa E_3 do algoritmo de desembaraçamento. O algoritmo 4.5 sumariza tudo o que foi realizado durante esta etapa.

```

1 Inicializar a tabela de regiões e curvas com os dados das curvas que já existiam antes da Etapa  $E_2$ ;
2 Enquanto ( houver entradas indefinidas na tabela ) faça
3
4   Enquanto ( tabela sofrer modificações ) faça
5     Para ( cada nó de cruzamento da interface ) faça
6       Aplicar o algoritmo 4.4 ao nó;
7     Fim Para
8   Fim Enquanto
9
10  Se ( houver entradas indefinidas na tabela )
11    Preencha uma entrada indefinida aleatória com uma nova região;
12  Fim Se
13
14 Fim Enquanto

```

Código 4.5: Algoritmo que contém todos os passos da Etapa E_3 .

Iteração	Comando	Estado atual das curvas																					
	n:= nó de cruzamento superior; curvas := {2, 3, 4, 5};																						
i=1	curva := curva 2; proxCurva := curva 3; antCurva := curva 5; regiao := r ₃ ; antCurva.regiaoDir := regiao; regiao := r ₁ proxCurva.regiaoDir := regiao;	<table border="1"> <tr><td>curva</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> <tr><td>Esq</td><td>r₂</td><td>r₃</td><td>?</td><td>?</td><td>?</td><td>?</td></tr> <tr><td>Dir</td><td>r₁</td><td>r₁</td><td>?</td><td>?</td><td>?</td><td>?</td></tr> </table>	curva	1	2	3	4	5	6	Esq	r ₂	r ₃	?	?	?	?	Dir	r ₁	r ₁	?	?	?	?
curva	1	2	3	4	5	6																	
Esq	r ₂	r ₃	?	?	?	?																	
Dir	r ₁	r ₁	?	?	?	?																	
i=2	curva := curva 3; proxCurva := curva 4; antCurva := curva 2; regiao := r ₁ ; antCurva.regiaoDir := regiao; regiao := ?;	<table border="1"> <tr><td>curva</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> <tr><td>Esq</td><td>r₂</td><td>r₃</td><td>?</td><td>?</td><td>?</td><td>?</td></tr> <tr><td>Dir</td><td>r₁</td><td>r₁</td><td>r₁</td><td>?</td><td>r₃</td><td>?</td></tr> </table>	curva	1	2	3	4	5	6	Esq	r ₂	r ₃	?	?	?	?	Dir	r ₁	r ₁	r ₁	?	r ₃	?
curva	1	2	3	4	5	6																	
Esq	r ₂	r ₃	?	?	?	?																	
Dir	r ₁	r ₁	r ₁	?	r ₃	?																	
i=3	curva := curva 4; proxCurva := curva 5; antCurva := curva 3; regiao := ?; regiao := ?;	<table border="1"> <tr><td>curva</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> <tr><td>Esq</td><td>r₂</td><td>r₃</td><td>?</td><td>?</td><td>?</td><td>?</td></tr> <tr><td>Dir</td><td>r₁</td><td>r₁</td><td>r₁</td><td>?</td><td>r₃</td><td>?</td></tr> </table>	curva	1	2	3	4	5	6	Esq	r ₂	r ₃	?	?	?	?	Dir	r ₁	r ₁	r ₁	?	r ₃	?
curva	1	2	3	4	5	6																	
Esq	r ₂	r ₃	?	?	?	?																	
Dir	r ₁	r ₁	r ₁	?	r ₃	?																	
i=4	curva := curva 5; proxCurva := curva 2; antCurva := curva 4; regiao := ?; regiao := r ₃ proxCurva.regiaoEsq := regiao;	<table border="1"> <tr><td>curva</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> <tr><td>Esq</td><td>r₂</td><td>r₃</td><td>?</td><td>?</td><td>?</td><td>?</td></tr> <tr><td>Dir</td><td>r₁</td><td>r₁</td><td>r₁</td><td>?</td><td>r₃</td><td>?</td></tr> </table>	curva	1	2	3	4	5	6	Esq	r ₂	r ₃	?	?	?	?	Dir	r ₁	r ₁	r ₁	?	r ₃	?
curva	1	2	3	4	5	6																	
Esq	r ₂	r ₃	?	?	?	?																	
Dir	r ₁	r ₁	r ₁	?	r ₃	?																	
Estado final		<table border="1"> <tr><td>curva</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> <tr><td>Esq</td><td>r₂</td><td>r₃</td><td>?</td><td>?</td><td>?</td><td>?</td></tr> <tr><td>Dir</td><td>r₁</td><td>r₁</td><td>r₁</td><td>?</td><td>r₃</td><td>?</td></tr> </table>	curva	1	2	3	4	5	6	Esq	r ₂	r ₃	?	?	?	?	Dir	r ₁	r ₁	r ₁	?	r ₃	?
curva	1	2	3	4	5	6																	
Esq	r ₂	r ₃	?	?	?	?																	
Dir	r ₁	r ₁	r ₁	?	r ₃	?																	

Tabela 4.1: Execução passo-a-passo do algoritmo 4.4 aplicado ao nó de cruzamento superior da figura 4.9c.

4.2.2.4 Etapa E_4 : Classificar regiões como físicas e não-físicas

Neste momento possuímos uma interface com novas curvas, que foram criadas na etapa E_2 , e com novas regiões, que foram criadas na etapa E_3 . Iremos agora buscar quais destas regiões são físicas e não-físicas.

curva	1	2	3	4	5	6
Esq	r_2	r_3	r_2	?	?	r_3
Dir	r_1	r_1	r_1	r_2	r_3	r_1

Tabela 4.2: Tabela de regiões e curvas relacionada a Figura 4.9c após múltiplas aplicações do algoritmo 4.4 aos nós de cruzamento.

Para realizar esta busca, iremos utilizar cinco observações que seguem imediatamente das quatro hipóteses $A_1 - A_4$ vistas no início desta seção. A demonstração da veracidade destas observações é comentada em [14]. As observações são:

- O_1 . Cada nó de cruzamento sempre possui duas curvas de entrada e duas de saída, dividindo a área ao redor do nó em quatro setores;
- O_2 . Um, e somente um, dos quatro setores ao redor de um nó de cruzamento é não-físico;
- O_3 . Se uma curva começa e termina em um mesmo nó de cruzamento (é um *loop*), os dois setores do lado de fora da curva são físicos;
- O_4 . Um setor entre duas curvas, cada uma ligando um nó de cruzamento n_1 a nós distintos $n_2 \neq n_3$, é físico se n_2 ou n_3 não é um nó de cruzamento.
- O_5 . Um setor não físico deve ser delimitado por uma das duas opções abaixo:
 - uma única curva que começa e termina em um mesmo nó de cruzamento;
 - duas curvas que vão de um mesmo nó n_1 a um outro nó n_2 , com um deles sendo um nó de cruzamento.

No momento da implementação, as observações mais importantes serão as O_2 e O_4 . O processo de classificar as regiões como físicas e não-físicas seguirá os seguintes passos:

1. Classificar todas as regiões como indefinidas;
2. Enquanto houver regiões indefinidas, repetir:
 - (a) Aplicar as observações O_2 e O_4 em cada nó de cruzamento para procurar regiões físicas;
 - (b) Aplicar as observações O_2 e O_4 em cada nó de cruzamento para procurar regiões não-físicas.

O passo 2a tenta classificar regiões como físicas. Ele é executado uma vez em cada nó de cruzamento da seguinte forma:

1. $n_1 :=$ nó de cruzamento atual;
2. Para cada um dos quatro setores de n_1 , fazer:
 - (a) Obter as duas curvas que formam este setor;
 - (b) Obter n_2 , que é o outro nó da primeira curva;
 - (c) Obter n_3 , que é o outro nó da segunda curva;
 - (d) Aplicar a observação O_4 aos nós n_1 , n_2 , n_3 e, se ela for verificada, classificar a região deste setor como física.

- (e) Se este setor já tiver sido classificado como não-físico em algum passo anterior, usar a observação O_2 e classificar os outros três setores como físicos.

Já o passo 2b tenta classificar regiões como não-físicas. Apenas a observação O_2 será usada para este propósito. Para cada nó de cruzamento, iremos contar quantos setores já foram previamente classificados como físicos. Se, por acaso, três setores já tiverem sido classificados, então o setor restante é não-físico.

Repetindo este processo é possível classificar todas regiões como físicas e não-físicas. Este é o fim da etapa E_4 .

4.2.2.5 Etapa E_5 : Remover regiões não-físicas

Como toda a classificação de regiões já foi feita, esta etapa se torna uma das mais simples. Para entender como ela foi implementada, vamos utilizar novamente a Figura 4.9c como exemplo. Nesta figura, observe que possuímos quatro regiões: r_1 , r_2 , r_3 e r_4 , sendo que r_4 é a região não-física central que foi encontrada e classificada pelas etapas E_3 e E_4 .

Devido a etapa E_4 , já sabemos que a região r_4 é não-física. Iremos então obter as duas curvas que delimitam esta região e removê-las da interface. Observe que, ao realizar esta remoção, as regiões r_2 e r_3 se unem, formando uma única. Sendo assim, o processo de remoção possui os seguintes passos:

1. Para cada região não-física r_{nf} da interface, faça:
 - (a) Obter as duas curvas, c_1 e c_2 , que delimitam a região r_{nf} ;
 - (b) Obter a região física $r_{f,1}$ que está do outro lado da curva c_1 (lembre-se que, de um lado de c_1 , está r_{nf} . Basta obter a curva que está do outro lado);
 - (c) Obter a região física $r_{f,2}$ que está do outro lado da curva c_2 ;
 - (d) Trocar o nome das regiões $r_{f,1}$ e $r_{f,2}$ para r_{nf} (isso realiza a união das regiões);
 - (e) Deletar as curvas c_1 e c_2 da interface.

Após este processo, a interface da Figura 4.9c possuirá apenas duas regiões, r_1 e r_4 . Esta nova interface é apresentada na figura 4.11. Observe que, neste momento a interface não está mais embarçada, concluindo o algoritmo de desembarçamento.

4.3 Mudanças topológicas em escoamentos com o método MAC

Nas seções anteriores foi apresentado um método que realiza uma mudança topológica a partir de interfaces que contenham um embarçamento. Iremos agora mostrar como esse algoritmo interage com simulações numéricas de escoamentos realizadas pelo método MAC.

Vimos na seção 3.3 que o método MAC classifica as células da malha computacional em três categorias, de acordo com a posição das partículas marcadoras da interface. Essas categorias são: EMPTY, FULL e SURFACE. Vale lembrar que células SURFACE são aquelas nas quais condições de superfície livre serão aplicadas, e também que células FULL são todas aquelas que não possuem nenhuma EMPTY ao seu redor.

Com essas informações sobre a classificação de células em mente, vamos agora exemplificar como a mudança topológica é realizada no método MAC nos dois casos possíveis: o de união de blocos de fluido, e o da separação de um bloco de fluido que torna-se dois.

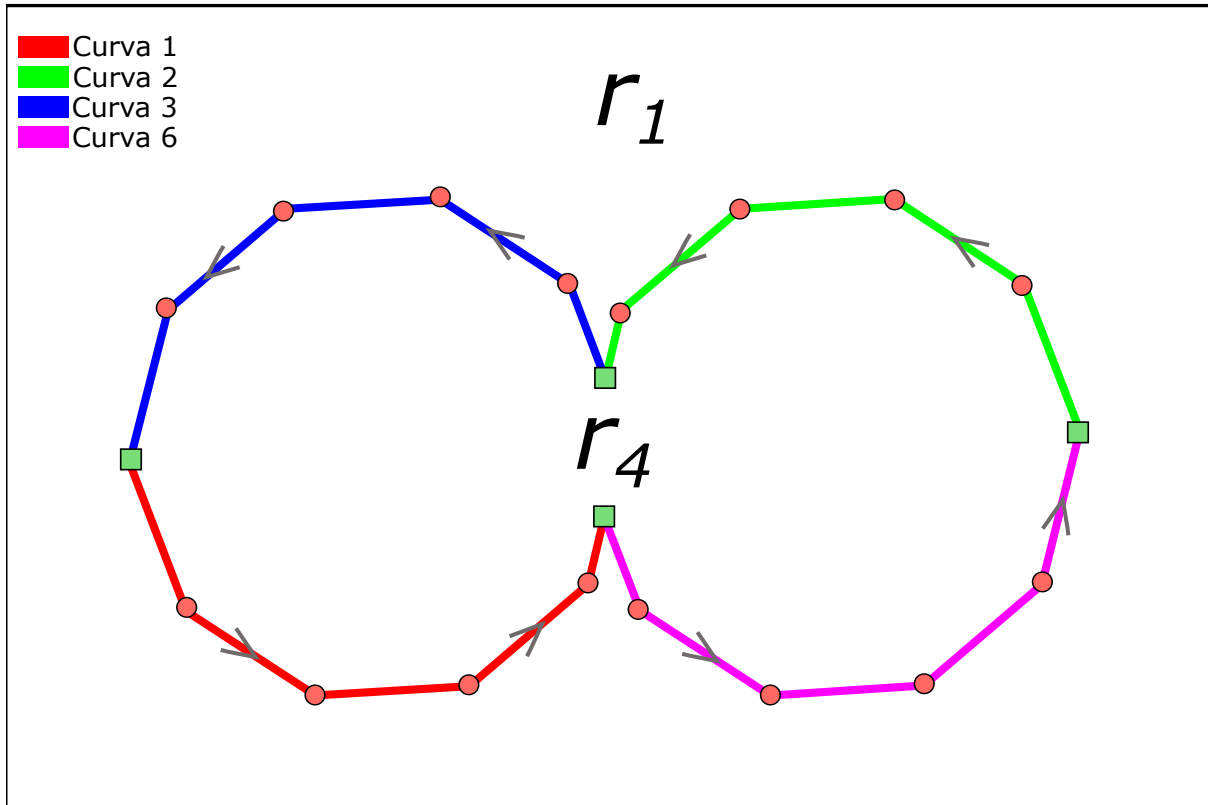


Figura 4.11: Interface do exemplo de união de bolhas (Figura 4.9c) após todas as etapas do algoritmo de embarçamento.

4.3.1 Caso 1: União

Iremos ilustrar o caso da união de superfícies através de um exemplo no qual duas gotas se juntam em um único bloco de fluido. Nesta simulação, duas gotas são lançadas uma acima da outra e, com o efeito da gravidade, atingem uma parede rígida inferior. A primeira gota atinge a parede diretamente, enquanto que a segunda gota se encontrará com a primeira, gerando uma união. A figura 4.12a mostra o estado inicial desta simulação e a figura 4.12b mostra um estado mais avançado no qual as gotas já se encontraram. Nestas duas figuras, células azuis, cinzas e vermelhas representam as categorias FULL, EMPTY e SURFACE, respectivamente.

Quando as partículas marcadoras das duas gotas ficam muito próximas, isto é, quando estão nas mesmas células ou em células adjacentes, o critério de classificação de células do método MAC faz com que estas células sejam classificadas como FULL. Isto ocorre, pois não haverá nenhuma célula EMPTY ao redor desta região, consequentemente transformando estas células em FULL. Neste momento, nenhuma condição de contorno será aplicada mais nesta região e as partículas de ambas as gotas serão advectadas com velocidades muito similares, impedindo que elas, de fato, se embarquem.

A figura 4.12c apresenta uma visualização ampliada deste estágio de união (os pontos destacados em verde e as setas serão explicados em breve). Podemos ver que há uma grande região com partículas marcadoras, mas que não possuem nenhuma célula SURFACE, apenas FULL. Essas partículas nunca geram um embarçamento na interface, pois sempre serão advectadas em conjunto. Desta forma, precisaremos forçar um embarçamento na interface para que o algoritmo de mudança topológica possa ser utilizado.

Sempre que houver uma região com muitas partículas marcadoras cujas células são FULL, iremos interpretar que tal região precisa sofrer uma mudança topológica de união.

Quando isto acontecer, um embaraçamento será criado na interface seguindo os seguintes passos:

1. Sejam A e B as curvas que compõem cada um dos dois blocos de fluido, respectivamente.
2. Percorreremos as partículas da curva A que estão dentro da região de células FULL, e armazenaremos aquelas que delimitam esta região. Isto nos dará duas partículas que chamaremos de A_1 e A_2 . Na figura 4.12c, por exemplo, se a curva A for a gota superior, A_1 seria a partícula mais a esquerda dentro da região FULL, e A_2 a partícula mais a direita (ou vice-versa).
3. Analogamente, percorreremos a curva B para encontrar as partículas B_1 e B_2 que delimitam sua região de células FULL.
4. As partículas da curva A que estiverem entre A_1 e A_2 irão trocar de posição com as partículas de B que estão entre B_1 e B_2 . Isto criará um embaraçamento das duas curvas como mostra a figura 4.12d.

Após os passos anteriores a interface estará embaraçada, e os cruzamentos ocorrem nos extremos da região de células FULL. Neste momento, podemos executar o algoritmo de mudança topológica e obteremos um único bloco de fluido, cuja interface pode ser vista na figura 4.12e. Em alguns testes, optamos também por finalizar este processo aplicando uma suavização pela curva de Bézier, o que nos dá a interface da figura 4.12f.

4.3.2 Caso 2: Separação

Vamos agora ilustrar como uma simulação com mudança topológica de separação foi realizada através do método MAC. Este caso será ilustrado com uma simulação em que um jato de fluido é injetado com uma velocidade prescrita no topo de um domínio computacional. Sob efeito da velocidade prescrita, da gravidade e da tensão superficial, este jato ficará cada vez mais fino ao longo do tempo, até que sofre uma quebra. A quebra gera uma parte de fluido ainda conectada com o injetor, e também uma gota que cai separadamente. Dois estados desta simulação podem ser vistos nas figuras 4.13a e 4.13b.

Da mesma forma como no caso da união de superfícies, o método MAC também não gera um embaraçamento da interface automaticamente neste caso. As curvas se tornam muito próximas, mas nunca se cruzam. Precisamos novamente utilizar algum critério e alguma metodologia para forçar um embaraçamento. O critério utilizado em nossas simulações é novamente baseado na classificação das células da malha. Desta vez, consideramos que uma mudança topológica deve ser realizada sempre que o jato estiver fino o suficiente de modo que ocupe apenas uma região de 2 células adjacentes. A figura 4.13c mostra uma ampliação da região em que isto ocorre em uma simulação.

Quando o critério for identificado, o embaraçamento forçado será realizado através dos seguintes passos:

1. Encontra-se o índice j (eixo y) das células onde o critério foi identificado;
2. Encontra-se dois pontos A_1 e B_1 , ambos localizados em células $j + 1$ e tais que: A_1 está na célula mais a esquerda do jato, e B_1 está na célula mais a direita do jato;
3. Os pontos A_1 e B_1 têm suas posições trocadas. Isto gera o embaraçamento superior visto na figura 4.13d.

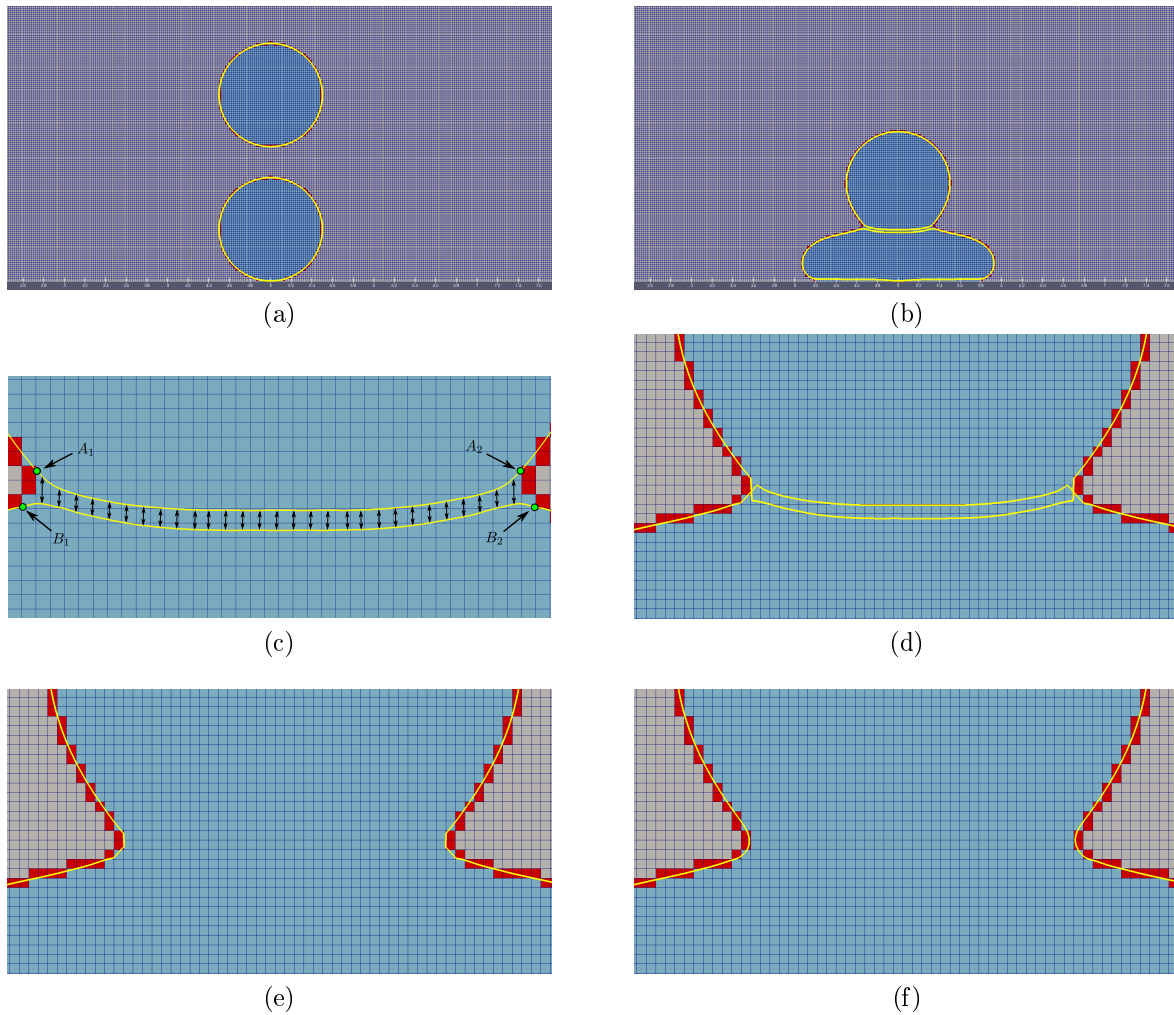


Figura 4.12: Ilustração da superfície livre durante o processo que gera o embaraçamento da interface no caso da união de blocos de fluido.

4. Encontra-se dois pontos A_2 e B_2 , ambos localizados em células $j - 1$ e tais que: A_2 está na célula mais a esquerda do jato, e B_2 está na célula mais a direita do jato;
5. Os pontos A_2 e B_2 têm suas posições trocadas. Isto gera o embaraçamento inferior também visto na figura 4.13d.

Neste momento, a interface da figura 4.13d já está embaraçada. Executando o algoritmo de mudança topológica, obtemos a interface da figura 4.13e. Esta interface possui os dois blocos de fluidos que desejamos e um terceiro extremamente pequeno (com tamanho pouco maior que 1 célula) entre eles. Este terceiro bloco será simplesmente removido, gerando um espaço de célula EMPTY entre os outros dois como mostra a interface da figura 4.13f.

4.4 Aproximação da curvatura da superfície livre

Vimos anteriormente que a equação sobre a superfície livre (2.45) é usada como condição de contorno para a equação de Poisson (3.4). Um dos termos presentes em (2.45) envolve a curvatura da superfície livre e, portanto, este valor precisa ser conhecido. Nesta seção será descrito o método utilizado para aproximar o valor desta curvatura. Este método foi proposto em [26].

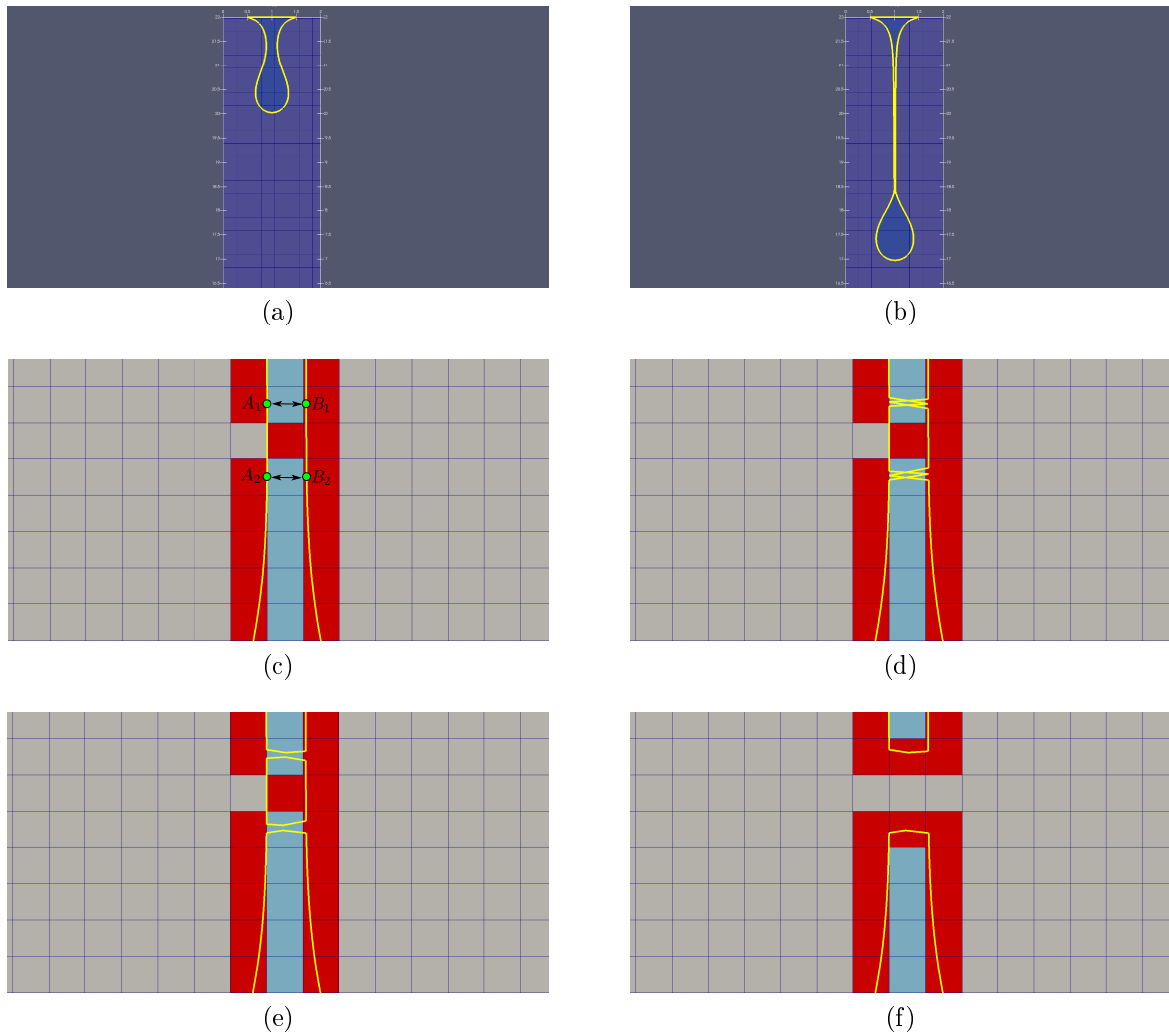


Figura 4.13: Ilustração da superfície livre durante o processo que gera o embaraçamento da interface no caso da separação de um bloco de fluido, tornando-se dois.

4.4.1 Algoritmo para o cálculo da curvatura

A descrição será feita dividindo o método em etapas, e cada etapa é ilustrada separadamente na figura 4.14.

- **Etapa 1: Escolha da célula**

A equação de contorno (2.45) é sempre aplicada no centro de uma célula, ou seja, em um ponto $\mathbf{x}_{i,j}$. Sendo assim, a primeira etapa do método é definir em qual célula desejamos aproximar a curvatura. Obrigatoriamente, esta célula deve possuir a classificação SURFACE, pois as etapas seguintes supõem que existam partículas marcadoras ao seu redor. A figura 4.14a ilustra uma célula (i, j) que pode ser usada neste método.

- **Etapa 2: Seleção de partículas**

A superfície livre completa pode ser definida por milhares de partículas marcadoras. Sendo assim, a segunda etapa do algoritmo é selecionar apenas algumas partículas que estejam próximas ao ponto central $\mathbf{x}_{i,j}$. Apenas as partículas selecionadas serão usadas nas próximas etapas do método. O critério de seleção utilizado aqui consiste na definição de uma circunferência com centro $\mathbf{x}_{i,j}$ e raio r (que deve ser escolhido). Todas as partículas dentro desta circunferência serão selecionadas para as próximas

etapas. Em nossos testes, utilizamos o raio $r = \min\{1.5\Delta_x, 1.5\Delta_y\}$. Outros valores também foram testados, e observamos que este apresentava bons resultados. É interessante frisar também que a quantidade de partículas utilizadas na criação da interface é relevante nesta etapa. Em nossos testes, nunca permitimos que a distância entre duas partículas consecutivas seja maior que $\min\{0.2\Delta_x, 0.2\Delta_y\}$, de modo que cada célula possui aproximadamente 5 partículas.

Esta etapa do método está ilustrada na figura 4.14b. Os pontos em azul foram selecionados para o método.

- **Etapa 3: Cálculo do vetor normal**

Na terceira etapa, desejamos encontrar o vetor normal a superfície livre nesta célula. Para isso, iremos aproximar a superfície por uma reta, que será calculada pelo método dos mínimos quadrados.

Seja $\mathbf{x}_k = (x_k, y_k)$ uma partícula marcadora selecionada na etapa anterior, com $k = 1, 2, \dots, m$. Gostaríamos de encontrar coeficientes a e b tais que a equação de reta $y_k = ax_k + b$ seja satisfeita para todos $k = 1, 2, \dots, m$. Obviamente, estes coeficientes não existem na grande maioria dos casos e, portanto, precisamos usar uma aproximação por mínimos quadrados para a solução deste sistema linear. Sendo assim, os coeficientes a e b serão calculados através do seguinte sistema linear

$$\begin{bmatrix} \mathbf{A}^t \mathbf{A} & \mathbf{A}^t \mathbf{B} \\ \mathbf{B}^t \mathbf{A} & \mathbf{B}^t \mathbf{B} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{Y}^t \mathbf{A} \\ \mathbf{Y}^t \mathbf{B} \end{bmatrix}, \quad (4.7)$$

onde

$$\mathbf{A} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}. \quad (4.8)$$

Com os coeficientes a e b já calculados, resta encontrar o vetor normal a reta $y = ax + b$. Este vetor é dado por

$$\mathbf{n} = (n_x, n_y) = \left(\frac{a}{\sqrt{a^2 + 1}}, \frac{-1}{\sqrt{a^2 + 1}} \right). \quad (4.9)$$

Se o sistema (4.7) for singular, o vetor normal será tomado como $\mathbf{n} = (1, 0)$.

Esta etapa do algoritmo é ilustrada na figura 4.14c. A reta $y = ax + b$ é apresentada em verde, e seu vetor normal em marrom.

- **Etapa 4: Mudança de base no plano cartesiano**

Nesta etapa iremos criar um novo sistema de coordenadas para o plano. Este novo sistema terá como origem o ponto $\mathbf{x}_{i,j}$ (centro da célula) e sua base é o conjunto de vetores $\{\mathbf{m}, \mathbf{n}\}$, em que \mathbf{n} é o vetor vindo da etapa anterior e $\mathbf{m} = (n_y, -n_x)$.

Chamaremos as coordenadas deste sistema de ξ e η , e elas se relacionam às coordenadas x e y pelas seguintes equações

$$\xi = (x - x_{i,j})n_y - (y - y_{i,j})n_x, \quad (4.10)$$

$$\eta = (x - x_{i,j})n_x + (y - y_{i,j})n_y, \quad (4.11)$$

em que $\mathbf{x}_{i,j} = (x_{i,j}, y_{i,j})$.

A definição deste novo sistema de coordenadas está ilustrada na figura 4.14d.

- **Etapa 5: Aproximação da superfície por uma parábola**

Na quinta etapa do método, desejamos encontrar uma parábola que aproxime a superfície livre ao redor desta célula, e esta parábola será criada no novo sistema de coordenadas da etapa anterior.

Para cada ponto selecionado \mathbf{x}_k , possuímos suas coordenadas equivalentes (ξ_k, η_k) , dadas pelas equações (4.10)-(4.11). Gostaríamos de encontrar coeficientes $\{c, d, e\}$ tais que a equação de parábola

$$\eta_k = c\xi_k^2 + d\xi_k + e \quad (4.12)$$

seja satisfeita para todos $k = 1, 2, \dots, m$. Claramente, estes coeficientes só existem em casos muitos particulares. Desta forma, iremos encontrar por mínimos quadrados a aproximação que melhor soluciona este sistema linear. Sendo assim, os coeficientes $\{c, d, e\}$ serão dados pelo sistema

$$\begin{bmatrix} \mathbf{C}^t \mathbf{C} & \mathbf{C}^t \mathbf{D} & \mathbf{C}^t \mathbf{B} \\ \mathbf{D}^t \mathbf{C} & \mathbf{D}^t \mathbf{D} & \mathbf{D}^t \mathbf{B} \\ \mathbf{B}^t \mathbf{C} & \mathbf{B}^t \mathbf{D} & \mathbf{B}^t \mathbf{B} \end{bmatrix} \begin{bmatrix} c \\ d \\ e \end{bmatrix} = \begin{bmatrix} \mathbf{E}^t \mathbf{C} \\ \mathbf{E}^t \mathbf{D} \\ \mathbf{E}^t \mathbf{B} \end{bmatrix}, \quad (4.13)$$

onde

$$\mathbf{C} = \begin{bmatrix} \xi_1^2 \\ \xi_2^2 \\ \vdots \\ \xi_m^2 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_m \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_m \end{bmatrix}. \quad (4.14)$$

A parábola resultante desta etapa é dada pela equação $\eta = c\xi^2 + d\xi + e$ e é ilustrada na figura 4.14e. Se o sistema (4.13) for singular, o método é interrompido e consideramos que a curvatura é nula.

- **Etapa 6: Cálculo da curvatura**

A última etapa consiste no cálculo direto da curvatura através da função $\eta = \eta(\xi)$ que define a parábola em (4.12). Como pode ser visto em [23], é possível calcular a curvatura de uma função através da expressão

$$\kappa = -\frac{\frac{\partial^2 \eta}{\partial \xi^2}}{\left(1 + \frac{\partial \eta}{\partial \xi}\right)^{\frac{3}{2}}}. \quad (4.15)$$

A curvatura obtém seu valor máximo (em módulo) no seu ponto crítico, ou seja, com $\frac{\partial \eta}{\partial \xi} = 0$, e é a curvatura deste ponto que iremos tomar como nossa aproximação. Desta forma, segue que a curvatura aproximada é dada por

$$\kappa = -\frac{\frac{\partial^2 \eta}{\partial \xi^2}}{(1 + 0)^{\frac{3}{2}}} = -\frac{\partial^2 \eta}{\partial \xi^2} = -\frac{\partial^2 (c\xi^2 + d\xi + e)}{\partial \xi^2} = -2c. \quad (4.16)$$

A expressão (4.16) determina a curvatura a não ser pelo sinal. O sinal pode ser corrigido, se necessário, comparando a normal \mathbf{n} obtida na etapa 3 com a normal \mathbf{n}_c aproximada pela classificação de células da figura 3.3. Se $\mathbf{n}^t \mathbf{n}_c < 0$, então o sinal da curvatura precisa ser invertido.

O valor obtido neste momento, é a curvatura aproximada que será usada na condição de contorno de superfície livre.

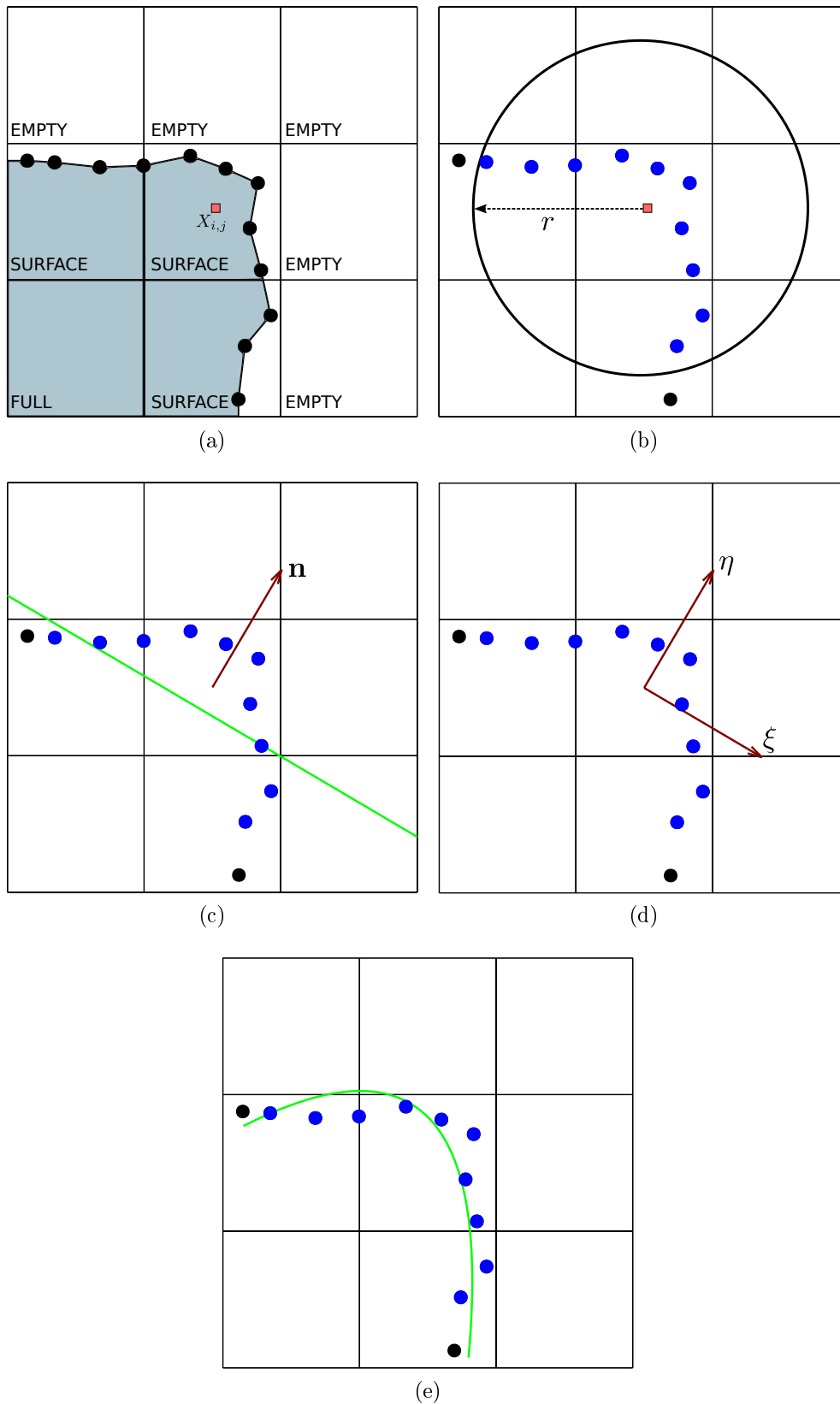
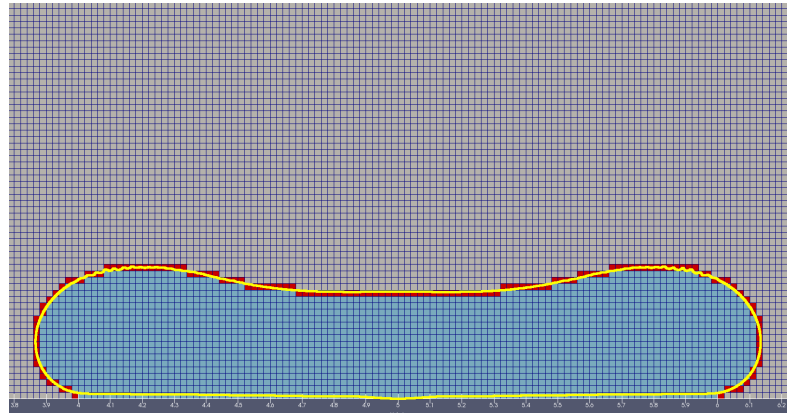


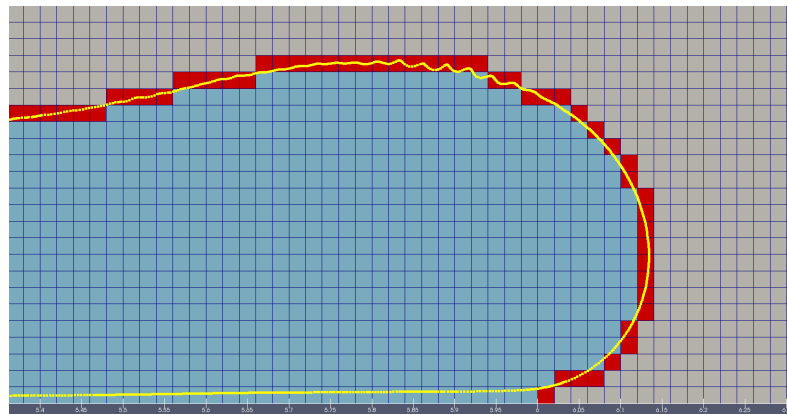
Figura 4.14: Ilustração das etapas realizadas para aproximação da curvatura da superfície livre em uma célula.

4.4.2 Dificuldades encontradas com o algoritmo

O algoritmo descrito acima para cálculo da curvatura é efetivo, porém algumas dificuldades também surgiram. Observamos que, ao realizarmos simulações nas quais a



(a) Visão geral



(b) Visão ampliada de uma região com ondulação

Figura 4.15: Exemplo de caso no qual a superfície livre apresenta ondulação geradas pelo cálculo da curvatura.

influência da tensão superficial é muito significativa, a superfície livre apresentava algumas ondulações inesperadas. A figura 4.15 mostra uma exemplo destas ondulações na simulação de uma gota impactando sobre uma parede rígida.

Analisando as células em que ondulações ocorrem, foi possível notar que sempre havia ao menos uma célula com uma descontinuidade extremamente grande no valor de sua curvatura em relação às suas vizinhas. Observamos também que estas células com curvatura inadequada sempre possuíam o sistema linear (4.13) quase singular. Devido a erros de arredondamento, a solução deste sistema leva a uma curvatura extremamente alta e muito distante dos valores vizinhos, gerando uma ondulação que se amplifica nos próximos passos temporais.

A quase singularidade do sistema (4.13) é um grande problema na utilização deste método. Mesmo em células nas quais a curvatura é calculada de forma adequada, ou seja, nos melhores casos, o determinante da matriz deste sistema sempre fica por volta de 10^{-7} .

Até o momento não conseguimos encontrar uma modificação deste método que eliminasse este problema. Pretendemos futuramente pensar em alguma outra alternativa para o cálculo da curvatura que não dependa de um sistema linear como este. Para que simulações numéricas pudessem ser realizadas mesmo com a existência destas imprecisões, optamos por suavizar a curva que compõe a superfície livre, eliminando tais ondulações. A estratégia utilizada até aqui foi suavizar a superfície através de uma curva de Bézier.

Uma curva de Bézier é uma curva definida de forma paramétrica a partir de uma quantidade finita de pontos dados, mas que não interpola estes pontos. Ou seja, diferente da interpolação, a curva de Bézier não irá passar exatamente sobre estes pontos. Sejam

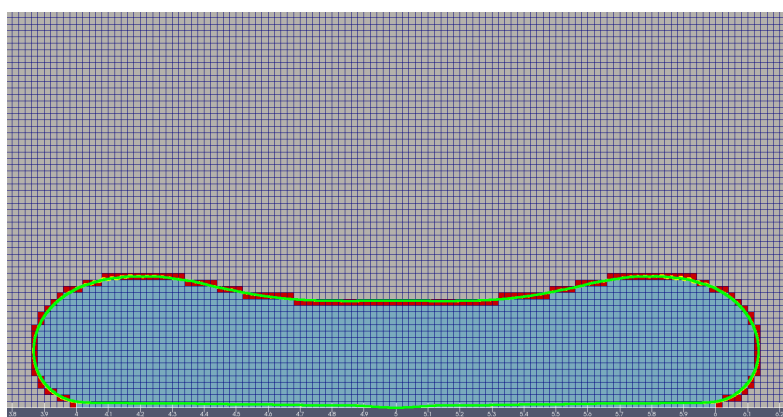
$\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_n$ pontos no plano cartesiano, e $\mathbf{B}_{\mathbf{x}_0\mathbf{x}_1\dots\mathbf{x}_n}$ a curva de Bézier definida por estes pontos. Esta curva é dada pela seguinte fórmula de recursão

$$\begin{cases} \mathbf{B}_{\mathbf{x}_i}(t) = \mathbf{x}_i, & i = 0, 1, \dots, n, \\ \mathbf{B}_{\mathbf{x}_0\mathbf{x}_1\dots\mathbf{x}_n}(t) = (1-t)\mathbf{B}_{\mathbf{x}_0\mathbf{x}_1\dots\mathbf{x}_{n-1}}(t) + t\mathbf{B}_{\mathbf{x}_1\dots\mathbf{x}_n}(t), & 0 \leq t \leq 1. \end{cases} \quad (4.17)$$

Pela fórmula (4.17), podemos notar que a curva de Bézier $\mathbf{B}(t) = \mathbf{B}_{\mathbf{x}_0\mathbf{x}_1\dots\mathbf{x}_n}$ é tal que $\mathbf{B}(0) = \mathbf{x}_0$ e $\mathbf{B}(1) = \mathbf{x}_n$. Para valores em $(0, 1)$ ela não irá necessariamente passar sobre os pontos $\{\mathbf{x}_1, \dots, \mathbf{x}_{n-1}\}$, o que gera a suavização da interface. A figura 4.16 mostra as partículas de uma interface original com ondulações (em amarelo) e pontos calculados através da curva de Bézier obtida de tais partículas (em verde). Mais detalhes sobre a construção e propriedades das curvas de Bézier podem ser vistos em [1] e [18].

Uma dificuldade na utilização da curva de Bézier diz respeito a conservação de massa na simulação. Essa aproximação não possui nenhuma propriedade de conservação e notamos que, de fato, a massa do fluido é alterada. Se a curva de Bézier for aplicada muitas vezes durante uma simulação, observamos que a alteração na massa se torna muito significativa, danificando a solução. Desta forma, sempre aplicávamos a suavização por curva de Bézier apenas em alguns passos temporais da simulação, de modo a minimizar a alteração na massa.

Uma técnica de suavização conhecida como “*Trapezoidal subgrid undulations removal*” (TSUR) foi proposta também em [26]. Esta técnica garante a conservação de massa no fluido, porém, por falta de tempo, ela ainda não foi testada aqui. Pretendemos realizar a implementação desta técnica futuramente.



(a) Visão geral



(b) Visão ampliada de uma região com ondulação

Figura 4.16: Comparação entre uma superfície livre com ondulação (amarelo) e a nova superfície calculada através de sua curva de Bézier (verde).

Resultados Numéricos

5.1 Escoamentos confinados

Como comentado na introdução, as duas principais motivações desse projeto foram a construção de um código que utilize uma malha não-uniforme e o estudo e implementação de técnicas para tratamento de escoamentos com mudanças topológicas no domínio do problema. Desta forma, nesta seção iremos abordar dois tipos de escoamentos: confinados e com superfícies livres.

Para validar o código que foi criado no projeto, vamos resolver alguns testes clássicos em Mecânica e Reologia Computacional. A implementação deste código foi toda feita em linguagem C e a biblioteca PETSc [5] foi utilizada para resolução de sistemas lineares. A visualização dos dados foi feita através dos *softwares* Gnuplot e Paraview [2].

5.1.1 Escoamento na cavidade (*lid-driven cavity*)

Este problema é bastante conhecido na literatura de dinâmica de fluidos e muito usado na validação de métodos numéricos. Trata-se de um escoamento dentro de uma caixa quadrada cuja tampa se move com velocidade constante.

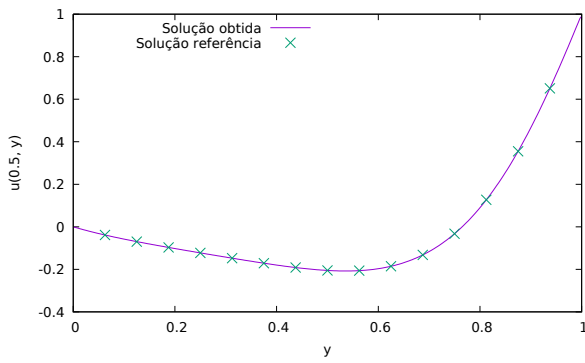
Para verificar os resultados obtidos com este teste, criamos um modelo idêntico ao apresentado em [27]. Este modelo foi criado com os seguintes parâmetros adimensionais para um fluido Newtoniano:

- Largura e altura da caixa: $\frac{D}{L} = 1$;
- Velocidade constante da tampa: $\frac{U_{tampa}}{U} = 1$;
- Instante de tempo final: $t_{max} = 30$;
- Passo temporal: $\Delta_t = 10^{-2}$;
- Número de Reynolds: varia-se em cada simulação.

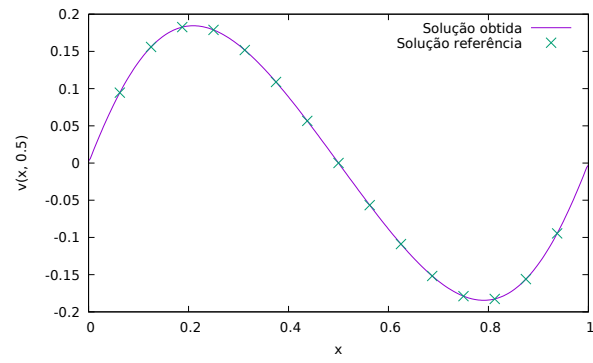
A discretização neste teste foi feita com uma malha uniforme com $N_x = N_y = 300$.

Em [27] são apresentados valores das velocidades u e v em dois cortes do domínio. A velocidade u é apresentada em um corte vertical feito em $x = 0.5$. A velocidade v é dada em um corte horizontal em $y = 0.5$. Utilizamos estes valores dados no artigo como uma referência para validar o resultado obtido pelo código implementado aqui.

$Re = 0.01$

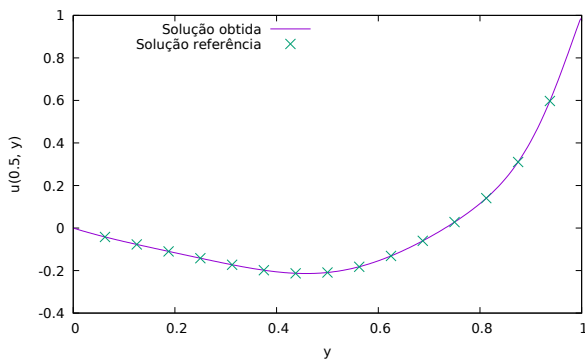


(a)

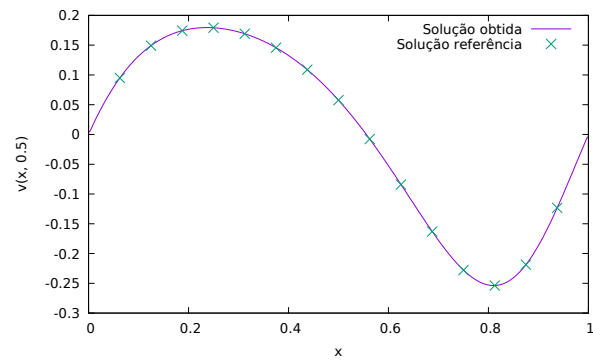


(b)

$Re = 100$

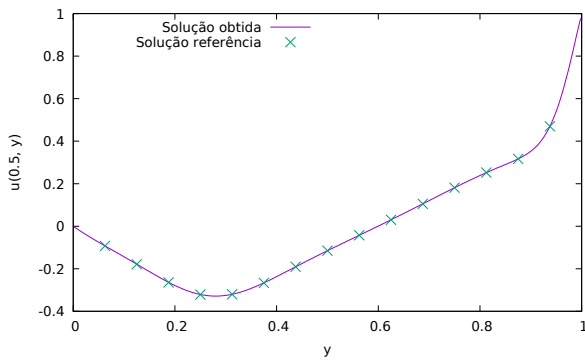


(c)

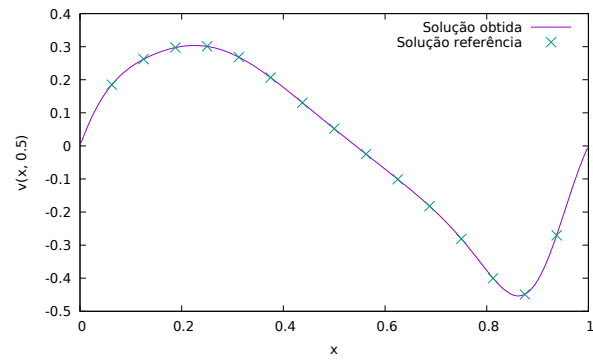


(d)

$Re = 400$



(e)



(f)

Figura 5.1: Comparação entre solução numérica obtida pelo método implementado e solução de referência obtida de [27].

A Figura 5.1 apresenta a comparação do resultado obtido aqui com a solução de referência em [27]. Observa-se que o método funciona como esperado tanto para valores baixos de Re , quanto para valores altos.

Na Figura 5.2 mostramos as linhas de corrente deste escoamento com os diferentes números de Reynolds. Como esperado, os vórtices crescem em tamanho junto com Re .

5.1.2 Escoamento com contração

Nesta seção iremos aplicar as formulações CSF e NSF para resolver um escoamento com quina bastante conhecido: o escoamento com contração. Para isso, foi adotada uma

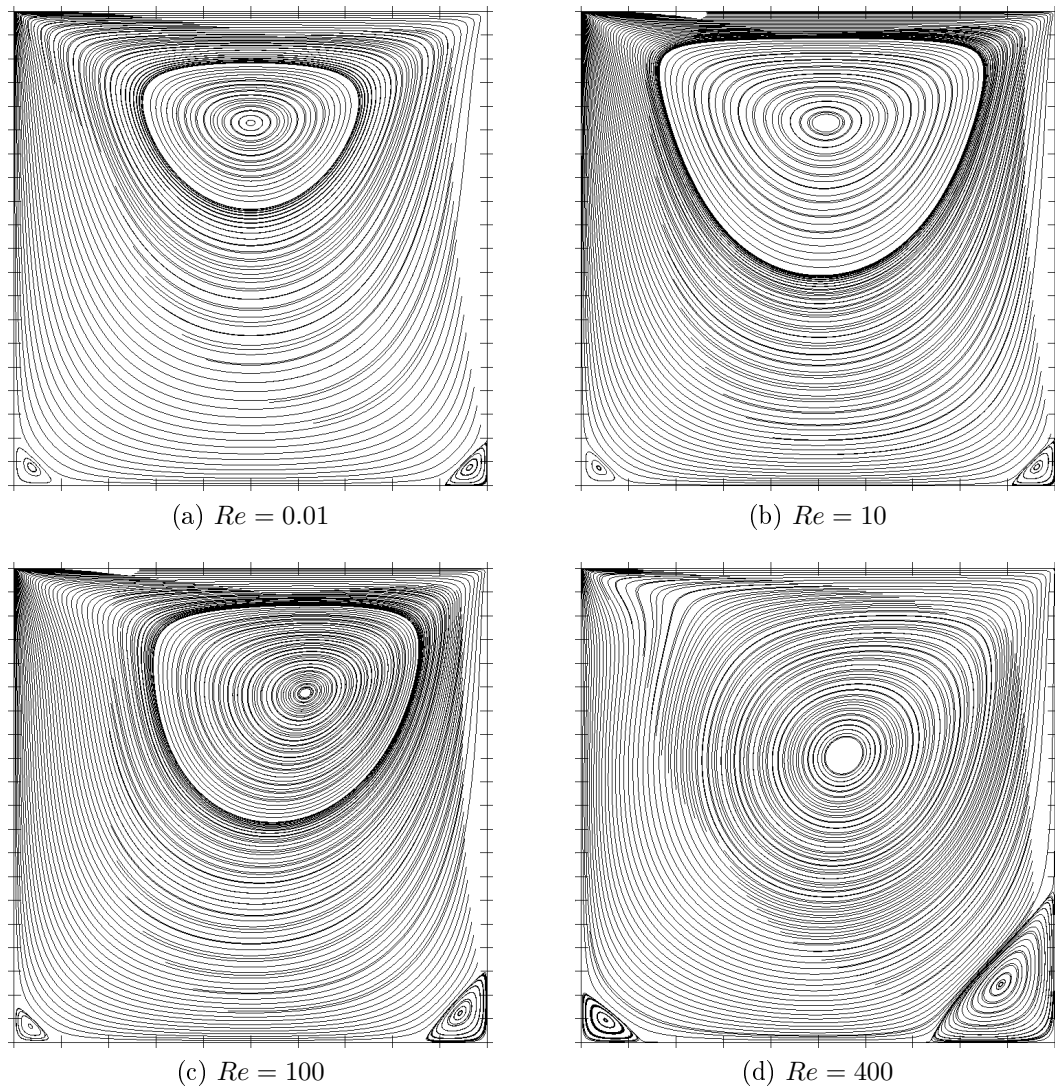


Figura 5.2: Linhas de corrente do escoamento na cavidade com diferentes números de Reynolds.

geometria com contração conforme a Figura 5.3. O domínio possui um canal de entrada e um de saída, cada um com comprimento $20L$, onde L (usado na adimensionalização) é metade da altura do canal de saída.

Consideramos que o fluido deste escoamento é viscoelástico e os seguintes parâmetros adimensionais foram fixados em todas as simulações:

- Número de Reynolds: $Re = 0.01$;
- Número de Weissenberg: $Wi = 1.0$;
- Razão das viscosidades: $\beta = 0.5$;
- Tolerância NSF: $tol = 10^{-6}$;
- *Inflow*: perfil desenvolvido de \mathbf{u} e \mathbf{T} , com velocidade máxima $\frac{u_{max}}{U} = 0.375$;
- Passo temporal: $\Delta_t = 10^{-4}$;

Três malhas foram utilizadas: uma uniforme (M1) e duas não-uniformes (M2 e M3). Detalhes sobre estas malhas são apresentados na tabela 5.1. A estratégia para construir

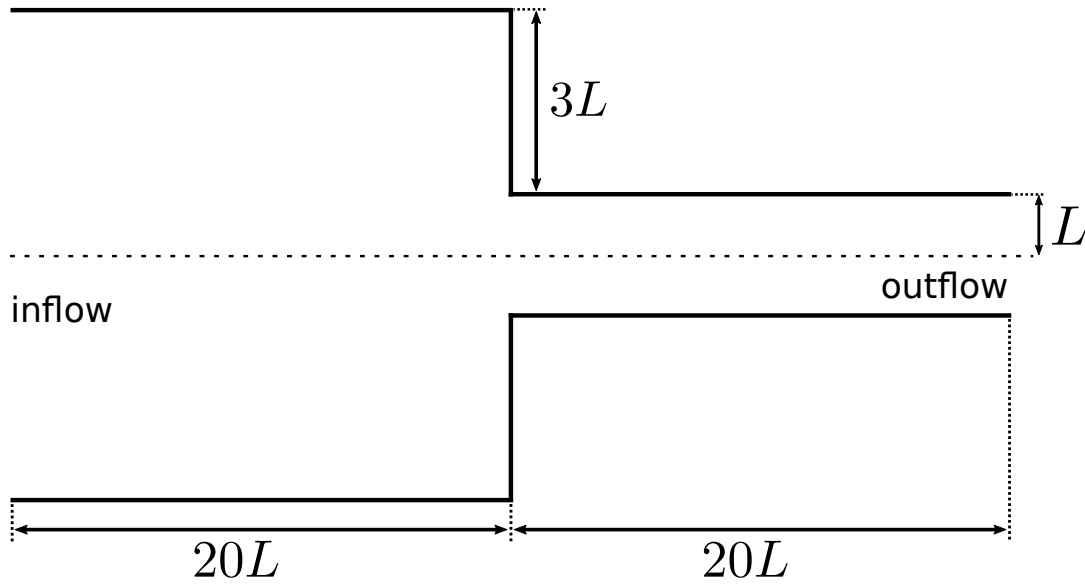


Figura 5.3: Geometria da contração com razão 4:1 usada nas simulações.

as malhas não-uniformes é similar às presentes nos artigos recentes [10, 36], isto é, a malha é mais refinada nas regiões da geometria que possuem quinas. A Figura 5.4 apresenta a estrutura das malhas não-uniformes que foram utilizadas.

Tabela 5.1: Detalhes das malhas uniforme e não-uniformes para a geometria com contração.

Malha	Espaçamento	Núm. de células	Núm. de células em x e y
M1 (uniforme)	$\delta x = \delta y = 0.05$	32000	320x160
M2 (não-uniforme)	$\delta x_{min} = \delta y_{min} = 0.008$	23800	140x260
M3 (não-uniforme)	$\delta x_{min} = \delta y_{min} = 0.004$	39600	180x340

O principal estudo feito para validar os resultados na contração, foi a análise assintótica das propriedades próximas à quina. Esse estudo foi conduzido de forma analítica em [19, 13] e diz que, ao se aproximar da quina de contração, temos

$$\mathbf{T} \propto r^{-\frac{2}{3}} \quad \text{e} \quad \mathbf{u} \propto r^{\frac{5}{9}}, \quad (5.1)$$

em que r é a distância de um ponto até a quina.

Iremos agora realizar uma comparação entre os resultados obtidos através das formulações NSF e CSF ao redor da quina. Para isso, vamos verificar se as componentes de \mathbf{T} se aproximam da quina de acordo com a previsão analítica (5.1). Seja r a distância de um ponto até a quina superior da geometria, queremos visualizar o valor de cada componente de \mathbf{T} quando r tende a 0. Para isso, vamos fixar um eixo vertical para r conforme a figura 5.5 e investigar o valor do tensor sobre este eixo.

A Figura 5.6 traz o valor de cada uma das propriedades de interesse em função da distância à quina (r). Esta figura deixa evidente que a NSF produziu resultados para $\log(|T^{xx}|)$ e $\log(|T^{xy}|)$ mais precisos que aqueles obtidos pela CSF. Em particular, de acordo com 5.6d, podemos observar uma melhora significativa para a pressão próxima da quina quando a formulação NSF é aplicada. Quando se trata do comportamento numérico da pressão comparado a resultados teóricos próximos a quina, a literatura atual é

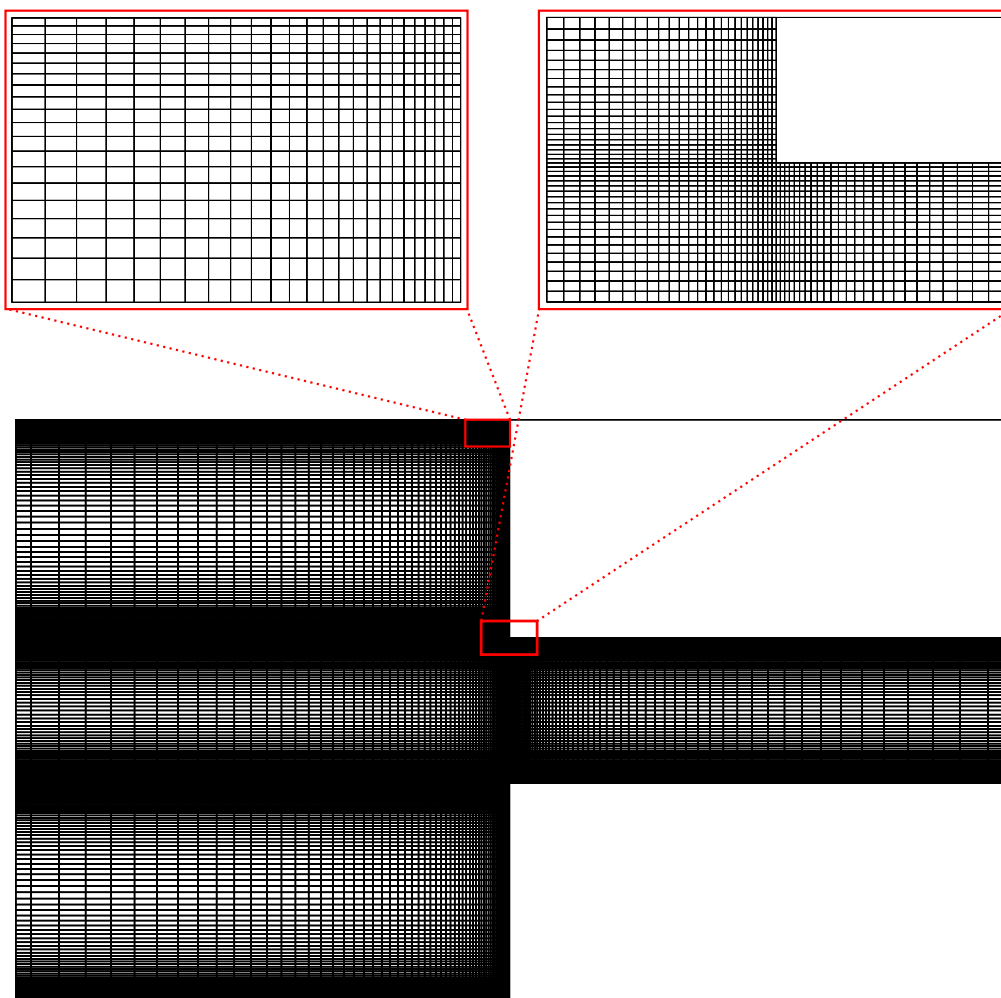


Figura 5.4: Estrutura das malhas não-uniformes usadas na geometria com contração.

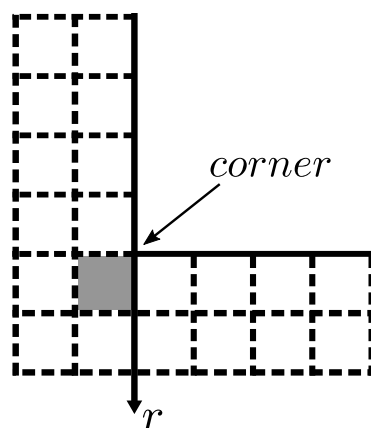


Figura 5.5: Representação da quina de contração e da direção r na qual a análise assintótica será feita.

muito escassa. Os resultados numéricos apresentados em [22] usando CSF, por exemplo, mostraram uma dificuldade para capturar um comportamento correto para a pressão próxima da quina. Desta forma, os resultados numéricos apresentados nas figuras 5.6a-5.6d evidenciam o potencial da formulação NSF para resolver com maior precisão escoamentos com quinas na geometria.

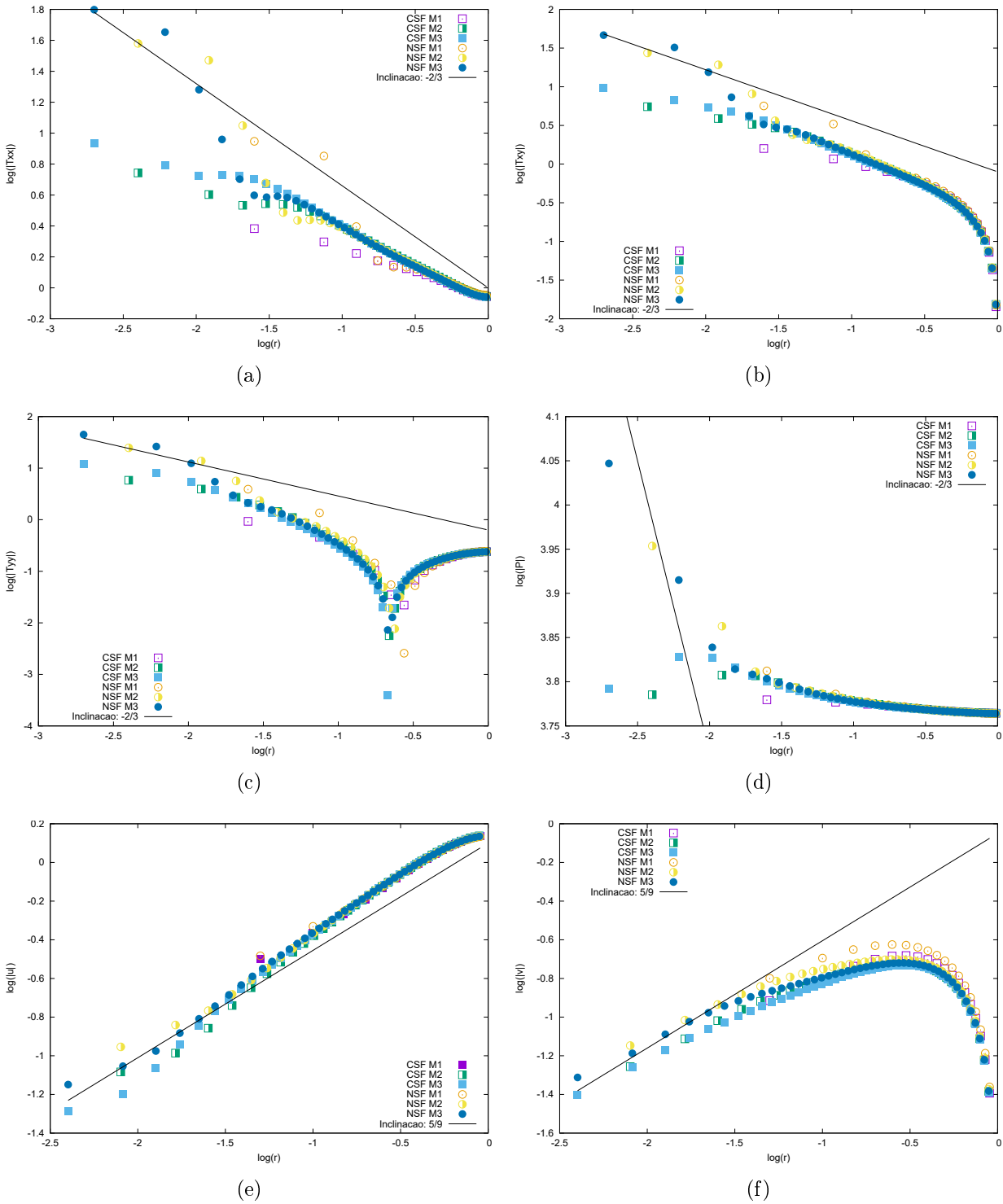


Figura 5.6: Valores de diversas propriedades de interesse em pontos próximos a uma quina da contração. O eixo horizontal representa a distância (r) de um dado ponto até a quina, e o eixo vertical é o valor da propriedade em questão neste ponto. As inclinações destacadas em cada imagem vêm da análise teórica realizada em [19, 13].

5.1.3 Escoamento no *cross-slot*

O último teste que será apresentado aqui é conhecido como *Cross-Slot*. Nesta simulação o escoamento acontece em um domínio com geometria de cruz. O fluido é injetado pelos extremos esquerdo e direito do domínio, e é ejetado pelos extremos superior e inferior.

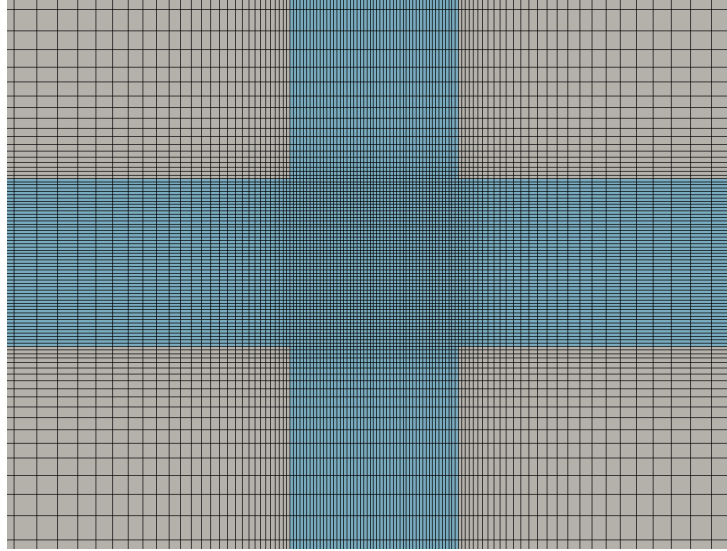


Figura 5.7: Malha não-uniforme utilizada na simulação *Cross-Slot*. A imagem mostra apenas a malha ao redor da região de maior refinamento. As células em azul são aquelas pelas quais há escoamento.

O modelo deste problema foi feito de forma idêntica ao apresentado em [38]. Os parâmetros adimensionais usados neste modelo são:

- Diâmetro do canal por onde há escoamento: $\frac{D}{L} = 1$;
- Comprimento (direção x) total do canal: $\frac{21D}{L}$;
- Altura (direção y) total do canal: $H = \frac{121D}{L}$;
- Velocidade no inflow: perfil parabólico com velocidade máxima $\frac{u_{max}}{U} = 1.5$;
- Passo temporal: $\Delta_t = 2 \times 10^{-4}$;
- Número de Reynolds: será variado em cada teste.

A malha computacional foi criada de modo não-uniforme. Em ambas as direções o maior refinamento é feito no centro da cruz e, gradualmente, o espaçamento é aumentado em direção aos extremos do domínio. Ao todo, são utilizadas $N_x = 300$ células na direção x e $N_y = 1300$ na direção y , com um espaçamento mínimo de $\Delta_x = \Delta_y = 0.01$ localizado no centro da cruz.

A Figura 5.7 mostra a malha criada para esta simulação ao redor da região de maior refinamento.

O artigo [38] apresenta um estudo do comprimento dos vórtices que este escoamento gera nos quatro cantos próximos ao centro da cruz. Observa-se que este comprimento (na direção y) cresce conforme o número de Reynolds é aumentado. Sendo assim, realizamos diversas simulações, cada uma com um número de Reynolds, e comparamos o comprimento obtido com aquele apresentado em [38].

A tabela 5.2 mostra essa comparação para os quatro cantos da cruz e para vários números de Reynolds. As colunas com título C_{obt} trazem o comprimento obtido pelas simulações deste trabalho. As colunas C_{ref} são os comprimentos de referência dados em [38]. Notamos que o método apresentou resultados com comportamento semelhante ao da referência.

Re	Vórtice Inferior Esquerdo		Vórtice Inferior Direito		Vórtice Superior Esquerdo		Vórtice Superior Direito	
	C_{obt}	C_{ref}	C_{obt}	C_{ref}	C_{obt}	C_{ref}	C_{obt}	C_{ref}
200	0.280648	0.331	0.280648	0.331	0.269020	0.331	0.269020	0.331
400	0.692302	0.744	0.692302	0.744	0.678199	0.744	0.678199	0.744
600	1.073996	1.137	1.057597	1.137	1.057597	1.137	1.041298	1.137
800	1.386725	1.534	1.386725	1.534	1.386725	1.534	1.386725	1.534

Tabela 5.2: Comparação entre o comprimento do vórtice obtido neste trabalho com valores de referência obtidos em [38] para diferentes números de Reynolds.

5.2 Escoamentos com superfície livre

Como segunda etapa de verificação do código, vamos apresentar alguns resultados de escoamentos com superfícies livres. Inicialmente, antes de mostrar a eficiência do método numérico para tratar mudanças topológicas, apresentaremos um teste clássico de escoamentos com superfícies, o inchamento do extrudado. A seguir, apresentaremos os resultados numéricos de problemas com mudanças topológicas.

5.2.1 Inchamento do extrudado (*die-swell*)

O *die-swell* é um escoamento com superfície livre no qual fluido é injetado por um canal de placas paralelas e, após uma certa distância, sai do canal. Experimentos e simulações da literatura mostram que, no momento da saída do fluido, há um *inchamento* que aumenta a espessura do jato. A literatura aponta também que a intensidade deste inchamento está relacionada às características não-newtonianas do fluido.

Para verificar a solução obtida neste escoamento não-newtoniano, realizamos uma modelagem idêntica à feita em [30], usando os seguintes parâmetros adimensionais:

- Altura do canal de entrada: $\frac{H}{L} = 1$;
- Comprimento do canal de entrada: $\frac{4H}{L}$;
- Distância entre o final do canal e o *Outflow*: $\frac{6H}{L}$;
- Velocidade no *inflow*: perfil parabólico com velocidade máxima $\frac{u_{max}}{U} = 1.0$;
- Número de Reynolds: $Re = 0.1$;
- Parâmetros viscoelásticos: $Wi = 0.5$ e $\beta = 0.5$;
- Tensão superficial: efeito desconsiderado (usa-se $\kappa = 0$);
- Malha computacional: uniforme com $\Delta_x = \Delta_y = 0.05$;
- Passo temporal: $\Delta_t = 5 \times 10^{-5}$.

A medida mais utilizada para validar este escoamento é a espessura do jato após o processo de inchamento. Na tabela 5.3 mostramos os valores desta medida obtidos em nossas simulações com as formulações NSF e CSF. O valor de referência vindo de [30] está bastante próximo do obtido com a formulação CSF em nossa simulação, já que esta é a formulação utilizada também na referência. Com a NSF, o diâmetro do jato se torna

Tabela 5.3: Espessura do jato após o processo de inchamento pelos métodos CSF e NSF.

Formulação	Espessura
CSF	1.476096
NSF	1.592435
Referência [30]	1.49

consideravelmente maior, mostrando que o tratamento preciso do escoamento nas quinas traz modificações significativas no resultado.

Até o presente momento da literatura, aparentemente não há nenhum resultado numérico com a formulação NSF aplicada ao problema do inchamento do extrado. Portanto, com o objetivo de ilustrar a aplicação dessa técnica, apresentamos na Figura 5.8 a superfície livre do escoamento *die-swell* em diferentes instantes de tempo.

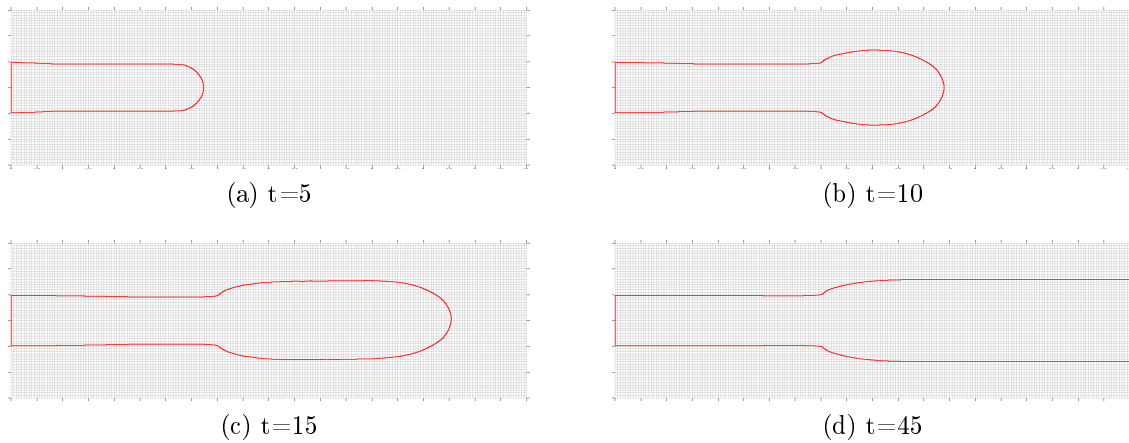


Figura 5.8: Visualização da superfície livre na simulação *die-swell* em diferentes instantes de tempo.

5.2.2 Impacto entre gota e uma camada de fluido

Este primeiro teste com superfície livre, tem apenas o objetivo de ilustrar as alterações geradas pelo algoritmo de mudança topológica unido a solução das equações governantes. Para isso, realizamos duas simulações: uma delas resolve as equações sem usar o algoritmo de mudança, e a outra resolve usando-o. A comparação entre as duas soluções nos permite visualizar o efeito do algoritmo.

Neste teste, uma gota é lançada sob o efeito da gravidade até cair dentro de uma camada de fluido. Testes deste tipo foram realizados, por exemplo, em referências como [24, 25, 53]. Nestas simulações, consideramos um fluido Newtoniano e os seguintes parâmetros adimensionais foram usados:

- Diâmetro da gota: $\frac{D}{L} = 1.0$;
- Posição inicial do centro da gota: $(1.25, 2)$;
- Largura da camada de fluido: $\frac{P}{L} = 2.5$;
- Altura da camada de fluido: $\frac{H}{L} = 0.5$;

- Número de Reynolds: $Re = 5.0$
- Número de Froude: $Fr = 2.2575$
- Tensão superficial: efeito desconsiderado (usa-se $\kappa = 0$);
- Velocidade inicial da bolha $\frac{\mathbf{u}}{U} = (0, -1)$;
- Passo temporal: $\Delta_t = 10^{-3}$.

A malha computacional foi criada da seguinte forma:

- Na direção x utilizamos uma discretização uniforme com $N_x = 100$ células;
- Na direção y utilizamos uma discretização não-uniforme que é mais refinada na região onde há o impacto da gota ($y = 0.5$). O menor espaçamento na direção y é $\Delta_y^{min} = 0.01$ e, ao todo, a malha tem $N_y = 70$ células nessa direção.

As Figuras 5.9 e 5.10 trazem, visualmente, o resultado desta simulação. Seis instantes de tempo são apresentados. Para destacar o funcionamento do algoritmo de mudança topológica, apresentamos, lado a lado, o resultado da simulação com e sem a utilização do algoritmo. Observamos que os resultados são bem semelhantes no que tange a efeitos físicos, ou seja, o algoritmo apenas modifica a região de contato entre as interfaces da gota e da camada de fluido.

A seguir, verificaremos qualitativamente se a solução obtida aqui se comporta de forma semelhante a apresentada em artigos como [24, 25, 53] conforme o número de Reynolds é variado. Espera-se que, ao aumentar o valor do número de Reynolds, o impacto crie uma tendência a formação do efeito de *splash* que gera uma coroa de fluido ao redor da região de impacto.

Novamente, consideramos um fluido Newtoniano e os seguintes parâmetros adimensionais foram utilizados na criação do teste:

- Diâmetro da gota: $\frac{D}{L} = 0.2$;
- Posição inicial do centro da gota: $(1.25, 0.3)$;
- Largura da camada de fluido: $\frac{P}{L} = 2.5$;
- Altura da camada de fluido: $\frac{H}{L} = 0.1$;
- Número de Reynolds: varia-se;
- Número de Froude: $Fr = 2.26$;
- Tensão superficial: efeito desconsiderado (usa-se $\kappa = 0$);
- Velocidade inicial da bolha: $\frac{\mathbf{u}}{U} = (0, -1)$;
- Passo temporal: $\Delta_t = 10^{-3}$;
- Malha computacional: uniforme com $\Delta_x = 10^{-2}$ e $\Delta_y = 10^{-2}$.

Os resultados deste teste são apresentados na Figura 5.11. Observamos que, de fato, o comportamento descrito na literatura é capturado pelo método implementado aqui. A presença da coroa de fluido é bem visível e cresce com o aumento do número de Reynolds.

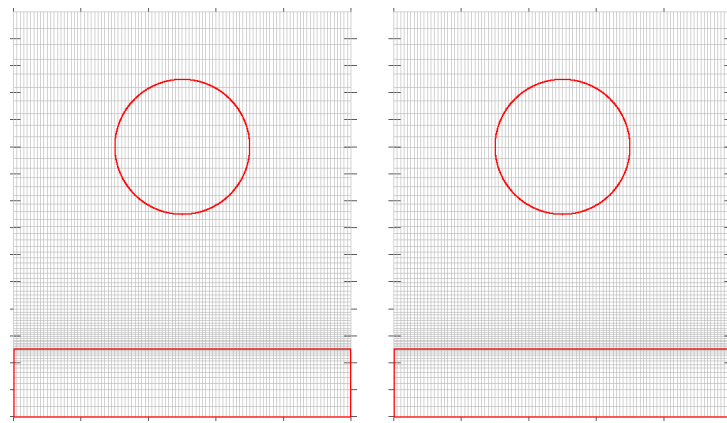
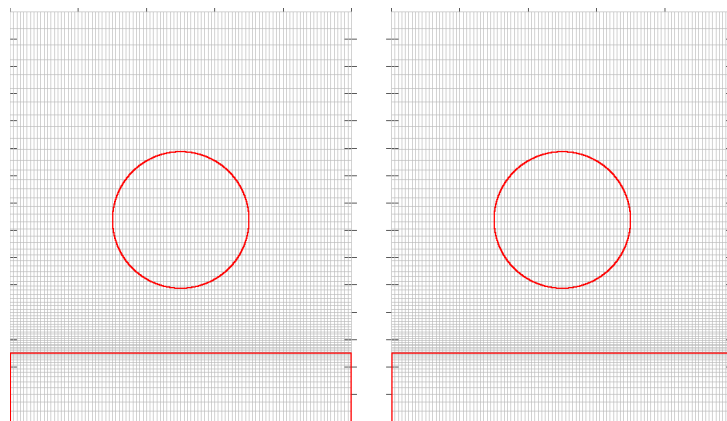
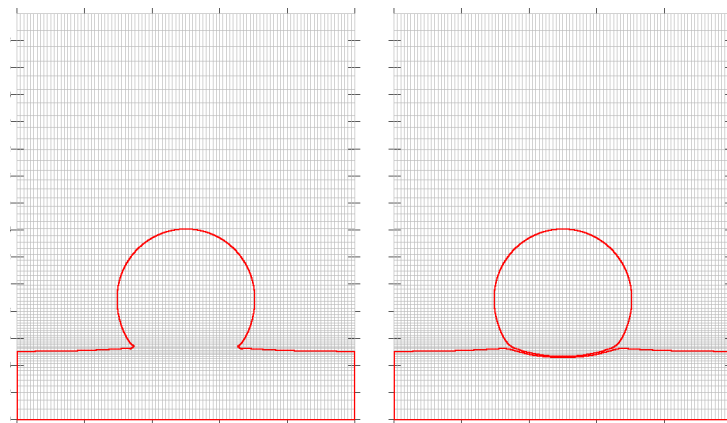
(a) Instante: $t = 0$.(b) Instante: $t = 0.5$.(c) Instante: $t = 1.0$.

Figura 5.9: Instantes da simulação do impacto de uma gota. A coluna da esquerda representa a simulação realizada usando o algoritmo de mudança topológica. A coluna da direita não usa o algoritmo de mudança topológica.

5.2.3 Escoamento no *cross-slot* com superfície livre

Para testar novamente o funcionamento do algoritmo de mudanças topológicas, realizamos uma segunda simulação na geometria *cross-slot*, mas dessa vez utilizando superfície livre.

Considera-se que o canal não possui fluido no tempo inicial e, ao longo do tempo, o fluido será injetado até cobrir toda a geometria. Em determinado momento, os dois jatos se encontram e a mudança topológica é realizada.

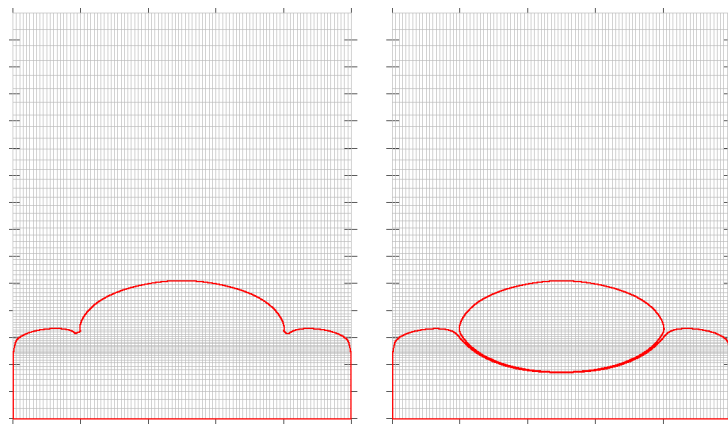
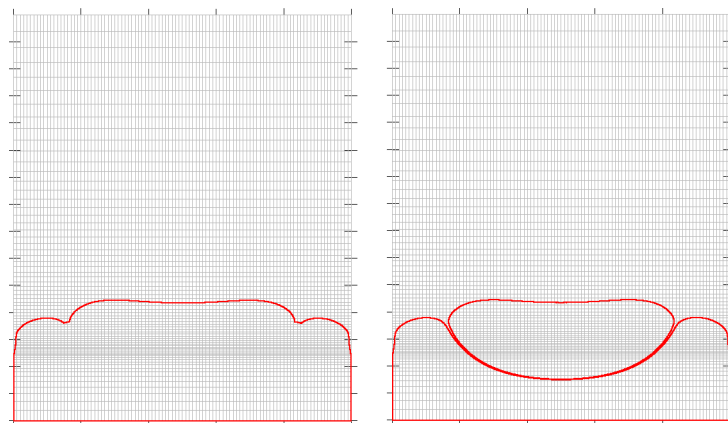
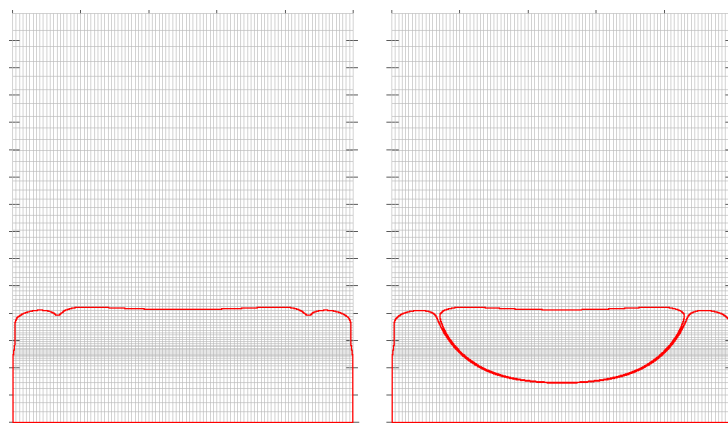
(a) Instante: $t = 1.5$.(b) Instante: $t = 3$.(c) Instante: $t = 21$.

Figura 5.10: Instantes da simulação do impacto de uma gota. A coluna da esquerda representa a simulação realizada usando o algoritmo de mudança topológica. A coluna da direita não usa o algoritmo de mudança topológica.

Os seguintes parâmetros foram usados nesta simulação para um fluido viscoelástico

- Diâmetro do canal por onde há escoamento: $\frac{D}{L} = 1$;
- Distância entre *inflow* e centro da cruz: $\frac{3.5D}{L}$;
- Distância entre *outflow* e centro da cruz: $\frac{3.5D}{L}$;
- Velocidade no inflow: perfil parabólico com velocidade máxima $\frac{u_{max}}{U} = 1$;

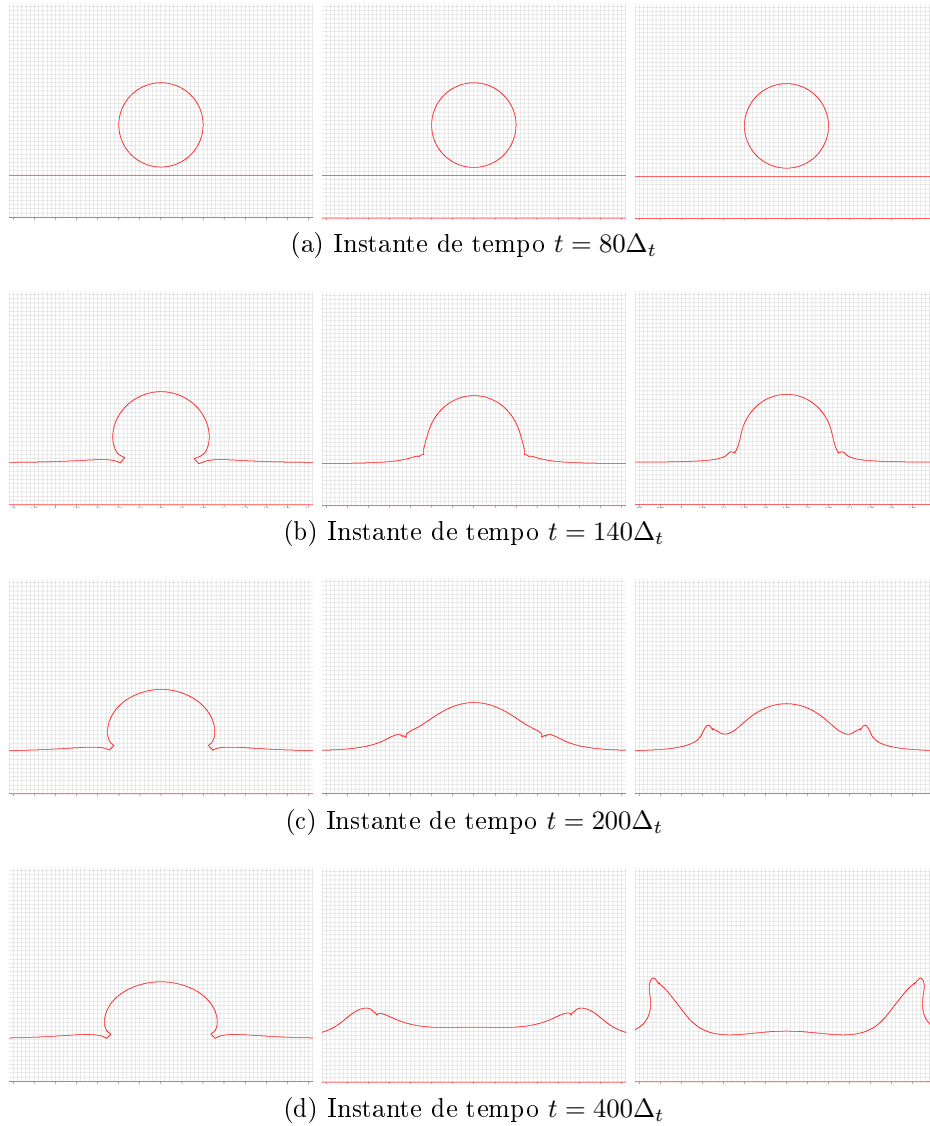


Figura 5.11: Visualização do momento de impacto da gota para diferentes valores de Reynolds (Re). A coluna da esquerda é uma simulação com $Re = 10$. A coluna central com $Re = 80$, e a da direita com $Re = 200$.

- Número de Reynolds: $Re = 0.01$;
- Parâmetros viscoelásticos: $Wi = 1.0$ e $\beta = 0.5$ com formulação CSF;
- Tensão superficial: efeito desconsiderado (usa-se $\kappa = 0$);
- Malha computacional: não-uniforme com $\Delta_x^{min} = \Delta_y^{min} = 0.02$ no centro da cruz;
- Passo temporal: $\Delta_t = 10^{-3}$.

A figura 5.12 mostra o movimento das interfaces envolvidas nesse escoamento para diferentes instantes de tempo. Nota-se que o algoritmo de mudança topológica consegue capturar o encontro das duas frentes de fluido, unindo e mantendo a física consistente desse problema. Além disso, na figura 5.13, apresentamos os campos de velocidade e pressão do escoamento antes do preenchimento total do domínio.

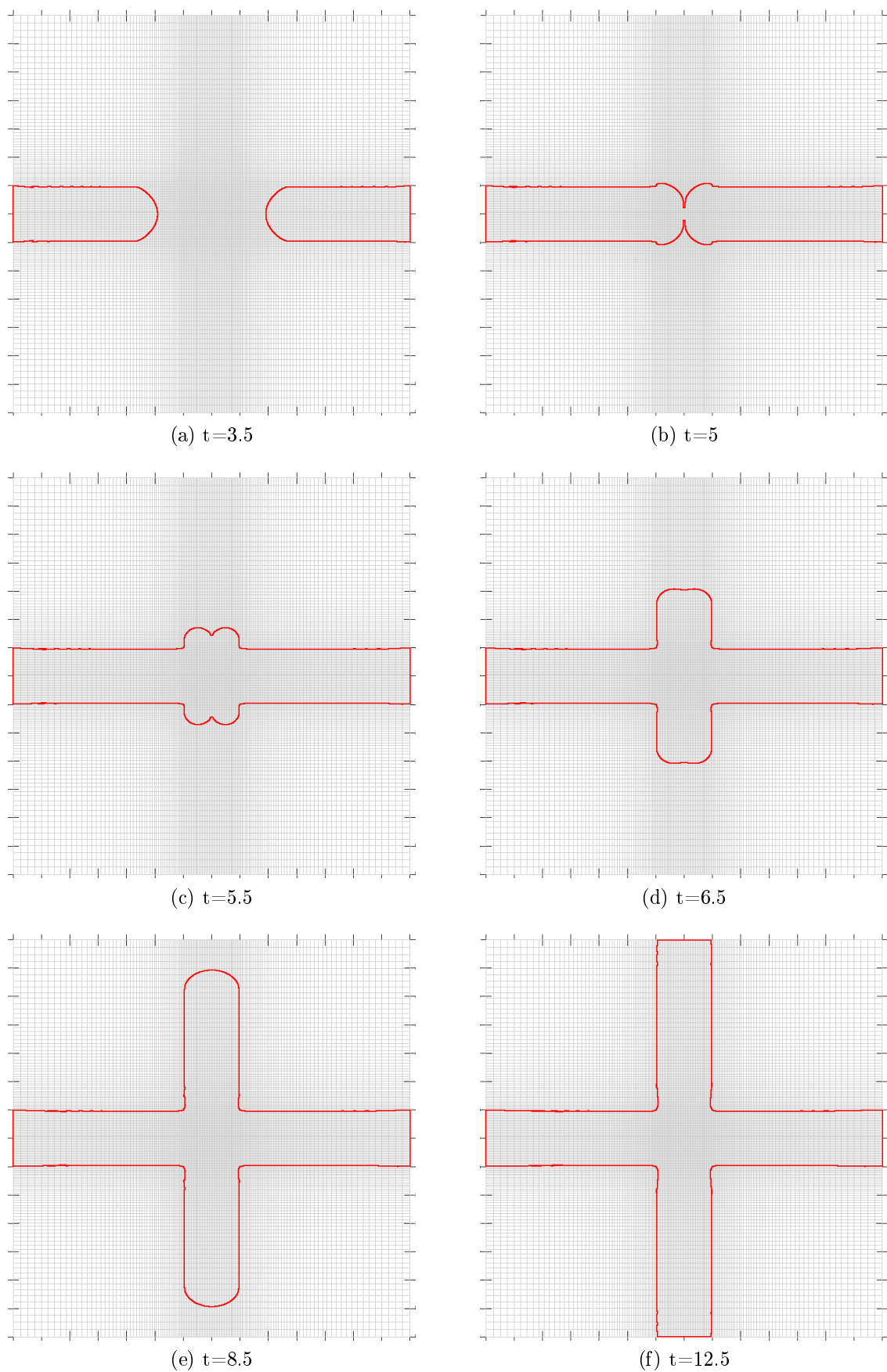


Figura 5.12: Visualização da superfície livre na simulação *cross-slot* em diferentes instantes de tempo.

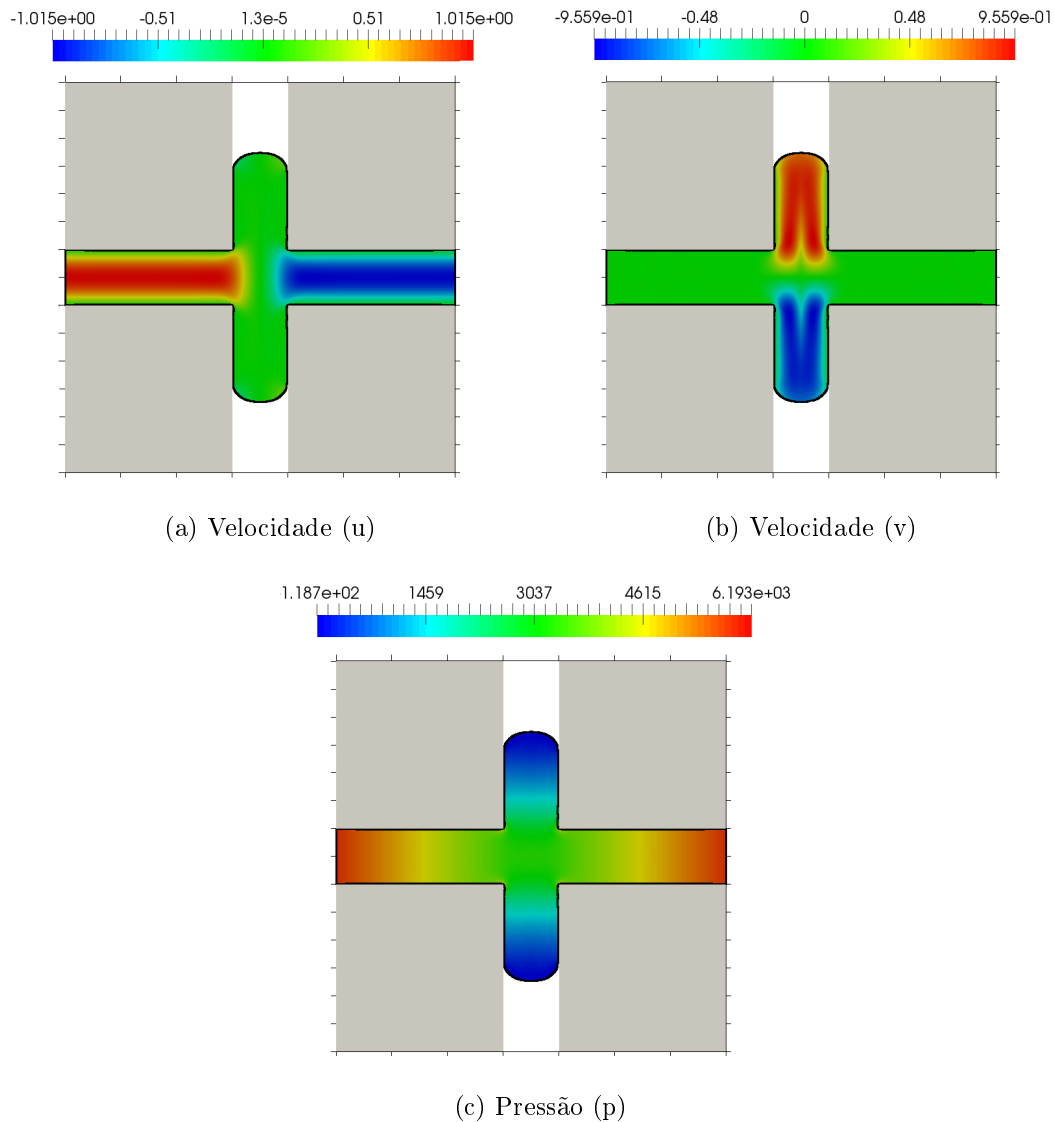


Figura 5.13: Visualização dos campos de velocidade e pressão para a geometria *cross-slot* no instante $t=7.5$, antes do preenchimento total do canal.

5.2.4 Impacto entre gotas e uma parede rígida

Nesta seção vamos descrever simulações realizadas nas quais uma ou duas gotas são lançadas de encontro a uma parede rígida. A figura 5.14 mostra o estado inicial destas simulações tanto para o caso de uma única gota, quanto para o de duas. No caso de duas gotas, uma mudança topológica será realizada quando ambas se encontrarem. Algumas referências que também realizaram testes como este são [21, 39, 48, 52].

A principal medida utilizada na literatura para estudar este problema é a largura das gotas ao longo do tempo. Iremos quantificar estes resultados em nossas simulações para verificar o comportamento do método.

Primeiramente realizaremos uma comparação com resultados apresentados em [21]. Esta referência resolve numericamente o problema de interface móveis via métodos de partículas que, conseqüentemente, não necessitam de um algoritmo de mudança topológica. Mesmo após o encontro entre as gotas, os autores continuam calculando a largura de cada uma delas separadamente. Para fins de comparação, optamos por fazer o mesmo e não executar o algoritmo de mudança topológica em nossa superfície livre.

Os seguintes parâmetros adimensionais são usados nesta simulação:

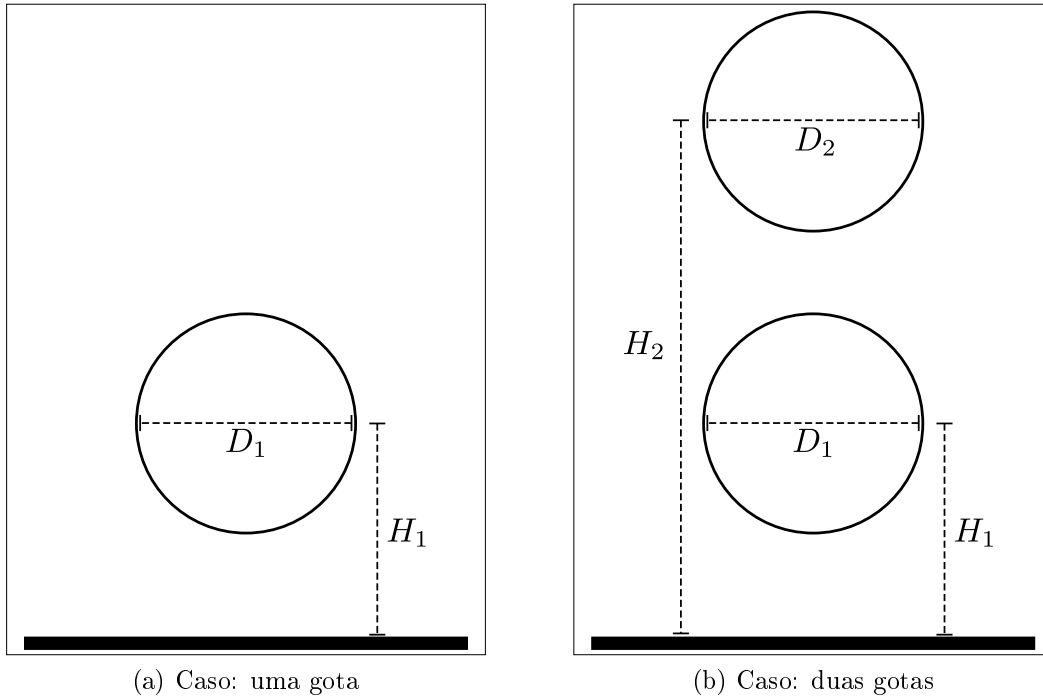


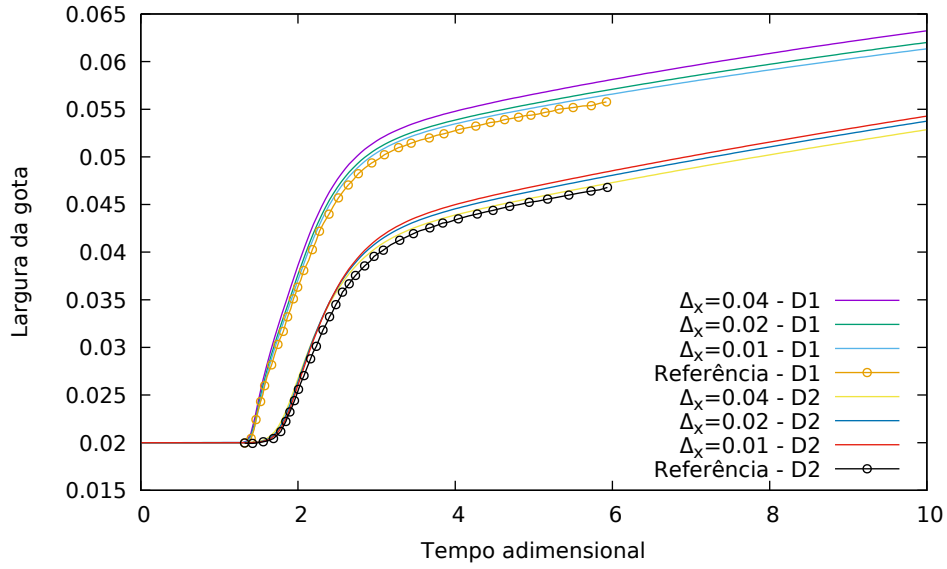
Figura 5.14: Estado inicial das simulações com gotas que impactam sobre uma superfície rígida. À esquerda, uma única gota é lançada. À direita, duas gotas são lançadas.

- Diâmetro inicial das gotas: $\frac{D_1}{L} = \frac{D_2}{L} = 1$;
- Altura das gotas: $\frac{H_1}{L} = 2$ e $\frac{H_2}{L} = 3$;
- Velocidade de lançamento das gotas: $\frac{u_{inicial}}{U} = 1$;
- Número de Reynolds: $Re = 5$;
- Número de Froude: $Fr = 2.2576$;
- Tensão superficial: efeito desconsiderado (usa-se $\kappa = 0$);
- Parâmetros do caso viscoelástico: $Wi = 1$ e $\beta = 0.1$ com formulação CSF;
- Passo temporal: $\Delta_t = 5 \cdot 10^{-4}$.

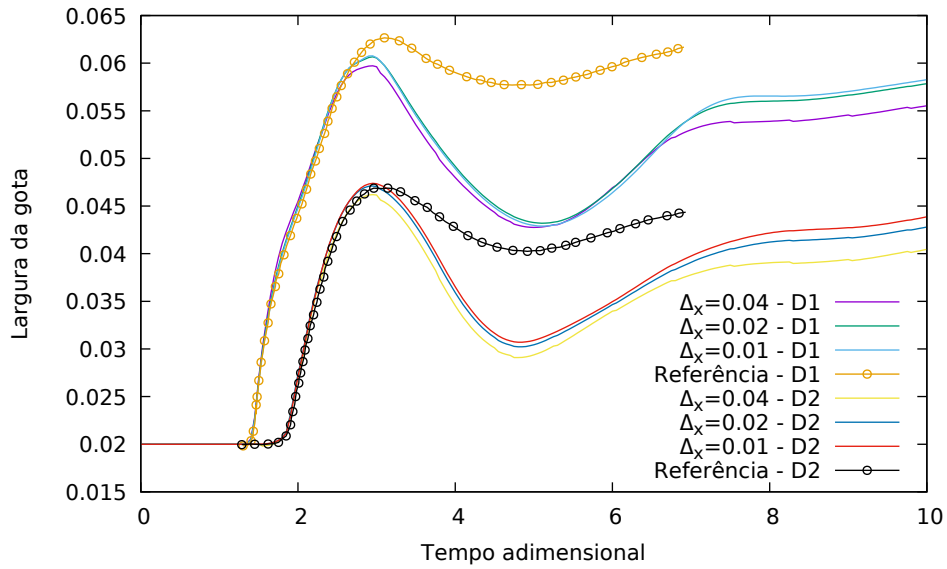
A figura 5.15 mostra a variação do diâmetro, como função do tempo, de cada uma das gotas calculadas com três malhas diferentes. As malhas são uniformes e possuem $\Delta_x = 0.01$, $\Delta_x = 0.02$ e $\Delta_x = 0.04$, respectivamente. A figura contém também a solução numérica de referência obtida do artigo [21]. Os resultados indicam que o método tende a convergir com o refinamento de malha, mas a solução, em especial no caso viscoelástico, não está muito próxima da referência. Como a solução de referência também é numérica, acreditamos que a diferença possa ter surgido devido às grandes diferenças presentes na metodologia sem malha utilizada pelos autores.

A figura 5.16 mostra a interface desta simulação em diferentes instantes de tempo para o caso viscoelástico considerando o modelo Oldroyd-B.

Iremos agora partir para testes que considerem também os efeitos da tensão superficial. Iniciaremos realizando uma comparação da variação dos diâmetros obtidos quando lançamos uma única gota ou duas. Este teste foi baseado na referência [39], no qual esta comparação também é feita. No artigo de referência, os autores utilizam valores altos de



(a) Caso Newtoniano



(b) Caso viscoelástico Oldroyd-B

Figura 5.15: Largura das gotas calculadas com diferentes malhas e comparadas com a referência [21].

Reynolds e também um modelo de ângulo de contato, o que dificulta a nossa comparação. Desta forma, utilizamos a referência como base para uma comparação apenas qualitativa.

Os parâmetros adimensionais usados neste teste foram:

- Diâmetro inicial das gotas: $\frac{D_1}{L} = \frac{D_2}{L} = 1$;
- Altura das gotas: $\frac{H_1}{L} = 0.5$ e $\frac{H_2}{L} = 1.8$;
- Velocidade de lançamento das gotas: $\frac{u_{inicial}}{U} = 1$;
- Número de Reynolds: $Re = 20$;
- Número de Froude: $Fr = 1$;
- Número de Weber: $We = 10$;

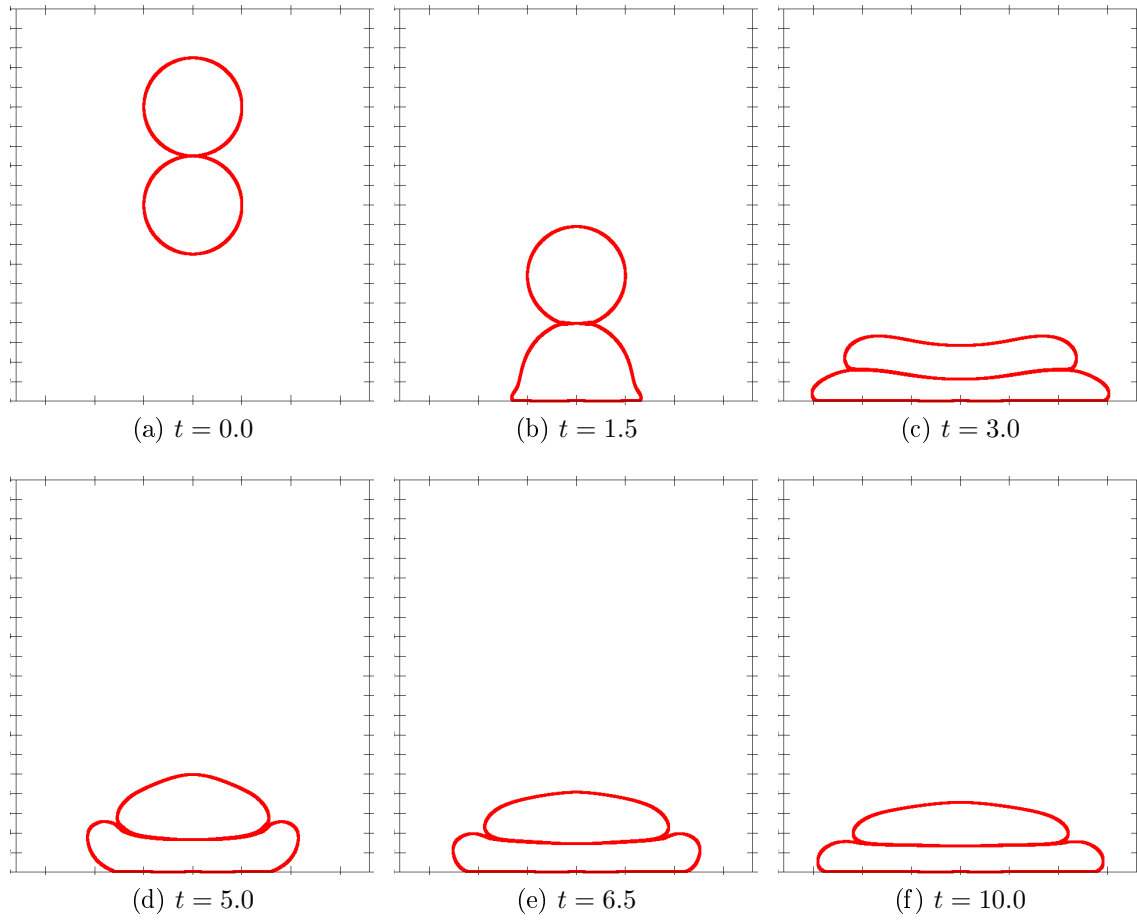


Figura 5.16: Interface da simulação não-Newtoniana em diferentes instantes de tempo.

- Parâmetros do caso viscoelástico: $\beta = 0.5$ com vários Wi e formulação CSF;
- Malha computacional: uniforme com $\Delta_x = \Delta_y = 0.02$
- Passo temporal: $\Delta_t = 5 \cdot 10^{-4}$.

A figura 5.17 traz estas comparações. É visível que, nas simulações com duas gotas, a largura alcança um pico muito mais alto. Isto é natural, já que a velocidade adicional vinda da segunda gota potencializa o efeito do impacto. Podemos ver também que, ao longo do tempo, a largura da gota cresce e diminui em ciclos até que, eventualmente, alcançaria um estado estacionário. Este efeito de crescimento e diminuição consecutivos é gerado tanto pela tensão superficial quanto pelas propriedades elásticas (no caso não-Newtoniano) e, por isso, pudemos notar que nas simulações com maior Wi este efeito é acentuado.

Com o objetivo de mostrar novamente a aplicação do algoritmo de mudança topológica, a figura 5.18 mostra a interface em alguns instantes de tempo desta simulação com $Wi = 5$. Observamos que, ao aumentar o Wi , temos o efeito de inversão das velocidades, de forma que a gota contrai e posteriormente relaxa, repetindo este movimento mais de uma vez. Essa variação do diâmetro também pode ser observada pelas sub-figuras em 5.17 quando o escoamento é viscoelástico.

O último teste realizado nesta seção foi feito para analisar o efeito que a distância entre as gotas no lançamento tem na simulação. Um teste similar é feito em [48]. Estes testes foram realizados com os mesmos parâmetros do caso anterior, alterando apenas a altura inicial da gota superior (H_2) da seguinte forma:

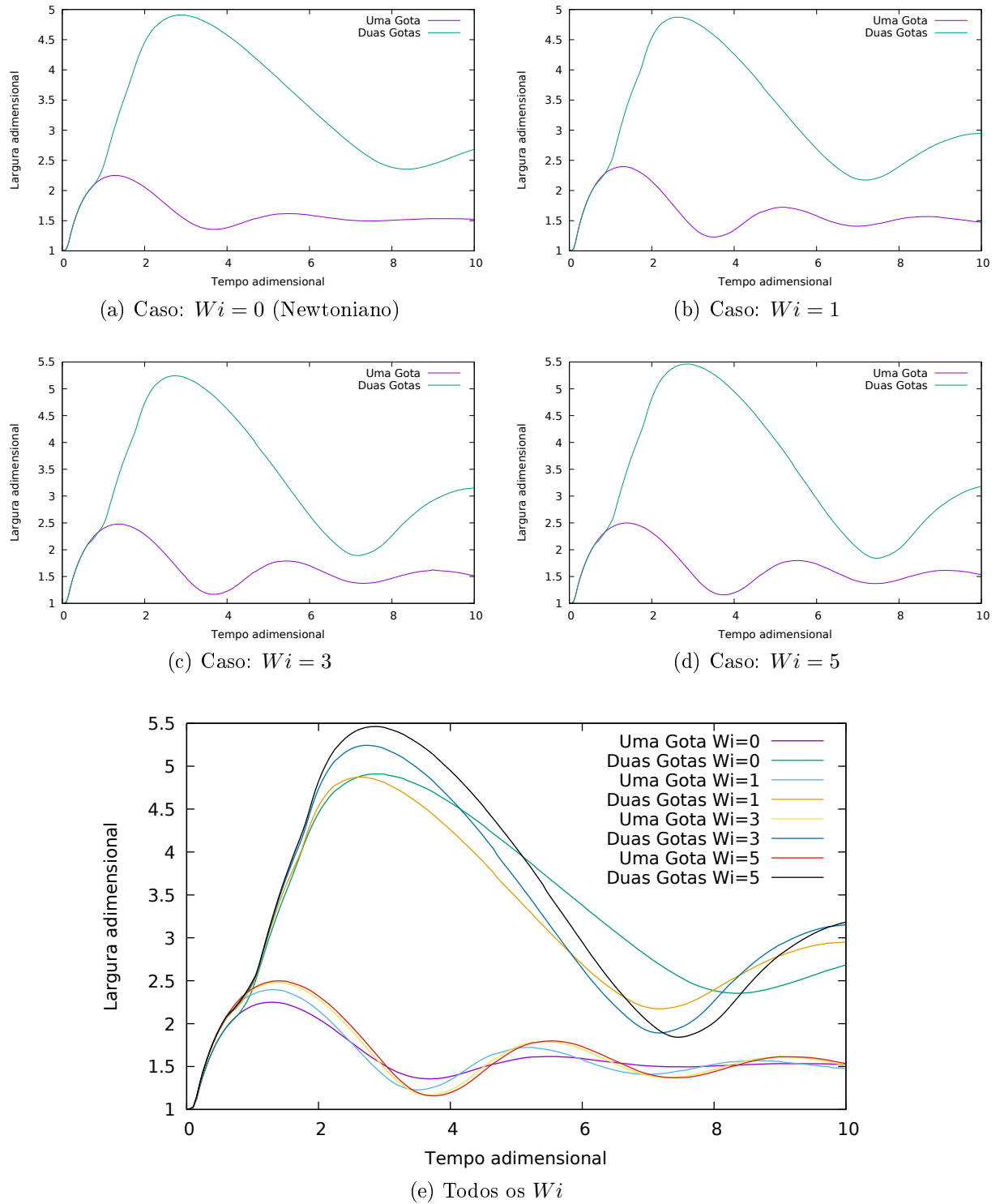


Figura 5.17: Largura das gotas calculadas com diferentes Wi . São comparadas simulações com apenas uma gota e simulações com duas.

- Caso 1: $\frac{H_2}{L} = 1.65$;
- Caso 2: $\frac{H_2}{L} = 1.8$;
- Caso 3: $\frac{H_2}{L} = 2.1$;
- Caso 4: $\frac{H_2}{L} = 2.4$.

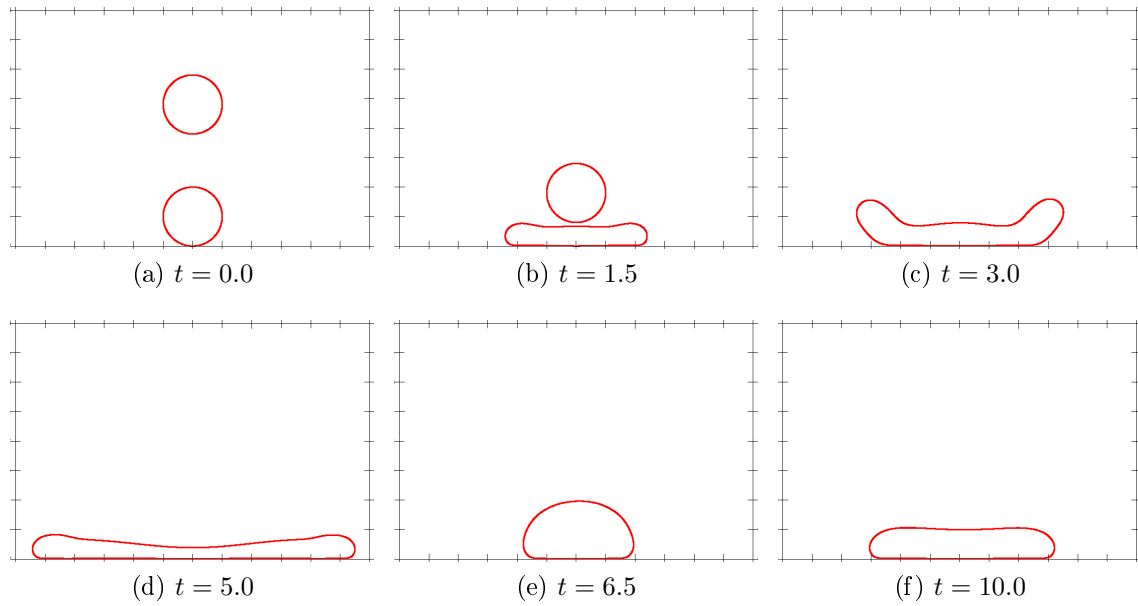


Figura 5.18: Interface da simulação de duas gotas com $Wi = 5$ em diferentes instantes de tempo.

A figura 5.19 mostra o efeito da distância entre as gotas para diferentes valores de Wi . Observamos que o caso Newtoniano foi o mais sensível à variação desta distância, sugerindo que os efeitos elásticos conseguem segurar o valor final da largura mais facilmente, alterando a energia interna das gotas.

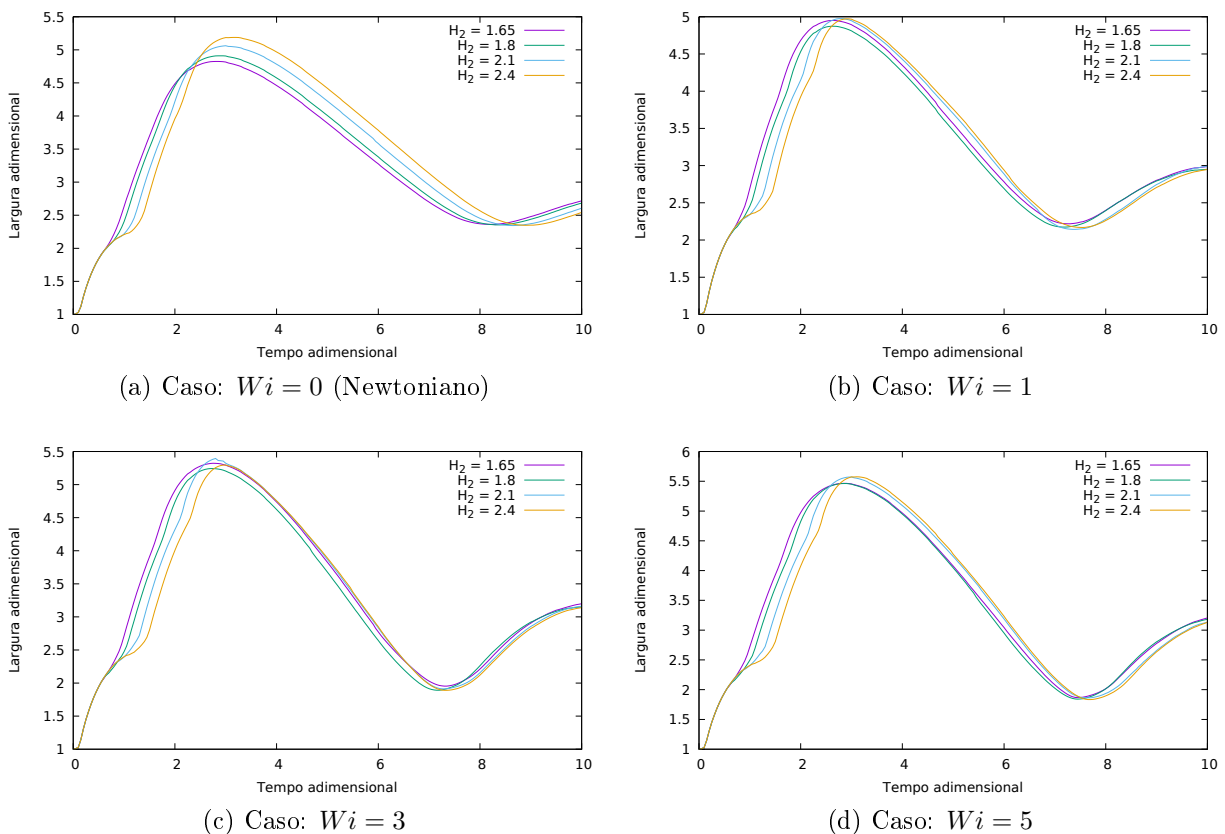


Figura 5.19: Efeito da variação da distância entre gotas no valor da largura.

5.2.5 Jato com tensão superficial

As últimas simulações realizadas neste trabalho têm o intuito de testar o algoritmo de mudanças topológicas em escoamentos nos quais ocorre uma separação da interface. O problema escolhido envolve um jato de fluido que é inserido na parte superior do domínio computacional com uma velocidade prescrita. Devido aos efeitos da velocidade prescrita, gravidade e tensão superficial, este jato cai e fica cada vez mais fino ao longo do tempo. Eventualmente, uma quebra ocorre na região mais fina, gerando blocos separados de fluido. A figura 5.20 mostra a geometria utilizada nas simulações deste escoamento.

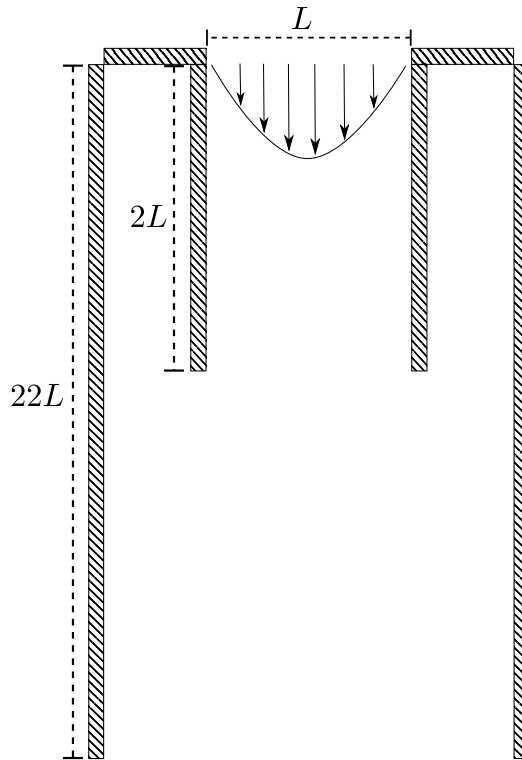


Figura 5.20: Esboço da geometria utilizada no problema do jato.

Os parâmetros utilizados nas simulações a seguir foram baseados nas propriedades do fluido Newtoniano adotado também em [43]. Estes parâmetros dimensionais são:

- Largura do injetor: $L = 5 \cdot 10^{-3} \text{m}$ (usado como escala para adimensionalização);
- Densidade: $\rho = 1.12 \cdot 10^3 \text{kg/m}^3$;
- Viscosidade: $\eta_s = 3.16 \cdot 10^{-2} \text{kg m}^{-1} \text{s}^{-1}$;
- Tensão superficial: $\sigma = 4.4 \cdot 10^{-2} \text{N/m}$;
- Aceleração da gravidade: $g_0 = -9.81 \text{m/s}^2$ ($|g_0|$ usado como escala na adimensionalização).

O último parâmetro a ser escolhido é a velocidade máxima (U) imposta no perfil parabólico do inflow. Realizamos quatro variações, cada uma com um valor para U . Estes quatro casos e seus respectivos parâmetros adimensionais são dados na tabela 5.4.

A discretização do domínio foi feita com uma malha uniforme e $\Delta_t = 10^{-5}$ foi usado como passo temporal.

O primeiro estudo realizado nestes testes visa analisar o efeito do refinamento de malha sobre as simulações com jato. Este estudo é muito relevante aqui, pois, como descrito na

Tabela 5.4: Quatro casos utilizados nas simulações desta seção. Cada caso possui uma velocidade de injeção (U), levando a parâmetros adimensionais diferentes.

Caso	U (m/s)	Re	Fr	We
A	$8.88 \cdot 10^{-2}$	15.737	0.40095	1.0036
B	$2.22 \cdot 10^{-2}$	7.8933	0.20048	0.2509
C	$4.44 \cdot 10^{-2}$	3.9467	0.10024	0.062725
D	$1.11 \cdot 10^{-2}$	1.9733	0.050119	0.015681

seção 4.3.2, a quebra do jato é forçada sempre que sua espessura atinge um limite de duas células. Desta forma, é intuitivo esperar que uma mudança no tamanho das células iria alterar o instante no qual a quebra acontece.

Na figura 5.21 mostramos uma comparação entre simulações do caso B realizadas com três malhas uniformes diferentes. Em vermelho, amarelo e azul estão as simulações realizadas com $\Delta_x = 0.033$, $\Delta_x = 0.025$ e $\Delta_x = 0.02$ respectivamente. No primeiro quadro da figura ainda não houve nenhuma quebra do jato, e as três malhas apresentam superfícies muito próximas. A partir do segundo quadro as quebras começam a acontecer, e é a partir daqui que as diferenças se destacam. Em cada quadro pode-se ver que as simulações quebram em tempos bastantes distintos e que, quanto maior o espaçamento, mais rápido ocorre a quebra. Este comportamento está de acordo com o esperado, já que o critério de quebra está diretamente ligado ao tamanho da célula. Em todas as análises que virão a seguir, foi utilizada a malha com $\Delta_x = 0.02$.

O próximo estudo realizado aqui foi o da influência da velocidade de injeção na simulação, ou seja, queremos observar o comportamento de cada um dos casos dados na tabela 5.4. As figuras 5.22, 5.23, 5.24, 5.25 mostram, respectivamente, as simulações A, B, C e D em diferentes instantes de tempo. Notamos que, na simulação com velocidade mais elevada (caso A), o jato chega ao final do domínio computacional sem que nenhuma mudança topológica aconteça, já que nunca ficou fino o bastante para isso. Ao diminuir a velocidade de injeção (casos B, C, D) observamos que o processo de afinamento ocorre cada vez mais rápido, de modo que até múltiplas gotas são formadas em alguns casos. Com velocidades de injeção mais baixas, os efeitos da tensão superficial e gravidade tendem a dominar o escoamento, levando ao comportamento de separação. Os efeitos capturados nessas simulações estão em concordância qualitativa com aqueles apresentados em [43], isto é, ao reduzirmos o valor de We , o jato tende a forma de gotejamentos (*dripping*).

Ainda nas figuras 5.22-5.25 é possível também notar um efeito causado pelas imprecisões no algoritmo que calcula a curvatura da interface, como foi comentado na seção 4.4.2. Observe que, nos casos com maior influência da tensão superficial, o jato principal e algumas gotas perdem sua simetria e saem do eixo central do domínio. Este efeito é não-físico e gerado por erros numéricos. Analisando os instantes e as regiões em que as assimetrias começam a aparecer, pudemos notar que elas coincidiam com regiões nas quais um valor de curvatura muito impreciso era calculado pelo método.

Para tentar verificar os resultados destas simulações, realizamos também uma comparação entre o que foi obtido aqui e um resultado apresentado em [41]. Seja H o comprimento do jato a partir do injetor até a sua ponta; foi observado numericamente e experimentalmente em [41] que o valor de H é proporcional a $We^{0.28}Fr^{0.78}$, isto é, existe uma constante α tal que

$$H = \alpha We^{0.28} Fr^{0.78} \quad (5.2)$$

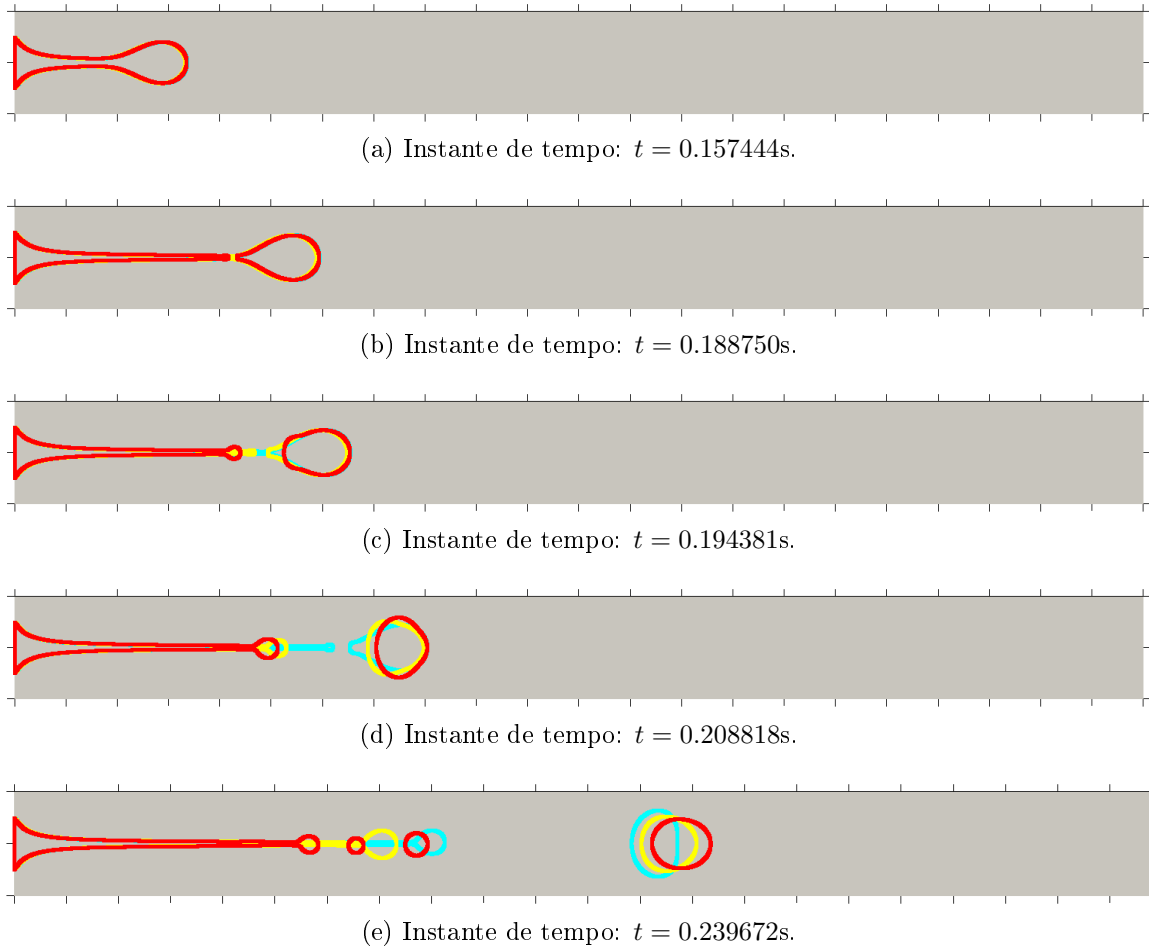


Figura 5.21: Comparação entre malhas para o caso B do jato com tensão superficial. As malhas em vermelho, verde e azul possuem $\Delta_x = 0.033$, $\Delta_x = 0.025$ e $\Delta_x = 0.02$ respectivamente. Todas elas são uniformes.

Os autores de [41] calculam o valor H definindo um tempo final t_{final} , e realizando uma média aritmética do comprimento medido em vários instantes de tempo no intervalo $[0, t_{final}]$. Utilizamos a mesma estratégia com $t_{final} = 0.1\text{s}$ e $t_{final} = 0.2\text{s}$. Os resultados desta medição são apresentados na figura 5.26. Quatro pontos presentes nos gráficos desta figura são referentes às simulações A, B, C, D; os outros três foram simulações adicionais realizadas com $U = 0.074\text{m/s}$, $U = 0.0592\text{m/s}$ e $U = 0.0333\text{m/s}$, que correspondem a $We = 0.69695$, $We = 0.44605$ e $We = 0.14113$, respectivamente. Aproximando os pontos destes gráficos com mínimos quadrados, conseguimos encontrar retas da forma (5.2) que realmente estão muito próximas dos pontos obtidos. Isto indica que o método foi capaz de capturar o comportamento previsto em [41].

Por fim, realizamos um último teste no qual utilizamos as simulações A, B, C, D para verificar um resultado teórico apresentado em [43]. Uma expressão analítica é dada em [43] que mede a espessura do jato antes da formação da gota em função da distância até o injetor de fluido. Esta expressão é dada por

$$E(y) = L \left[1 + 2 \frac{Fr^{-2}y}{L} \left(1 - \frac{\rho_g}{\rho} \right) \right]^{-1/2}, \quad (5.3)$$

em que L é o tamanho do injetor, ρ é a densidade do fluido no jato, $\rho_g = 1.2\text{kg/m}^3$ é a densidade do ar, y é a distância até o injetor, e E é a espessura do jato.

A figura 5.27 mostra a comparação entre solução numérica e analítica imediatamente antes da primeira quebra para os quatro casos estudados. Em vermelho e azul estão,

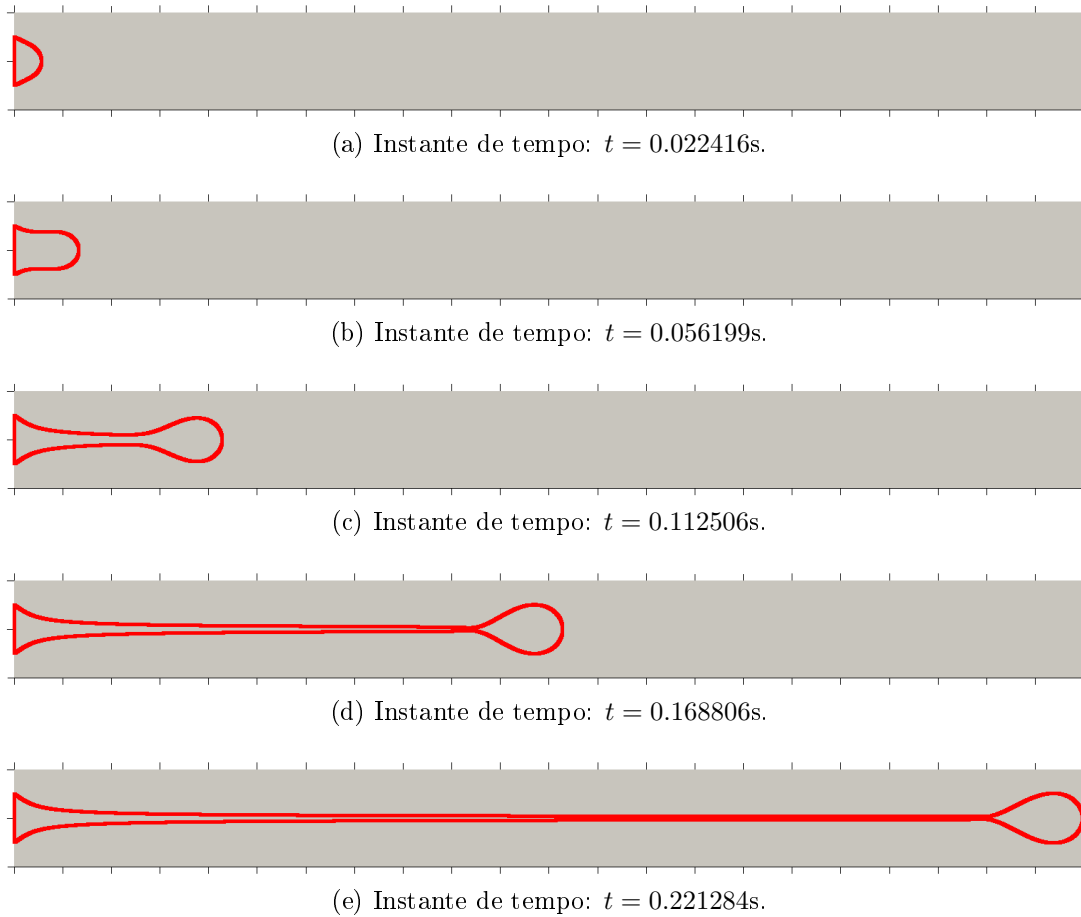


Figura 5.22: Caso A da simulação do jato com tensão superficial em diferentes instantes de tempo.

respectivamente, a interface numérica e a analítica. Observamos que o resultado numérico se aproxima bem da espessura prevista analiticamente e, quanto menor a velocidade de injeção, mais fino fica o jato. Nos casos com maior efeito da tensão superficial já existe um início de assimetria na simulação, mas a espessura do jato ainda se mantém próxima do esperado analiticamente.

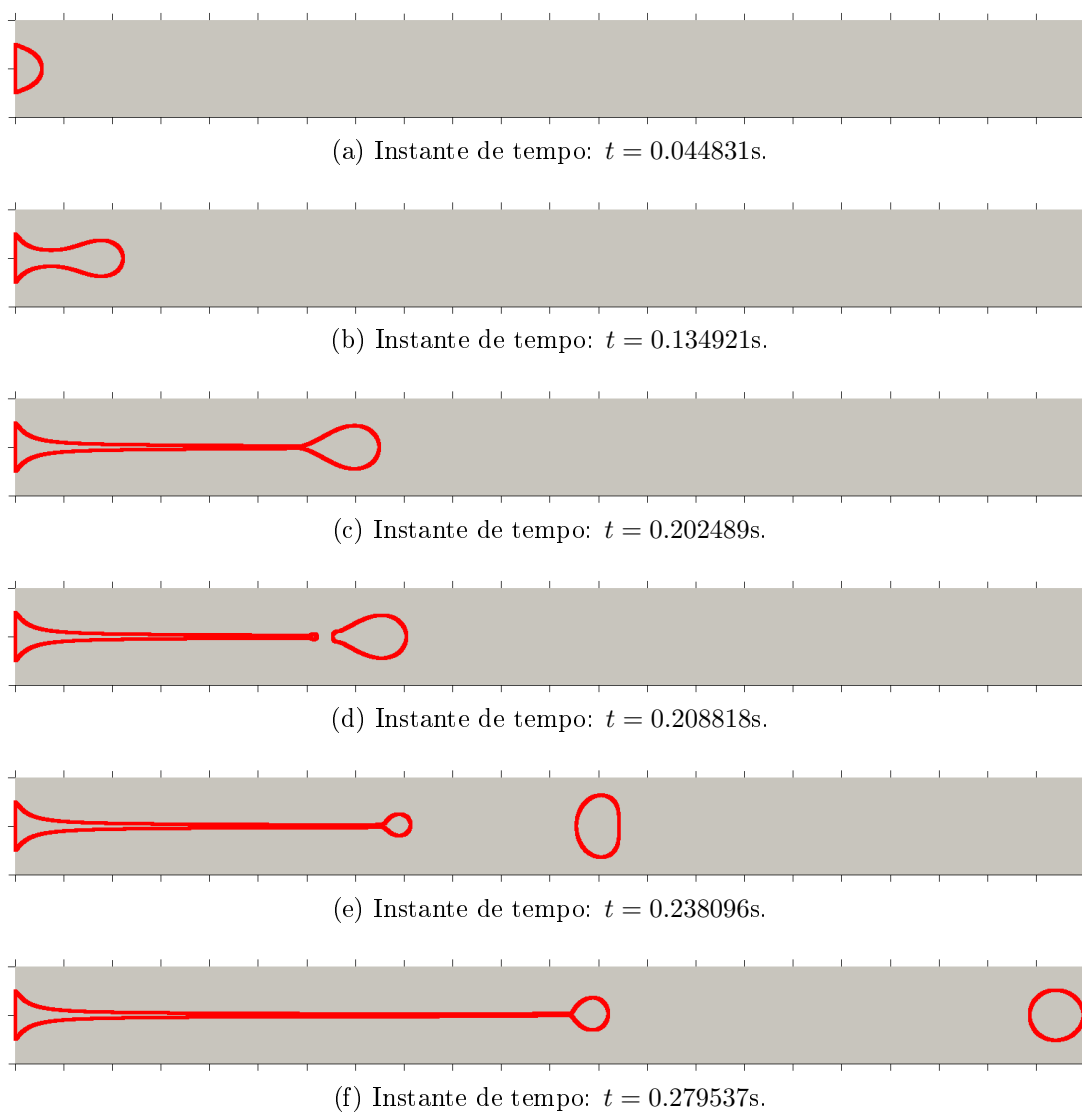


Figura 5.23: Caso B da simulação do jato com tensão superficial em diferentes instantes de tempo.

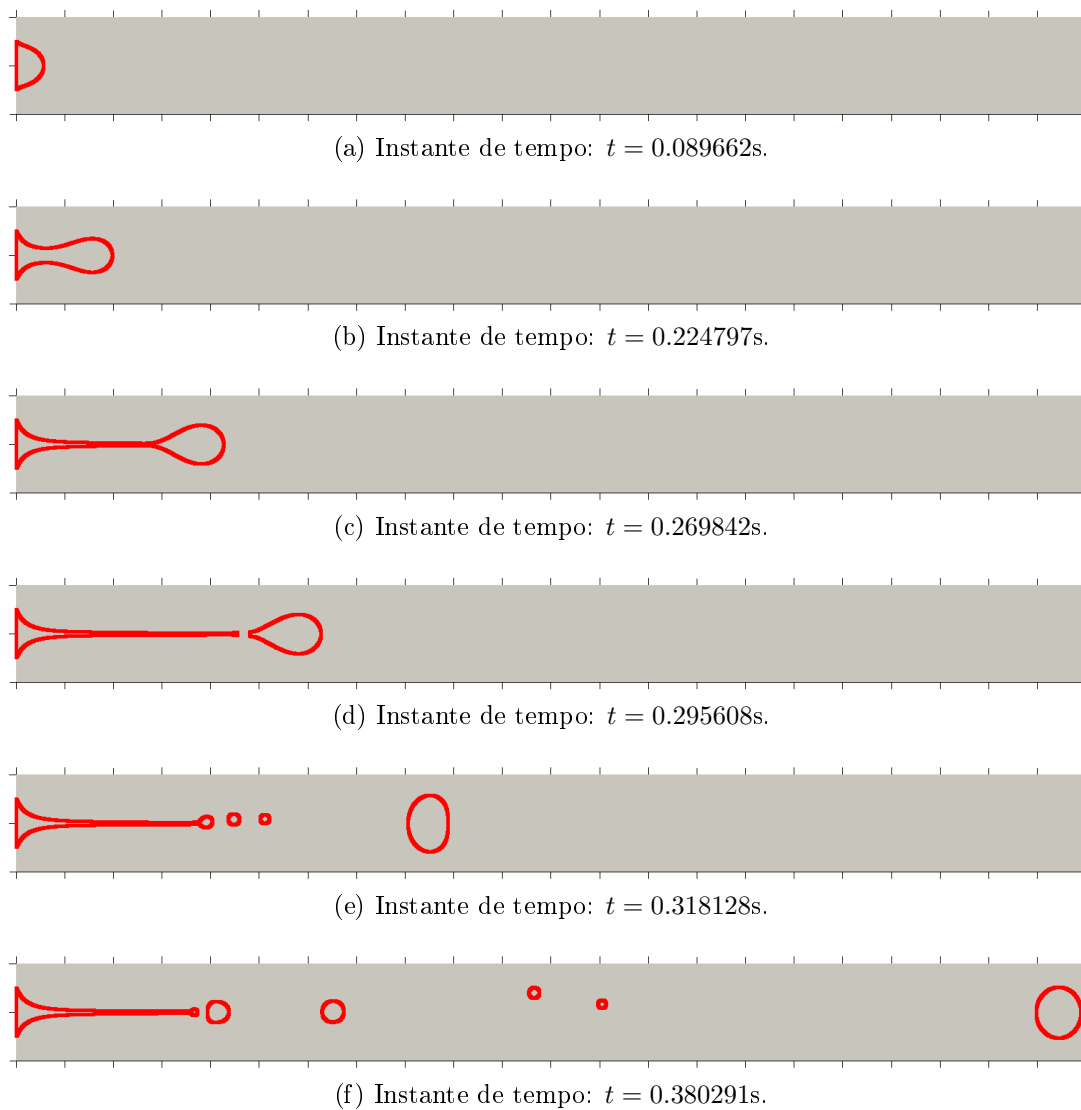


Figura 5.24: Caso C da simulação do jato com tensão superficial em diferentes instantes de tempo.

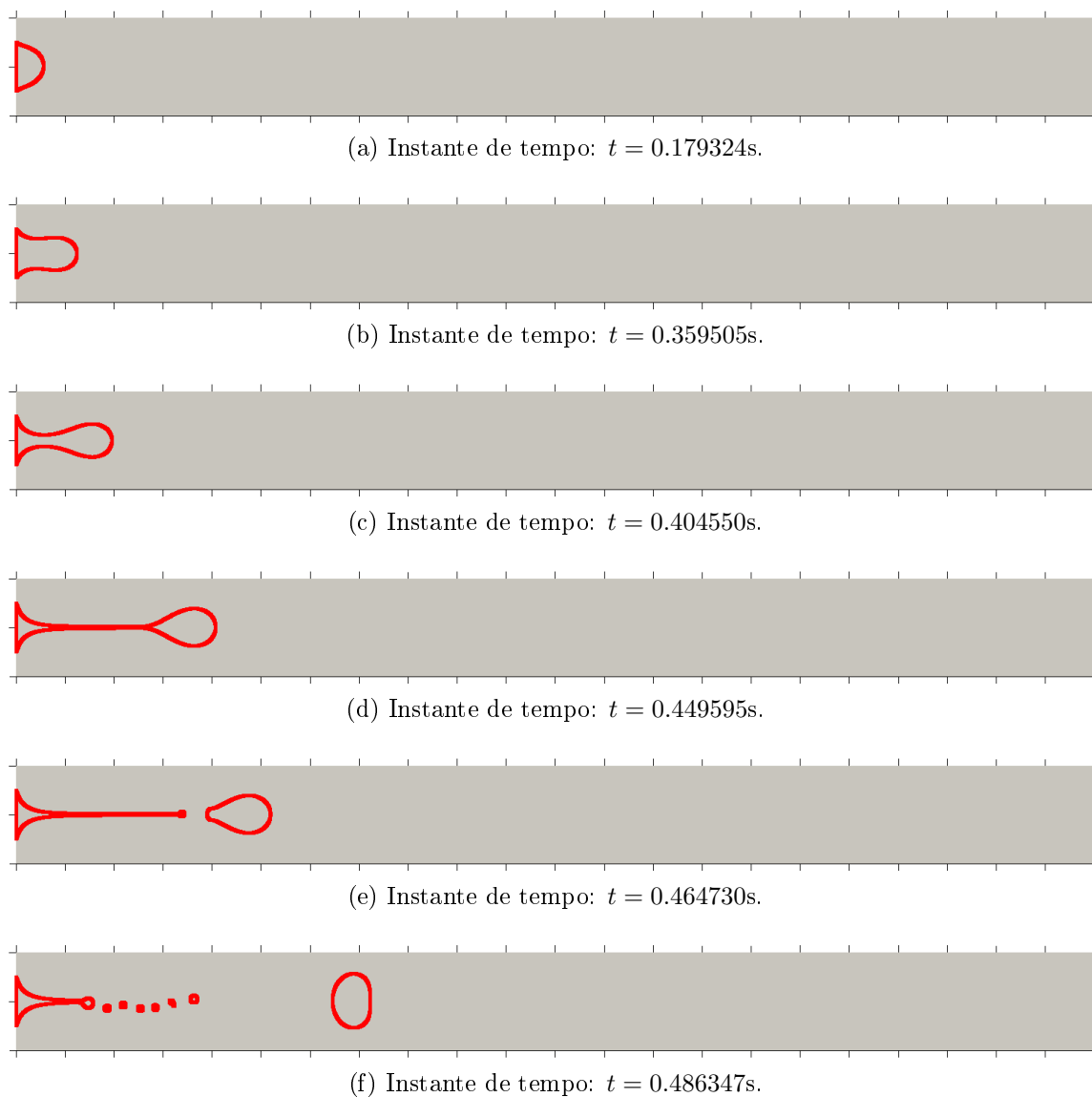


Figura 5.25: Caso D da simulação do jato com tensão superficial em diferentes instantes de tempo.

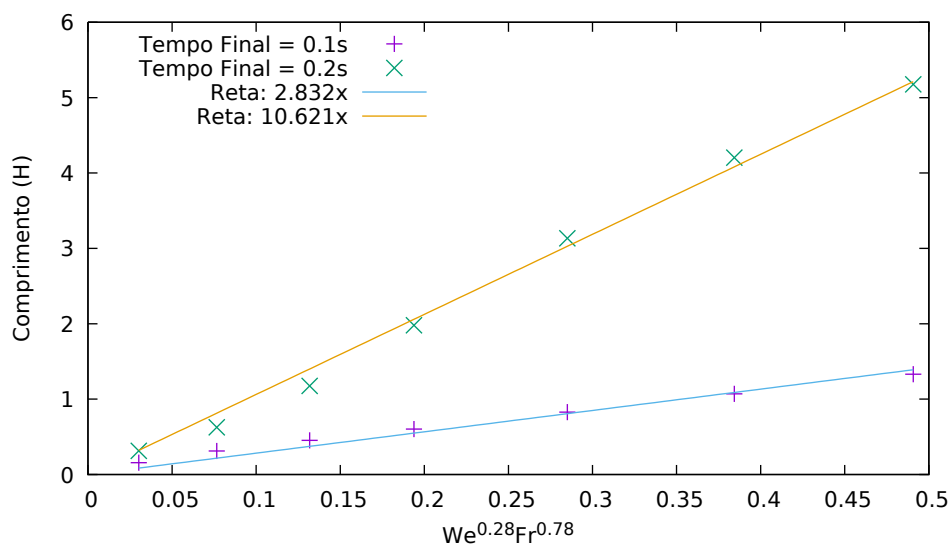
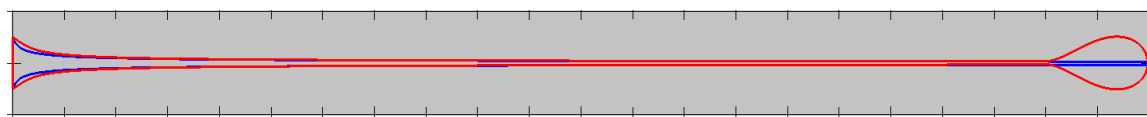
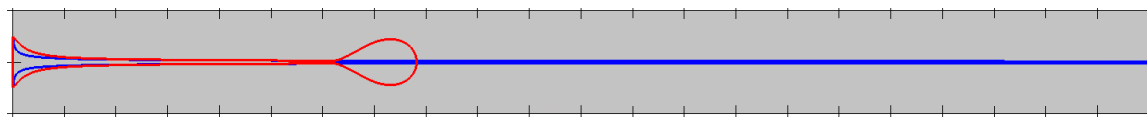


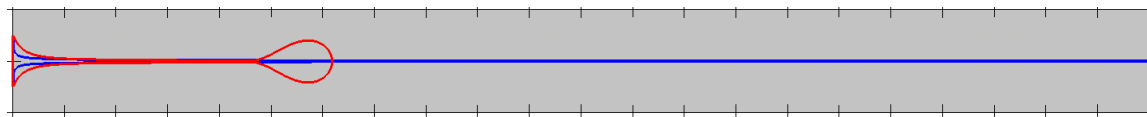
Figura 5.26: Comprimento do jato em função de $We^{0.28}Fr^{0.78}$.



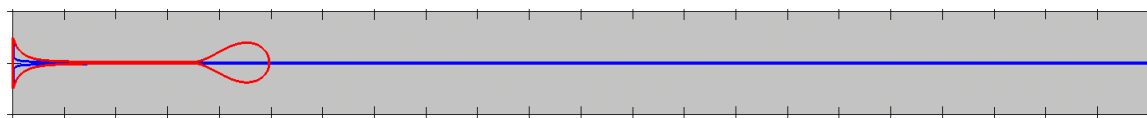
(a) Caso A



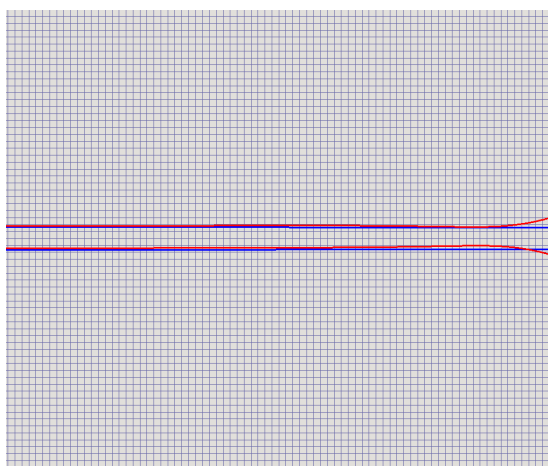
(b) Caso B



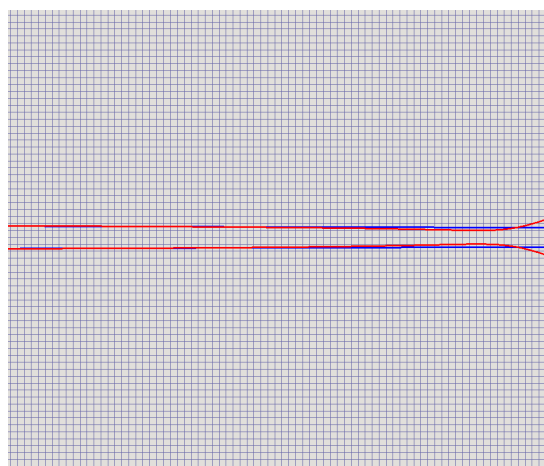
(c) Caso C



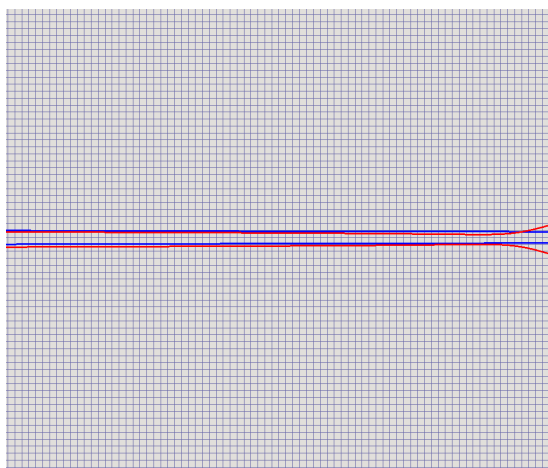
(d) Caso D



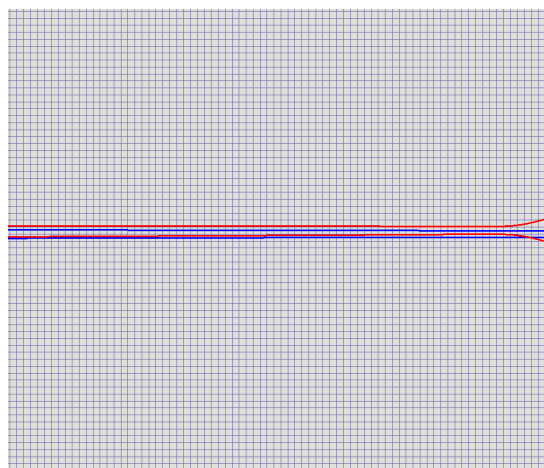
(e) Caso A - Ampliação



(f) Caso B - Ampliação



(g) Caso C - Ampliação



(h) Caso D - Ampliação

Figura 5.27: Comparação da espessura do jato antes da quebra para cada um dos quatro casos. Em vermelho e azul são apresentadas as soluções numérica e analítica, respectivamente.

Conclusão e etapas seguintes

Neste projeto de mestrado foi possível criar um código para a solução das equações de Navier-Stokes com superfície livre para escoamentos incompressíveis. Todo o desenvolvimento foi feito com dois objetivos principais em mente: a discretização das equações utilizando uma malha não uniforme, e a implementação de um algoritmo que realize mudanças topológicas.

A utilização de uma malha não uniforme foi muito importante no projeto devido aos testes confinados realizados em geometrias que possuem quinas. Geometrias deste tipo, como a *cross-slot* ou o canal com contração, possuem regiões específicas nas quais as equações precisam ser resolvidas com mais precisão. A presença de uma malha não-uniforme permitiu que esta precisão pudesse ser obtida sem que o custo computacional fosse elevado muito drasticamente.

Para simulações de fluidos viscoelásticos, foi feita também a implementação de uma técnica conhecida como *Natural Stress Formulation*, que reescreve o tensor \mathbf{T} sobre uma base de vetores baseada na velocidade do escoamento. Esta formulação foi bastante testada em escoamentos confinados, em especial na geometria com contração. Os resultados obtidos destes testes foram muito promissores, pois eram mais precisos que aqueles obtidos pela formulação tradicional quando comparados a previsões teóricas. Resultados obtidos com esta formulação ainda são muito escassos na literatura e, dessa forma, um artigo foi elaborado reportando estas simulações e será submetido em breve.

Os escoamentos com superfície livre foram resolvidos através de uma implementação baseada no método GENSMAC. Este método representa a interface por partículas marcadoras, e classifica as células da malha computacional de acordo com a posição das mesmas. Simulações para o problema do inchamento do extrudado (*die-swell*) foram realizadas e permitiram validar esta etapa do código. Em seguida, um algoritmo de mudanças topológicas foi implementado baseado na técnica apresentada em [14] que detecta embaraçamentos na interface. Esta técnica foi testada em escoamentos como: o impacto de uma gota em uma camada de fluido, e o escoamento na geometria *cross-slot* com superfície livre.

Por fim, foi feita também a inclusão dos efeitos da tensão superficial no código. A principal dificuldade nesta etapa foi o cálculo da curvatura da interface, que foi feito com um algoritmo proposto em [26]. Para testar os efeitos da tensão superficial nas simulações, realizamos escoamentos como: o impacto de gotas sobre uma placa rígida, e o afinamento de um jato sob tensão superficial. Estes testes apresentaram resultados que se comportavam de forma coerente com aqueles estudados na literatura.

Em trabalhos futuros, pretendemos aprimorar algumas técnicas utilizadas aqui e procurar soluções para dificuldades encontradas durante este projeto. Dentre estes aprimoramentos, podemos citar:

- a utilização de vetores normais mais precisos, isto é, que não estejam limitados às oito possibilidades da figura 3.3;
- melhorar a precisão no cálculo da curvatura da interface. Testar também outras formas de suavização para a interface, como a técnica TSUR, proposta em [26] e que preserva a massa no escoamento;
- estudar e melhorar os critérios utilizados para determinar quando uma mudança topológica deve ser realizada. Em todos os testes deste projeto utilizamos critérios baseados na classificação das células da malha. Isto faz com que as mudanças topológicas sejam realizadas em instantes que dependem diretamente da discretização escolhida, ou seja, os critérios são muito artificiais.

Referências

- [1] M.K. Agoston. *Computer Graphics and Geometric Modelling: Implementation & Algorithms*. Computer Graphics and Geometric Modeling. Springer London, 2005.
- [2] J. Ahrens, B. Geveci, and C. Law. *ParaView: An End-User Tool for Large Data Visualization, Visualization Handbook*. Elsevier, 2005.
- [3] M. A. Alves, P. J. Oliveira, and F. T. Pinho. A convergent and universally bounded interpolation scheme for the treatment of advection. *International Journal for Numerical Methods in Fluids*, 41(1):47–75, 2003.
- [4] N. Ashgriz. *Handbook of Atomization and Sprays: Theory and Applications*. SpringerLink : Bücher. Springer US, 2011.
- [5] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, D.A. May, L.C. McInnes, R.T. Mills, T. Munson, K. Rupp, P. Sanan, B.F. Smith, and S. Zampini H. Zhang. PETSc Web page, 2018.
- [6] O.A. Basaran, H. Gao, and P.P. Bhat. Nonstandard inkjets. *Annual Review of Fluid Mechanics*, 45(1):85–113, 2013.
- [7] S. Basting and M. Weismann. A hybrid level set-front tracking finite element approach for fluid-structure interaction and two-phase flow applications. *Journal of Computational Physics*, 255:228 – 244, 2013.
- [8] H.D. Ceniceros, A.M. Roma, A. Silveira-Neto, and M.M. Villar. A robust, fully adaptive hybrid level-set/front-tracking method for two-phase flows with an accurate surface tension computation. *Communications in Computational Physics*, 8(1):51–94, 2010.
- [9] A.J. Chorin and J.E. Marsden. *A mathematical introduction to fluid mechanics*. Springer-Verlag, 1979.
- [10] R. Comminal, J.H. Hattel, M.A. Alves, and J. Spangenberg. Vortex behavior of the Oldroyd-B fluid in the 4-1 planar contraction simulated with the streamfunction-log-conformation formulation. *Journal of Non-Newtonian Fluid Mechanics*, 237(Supplement C):1 – 15, 2016.
- [11] W.C. de Jesus, A.M. Roma, M.R. Pivello, M.M. Villar, and A. Silveira-Neto. A 3D front-tracking approach for simulation of a two-phase fluid with insoluble surfactant. *Journal of Computational Physics*, 281:403 – 420, 2015.
- [12] F.S. de Sousa, N. Mangiavacchi, L.G. Nonato, A. Castelo, M.F. Tomé, V.G. Ferreira, J.A. Cuminato, and S. McKee. A front-tracking/front-capturing method for

- the simulation of 3D multi-fluid flows with free surfaces. *Journal of Computational Physics*, 198(2):469 – 499, 2004.
- [13] J.D Evans. Re-entrant corner flows of Oldroyd-B fluids. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 461(2060):2573–2603, 2005.
- [14] J. Glimm, J. Grove, B. Lindquist, O.A. McBryan, and G. Tryggvason. The bifurcation of tracked scalar waves. *SIAM Journal on Scientific and Statistical Computing*, 9(1):61–79, 1988.
- [15] J. Glimm, J. W. Grove, and Y. Zhang. Interface tracking for axisymmetric flows. *SIAM Journal on Scientific Computing*, 24(1):208–236, 2002.
- [16] J Glimm, D Marchesin, and O McBryan. A numerical method for two phase flow with an unstable interface. *Journal of Computational Physics*, 39(1):179 – 200, 1981.
- [17] J. Glimm, O. McBryan, R. Menikoff, and D. H. Sharp. Front tracking applied to Rayleigh-Taylor instability. *SIAM Journal on Scientific and Statistical Computing*, 7(1):230–251, 1986.
- [18] T. Gonzalez, J. Diaz-Herrera, and A. Tucker. *Computing Handbook, Third Edition: Computer Science and Software Engineering*. Computing Handbook. Taylor & Francis, 2014.
- [19] E.J. Hinch. The flow of an Oldroyd fluid around a sharp corner. *Journal of Non-Newtonian Fluid Mechanics*, 50(2):161 – 171, 1993.
- [20] C.W Hirt and B.D Nichols. Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201 – 225, 1981.
- [21] T. Jiang, L.G. Lu, and W.G. Lu. The numerical investigation of spreading process of two viscoelastic droplets impact problem by using an improved SPH scheme. *Computational Mechanics*, 53(5):977–999, May 2014.
- [22] J.M. Kim, C. Kim, J.H Kim, C. Chung, K.H Ahn, and S.J Lee. High-resolution finite element simulation of 4:1 planar contraction flow of viscoelastic fluid. *Journal of Non-Newtonian Fluid Mechanics*, 129(1):23 – 37, 2005.
- [23] M. Kline. *Calculus: An Intuitive and Physical Approach (Second Edition)*. Dover Books on Mathematics. Dover Publications, 2013.
- [24] S.H Lee, N. Hur, and S. Kang. A numerical analysis of drop impact on liquid film by using a level set method. *Journal of Mechanical Science and Technology*, 25(10):2567 – 2572, 2011.
- [25] G. Liang, Y. Guo, and S. Shen. Gas properties on crown behavior and drop coalescence. *Numerical Heat Transfer, Part B: Fundamentals*, 65(6):537–553, 2014.
- [26] N. Mangiavacchi, A. Castelo, M. Tomé, J. Cuminato, M. de Oliveira, and S. McKee. An effective implementation of surface tension using the marker and cell method for axisymmetric and planar flows. *SIAM Journal on Scientific Computing*, 26(4):1340–1368, 2005.

- [27] C.H. Marchi, R. Suero, and L.K. Araki. The lid-driven square cavity flow: numerical solution with a 1024 x 1024 grid. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 31:186 – 198, 09 2009.
- [28] T. Marić, H. Marschall, and D. Bothe. lentfoam - a hybrid level set/front tracking method on unstructured meshes. *Computers & Fluids*, 113:20 – 31, 2015. Small scale simulation of multiphase flows.
- [29] F.P. Martins. Desenvolvimento de um método numérico implícito para a simulação de escoamentos viscoelásticos com superfícies livres. Master's thesis, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2009.
- [30] F.P. Martins, C.M. Oishi, A.M. Afonso, and M.A. Alves. A numerical study of the kernel-conformation transformation for transient viscoelastic fluid flows. *Journal of Computational Physics*, 302(Supplement C):653 – 673, 2015.
- [31] S. McKee, M.F. Tomé, V.G. Ferreira, J.A. Cuminato, A. Castelo, F.S. Sousa, and N. Mangiavacchi. The MAC method. *Computers & Fluids*, 37(8):907 – 930, 2008.
- [32] R.H. Nochetto and S.W. Walker. A hybrid variational front tracking-level set mesh generator for problems exhibiting large deformations and topological changes. *Journal of Computational Physics*, 229(18):6243 – 6269, 2010.
- [33] C.M. Oishi, F.P. Martins, M.F. Tomé, and M.A. Alves. Numerical simulation of drop impact and jet buckling problems using the eXtended pom-pom model. *Journal of Non-Newtonian Fluid Mechanics*, 169-170:91 – 103, 2012.
- [34] C.M. Oishi, M.F. Tomé, J.A. Cuminato, and S. McKee. An implicit technique for solving 3D low Reynolds number moving free surface flows. *Journal of Computational Physics*, 227(16):7446 – 7468, 2008.
- [35] J.R.A. Pearson and S.M. Richardson. *Computational analysis of polymer processing*. Applied Science Publishers, 1983.
- [36] F. Pimenta and M.A. Alves. Stabilization of an open-source finite-volume solver for viscoelastic fluid flows. *Journal of Non-Newtonian Fluid Mechanics*, 239(Supplement C):85 – 104, 2017.
- [37] M.R. Pivello, M.M. Villar, R. Serfaty, A.M. Roma, and A. Silveira-Neto. A fully adaptive front tracking method for the simulation of two phase flows. *International Journal of Multiphase Flow*, 58:72 – 82, 2014.
- [38] R.J. Poole, G.N. Rocha, and P.J. Oliveira. A symmetry-breaking inertial bifurcation in a cross-slot flow. *Computers & Fluids*, 93:91 – 99, 2014.
- [39] K.A. Raman, R.K. Jaiman, T. Lee, and H. Low. Lattice Boltzmann study on the dynamics of successive droplets impact on a solid surface. *Chemical Engineering Science*, 145:181 – 195, 2016.
- [40] M. Renardy. How to integrate the upper convected Maxwell (UCM) stresses near a singularity (and maybe elsewhere, too). *Journal of Non-Newtonian Fluid Mechanics*, 52(1):91 – 95, 1994.
- [41] K. Shibata, S. Koshizuka, and Y. Oka. Numerical analysis of jet breakup behavior using particle method. *Journal of Nuclear Science and Technology*, 41(7):715–722, 2004.

-
- [42] S. Shin and D. Juric. Modeling three-dimensional multiphase flow using a level contour reconstruction method for front tracking without connectivity. *Journal of Computational Physics*, 180(2):427 – 470, 2002.
- [43] F.V. Sirotkin and J.J. Yoh. A new particle method for simulating breakup of liquid jets. *Journal of Computational Physics*, 231(4):1650 – 1674, 2012.
- [44] F.S. Sousa, C.M. Oishi, and G.C. Buscaglia. Spurious transients of projection methods in microflow simulations. *Computer Methods in Applied Mechanics and Engineering*, 285:659 – 693, 2015.
- [45] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 114(1):146 – 159, 1994.
- [46] H. Tan, E. Torniainen, D.P. Markel, and R.N.K. Browning. Numerical simulation of droplet ejection of thermal inkjet printheads. *International Journal for Numerical Methods in Fluids*, 77(9):544–570, 2015.
- [47] M. F Tomé. *GENSMAC: a multiple free surface fluid flow solver*. PhD thesis, University of Atrathclyde, 1993.
- [48] A.Y. Tong, S. Kasliwal, and H. Fujimoto. On the successive impingement of droplets onto a substrate. *Numerical Heat Transfer, Part A: Applications*, 52(6):531–548, 2007.
- [49] D.J. Torres and J.U. Brackbill. The point-set method: Front-tracking without connectivity. *Journal of Computational Physics*, 165(2):620 – 644, 2000.
- [50] G. Tryggvason, B. Bunner, A. Esmarelli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan. A front-tracking method for the computations of multiphase flow. *Journal of Computational Physics*, 169(2):708 – 759, 2001.
- [51] S.O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics*, 100(1):25 – 37, 1992.
- [52] H. Xiang and B. Chen. A moving particle semi-implicit method for free surface flow: Improvement in inter-particle force stabilization and consistency restoring. *International Journal for Numerical Methods in Fluids*, 84(7):409–442, 2016.
- [53] H. Xie, S. Koshizuka, and Y. Oka. Modelling of a single drop impact onto liquid film using particle method. *International Journal for Numerical Methods in Fluids*, 45(9):1009–1023, 2004.

Criando fórmulas de diferenças sobre malhas não-uniformes

Neste apêndice iremos descrever uma importante modificação feita na etapa de discretização do método numérico. O objetivo desta alteração é tornar possível a utilização de malhas com espaçamento não-uniforme.

Na seção 3.2, a malha computacional foi criada com espaçamentos constantes Δ_x e Δ_y . Neste momento, deixaremos de impor que estes valores sejam constantes, isto é, dados N_x e N_y , existirão duas sequências de pontos

$$\begin{aligned} x_0, x_1, x_2, \dots, x_{N_x} \quad \text{e} \\ y_0, y_1, y_2, \dots, y_{N_y}, \end{aligned} \tag{A.1}$$

tais que os espaçamentos

$$\begin{aligned} \Delta_{x_i} &= x_i - x_{i-1}, & i &= 1, 2, \dots, N_x \\ \Delta_{y_j} &= y_j - y_{j-1}, & j &= 1, 2, \dots, N_y \end{aligned} \tag{A.2}$$

não são necessariamente iguais.

O primeiro passo para a utilização de malhas não-uniforme é a criação de fórmulas de diferenças finitas que considerem a presença de espaçamentos variados. Vamos agora apresentar a estratégia utilizada para construir tais fórmulas.

A.1 Fórmulas de diferenças finitas em uma malha não-uniforme

O objetivo desta seção é criar fórmulas de diferenças finitas que aproximem os valores da primeira e segunda derivada de uma função. Iremos seguir uma abordagem baseada na utilização de polinômios interpoladores para criar tais fórmulas.

Considere uma sequência de pontos

$$x_0, x_1, \dots, x_N,$$

e os espaçamentos

$$\Delta_{x_i} = x_i - x_{i-1}, \quad i = 1, 2, \dots, N_x. \tag{A.3}$$

Suponha que, dada uma função $u = u(x)$, desejamos aproximar os valores de

$$\left. \frac{\partial u}{\partial x} \right|_{x=x_i} \quad \text{e} \quad \left. \frac{\partial^2 u}{\partial x^2} \right|_{x=x_i}. \tag{A.4}$$

Para criar uma fórmula de diferenças finitas, primeiramente é preciso definir quantos e quais pontos deseja-se inserir na fórmula. Neste trabalho, vamos utilizar fórmulas centradas de três pontos, isto é, para aproximar as derivadas em x_i queremos utilizar valores da função apenas nos pontos x_{i-1} , x_i e x_{i+1} .

Com três pontos é possível criar um polinômio interpolador de segundo grau para a função u . Este polinômio é dado por

$$\begin{aligned} p_2(x) &= \frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})}u_{i-1} + \frac{(x-x_{i-1})(x-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})}u_i + \frac{(x-x_{i-1})(x-x_i)}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)}u_{i+1} \\ &= \frac{(x-x_i)(x-x_{i+1})}{\Delta_{x_i}(\Delta_{x_i}+\Delta_{x_{i+1}})}u_{i-1} - \frac{(x-x_{i-1})(x-x_{i+1})}{\Delta_{x_i}\Delta_{x_{i+1}}}u_i + \frac{(x-x_{i-1})(x-x_i)}{\Delta_{x_{i+1}}(\Delta_{x_i}+\Delta_{x_{i+1}})}u_{i+1}. \end{aligned} \quad (\text{A.5})$$

Calculando a primeira e segunda derivada de p_2 , obtemos

$$p_2'(x) = \frac{2x-x_i-x_{i+1}}{\Delta_{x_i}(\Delta_{x_i}+\Delta_{x_{i+1}})}u_{i-1} - \frac{2x-x_{i-1}-x_{i+1}}{\Delta_{x_i}\Delta_{x_{i+1}}}u_i + \frac{2x-x_{i-1}-x_i}{\Delta_{x_{i+1}}(\Delta_{x_i}+\Delta_{x_{i+1}})}u_{i+1}, \quad (\text{A.6})$$

e

$$p_2''(x) = \frac{2}{\Delta_{x_i}(\Delta_{x_i}+\Delta_{x_{i+1}})}u_{i-1} - \frac{2}{\Delta_{x_i}\Delta_{x_{i+1}}}u_i + \frac{2}{\Delta_{x_{i+1}}(\Delta_{x_i}+\Delta_{x_{i+1}})}u_{i+1}. \quad (\text{A.7})$$

Como o objetivo é aproximar a derivada no ponto x_i , devemos aplicar (A.6) e (A.7) neste ponto. Com isto, o valor das derivadas são

$$p_2'(x_i) = -\frac{\Delta_{x_{i+1}}}{\Delta_{x_i}(\Delta_{x_i}+\Delta_{x_{i+1}})}u_{i-1} + \frac{\Delta_{x_{i+1}}-\Delta_{x_i}}{\Delta_{x_i}\Delta_{x_{i+1}}}u_i + \frac{\Delta_{x_i}}{\Delta_{x_{i+1}}(\Delta_{x_i}+\Delta_{x_{i+1}})}u_{i+1}, \quad (\text{A.8})$$

e

$$p_2''(x_i) = \frac{2}{\Delta_{x_i}(\Delta_{x_i}+\Delta_{x_{i+1}})}u_{i-1} - \frac{2}{\Delta_{x_i}\Delta_{x_{i+1}}}u_i + \frac{2}{\Delta_{x_{i+1}}(\Delta_{x_i}+\Delta_{x_{i+1}})}u_{i+1}. \quad (\text{A.9})$$

Portanto, apenas para escrever as fórmulas de forma mais genérica, considere que x_e é um ponto mais a esquerda, x_d um mais a direita e x_c o ponto central. Desta forma, as fórmulas de diferenças finitas que serão usadas para aproximar as derivadas em x_c são:

$$\begin{aligned} \left. \frac{\partial u}{\partial x} \right|_{x=x_c} &\approx -\frac{h_2}{h_1(h_1+h_2)}u_e + \frac{h_2-h_1}{h_1h_2}u_c + \frac{h_1}{h_2(h_1+h_2)}u_d, \\ \left. \frac{\partial^2 u}{\partial x^2} \right|_{x=x_c} &\approx \frac{2}{h_1(h_1+h_2)}u_e - \frac{2}{h_1h_2}u_c + \frac{2}{h_2(h_1+h_2)}u_d, \end{aligned} \quad (\text{A.10})$$

onde $h_1 = (x_c - x_e)$ e $h_2 = (x_d - x_c)$.

Aproximação dos termos convectivos pelo método CUBISTA

Nesta seção será descrita como o método CUBISTA foi utilizado para a aproximação dos termos convectivos presentes nas equações de quantidade de movimento e nas equações constitutivas. O desenvolvimento desta seção foca apenas na utilização prática do método, de modo que a construção teórica do mesmo pode ser vista em [3].

Os termos convectivos que advectam as variáveis presentes neste trabalho sempre possuem a forma

$$\frac{\partial(u\phi)}{\partial x}, \quad (\text{B.1})$$

onde ϕ é a variável sendo advectada, u é a velocidade de advecção e x é a coordenada do plano na qual a advecção ocorre.

Considere que o eixo da coordenada x foi discretizado com uma malha não-uniforme, conforme a Figura B.1. Considere também que os valores de ϕ são armazenados nos vértices da malha $(\dots, i-1, i, i+1, \dots)$ e que desejamos calcular a derivada (B.1) no vértice i .

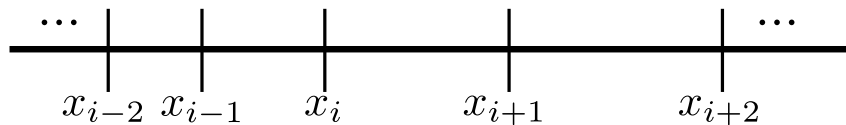


Figura B.1: Malha não-uniforme usada como exemplo na discretização.

O primeiro passo é aproximar a derivada (B.1) com uma fórmula de diferenças finitas. Neste trabalho sempre foi usada uma aproximação centrada portanto, utilizando a expressão (A.10), obtemos

$$\frac{\partial(u\phi)}{\partial x} \Big|_{x=x_i} \approx -\frac{h_2}{h_1(h_1+h_2)}u_{i-\frac{1}{2}}\phi_{i-\frac{1}{2}} + \frac{h_2-h_1}{h_1h_2}u_i\phi_i + \frac{h_1}{h_2(h_1+h_2)}u_{i+\frac{1}{2}}\phi_{i+\frac{1}{2}}, \quad (\text{B.2})$$

onde $h_1 = \frac{1}{2}(x_i - x_{i-1})$ e $h_2 = \frac{1}{2}(x_{i+1} - x_i)$.

Observe que os valores $\phi_{i-\frac{1}{2}}$ e $\phi_{i+\frac{1}{2}}$ da equação (B.2) estão localizados nas faces da malha, ou seja, em pontos nos quais a variável não está armazenada. É neste momento que o método CUBISTA será utilizado para aproximar estes dois valores.

Seja ϕ_f o valor da variável que se deseja calcular e que está localizado em uma face, ou seja, ϕ_f pode ser tanto $\phi_{i-\frac{1}{2}}$ quanto $\phi_{i+\frac{1}{2}}$. Vamos nomear os pontos ao redor de ϕ_f

de acordo com a direção da velocidade u_f conforme a Figura B.2. Os índices D, U e R significam, respectivamente, *Downstream*, *Upstream* e *Remote-upstream*.

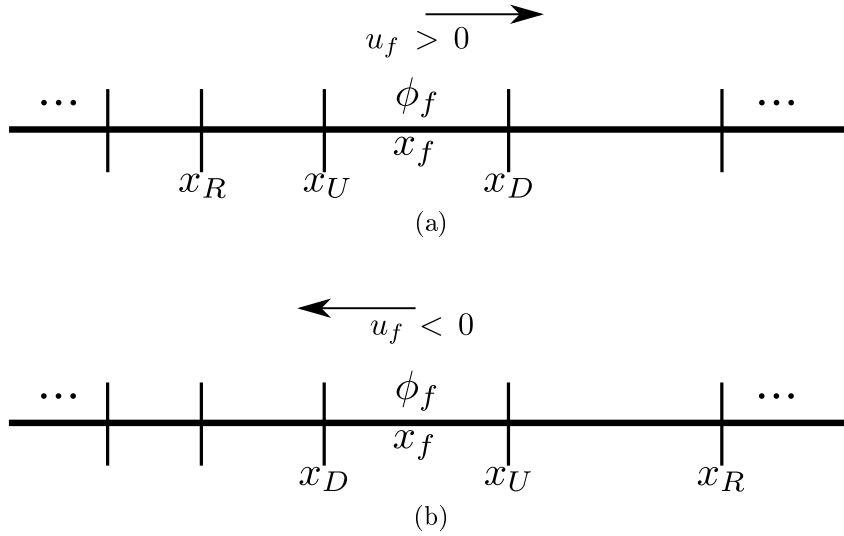


Figura B.2: Definição dos pontos x_D , x_U e x_R que serão usados no método CUBISTA. (a) caso com $u_f > 0$ e (b) caso com $u_f < 0$.

Para que as expressões do método CUBISTA fiquem mais simples, iremos agora realizar uma mudança de variáveis. Iremos definir uma variável normalizada $\hat{\phi}$ como

$$\hat{\phi} = \frac{\phi - \phi_R}{\phi_D - \phi_R}. \quad (\text{B.3})$$

Analogamente, a coordenada normalizada \hat{x} será dada por

$$\hat{x} = \frac{x - x_R}{x_D - x_R}. \quad (\text{B.4})$$

Observe que, nosso objetivo é calcular o valor de ϕ_f . Porém, se o valor de $\hat{\phi}_f$ for calculado, podemos facilmente recuperar ϕ_f através de (B.3).

Com estes valores definidos, o método CUBISTA diz que o valor de $\hat{\phi}_f$ é dado por

$$\hat{\phi}_f = \begin{cases} \left[1 + \frac{\hat{x}_f - \hat{x}_U}{3(1 - \hat{x}_U)} \right] \frac{\hat{x}_f}{\hat{x}_U} \hat{\phi}_U, & \text{se } 0 < \hat{\phi}_U < \frac{3}{4}\hat{x}_U \\ \frac{\hat{x}_f(1 - \hat{x}_f)}{\hat{x}_U(1 - \hat{x}_U)} \hat{\phi}_U + \frac{\hat{x}_f(\hat{x}_f - \hat{x}_U)}{1 - \hat{x}_U}, & \text{se } \frac{3}{4}\hat{x}_U \leq \hat{\phi}_U \leq \frac{1 + 2(\hat{x}_f - \hat{x}_U)}{2\hat{x}_f - \hat{x}_U} \hat{x}_U \\ 1 - \frac{1 - \hat{x}_f}{2(1 - \hat{x}_U)} (1 - \hat{\phi}_U), & \text{se } \frac{1 + 2(\hat{x}_f - \hat{x}_U)}{2\hat{x}_f - \hat{x}_U} \hat{x}_U < \hat{\phi}_U < 1 \\ \hat{\phi}_U, & \text{nos demais casos.} \end{cases} \quad (\text{B.5})$$

B.1 Exemplo 1: Equação de quantidade de movimento

Nesta seção iremos apresentar um exemplo de como uma das derivadas da equação (C.2) é aproximada usando o método CUBISTA.

Vimos que a derivada convectiva na direção x desta equação foi discretizada através da fórmula (C.3). Observe que, precisamos agora, usar o método CUBISTA para aproximar os valores de $U_{i-1,j}$ e $U_{i,j}$, que estão fora dos pontos onde U está armazenada.

Primeiramente, vamos aproximar o valor de $\phi_f = U_{i-1,j}$. Neste caso, note que a velocidade de advecção também é $u_f = U_{i-1,j}$, pois o termo dentro da derivada é U^2 . O valor da velocidade de advecção u_f precisa ser conhecido, portanto iremos apenas usar uma interpolação linear de termos vizinhos para calculá-lo, ou seja,

$$u_f = U_{i-1,j} = \frac{U_{i-\frac{3}{2},j} + U_{i-\frac{1}{2},j}}{2}. \quad (\text{B.6})$$

Em seguida, iremos definir quais são os pontos x_D , x_U e x_R , de acordo com o sinal de u_f . Teremos que:

$$\begin{cases} x_D = x_{i-\frac{1}{2}}, & x_U = x_{i-\frac{3}{2}} & x_R = x_{i-\frac{5}{2}}, & \text{se } u_f \geq 0 \\ x_D = x_{i-\frac{3}{2}}, & x_U = x_{i-\frac{1}{2}} & x_R = x_{i+\frac{1}{2}}, & \text{se } u_f < 0. \end{cases} \quad (\text{B.7})$$

Por fim, a fórmula CUBISTA (B.5) pode ser usada para calcular $\hat{\phi}_f$ e, conseqüentemente, calcular $\phi_f = U_{i-1,j}$ através de (B.3).

Precisamos agora aproximar o valor de $\phi_f = U_{i,j}$. Novamente, a velocidade de advecção também é $u_f = U_{i,j}$. Ela será interpolada por

$$u_f = U_{i,j} = \frac{U_{i-\frac{1}{2},j} + U_{i+\frac{1}{2},j}}{2}. \quad (\text{B.8})$$

Em seguida, os pontos x_D , x_U e x_R são definidos de acordo com o sinal de u_f . Temos que:

$$\begin{cases} x_D = x_{i+\frac{1}{2}}, & x_U = x_{i-\frac{1}{2}} & x_R = x_{i-\frac{3}{2}}, & \text{se } u_f \geq 0 \\ x_D = x_{i-\frac{1}{2}}, & x_U = x_{i+\frac{1}{2}} & x_R = x_{i+\frac{3}{2}}, & \text{se } u_f < 0. \end{cases} \quad (\text{B.9})$$

E, finalmente, o valor de $\phi_f = U_{i,j}$ é calculado através da fórmula CUBISTA (B.5) e da mudança de variáveis (B.3).

Agora que os valores de $U_{i-1,j}$ e $U_{i,j}$ são conhecidos, a derivada pode ser calculada normalmente através da fórmula de diferenças finitas (C.3).

B.2 Exemplo 2: Equações constitutivas

O exemplo desta seção aplica o método CUBISTA na aproximação da derivada (C.27) presente em várias equações constitutivas. Observe que, nosso objetivo é encontrar os valores de $T_{i+\frac{1}{2},j}$ e $T_{i,j}$, lembrando que T só está definido no centro de cada célula.

Primeiramente, vamos calcular o valor de $\phi_f = T_{i+\frac{1}{2},j}$. Neste caso, a velocidade de advecção é $u_f = U_{i+\frac{1}{2},j}$ e já está definida no ponto adequado. Agora, de acordo o sinal de u_f , definiremos os pontos x_D , x_U e x_R . Teremos que:

$$\begin{cases} x_D = x_{i+1}, & x_U = x_i & x_R = x_{i-1}, & \text{se } u_f \geq 0 \\ x_D = x_i, & x_U = x_{i+1} & x_R = x_{i+2}, & \text{se } u_f < 0. \end{cases} \quad (\text{B.10})$$

Por fim, a fórmula CUBISTA (B.5) é usada para calcular o valor de $\phi_f = T_{i+\frac{1}{2},j}$.

Agora precisamos calcular o valor de $\phi_f = T_{i-\frac{1}{2},j}$. Neste caso, a velocidade de advecção é $u_f = U_{i-\frac{1}{2},j}$ e já está definida no ponto adequado. Agora, de acordo o sinal de u_f , definiremos os pontos x_D , x_U e x_R . Temos que:

$$\begin{cases} x_D = x_i, & x_U = x_{i-1} & x_R = x_{i-2}, & \text{se } u_f \geq 0 \\ x_D = x_{i-1}, & x_U = x_i & x_R = x_{i+1}, & \text{se } u_f < 0. \end{cases} \quad (\text{B.11})$$

Por fim, a fórmula CUBISTA (B.5) é usada para calcular o valor de $\phi_f = T_{i-\frac{1}{2},j}$.

Com os valores de $T_{i-\frac{1}{2},j}$ e $T_{i+\frac{1}{2},j}$ calculados, a fórmula de diferenças finitas (C.27) pode ser usada normalmente.

Discretização espacial das equações

Neste apêndice iremos descrever a discretização espacial das equações realizada pelo método de diferenças finitas sobre uma malha não-uniforme.

C.1 Discretização da equação de quantidade de movimento

Inicialmente, vamos trabalhar com a equação referente a componente u da velocidade. Escrevendo (3.13) em coordenadas cartesianas, essa equação será dada por

$$\begin{aligned} \frac{\tilde{u}^{n+1} - u^n}{\Delta t} + \frac{\partial(u^n u^n)}{\partial x} + \frac{\partial(u^n V^n)}{\partial y} = -\frac{\partial p^n}{\partial x} + \frac{\beta}{Re} \left(\frac{\partial^2 \tilde{u}^{n+1}}{\partial x^2} + \frac{\partial^2 \tilde{u}^{n+1}}{\partial y^2} \right) + \\ \frac{1}{Re} \left(\frac{\partial T^{xx,n}}{\partial x} + \frac{\partial T^{xy,n}}{\partial y} \right) + \frac{1}{Fr^2} g_x^{n+1}, \end{aligned} \quad (C.1)$$

Aplicando (C.1) no ponto $X_{i-\frac{1}{2},j}$, temos:

$$\begin{aligned} \frac{\tilde{u}_{i-\frac{1}{2},j}^{n+1} - u_{i-\frac{1}{2},j}^n}{\Delta t} + \frac{\partial(u^2)}{\partial x} \Big|_{i-\frac{1}{2},j}^n + \frac{\partial(uv)}{\partial y} \Big|_{i-\frac{1}{2},j}^n = -\frac{\partial p}{\partial x} \Big|_{i-\frac{1}{2},j}^n + \frac{\beta}{Re} \left(\frac{\partial^2 \tilde{u}}{\partial x^2} \Big|_{i-\frac{1}{2},j}^{n+1} + \frac{\partial^2 \tilde{u}}{\partial y^2} \Big|_{i-\frac{1}{2},j}^{n+1} \right) + \\ \frac{1}{Re} \left(\frac{\partial T^{xx}}{\partial x} \Big|_{i-\frac{1}{2},j}^n + \frac{\partial T^{xy}}{\partial y} \Big|_{i-\frac{1}{2},j}^n \right) + \frac{1}{Fr^2} g_x^{n+1}(X_{i-\frac{1}{2},j}). \end{aligned} \quad (C.2)$$

O próximo passo será aproximar todas as derivadas de (C.2) por fórmulas de diferenças finitas centrais de segunda ordem. De modo a simplificar a ilustração desse processo, vamos aplicar as fórmulas de diferenças em cada um dos termos separadamente.

Vamos começar pelos termos não-lineares da equação, ou seja, os termos nos quais aparece uma multiplicação entre incógnitas. A aproximação destas derivadas será feita com fórmulas centrais. Sendo assim, as aproximações são dadas por

$$\frac{\partial(u^2)}{\partial x} \Big|_{i-\frac{1}{2},j}^n \approx -\frac{h_2}{h_1(h_1+h_2)}(u_{i-1,j}^n)^2 + \frac{h_2-h_1}{h_1 h_2}(u_{i-\frac{1}{2},j}^n)^2 + \frac{h_1}{h_2(h_1+h_2)}(u_{i,j}^n)^2, \quad (C.3)$$

$$\frac{\partial(uv)}{\partial y} \Big|_{i-\frac{1}{2},j}^n \approx \frac{u_{i-\frac{1}{2},j+\frac{1}{2}}^n v_{i-\frac{1}{2},j+\frac{1}{2}}^n - u_{i-\frac{1}{2},j-\frac{1}{2}}^n v_{i-\frac{1}{2},j-\frac{1}{2}}^n}{\Delta y_j}, \quad (C.4)$$

onde $h_1 = \frac{1}{2}\Delta_{x_{i-1}}$ e $h_2 = \frac{1}{2}\Delta_{x_i}$. Note que, quando aplicamos as fórmulas de diferenças finitas, obtivemos valores de u e v em pontos nos quais eles não estão definidos. O valor destas variáveis serão aproximados através do método CUBISTA para termos convectivos. Mais detalhes deste método são dados no Apêndice B.

Agora vamos discretizar as derivadas que aparecem no lado direito da equação (C.2), ou seja, a derivada da pressão, os termos viscosos (derivadas de segundo grau), e as derivadas do tensor \mathbf{T} . As fórmulas de diferenças que vamos utilizar para essas derivadas são:

$$\frac{\partial^2 u}{\partial x^2} \Big|_{i-\frac{1}{2},j}^{n+1} \approx \frac{2}{h_1(h_1+h_2)} u_{i-\frac{3}{2},j}^{n+1} - \frac{2}{h_1 h_2} u_{i-\frac{1}{2},j}^{n+1} + \frac{2}{h_2(h_1+h_2)} u_{i+\frac{1}{2},j}^{n+1}, \quad (\text{C.5})$$

onde $h_1 = \Delta_{x_{i-1}}$ e $h_2 = \Delta_{x_i}$.

$$\frac{\partial^2 u}{\partial y^2} \Big|_{i-\frac{1}{2},j}^{n+1} \approx \frac{2}{h_1(h_1+h_2)} u_{i-\frac{1}{2},j-1}^{n+1} - \frac{2}{h_1 h_2} u_{i-\frac{1}{2},j}^{n+1} + \frac{2}{h_2(h_1+h_2)} u_{i-\frac{1}{2},j+1}^{n+1}, \quad (\text{C.6})$$

onde $h_1 = \frac{1}{2}(\Delta_{y_{j-1}} + \Delta_{y_j})$ e $h_2 = \frac{1}{2}(\Delta_{y_j} + \Delta_{y_{j+1}})$.

$$\frac{\partial p}{\partial x} \Big|_{i-\frac{1}{2},j}^n \approx -\frac{h_2}{h_1(h_1+h_2)} p_{i-1,j}^n + \frac{h_2-h_1}{h_1 h_2} p_{i-\frac{1}{2},j}^n + \frac{h_1}{h_2(h_1+h_2)} p_{i,j}^n, \quad (\text{C.7})$$

onde $h_1 = \frac{1}{2}\Delta_{x_{i-1}}$ e $h_2 = \frac{1}{2}\Delta_{x_i}$.

$$\frac{\partial T}{\partial x} \Big|_{i-\frac{1}{2},j}^n \approx -\frac{h_2}{h_1(h_1+h_2)} T_{i-1,j}^n + \frac{h_2-h_1}{h_1 h_2} T_{i-\frac{1}{2},j}^n + \frac{h_1}{h_2(h_1+h_2)} T_{i,j}^n, \quad (\text{C.8})$$

onde $h_1 = \frac{1}{2}\Delta_{x_{i-1}}$ e $h_2 = \frac{1}{2}\Delta_{x_i}$

$$\frac{\partial T}{\partial y} \Big|_{i-\frac{1}{2},j}^n \approx -\frac{h_2}{h_1(h_1+h_2)} T_{i-\frac{1}{2},j-1}^n + \frac{h_2-h_1}{h_1 h_2} T_{i-\frac{1}{2},j}^n + \frac{h_1}{h_2(h_1+h_2)} T_{i-\frac{1}{2},j+1}^n, \quad (\text{C.9})$$

onde $h_1 = \frac{\Delta_{y_{j-1}} + \Delta_{y_j}}{2}$ e $h_2 = \frac{\Delta_{y_j} + \Delta_{y_{j+1}}}{2}$.

Substituindo todas as derivadas de (C.2) pelas fórmulas acima, criamos a equação de diferenças que deverá ser resolvida.

Analogamente, vamos agora discretizar a equação de transporte da componente da velocidade na direção y . Escrevendo (3.13) em coordenadas cartesianas, a equação na direção y é dada por

$$\begin{aligned} \frac{\tilde{v}^{n+1} - v^n}{\Delta_t} + \frac{\partial(u^n v^n)}{\partial x} + \frac{\partial(v^n v^n)}{\partial y} &= -\frac{\partial p^n}{\partial y} + \frac{\beta}{Re} \left(\frac{\partial^2 \tilde{v}^{n+1}}{\partial x^2} + \frac{\partial^2 \tilde{v}^{n+1}}{\partial y^2} \right) + \\ \frac{1}{Re} \left(\frac{\partial T^{xy,n}}{\partial x} + \frac{\partial T^{yy,n}}{\partial y} \right) &+ \frac{1}{Fr^2} g_y^{n+1}, \end{aligned} \quad (\text{C.10})$$

Aplicando (C.10) no ponto $X_{i,j-\frac{1}{2}}$, obtemos

$$\begin{aligned} \frac{\tilde{v}_{i,j-\frac{1}{2}}^{n+1} - v_{i,j-\frac{1}{2}}^n}{\Delta_t} + \frac{\partial(uv)}{\partial x} \Big|_{i,j-\frac{1}{2}}^n + \frac{\partial(v^2)}{\partial y} \Big|_{i,j-\frac{1}{2}}^n &= -\frac{\partial p}{\partial y} \Big|_{i,j-\frac{1}{2}}^n + \frac{\beta}{Re} \left(\frac{\partial^2 \tilde{v}}{\partial x^2} \Big|_{i,j-\frac{1}{2}}^{n+1} + \frac{\partial^2 \tilde{v}}{\partial y^2} \Big|_{i,j-\frac{1}{2}}^{n+1} \right) + \\ \frac{1}{Re} \left(\frac{\partial T^{xy}}{\partial x} \Big|_{i,j-\frac{1}{2}}^n + \frac{\partial T^{yy}}{\partial y} \Big|_{i,j-\frac{1}{2}}^n \right) &+ \frac{1}{Fr^2} g_y^{n+1}(X_{i,j-\frac{1}{2}}). \end{aligned} \quad (\text{C.11})$$

As derivadas de (C.11) serão discretizadas utilizando fórmulas criadas de forma idêntica ao que foi feito para a equação na direção x . As fórmulas de diferenças finitas usadas a serem usadas nesta equação são dadas por:

$$\left. \frac{\partial^2 v}{\partial x^2} \right|_{i,j-\frac{1}{2}}^{n+1} \approx \frac{2}{h_1(h_1+h_2)} v_{i-1,j-\frac{1}{2}}^{n+1} - \frac{2}{h_1 h_2} v_{i,j-\frac{1}{2}}^{n+1} + \frac{2}{h_2(h_1+h_2)} v_{i+1,j-\frac{1}{2}}^{n+1}, \quad (\text{C.12})$$

onde $h_1 = \frac{1}{2}(\Delta_{x_{i-1}} + \Delta_{x_i})$ e $h_2 = \frac{1}{2}(\Delta_{x_i} + \Delta_{x_{i+1}})$.

$$\left. \frac{\partial^2 v}{\partial y^2} \right|_{i,j-\frac{1}{2}}^{n+1} \approx \frac{2}{h_1(h_1+h_2)} v_{i,j-\frac{3}{2}}^{n+1} - \frac{2}{h_1 h_2} v_{i,j-\frac{1}{2}}^{n+1} + \frac{2}{h_2(h_1+h_2)} v_{i,j+\frac{1}{2}}^{n+1}, \quad (\text{C.13})$$

onde $h_1 = \Delta_{y_{j-1}}$ e $h_2 = \Delta_{y_j}$.

$$\left. \frac{\partial(v^2)}{\partial y} \right|_{i,j-\frac{1}{2}}^n \approx -\frac{h_2}{h_1(h_1+h_2)} (v_{i,j-1}^n)^2 + \frac{h_2-h_1}{h_1 h_2} (v_{i,j-\frac{1}{2}}^n)^2 + \frac{h_1}{h_2(h_1+h_2)} (v_{i,j}^n)^2, \quad (\text{C.14})$$

onde $h_1 = \frac{1}{2}\Delta_{y_{j-1}}$ e $h_2 = \frac{1}{2}\Delta_{y_j}$.

$$\left. \frac{\partial(uv)}{\partial x} \right|_{i,j-\frac{1}{2}}^n \approx \frac{u_{i+\frac{1}{2},j-\frac{1}{2}}^n v_{i+\frac{1}{2},j-\frac{1}{2}}^n - u_{i-\frac{1}{2},j-\frac{1}{2}}^n v_{i-\frac{1}{2},j-\frac{1}{2}}^n}{\Delta_{x_i}}. \quad (\text{C.15})$$

$$\left. \frac{\partial p}{\partial y} \right|_{i,j-\frac{1}{2}}^n \approx -\frac{h_2}{h_1(h_1+h_2)} p_{i,j-1}^n + \frac{h_2-h_1}{h_1 h_2} p_{i,j-\frac{1}{2}}^n + \frac{h_1}{h_2(h_1+h_2)} p_{i,j}^n, \quad (\text{C.16})$$

onde $h_1 = \frac{1}{2}\Delta_{y_{j-1}}$ e $h_2 = \frac{1}{2}\Delta_{y_j}$.

$$\left. \frac{\partial T}{\partial x} \right|_{i,j-\frac{1}{2}}^n \approx -\frac{h_2}{h_1(h_1+h_2)} T_{i-1,j-\frac{1}{2}}^n + \frac{h_2-h_1}{h_1 h_2} T_{i,j-\frac{1}{2}}^n + \frac{h_1}{h_2(h_1+h_2)} T_{i+1,j-\frac{1}{2}}^n, \quad (\text{C.17})$$

onde $h_1 = \frac{\Delta_{x_{i-1}} + \Delta_{x_i}}{2}$ e $h_2 = \frac{\Delta_{x_i} + \Delta_{x_{i+1}}}{2}$.

$$\left. \frac{\partial T}{\partial y} \right|_{i,j-\frac{1}{2}}^n \approx -\frac{h_2}{h_1(h_1+h_2)} T_{i,j-1}^n + \frac{h_2-h_1}{h_1 h_2} T_{i,j-\frac{1}{2}}^n + \frac{h_1}{h_2(h_1+h_2)} T_{i,j}^n, \quad (\text{C.18})$$

onde $h_1 = \frac{1}{2}\Delta_{y_{j-1}}$ e $h_2 = \frac{1}{2}\Delta_{y_j}$.

Portanto, o primeiro passo do método de projeção é resolver as equações de diferenças para a quantidade de movimento usando as fórmulas de aproximação acima. Com isso, ficam determinados os valores de \tilde{u}^{n+1} e \tilde{v}^{n+1} .

C.2 Discretização da equação de Poisson

Vamos agora discretizar a equação de Poisson (3.4). Escrevendo (3.4) em coordenadas cartesianas e aplicando-a no ponto $X_{i,j}$, temos

$$\left. \frac{\partial^2 \psi}{\partial x^2} \right|_{i,j}^{n+1} + \left. \frac{\partial^2 \psi}{\partial y^2} \right|_{i,j}^{n+1} = \left. \frac{\partial \tilde{u}}{\partial x} \right|_{i,j}^{n+1} + \left. \frac{\partial \tilde{v}}{\partial y} \right|_{i,j}^{n+1}. \quad (\text{C.19})$$

As derivadas desta equação serão aproximadas com as seguintes fórmulas de diferenças centradas:

$$\left. \frac{\partial^2 \psi}{\partial x^2} \right|_{i,j}^{n+1} \approx \frac{2}{h_1(h_1+h_2)} \psi_{i-1,j}^{n+1} - \frac{2}{h_1 h_2} \psi_{i,j}^{n+1} + \frac{2}{h_2(h_1+h_2)} \psi_{i+1,j}^{n+1}, \quad (\text{C.20})$$

onde $h_1 = \frac{1}{2}(\Delta_{x_{i-1}} + \Delta_{x_i})$ e $h_2 = \frac{1}{2}(\Delta_{x_i} + \Delta_{x_{i+1}})$.

$$\frac{\partial^2 \psi}{\partial y^2} \Big|_{i,j}^{n+1} \approx \frac{2}{h_1(h_1 + h_2)} \psi_{i,j-1}^{n+1} - \frac{2}{h_1 h_2} \psi_{i,j}^{n+1} + \frac{2}{h_2(h_1 + h_2)} \psi_{i,j+1}^{n+1}, \quad (\text{C.21})$$

onde $h_1 = \frac{1}{2}(\Delta_{y_{j-1}} + \Delta_{y_j})$ e $h_2 = \frac{1}{2}(\Delta_{y_j} + \Delta_{y_{j+1}})$.

$$\frac{\partial \tilde{u}}{\partial x} \Big|_{i,j}^{n+1} = \frac{\tilde{u}_{i+\frac{1}{2},j}^{n+1} - \tilde{u}_{i-\frac{1}{2},j}^{n+1}}{\Delta_{x_i}}, \quad (\text{C.22})$$

$$\frac{\partial \tilde{v}}{\partial y} \Big|_{i,j}^{n+1} = \frac{\tilde{v}_{i,j+\frac{1}{2}}^{n+1} - \tilde{v}_{i,j-\frac{1}{2}}^{n+1}}{\Delta_{y_j}}. \quad (\text{C.23})$$

Resolvendo a equação de Poisson com as aproximações acima, obteremos o valor do campo ψ^{n+1} em todo o domínio.

C.3 Discretização das equações de atualização da velocidade

Após o cálculo de $\tilde{\mathbf{u}}$ e ψ , o terceiro passo a ser realizado é a atualização da velocidade pela equação (3.3).

Escrevendo (3.3) em duas dimensões com coordenadas cartesianas obtemos duas equações. Essas equações, aplicadas no tempo t_{n+1} e respectivamente nos pontos $X_{i-\frac{1}{2},j}$ e $X_{i,j-\frac{1}{2}}$, são

$$\begin{cases} u_{i-\frac{1}{2},j}^{n+1} = \tilde{u}_{i-\frac{1}{2},j}^{n+1} - \frac{\partial \psi}{\partial x} \Big|_{i-\frac{1}{2},j}^{n+1}, \\ v_{i,j-\frac{1}{2}}^{n+1} = \tilde{v}_{i,j-\frac{1}{2}}^{n+1} - \frac{\partial \psi}{\partial y} \Big|_{i,j-\frac{1}{2}}^{n+1}. \end{cases} \quad (\text{C.24})$$

As derivadas de (C.24) serão substituídas pelas seguintes aproximações por diferenças finitas

$$\frac{\partial \psi}{\partial x} \Big|_{i-\frac{1}{2},j}^{n+1} \approx -\frac{h_2}{h_1(h_1 + h_2)} \psi_{i-1,j}^{n+1} + \frac{h_2 - h_1}{h_1 h_2} \psi_{i-\frac{1}{2},j}^{n+1} + \frac{h_1}{h_2(h_1 + h_2)} \psi_{i,j}^{n+1}, \quad (\text{C.25})$$

onde $h_1 = \frac{1}{2}\Delta_{x_{i-1}}$ e $h_2 = \frac{1}{2}\Delta_{x_i}$.

$$\frac{\partial \psi}{\partial x} \Big|_{i,j-\frac{1}{2}}^{n+1} \approx -\frac{h_2}{h_1(h_1 + h_2)} \psi_{i,j-1}^{n+1} + \frac{h_2 - h_1}{h_1 h_2} \psi_{i,j-\frac{1}{2}}^{n+1} + \frac{h_1}{h_2(h_1 + h_2)} \psi_{i,j}^{n+1}, \quad (\text{C.26})$$

onde $h_1 = \frac{1}{2}\Delta_{y_{j-1}}$ e $h_2 = \frac{1}{2}\Delta_{y_j}$.

C.4 Discretização das equações constitutivas

C.4.1 Formulação CSF

Cada uma das derivadas das equações (3.24)-(3.26) será substituída por uma fórmula de diferenças finitas. Estas fórmulas são dadas por

$$\frac{\partial(uT)}{\partial x} \Big|_{i,j}^n \approx \frac{u_{i+\frac{1}{2},j}^{n+1} T_{i+\frac{1}{2},j}^n - u_{i-\frac{1}{2},j}^{n+1} T_{i-\frac{1}{2},j}^n}{\Delta_{x_i}}, \quad (\text{C.27})$$

$$\left. \frac{\partial(vT)}{\partial y} \right|_{i,j}^n \approx \frac{v_{i,j+\frac{1}{2}}^{n+1} T_{i,j+\frac{1}{2}}^n - v_{i,j-\frac{1}{2}}^{n+1} T_{i,j-\frac{1}{2}}^n}{\Delta y_j}, \quad (\text{C.28})$$

$$\left. \frac{\partial u}{\partial x} \right|_{i,j}^{n+1} \approx \frac{u_{i+\frac{1}{2},j}^{n+1} - u_{i-\frac{1}{2},j}^{n+1}}{\Delta x_i}, \quad (\text{C.29})$$

$$\left. \frac{\partial v}{\partial y} \right|_{i,j}^{n+1} \approx \frac{v_{i,j+\frac{1}{2}}^{n+1} - v_{i,j-\frac{1}{2}}^{n+1}}{\Delta y_j}. \quad (\text{C.30})$$

$$\left. \frac{\partial v}{\partial x} \right|_{i,j}^{n+1} \approx -\frac{h_2}{h_1(h_1 + h_2)} v_{i-1,j}^{n+1} + \frac{(h_2 - h_1)}{h_1 h_2} v_{i,j}^{n+1} + \frac{h_1}{h_2(h_1 + h_2)} v_{i+1,j}^{n+1}, \quad (\text{C.31})$$

em que $h_1 = \frac{1}{2}(\Delta x_{i-1} + \Delta x_i)$ e $h_2 = \frac{1}{2}(\Delta x_i + \Delta x_{i+1})$. As incógnitas $v_{i-1,j}^{n+1}$, $v_{i,j}^{n+1}$, e $v_{i+1,j}^{n+1}$ são substituídas por interpolações lineares das duas vizinhas localizadas abaixo e acima. E

$$\left. \frac{\partial u}{\partial y} \right|_{i,j}^{n+1} \approx -\frac{h_2}{h_1(h_1 + h_2)} u_{i,j-1}^{n+1} + \frac{(h_2 - h_1)}{h_1 h_2} u_{i,j}^{n+1} + \frac{h_1}{h_2(h_1 + h_2)} u_{i,j+1}^{n+1}, \quad (\text{C.32})$$

em que $h_1 = \frac{1}{2}(\Delta y_{j-1} + \Delta y_j)$ e $h_2 = \frac{1}{2}(\Delta y_j + \Delta y_{j+1})$. As incógnitas $u_{i-1,j}^{n+1}$, $u_{i,j}^{n+1}$ e $u_{i+1,j}^{n+1}$ são substituídas por interpolações lineares das duas vizinhas localizadas a esquerda e direita.

Vale lembrar também, que os valores de T nas aproximações (C.27)-(C.28) estão em pontos nos quais o tensor não está definido. Estes valores são aproximados pelo método CUBISTA (ver Apêndice B).

C.4.2 Formulação NSF

Todas as derivadas presentes nas equações (3.27)-(3.29) serão substituídas por fórmulas de diferenças finitas. Com exceção das derivadas temporais, todas as outras serão idênticas ao que foi feito para a formulação CSF, ou seja, usaremos as aproximações (C.27)-(C.32), sendo que a variável genérica T pode ser substituída por λ , μ e ν . Já as derivadas temporais serão aproximadas por

$$\frac{\partial u_{i,j}^{n+1}}{\partial t} = \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} = \frac{u_{i+\frac{1}{2},j}^{n+1} + u_{i-\frac{1}{2},j}^{n+1} - u_{i+\frac{1}{2},j}^n - u_{i-\frac{1}{2},j}^n}{2\Delta t}, \quad (\text{C.33})$$

$$\frac{\partial v_{i,j}^{n+1}}{\partial t} = \frac{v_{i,j}^{n+1} - v_{i,j}^n}{\Delta t} = \frac{v_{i,j+\frac{1}{2}}^{n+1} + v_{i,j-\frac{1}{2}}^{n+1} - v_{i,j+\frac{1}{2}}^n - v_{i,j-\frac{1}{2}}^n}{2\Delta t}. \quad (\text{C.34})$$

C.5 Discretização das condições de contorno de superfície livre

Resta agora discretizar as duas equações (2.49)-(2.50) que são utilizadas nas células de superfície livre.

Primeiramente vamos discretizar a equação tangencial (2.50). Observe que, com as aproximações de vetor normal feitas na Figura 3.3, a equação (2.50) pode assumir uma de duas formas simplificadas, sendo elas:

$$\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} = -\frac{1}{\beta} T^{xy}, \quad (\text{C.35})$$

ou

$$\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x} = \frac{1}{2\beta} (T^{xx} - T^{yy}). \quad (\text{C.36})$$

Para que as velocidades coincidam com a distribuição da malha deslocada, a equação (C.35) será sempre aplicada em vértices da malha, isto é, em pontos $X_{i-\frac{1}{2},j-\frac{1}{2}}$. Sendo assim, as derivadas dessa equação serão aproximadas pelas seguintes fórmulas centradas:

$$\left. \frac{\partial v}{\partial x} \right|_{i-\frac{1}{2},j-\frac{1}{2}} = -\frac{h_2}{h_1(h_1+h_2)}v_{i-1,j-\frac{1}{2}} + \frac{(h_2-h_1)}{h_1h_2}v_{i-\frac{1}{2},j-\frac{1}{2}} + \frac{h_1}{h_2(h_1+h_2)}v_{i,j-\frac{1}{2}}, \quad (\text{C.37})$$

em que $h_1 = \frac{1}{2}\Delta_{x_{i-1}}$ e $h_2 = \frac{1}{2}\Delta_{x_i}$. A incógnita $v_{i-\frac{1}{2},j-\frac{1}{2}}$ é substituída por uma interpolação linear das duas vizinhas. E

$$\left. \frac{\partial u}{\partial y} \right|_{i-\frac{1}{2},j-\frac{1}{2}} = -\frac{h_2}{h_1(h_1+h_2)}u_{i-\frac{1}{2},j-1} + \frac{(h_2-h_1)}{h_1h_2}u_{i-\frac{1}{2},j-\frac{1}{2}} + \frac{h_1}{h_2(h_1+h_2)}u_{i-\frac{1}{2},j}, \quad (\text{C.38})$$

em que $h_1 = \frac{1}{2}\Delta_{y_{j-1}}$ e $h_2 = \frac{1}{2}\Delta_{y_j}$. A incógnita $u_{i-\frac{1}{2},j-\frac{1}{2}}$ é substituída por uma interpolação linear das duas vizinhas.

Já a equação (C.36) será aplicada sempre em centros de célula, isto é, em $X_{i,j}$. Dessa forma, as derivadas dessa equação serão aproximadas por

$$\begin{aligned} \left. \frac{\partial v}{\partial y} \right|_{i,j} &= \frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{\Delta_{y_j}}, \\ \left. \frac{\partial u}{\partial x} \right|_{i,j} &= \frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{\Delta_{x_i}}. \end{aligned} \quad (\text{C.39})$$

Vamos agora discretizar a equação de superfície livre na direção normal. Vimos anteriormente que esta equação foi usada para construir uma condição de contorno para a equação de Poisson. Essa condição de contorno construída é dada em (3.19). Para evitar que a discretização das derivadas em (3.19) leve a muitas incógnitas fora da região de fluido, iremos trabalhar com essa equação em casos separados. Cada caso corresponde a uma das possibilidade de vetor normal na Figura 3.3. Em todos os casos, que são dados abaixo, a equação é aplicada no centro de uma célula, isto é, em $X_{i,j}$.

C.5.1 Caso 1: $\mathbf{n} = (0, 1)$

Neste caso queremos evitar que a discretização das derivadas utilize incógnitas que estejam acima do ponto central (pois não há fluido na célula acima da central). Portanto, primeiramente vamos eliminar uma das derivadas na direção y usando a relação

$$\frac{\partial v}{\partial y} = -\frac{\partial u}{\partial x}, \quad (\text{C.40})$$

que vem da equação de continuidade. Substituindo (C.40) em (3.19), obtemos

$$\begin{aligned} p^n + \frac{\psi^{n+1}}{\Delta_t} &= \frac{2\beta}{Re} \left[(n_x^2 - n_y^2) \frac{\partial u^{n+1}}{\partial x} + n_x n_y \left(\frac{\partial u^{n+1}}{\partial y} + \frac{\partial v^{n+1}}{\partial x} \right) \right] + \\ &\quad \frac{1}{Re} [n_x^2 T^{xx,n} + 2n_x n_y T^{xy,n} + n_y^2 T^{yy,n}] + \frac{1}{We} \kappa. \end{aligned} \quad (\text{C.41})$$

Em seguida, usaremos a decomposição do método de projeção e a equação se tornará

$$\begin{aligned} p^n + \frac{\psi^{n+1}}{\Delta_t} &= \frac{2\beta}{Re} \left[(n_x^2 - n_y^2) \frac{\partial}{\partial x} \left\{ \tilde{u}^{n+1} - \left. \frac{\partial \psi^{n+1}}{\partial x} \right|_{i,j} \right\} + n_x n_y \left(\frac{\partial}{\partial y} \left\{ \tilde{u}^{n+1} - \left. \frac{\partial \psi^{n+1}}{\partial x} \right|_{i,j} \right\} + \right. \right. \\ &\quad \left. \left. \frac{\partial}{\partial x} \left\{ \tilde{v}^{n+1} - \left. \frac{\partial \psi^{n+1}}{\partial y} \right|_{i,j} \right\} \right) \right] + \frac{1}{Re} [n_x^2 T^{xx,n} + 2n_x n_y T^{xy,n} + n_y^2 T^{yy,n}] + \frac{1}{We} \kappa. \end{aligned} \quad (\text{C.42})$$

Agora, para evitar pontos acima do central, vamos aproximar as derivadas de (C.42) com diferenças centrais na direção x e regressivas na direção y . Sendo assim, as derivadas de (C.42) serão aproximadas por:

$$\left. \frac{\partial \tilde{u}^{n+1}}{\partial x} \right|_{i,j} = \frac{\tilde{U}_{i+\frac{1}{2},j}^{n+1} - \tilde{U}_{i-\frac{1}{2},j}^{n+1}}{\Delta x_i}, \quad (\text{C.43})$$

$$\left. \frac{\partial^2 \psi^{n+1}}{\partial x^2} \right|_{i,j} = \frac{2}{h_1(h_1 + h_2)} \psi_{i-1,j}^{n+1} - \frac{2}{h_1 h_2} \psi_{i,j}^{n+1} + \frac{2}{h_2(h_1 + h_2)} \psi_{i+1,j}^{n+1}, \quad (\text{C.44})$$

em que $h_1 = \frac{1}{2}(\Delta x_{i-1} + \Delta x_i)$ e $h_2 = \frac{1}{2}(\Delta x_i + \Delta x_{i+1})$.

$$\left. \frac{\partial \tilde{u}^{n+1}}{\partial y} \right|_{i,j} = -\frac{h_2}{h_1(h_1 + h_2)} \tilde{U}_{i,j-1}^{n+1} + \frac{(h_2 - h_1)}{h_1 h_2} \tilde{U}_{i,j}^{n+1} + \frac{h_1}{h_2(h_1 + h_2)} \tilde{U}_{i,j+1}^{n+1}, \quad (\text{C.45})$$

em que $h_1 = \frac{1}{2}(\Delta y_{j-1} + \Delta y_j)$ e $h_2 = \frac{1}{2}(\Delta y_j + \Delta y_{j+1})$. As três incógnitas são aproximadas com interpolação linear de dois valores vizinhos.

$$\left. \frac{\partial \tilde{v}^{n+1}}{\partial x} \right|_{i,j} = -\frac{h_2}{h_1(h_1 + h_2)} \tilde{V}_{i-1,j}^{n+1} + \frac{(h_2 - h_1)}{h_1 h_2} \tilde{V}_{i,j}^{n+1} + \frac{h_1}{h_2(h_1 + h_2)} \tilde{V}_{i+1,j}^{n+1}, \quad (\text{C.46})$$

em que $h_1 = \frac{1}{2}(\Delta x_{i-1} + \Delta x_i)$ e $h_2 = \frac{1}{2}(\Delta x_i + \Delta x_{i+1})$.

$$\begin{aligned} \left. \frac{\partial}{\partial y} \left(\frac{\partial \psi^{n+1}}{\partial x} \right) \right|_{i,j} &= \frac{1}{\frac{1}{2}(\Delta y_{j-1} + \Delta y_j)} \left[-\frac{h_2}{h_1(h_1 + h_2)} \psi_{i-1,j}^{n+1} + \frac{(h_2 - h_1)}{h_1 h_2} \psi_{i,j}^{n+1} + \frac{h_1}{h_2(h_1 + h_2)} \psi_{i+1,j}^{n+1} \right. \\ &\quad \left. + \frac{h_2}{h_1(h_1 + h_2)} \psi_{i-1,j-1}^{n+1} - \frac{(h_2 - h_1)}{h_1 h_2} \psi_{i,j-1}^{n+1} - \frac{h_1}{h_2(h_1 + h_2)} \psi_{i+1,j-1}^{n+1} \right], \end{aligned} \quad (\text{C.47})$$

em que $h_1 = \frac{1}{2}(\Delta x_{i-1} + \Delta x_i)$ e $h_2 = \frac{1}{2}(\Delta x_i + \Delta x_{i+1})$.

C.5.2 Caso 2: $\mathbf{n} = (0, -1)$

Novamente iremos usar a equação (C.42), utilizada no caso 1. Porém, desta vez as derivadas de ψ na direção y serão aproximadas com fórmulas progressivas, evitando que incógnitas abaixo da central sejam criadas.

Observe que apenas a derivada cruzada (C.47) será discretizada de forma diferente do que foi feito no caso 1, de modo que não precisamos reescrever todas as outras. Essa derivada será discretizada por

$$\begin{aligned} \left. \frac{\partial}{\partial y} \left(\frac{\partial \psi^{n+1}}{\partial x} \right) \right|_{i,j} &= \frac{1}{\frac{1}{2}(\Delta y_j + \Delta y_{j+1})} \left[-\frac{h_2}{h_1(h_1 + h_2)} \psi_{i-1,j+1}^{n+1} + \frac{(h_2 - h_1)}{h_1 h_2} \psi_{i,j+1}^{n+1} + \frac{h_1}{h_2(h_1 + h_2)} \psi_{i+1,j+1}^{n+1} \right. \\ &\quad \left. + \frac{h_2}{h_1(h_1 + h_2)} \psi_{i-1,j}^{n+1} - \frac{(h_2 - h_1)}{h_1 h_2} \psi_{i,j}^{n+1} - \frac{h_1}{h_2(h_1 + h_2)} \psi_{i+1,j}^{n+1} \right], \end{aligned} \quad (\text{C.48})$$

em que $h_1 = \frac{1}{2}(\Delta x_{i-1} + \Delta x_i)$ e $h_2 = \frac{1}{2}(\Delta x_i + \Delta x_{i+1})$.

C.5.3 Caso 3: $\mathbf{n} = (1, 0)$

Nesse caso queremos evitar que incógnitas sejam criadas a direita do ponto central. Começaremos substituindo em (3.19) a seguinte relação

$$\frac{\partial u}{\partial x} = -\frac{\partial v}{\partial y}, \quad (\text{C.49})$$

que vem da equação de continuidade. Essa substituição nos leva a equação

$$p^n + \frac{\psi^{n+1}}{\Delta_t} = \frac{2\beta}{Re} \left[(n_y^2 - n_x^2) \frac{\partial v^{n+1}}{\partial y} + n_x n_y \left(\frac{\partial u^{n+1}}{\partial y} + \frac{\partial v^{n+1}}{\partial x} \right) \right] + \frac{1}{Re} [n_x^2 T^{xx,n} + 2n_x n_y T^{xy,n} + n_y^2 T^{yy,n}] + \frac{1}{We} \kappa. \quad (\text{C.50})$$

Usando a decomposição do método de projeção, obtemos então

$$p^n + \frac{\psi^{n+1}}{\Delta_t} = \frac{2\beta}{Re} \left[(n_y^2 - n_x^2) \frac{\partial}{\partial y} \left\{ \tilde{v}^{n+1} - \frac{\partial \psi^{n+1}}{\partial y} \Big|_{i,j} \right\} + n_x n_y \left(\frac{\partial}{\partial y} \left\{ \tilde{u}^{n+1} - \frac{\partial \psi^{n+1}}{\partial x} \Big|_{i,j} \right\} + \frac{\partial}{\partial x} \left\{ \tilde{v}^{n+1} - \frac{\partial \psi^{n+1}}{\partial y} \Big|_{i,j} \right\} \right) \right] + \frac{1}{Re} [n_x^2 T^{xx,n} + 2n_x n_y T^{xy,n} + n_y^2 T^{yy,n}] + \frac{1}{We} \kappa. \quad (\text{C.51})$$

Apenas duas derivadas de (C.51) serão discretizadas de forma diferente do que foi feito no caso 1 e, sendo assim, não iremos reescrever as demais. Essas duas derivadas serão aproximadas por:

$$\frac{\partial \tilde{v}^{n+1}}{\partial y} \Big|_{i,j} = \frac{\tilde{V}_{i,j+\frac{1}{2}}^{n+1} - \tilde{V}_{i,j-\frac{1}{2}}^{n+1}}{\Delta_{y_j}}. \quad (\text{C.52})$$

$$\frac{\partial^2 \psi^{n+1}}{\partial y^2} \Big|_{i,j} = \frac{2}{h_1(h_1 + h_2)} \psi_{i,j-1}^{n+1} - \frac{2}{h_1 h_2} \psi_{i,j}^{n+1} + \frac{2}{h_2(h_1 + h_2)} \psi_{i,j+1}^{n+1}, \quad (\text{C.53})$$

em que $h_1 = \frac{1}{2}(\Delta_{y_{j-1}} + \Delta_{y_j})$ e $h_2 = \frac{1}{2}(\Delta_{y_j} + \Delta_{y_{j+1}})$.

$$\frac{\partial}{\partial y} \left(\frac{\partial \psi^{n+1}}{\partial x} \right) \Big|_{i,j} = \frac{1}{\frac{1}{2}(\Delta_{x_{i-1}} + \Delta_{x_i})} \left[-\frac{h_2}{h_1(h_1 + h_2)} \psi_{i,j-1}^{n+1} + \frac{(h_2 - h_1)}{h_1 h_2} \psi_{i,j}^{n+1} + \frac{h_1}{h_2(h_1 + h_2)} \psi_{i,j+1}^{n+1} + \frac{h_2}{h_1(h_1 + h_2)} \psi_{i-1,j-1}^{n+1} - \frac{(h_2 - h_1)}{h_1 h_2} \psi_{i-1,j}^{n+1} - \frac{h_1}{h_2(h_1 + h_2)} \psi_{i-1,j+1}^{n+1} \right], \quad (\text{C.54})$$

em que $h_1 = \frac{1}{2}(\Delta_{y_{j-1}} + \Delta_{y_j})$ e $h_2 = \frac{1}{2}(\Delta_{y_j} + \Delta_{y_{j+1}})$.

C.5.4 Caso 4: $\mathbf{n} = (-1, 0)$

A mesma equação do caso 3 (C.51) será aplicada aqui, porém agora queremos evitar que incógnitas sejam criadas a esquerda do ponto central. Apenas uma derivada será aproximada de forma diferente do caso 3, sendo ela

$$\frac{\partial}{\partial y} \left(\frac{\partial \psi^{n+1}}{\partial x} \right) \Big|_{i,j} = \frac{1}{\frac{1}{2}(\Delta_{x_i} + \Delta_{x_{i+1}})} \left[-\frac{h_2}{h_1(h_1 + h_2)} \psi_{i+1,j-1}^{n+1} + \frac{(h_2 - h_1)}{h_1 h_2} \psi_{i+1,j}^{n+1} + \frac{h_1}{h_2(h_1 + h_2)} \psi_{i+1,j+1}^{n+1} + \frac{h_2}{h_1(h_1 + h_2)} \psi_{i,j-1}^{n+1} - \frac{(h_2 - h_1)}{h_1 h_2} \psi_{i,j}^{n+1} - \frac{h_1}{h_2(h_1 + h_2)} \psi_{i,j+1}^{n+1} \right], \quad (\text{C.55})$$

em que $h_1 = \frac{1}{2}(\Delta_{y_{j-1}} + \Delta_{y_j})$ e $h_2 = \frac{1}{2}(\Delta_{y_j} + \Delta_{y_{j+1}})$.

C.5.5 Caso 5: $\mathbf{n} = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$

A equação do caso 1 (C.42) será aplicada aqui, mas dessa vez desejamos evitar que incógnitas sejam criadas a direita e acima do ponto central. Uma única derivada será aproximada de forma diferente do que foi feito no caso 1, sendo ela

$$\frac{\partial}{\partial x} \left(\frac{\partial \psi^{n+1}}{\partial y} \right) \Big|_{i,j} = \frac{\psi_{i,j}^{n+1} - \psi_{i,j-1}^{n+1} - \psi_{i-1,j}^{n+1} + \psi_{i-1,j-1}^{n+1}}{\frac{1}{4}(\Delta_{x_{i-1}} + \Delta_{x_i})(\Delta_{y_{j-1}} + \Delta_{y_j})}. \quad (\text{C.56})$$

C.5.6 Caso 6: $\mathbf{n} = \left(-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$

A mesma equação do caso 1 (C.42) será usada, mas vamos evitar criar incógnitas a esquerda e acima do ponto central. Uma única derivada será diferente do que foi feito no caso 1, sendo ela

$$\frac{\partial}{\partial x} \left(\frac{\partial \psi^{n+1}}{\partial y} \right) \Big|_{i,j} = \frac{\psi_{i+1,j}^{n+1} - \psi_{i+1,j-1}^{n+1} - \psi_{i,j}^{n+1} + \psi_{i,j-1}^{n+1}}{\frac{1}{4}(\Delta_{x_i} + \Delta_{x_{i+1}})(\Delta_{y_{j-1}} + \Delta_{y_j})}. \quad (\text{C.57})$$

C.5.7 Caso 7: $\mathbf{n} = \left(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)$

A mesma equação do caso 1 (C.42) será usada, mas vamos evitar criar incógnitas a direita e abaixo do ponto central. Uma única derivada será diferente do que foi feito no caso 1, sendo ela

$$\frac{\partial}{\partial x} \left(\frac{\partial \psi^{n+1}}{\partial y} \right) \Big|_{i,j} = \frac{\psi_{i,j+1}^{n+1} - \psi_{i,j}^{n+1} - \psi_{i-1,j+1}^{n+1} + \psi_{i-1,j}^{n+1}}{\frac{1}{4}(\Delta_{x_{i-1}} + \Delta_{x_i})(\Delta_{y_j} + \Delta_{y_{j+1}})}. \quad (\text{C.58})$$

C.5.8 Caso 8: $\mathbf{n} = \left(-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)$

A mesma equação do caso 1 (C.42) será usada, mas vamos evitar criar incógnitas a esquerda e abaixo do ponto central. Uma única derivada será diferente do que foi feito no caso 1, sendo ela

$$\frac{\partial}{\partial x} \left(\frac{\partial \psi^{n+1}}{\partial y} \right) \Big|_{i,j} = \frac{\psi_{i+1,j+1}^{n+1} - \psi_{i+1,j}^{n+1} - \psi_{i,j+1}^{n+1} + \psi_{i,j}^{n+1}}{\frac{1}{4}(\Delta_{x_i} + \Delta_{x_{i+1}})(\Delta_{y_j} + \Delta_{y_{j+1}})}. \quad (\text{C.59})$$