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

Estudos de Técnicas de Reinicialização em Métodos Level-Set e suas Aplicações em Escoamentos Bifásicos

Amauri Gabriel de Jesus Junior

Orientador: Prof. Dr. Cassio Machiaveli Oishi

Programa: Matemática Aplicada e Computacional

Presidente Prudente, Março de 2016

UNIVERSIDADE ESTADUAL PAULISTA

Faculdade de Ciências e Tecnologia de Presidente Prudente

Programa de Pós-Graduação em Matemática Aplicada e Computacional

Estudos de Técnicas de Reinicialização em Métodos Level-Set e suas Aplicações em Escoamentos Bifásicos

Amauri Gabriel de Jesus Junior

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, Março de 2016

FICHA CATALOGRÁFICA

Jesus Junior, Amauri Gabriel de.
J56e Estudos de técnicas de reinicialização em métodos level-set e suas aplicações em escoamentos bifásicos / Amauri Gabriel de Jesus Junior. - Presidente Prudente : [s.n.], 2016
104 f.

Orientador: Cássio Machiaveli Oishi
Dissertação (mestrado) - Universidade Estadual Paulista, Faculdade de Ciências e Tecnologia
Inclui bibliografia

1. Matemática aplicada. 2. Solução numérica. 3. Representação implícita de interfaces. I. Oishi, Cássio Machiaveli. II. Universidade Estadual Paulista. Faculdade de Ciências e Tecnologia. III. Título.

Dedico esse trabalho à minha família, sem a qual ele não teria sido realizado, e a todos os amigos que me ajudaram durante esse período.

Agradecimentos

Agradeço primeiramente a Deus pela força concedida nos momentos de dificuldade no desenvolvimento desse projeto. À professora dra. Vanessa Avansini Botta Pirani por ter me convidado a me inscrever no mestrado. Ao professor dr. Cássio Machiaveli Oishi pela orientação e pelos conselhos. Agradeço, também, a todos os professores que lecionaram as disciplinas que cursei pela paciência e dedicação. À CAPES pelo apoio financeiro concedido na forma de bolsa de estudos. À todos que de alguma maneira contribuíram para que esse trabalho pudesse ser realizado, meus sinceros agrecimentos.

O temor do Senhor é a instrução da sabedoria.
Provérbios, 15, v33

Resumo

Esse trabalho apresenta um estudo sobre técnicas de reinicialização para métodos level-set com aplicação em escoamentos de fluidos. O transporte da interface é feito com um método WENO de quinta ordem para as derivadas espaciais, e um Runge-Kutta de terceira ordem para a derivada temporal. Utilizamos malha uniforme, no contexto de diferenças finitas, e nos limitamos a estudar casos bidimensionais. Após o estudo da advecção da interface, apresentamos duas técnicas de reinicialização para restaurar a propriedade de função distância da função level-set sem, que haja perda de massa. A primeira das técnicas consiste na resolução de uma equação diferencial parcial, e a outra é puramente geométrica. Utilizando as Equações de Navier-Stokes, estudamos duas técnicas para o cálculo da força devido à tensão superficial: CSF e SSF. Apresentamos testes numéricos com velocidade prescrita para verificar as duas técnicas de reinicialização, e resolvemos problemas de escoamentos bifásicos com tensão superficial para verificar como as duas técnicas de reinicialização se comportam nesse tipo de problema.

Palavras-Chave: *Level Set, Técnicas de Reinicialização, Escoamentos Multifásicos, Representação Implícita de Interfaces.*

Abstract

This work presents a study about redistancing schemes for level-set methods with applications in fluid flow. The transport of the interface is performed with a fifth order WENO scheme combined with a third order accurate Runge-Kutta method. After the advection of the interface, we analise two redistancing methods to restore the distance function property of the level-set function. The first method consists in the solution of a partial differential equation, while the other is fully geometric. Using the Navier-Stokes equations, we study two different techniques for the calculation of the surface tension force: CSF and SSF. In order to numerically verify the two redistancing schemes, we present tests with prescribed velocity field. Finally, we apply the redistancing techniques for solving biphase fluid flows in the presence of the surface tension.

Keywords: *Level Set, Redistancing Schemes, Biphasic Flow, Implicit Surface.*

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo

1

Introdução

Muitos problemas em Mecânica dos Fluidos contam com a presença de dois ou mais fluidos. Para simular computacionalmente esses problemas, é necessário representar de alguma forma a região de fronteira (interface) entre os diferentes fluidos envolvidos. Exemplos de problemas que envolvem dois fluidos são problemas com sprays e atomização [19].

A representação matemática de interfaces pode ser feita de duas formas distintas: de maneira explícita e de maneira implícita. A representação explícita pode ser feita, por exemplo, com métodos do tipo *front-tracking* [43], nos quais a interface é representada por partículas de massa desprezível, e o contorno "real" da interface é reconstruído apenas através da posição dessas partículas (por exemplo, fazendo alguma interpolação). Algumas das vantagens e desvantagens da representação explícita são apresentadas na seção 2.1.

Na representação implícita, a interface é representada sem o uso de partículas marcadoras. Por exemplo, no método *Volume of Fluid* (VOF), utiliza-se a função fração de volume, que diz qual porção de cada célula computacional está preenchida com um determinado tipo de fluido. Já métodos level-set utilizam uma curva de nível qualquer de alguma função escalar para fazer essa representação. Tradicionalmente, essa função é uma função distância sinalizada, e a curva de nível adotada é o isocontorno nulo [39].

A utilização de métodos level-set para a propagação de interfaces vem desde 1988, quando Osher e Sethian [27] apresentaram a primeira formulação. Em 1990, Mulder e Osher [23] apresentaram estudos sobre a utilização de métodos level-set para escoamentos compressíveis, estudando um escoamento com hélio e ar.

A seguir, em 1994, Sussman *et al.* [39] propuseram um método level-set para ser utilizado com escoamentos incompressíveis bifásicos, realizando simulações com água e ar. Desde então, diversos autores têm utilizado métodos level-set para escoamentos bifásicos [5, 10, 18].

O método proposto por Sussman *et al.* utiliza uma função distância sinalizada para representar a interface. Contudo, conforme a interface é movimentada, ocorre o surgimento de gradientes muito altos (ou muito baixos) nas proximidades da interface, fazendo com que a função level-set perca a sua propriedade de função distância. Isso acarreta em imprecisões no cálculo da força devido à tensão superficial, conforme dito por Croce *et al.* [10].

Além disso, métodos level-set costumam apresentar grandes problemas de perda de massa [21], o que pode provocar comportamentos não-físicos durante as simulações numéricas. Luo *et al.* [21] fizeram uma descrição detalhada sobre formas de melhorar a conservação de massa. Entre elas, podemos destacar:

Transporte da função level-set com métodos de alta ordem Nourgaliev *et al.* [24] verificaram que a perda de massa quando a discretização é feita com um método WENO de quinta ordem é menor do que quando é feita com métodos de ordem mais baixa. Salih e Moulic [31] fizeram observação semelhante.

Utilização de métodos híbridos Métodos VOF são consideravelmente melhores que métodos level-set quanto à conservação de massa, embora sejam menos precisos no cálculo da curvatura e do vetor normal. Sendo assim, é natural pensar em alguma forma de combinar características dos dois métodos. Entre esses métodos, podemos destacar o método VOSET [34] e o CLSVOF [38].

Métodos level-set adaptativos Como, em problemas de fluidos, a simulação precisa ficar bem calculada apenas numa região vizinha à interface, surgiram métodos level-set adaptativos [20, 24, 36], que buscam melhorar a solução apenas nessas regiões, sob custo computacional aceitável.

Melhoria da reinicialização A reinicialização é necessária para restaurar a propriedade de função distância da função level-set. Todavia, ela também pode contribuir para a perda de massa. Em 1998, Sussman *et al.* [37] adicionaram uma correção à sua reinicialização por EDP, a fim de melhorar a conservação de massa. Algumas variações dessa reinicialização foram propostas na literatura [10, 45]. A reinicialização geométrica também possui alguns passos que devem ser seguidos para preservar a massa da interface [4].

Para restaurar a propriedade de função distância da função level-set, optamos por utilizar duas rotinas de reinicialização distintas. Esser *et al.* [11] classificou as reinicializações existentes na literatura em dois principais grupos: as baseadas em EDP e as geométricas. Nesse trabalho, estudamos uma reinicialização de cada um desses grupos. A primeira delas, que é uma reinicialização por EDP, foi proposta por Sussman *et al.* [39] já em 1994, e consiste de resolver uma nova equação de advecção a cada passo de tempo da equação de transporte da interface. Essa reinicialização segue bem utilizada até os dias de hoje. Em 2010 ela foi utilizada por Croce *et al.* [10], e em 2013 foi testada de maneira paralela em um cluster GPU por Zaspel e Griebel [46]. Lalanne *et al.* [18] também utilizaram abordagem semelhante em 2015. A reinicialização geométrica que escolhemos foi proposta por Ausas *et al.* [4]. Não há muitos trabalhos publicados utilizando essa reinicialização geométrica, mas Ausas *et al.* [3] e Buscaglia e Ausas [8] resolveram alguns problemas com escoamentos bifásicos no contexto de elementos finitos.

Além disso trabalho optamos por utilizar métodos de alta ordem para a aproximação da solução das equações de transporte da função level-set. Utilizamos WENO de quinta ordem para a aproximação das derivadas espaciais, e Runge-Kutta de terceira ordem para as derivadas temporais. Ainda, fizemos uma implementação cuidadosa dos passos de conservação de massa nas duas rotinas de reinicialização estudadas.

No Capítulo 2, dois paradigmas para a representação de interfaces são discutidos: a representação explícita e a implícita. O método level-set, que é um método de representação implícita, é, então, apresentado. Nesse mesmo Capítulo apresentamos as equações para o cálculo do vetor normal e da curvatura da interface. Mostramos a malha computacional utilizada, para que possamos apresentar a discretização das equações. Por fim, discretizamos a equação de transporte da interface com o método WENO de quinta ordem e o Runge-Kutta de terceira ordem mencionados anteriormente.

No Capítulo 3, detalhamos as duas técnicas de reinicialização estudadas: a reinicialização por EDP e a geométrica. Em ambas as técnicas, discutimos também detalhes adicionais que precisam ser seguidos para reduzir a perda de massa do método. Ainda

nesse Capítulo, apresentamos resultados numéricos que comparam essas duas técnicas em problemas com campo de velocidade prescrito.

Depois, no Capítulo 4, é feita a derivação das equações de Navier-Stokes, que modelam o comportamento dos fluidos. Elas são apresentadas em sua forma mais geral para escoamentos multifásicos, e depois particularizadas para fluidos Newtonianos e incompressíveis. Ainda nesse mesmo Capítulo, os métodos utilizados para a resolução numérica das equações de Navier-Stokes e para o cálculo da tensão superficial são mostrados com detalhes. Foi utilizado um método implícito para a integração temporal, e um método de projeção para o cálculo da velocidade e da pressão. A tensão superficial foi calculada através dos métodos *Sharp Surface tension Force* (SSF [12]) e *Continuous Surface tension Force* (CSF [7]).

Por fim, são exibidos testes numéricos para a verificação dos métodos estudados aplicados em escoamentos bifásicos. São comparadas as duas técnicas de reinicialização estudadas, os dois métodos para o cálculo da força devido à tensão superficial e três diferentes discretizações para o cálculo da curvatura da interface.

Capítulo

2

Conceitos Básicos em Métodos Level-Set

Nesse Capítulo, faremos um estudo teórico sobre métodos level-set. Inicialmente, é introduzido o conceito de representação implícita de interfaces e o de função distância. Depois, o método WENO de quinta ordem é apresentado para a aproximação das derivadas espaciais na equação do transporte da interface, bem como um método Runge-Kutta de terceira ordem para a integração temporal. Nesse Capítulo, e no Capítulo seguinte, Utilizaremos a notação $|\mathbf{x}|$ para representar o comprimento do vetor $\mathbf{x}$ (medido através da norma euclidiana).

2.1 Representação de Interfaces

Interfaces podem ser representadas de duas maneiras distintas: de forma explícita ou de forma implícita. Nesta seção é feita uma breve introdução sobre cada uma das duas abordagens, apresentando suas vantagens e desvantagens. Uma descrição mais completa sobre o assunto pode ser encontrada em [26], que é o texto base utilizado nesta seção.

2.1.1 Uma dimensão

Considere primeiramente que nosso domínio de interesse seja a reta real, e que desejamos dividi-lo em duas regiões distintas:

$$\Omega_1 = (-\infty, -1) \cup (1, \infty)$$
$$\Omega_2 = (-1, 1).$$

Denominemos Ω_2 a *região interna* do domínio, e Ω_1 a *região externa*. A interface entre as regiões interna e externa consiste dos pontos

$$\partial\Omega = \{-1, 1\}.$$

Essa forma de representar a interface, explicitando os pontos que pertencem a ela, é denominada *representação explícita*. Ao invés disso, esses mesmos dois pontos pertencentes a $\partial\Omega$ podem ser representados através do isocontorno nulo da função $\phi(x) = x^2 - 1$, como a seguir:

$$\partial\Omega = \{x \in \mathbb{R}; x^2 - 1 = 0\}.$$

A representação de uma interface através de algum isocontorno de uma função é denominada *representação implícita*.

Com base nesse exemplo, podemos enumerar algumas propriedades importantes das representações explícita e implícita. Primeiramente, se o domínio estudado possui dimensão n, a interface possui dimensão $n-1$. Isso pode ser visto facilmente no exemplo acima, em que o domínio é a reta real, que possui dimensão 1, e a interface é formada por apenas dois pontos, cada um deles com dimensão 0. Outra propriedade importante é que, se o domínio de interesse possui dimensão n, qualquer função implícita que seja utilizada para representar uma interface nesse domínio deve estar definida num espaço de dimensão n, enquanto que a representação explícita lida com problemas de dimensão $n-1$.

Por fim, vale dizer que a interface pode ser determinada implicitamente por qualquer isocontorno desejado, e não necessariamente pelo isocontorno nulo. Podemos representar a mesma interface $\partial\Omega$ acima pelo isocontorno 1 da função x^2. Contudo, é mais usual utilizar o isocontorno nulo, e ele será utilizado ao longo desse deste trabalho.

2.1.2 Duas Dimensões

Considere agora o caso bidimensional, em que nosso domínio de interesse é o $\mathbb{R}^2$. Nesse caso, a interface será uma (ou mais) curva(s) fechada(s) que separará(ão) $\mathbb{R}^2$ em duas ou mais regiões de áreas não nulas. Considere ainda que as regiões internas e externas do domínio sejam dadas por

$$\Omega_1 = \{\boldsymbol{x} \in \mathbb{R}^2; |\boldsymbol{x}| > 1\}$$
$$\Omega_2 = \{\boldsymbol{x} \in \mathbb{R}^2; |\boldsymbol{x}| < 1\}.$$

Assim, a interface é o círculo de raio 1 e centro na origem, que pode ser definido de maneira explícita por

$$\partial\Omega = \{\boldsymbol{x} \in \mathbb{R}^2; |\boldsymbol{x}| = 1\}. \tag{2.1}$$

Note que, num caso mais geral em que a interface seja uma curva mais complexa, não é possível escrever $\partial\Omega$ numa forma analítica, e precisamos armazenar todos os pontos da interface diretamente. Entretanto, como ela é uma curva que possui infinitos pontos, eles não podem ser todos armazenados num computador, de modo que é preciso utilizar uma discretização.

Primeiramente, devemos parametrizar a curva como uma função vetorial $\boldsymbol{\phi}(s)$, num intervalo fechado $I = [s_0, s_f]$, onde $\boldsymbol{\phi}(s_i)$ $(0 \leq i \leq f)$ correspondem aos pontos da curva que representa a interface. Observe que, como a curva é fechada, $\boldsymbol{\phi}(s_0) = \boldsymbol{\phi}(s_f)$. A seguir, discretizamos I em um conjunto finito de pontos $s_0 < s_1 < \ldots < s_f$, com os subintervalos $[s_i, s_{i+1}]$ não necessariamente de mesmo comprimento. Então, para cada ponto s_i, armazenamos o ponto da curva representado por $\boldsymbol{\phi}(s_i)$, para obter uma representação explícita discretizada da interface.

A representação implícita da interface descrita em (2.1) pode ser feita através do isocontorno nulo da função $\phi(\boldsymbol{x}) = |\boldsymbol{x}| - 1$. Note que, se $\phi(\boldsymbol{x}) = 0$, $\boldsymbol{x}$ pertence à interface $\partial\Omega$ descrita de forma explícita em (2.1). Contudo, assim como no caso explícito, pode ser bastante difícil obter uma função analítica ϕ que defina implicitamente uma curva mais complexa.

Para isso, a solução é semelhante à do caso explícito: devemos discretizar o domínio de ϕ, que é, no caso, o conjunto $\mathbb{R}^2$. Contudo, como $\mathbb{R}^2$ é ilimitado, é necessário restringir a discretização a um subdomínio $D \subset \mathbb{R}^2$. A seguir, escolhemos pontos $\boldsymbol{x}_i = (x_i, y_i) \in D$, e armazenamos os valores de $\phi(\boldsymbol{x}_i)$. Note que, apesar de ϕ não ser conhecida analiticamente, seus valores nos pontos discretizados devem ser conhecidos.

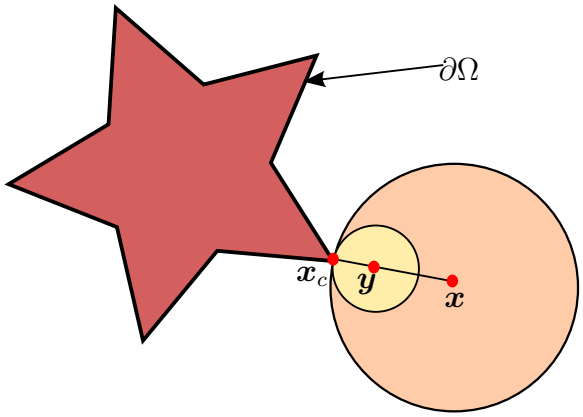


Figura 2.1: O contorno da estrela vermelha representa a interface $\partial\Omega$, o círculo alaranjado possui raio $|\boldsymbol{x}-\boldsymbol{x}_c|$ e o círculo em amarelo possui raio $|\boldsymbol{y}-\boldsymbol{x}_c|$. Se $\boldsymbol{x}_c$ é o ponto da interface mais próximo de $\boldsymbol{x}$, ele também é o ponto da interface mais próximo de qualquer ponto que pertença ao segmento de reta que vai de $\boldsymbol{x}$ a $\boldsymbol{x}_c$.

Nem a abordagem explícita nem a implícita nos dizem exatamente onde a interface está localizada; ao invés disso, elas contêm apenas as informações de alguns pontos dela. Para construir a interface completa, é necessário fazer aproximações numéricas, como interpolações. Em [26], é dito que o procedimento padrão para essa reconstrução com a interface representada de forma explícita é o uso de splines. Já na representação implícita, a situação é ainda mais complicada, porque, a menos que a discretização seja escolhida de maneira extremamente precisa, pode não haver nenhum ponto $\boldsymbol{x} \in D$ tal que $\phi(\boldsymbol{x}) = 0$ para que a interpolação possa ser feita. Nesse caso, a interpolação deve ser adaptada para que possa ser feita apenas com os valores conhecidos de ϕ.

2.2 Função Distância

Dado um domínio de interesse Ω separado em duas regiões por uma interface $\partial\Omega$, uma função distância $d : \Omega \to \mathbb{R}$ é definida como

$$d(\boldsymbol{x}) = \min_{\boldsymbol{x}_c \in \partial\Omega} \{|\boldsymbol{x} - \boldsymbol{x}_c|\}. \tag{2.2}$$

Observe que d se anula para todos os pontos da interface. Além disso, note que, se $\boldsymbol{x}_c$ é o ponto da interface mais próximo de um certo ponto $\boldsymbol{x}$, então $\boldsymbol{x}_c$ também é o ponto da interface mais próximo de todos os pontos do segmento de reta que conecta $\boldsymbol{x}$ a $\boldsymbol{x}_c$. Isso pode ser visto através da Figura 2.1. Como $\boldsymbol{x}_c$ é o ponto da interface mais próximo a $\boldsymbol{x}$, podemos desenhar um círculo centrado em $\boldsymbol{x}$ cuja única intersecção com a interface seja exatamente em $\boldsymbol{x}_c$. Seja agora $\boldsymbol{y}$ um ponto do segmento de reta que passe por $\boldsymbol{x}$ e $\boldsymbol{x}_c$, e desenhemos um novo círculo centrado em $\boldsymbol{y}$ que intersecte a interface em $\boldsymbol{x}_c$. Esse círculo menor é interno ao círculo maior, de modo que, se houver um segundo ponto da interface que pertença ao círculo menor, ele também pertencerá ao círculo maior, o que não é possível se $\boldsymbol{x}_c$ for o ponto da interface mais próximo de $\boldsymbol{x}$. Portanto, não pode haver ponto da interface mais próximo de $\boldsymbol{y}$ do que $\boldsymbol{x}_c$.

Uma propriedade útil e simples de ser verificada é que, para toda função distância d, vale

$$|\nabla d| = 1. \tag{2.3}$$

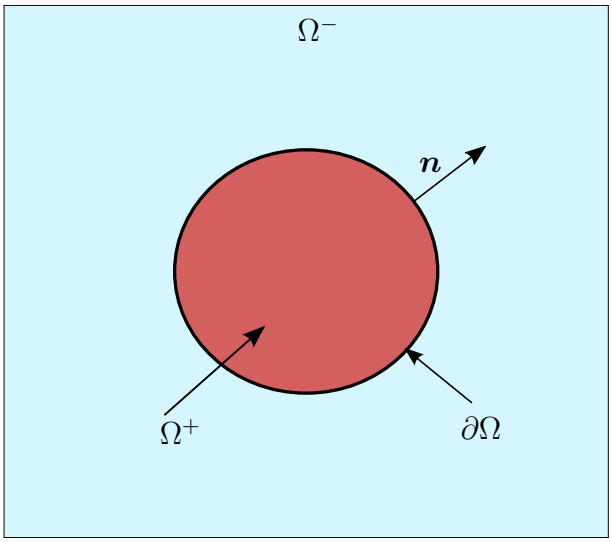


Figura 2.2: Duas regiões de um domínio separadas por uma interface. A região interna (Ω^+) está representada em vermelho, a região externa (Ω^-) em azul e a linha preta que as separa ($\partial\Omega$) é a interface. $\boldsymbol{n}$ é o vetor normal á interface.

Essa propriedade pode ser verificada intuitivamente, notando que, num ponto $\boldsymbol{x}$ cuja distância à interface seja o dobro da distância a partir de outro ponto $\boldsymbol{y}$, o valor $d(\boldsymbol{x})$ também será o dobro do valor $d(\boldsymbol{y})$. Infelizmente essa propriedade só é válida caso haja apenas um ponto $\boldsymbol{x}_c$ da interface que seja o mais próximo a $\boldsymbol{x}$. Caso hajam dois pontos equidistantes, a equação (2.3) não é válida. Além disso, quando o gradiente for calculado numericamente, ele não será, em geral, exatamente igual a 1, devido a erros numéricos.

Uma função distância sinalizada ϕ pode ser definida através de uma função distância d como

$$\phi(\boldsymbol{x}) = \begin{cases} d(\boldsymbol{x}), & \boldsymbol{x} \in \Omega^+ \cup \partial\Omega \\ -d(\boldsymbol{x}), & \boldsymbol{x} \in \Omega^- \end{cases} . \tag{2.4}$$

Note que a escolha entre a interface ser incluída em Ω^+ ou em Ω^- é irrelevante, pois $\phi(\boldsymbol{x}) = 0$ para todo $\boldsymbol{x}$ da interface, independentemente da escolha feita. Com base nessa definição, podemos definir as regiões interna e externa de Ω como a seguir:

$$\begin{aligned} \Omega^+ &= \{\boldsymbol{x} \in \Omega \mid \phi(\boldsymbol{x}) > 0\} \\ \Omega^- &= \{\boldsymbol{x} \in \Omega \mid \phi(\boldsymbol{x}) < 0\} \\ \partial\Omega &= \{\boldsymbol{x} \in \Omega \mid \phi(\boldsymbol{x}) = 0\}, \end{aligned}$$

sendo Ω^+ a região interior do domínio, Ω^- a exterior, e $\partial\Omega$ a interface, como mostra a Figura 2.2.

Uma função distância sinalizada ϕ também goza da propriedade (2.3), com as mesmas restrições quando houverem dois pontos da interface equidistantes a um certo ponto do domínio. Como seu gradiente é limitado e consideravelmente baixo, ela torna-se uma boa escolha para ser utilizada na representação implícita de interfaces [26]. Sua vantagem em relação às funções distância tradicional é que ela nos permite identificar facilmente se um ponto está na região interior ou na região exterior do domínio, bastando para isso avaliar o sinal de ϕ no ponto em questão.

2.3 Cálculo Numérico de Propriedades Geométricas da Interface

O objetivo dessa seção é apresentar detalhes do cálculo numérico do vetor normal e da curvatura da interface. Antes disso, porém, descrevemos a malha computacional utilizada para a discretização do domínio. Apenas uma discretização para o cálculo da curvatura é apresentado no texto principal do trabalho, mas outras duas discretizações alternativas são apresentadas no Apêndice A.

2.3.1 Malha Utilizada

Todas as aproximações para as derivadas presentes nesse trabalho foram feitas por métodos de diferenças finitas. A malha utilizada é apresentada na Figura 2.3. O domínio é discretizado em $N \times M$ células quadradas (ou seja, na figura citada, $\Delta x = \Delta y$), sendo N a quantidade de células na direção x, e M a quantidade de células na direção y. Assim, as células são identificadas por duas coordenadas inteiras: uma que vai de 1 a N na direção x, e outra que vai de 1 a M na direção y.

A origem do sistema de coordenadas está no vértice inferior esquerdo da célula de coordenadas $(1, 1)$, de forma que as coordenadas cartesianas do ponto central da célula (i, j) é $((i - 0.5)\Delta x, (j - 0.5)\Delta y)$. A função ϕ será armazenada no centro da célula, como descrito na Figura 2.3b. Nessa mesma figura, apresentamos a localização dos campos de pressão (a ser introduzida no Capítulo 4) e velocidade.

No centro de cada célula são armazenadas as funções ϕ e p (p é a pressão, que será introduzida no Capítulo 4). As velocidades são armazenadas nas faces de cada célula, conforme mostrado na Figura 2.3b.

2.3.2 Vetor Normal

O vetor normal pode ser obtido de maneira bastante simples. Utilizando o fato de que o gradiente de uma função é ortogonal à tangente num ponto qualquer das curvas de nível dessa função, $\nabla\phi$ é perpendicular à interface nos pontos em que $\phi = 0$. Logo, a normal unitária $\boldsymbol{n} = (n_x, n_y)$ pode ser obtida como a seguir:

$$\boldsymbol{n} = -\frac{\nabla\phi}{|\nabla\phi|}. \tag{2.5}$$

Na equação acima, o sinal de negativo implica que o vetor normal aponta "para fora" da interface, conforme mostra a Figura 2.2. De fato, o gradiente de uma função é um vetor que aponta para a direção de maior crescimento dessa função. Como ϕ é positiva na região interna do domínio, e negativa na região externa, $\nabla\phi$ aponta sempre para a região interna. Daí, o sinal de negativo força o vetor normal a apontar para a região externa do domínio, visto que ϕ é positiva na região interna e negativa na externa.

A equação acima só pode ser utilizada para o cálculo do vetor normal à interface nos pontos em que $\phi = 0$. Nos demais pontos, ela apenas permite calcular um vetor normal a uma curva de nível de ϕ. Em geral, o vetor normal é utilizado apenas para o cálculo da curvatura, de modo que a discretização dessa equação depende do método utilizado para o cálculo da curvatura. Um método é apresentado na seção a seguir, e dois outros podem ser encontrados no Apêndice A.

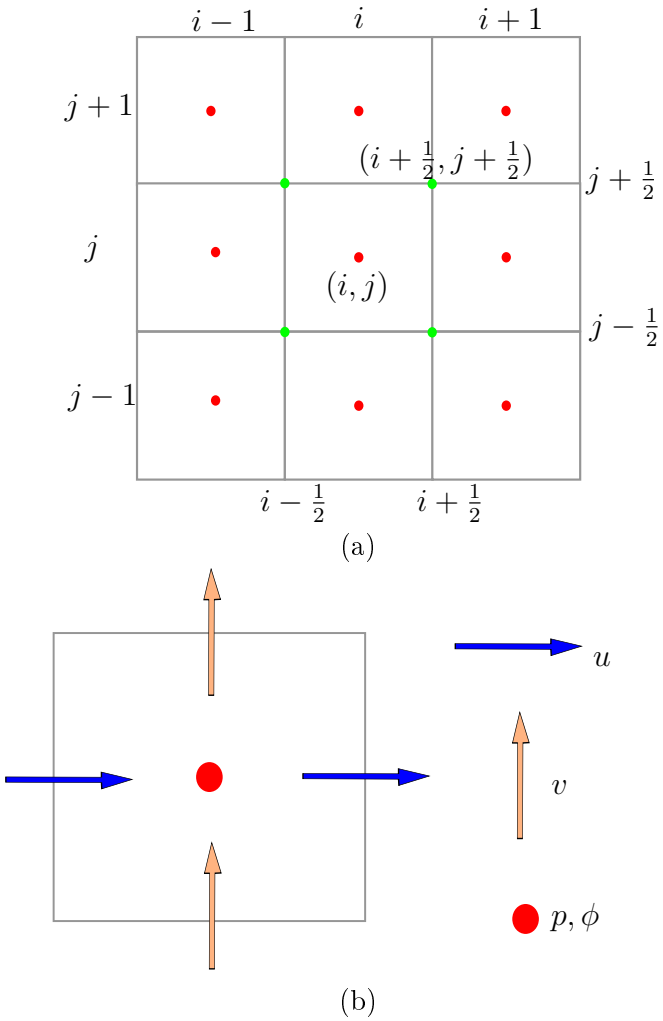


Figura 2.3: Malha utilizada nesse trabalho. **a)** Identificação dos nós da malha. O nó em vermelho da célula central é identificado pela posição (i, j), enquanto que o nó da quina superior direita da mesma célula possui coordenadas $(i + \frac{1}{2}, j + \frac{1}{2})$. **b)** no centro da célula são armazenadas a função ϕ e a pressão p (que será utilizada no Capítulo 4); a componente u do vetor velocidade $\boldsymbol{u} = (u, v)$ é armazenada nas faces verticais da célula, e a componente v é armazenada nas faces horizontais.

2.3.3 Curvatura

Pode ser mostrado [42] que a curvatura da interface é dada por

$$\kappa = -\nabla \cdot \boldsymbol{n}, \tag{2.6}$$

sendo $\boldsymbol{n}$ a normal unitária que aponta para fora da interface. Há diversas discretizações possíveis para essa expressão. Nesse trabalho, realizamos testes com três discretizações diferentes, que serão chamadas de LS1, LS2 e LS3, respectivamente. A seguir apresentamos o método LS1, que foi o que mostrou os melhores resultados. Os métodos LS2 e LS3 são apresentados no Apêndice A.

A discretização LS1 foi proposta por [33]. Utilizando as expressões (2.5) e (2.6), a curvatura pode ser calculada por

$$\kappa = \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right). \tag{2.7}$$

Desenvolvendo a equação acima em duas dimensões,

$$\begin{aligned}\kappa &= \frac{\partial}{\partial x}\left(\frac{\partial\phi}{\partial x}\frac{1}{|\nabla\phi|}\right) + \frac{\partial}{\partial y}\left(\frac{\partial\phi}{\partial y}\frac{1}{|\nabla\phi|}\right)\\ &= \frac{\partial^2\phi}{\partial x^2}\frac{1}{|\nabla\phi|} + \frac{\partial\phi}{\partial x}\frac{\partial|\nabla\phi|^{-1}}{\partial x} + \frac{\partial^2\phi}{\partial y^2}\frac{1}{|\nabla\phi|} + \frac{\partial\phi}{\partial y}\frac{\partial|\nabla\phi|^{-1}}{\partial y}\\ &= \frac{\nabla^2\phi}{|\nabla\phi|} - \boldsymbol{N}(\phi),\end{aligned} \tag{2.8}$$

com

$$\begin{aligned}\boldsymbol{N}(\phi) &= -\frac{\partial\phi}{\partial x}\frac{\partial|\nabla\phi|^{-1}}{\partial x} - \frac{\partial\phi}{\partial y}\frac{\partial|\nabla\phi|^{-1}}{\partial y}\\ &= -\nabla\phi\cdot\nabla(|\nabla\phi|^{-1})\\ &= \nabla\phi\cdot\frac{\nabla|\nabla\phi|}{|\nabla\phi|^2}\\ &= \frac{\nabla\phi\cdot\nabla|\nabla\phi|}{|\nabla\phi|^2}\\ &= \left(\frac{\phi_x}{|\nabla\phi|^2}, \frac{\phi_y}{|\nabla\phi|^2}\right)\cdot\left(|\nabla\phi|_x, |\nabla\phi|_y\right).\end{aligned}$$

Observe que

$$|\nabla\phi| = \left(\phi_x^2 + \phi_y^2\right)^{\frac{1}{2}} \tag{2.9}$$

$$|\nabla\phi|_x = \frac{\phi_x\phi_{xx} + \phi_y\phi_{xy}}{\left(\phi_x^2 + \phi_y^2\right)^{\frac{1}{2}}} \tag{2.10}$$

$$|\nabla\phi|_y = \frac{\phi_y\phi_{yy} + \phi_x\phi_{xy}}{\left(\phi_x^2 + \phi_y^2\right)^{\frac{1}{2}}}. \tag{2.11}$$

Daí, podemos escrever

$$\begin{aligned}\boldsymbol{N}(\phi) &= \left(\frac{\phi_x}{|\nabla\phi|^2}, \frac{\phi_y}{|\nabla\phi|^2}\right)\cdot\left(\frac{\phi_x\phi_{xx} + \phi_y\phi_{xy}}{|\nabla\phi|}, \frac{\phi_y\phi_{yy} + \phi_x\phi_{xy}}{|\nabla\phi|}\right)\\ &= \frac{\phi_x^2\phi_{xx} + 2\phi_x\phi_y\phi_{xy} + \phi_y^2\phi_{yy}}{|\nabla\phi|^3}.\end{aligned}$$

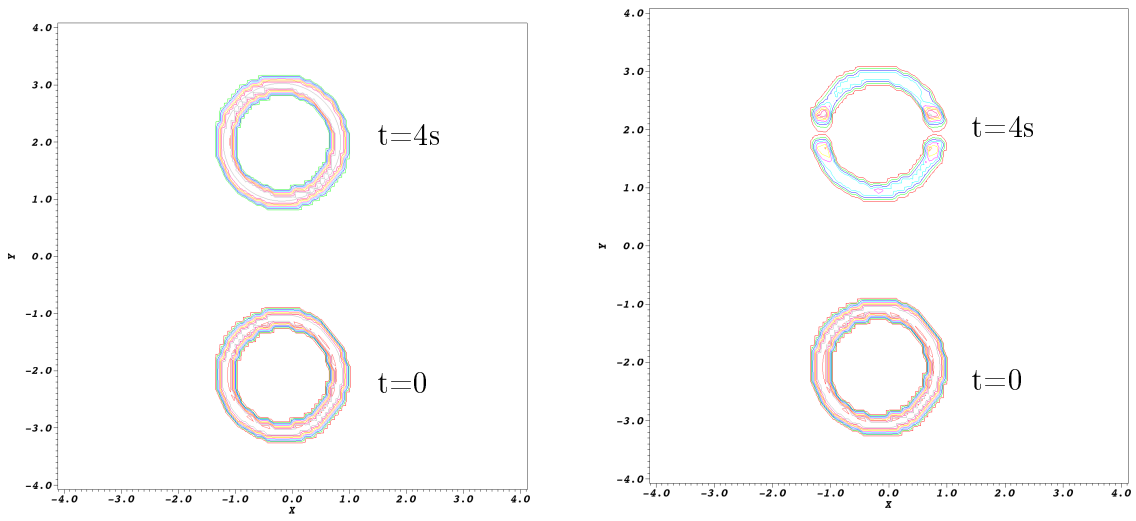
Portanto, (2.8) pode ser escrita como

$$\kappa = \frac{\phi_{xx} + \phi_{yy}}{|\nabla\phi|} - \frac{\phi_x^2\phi_{xx} + 2\phi_x\phi_y\phi_{xy} + \phi_y^2\phi_{yy}}{|\nabla\phi|^3} = \frac{\phi_x^2\phi_{yy} - 2\phi_x\phi_y\phi_{xy} + \phi_y^2\phi_{xx}}{|\nabla\phi|^3}. \tag{2.12}$$

No cálculo do módulo do gradiente, a expressão $\sqrt{\phi_x^2 + \phi_y^2 + \epsilon}$ foi utilizada, com $\epsilon = 10^{-12}$. Todas as derivadas, de primeira e segunda ordem, foram aproximadas por diferenças centradas.

2.4 Advecção da Interface

Agora estudaremos o problema de movimentar a interface definida implicitamente por $\phi(\boldsymbol{x}) = 0$. Problemas de advecção de interface surgem, por exemplo, no estudo de escoamentos multifásicos, nos quais, através das equações de Navier-Stokes, calcula-se o



(a) (b)

Figura 2.4: Curvas de nível da curvatura de ϕ quando um círculo é transladado verticalmente para cima. Em **a)** a translação é feita com o método WENO de quinta ordem. Em **b)**, é feita com um esquema CUBISTA [2].

campo de velocidade que dita o movimento de cada partícula do escoamento (veja [39]). A seguir, a função level-set, que é utilizada para determinar qual o fluido presente em cada ponto do domínio, é transportada através deste campo de velocidade já calculado.

A equação básica para descrever o transporte da interface é

$$\phi_t + \boldsymbol{u} \cdot \nabla\phi = 0. \tag{2.13}$$

Essa equação corresponde à movimentação da interface representada por ϕ quando submetida a um campo de velocidade $\boldsymbol{u}$. Por mais inocente que possa parecer, esse problema apresenta grandes complicações numéricas, podendo levar a resultados muito ruins se resolvido numericamente com métodos de baixa precisão (ver [31]). Veja, por exemplo, a Figura 2.4. Nela, um círculo é transladado verticalmente para cima, com a advecção sendo comparada entre os métodos WENO de quinta ordem e o método upwind de terceira ordem CUBISTA [2]. Como trata-se de um círculo, as curvas de nível da curvatura de ϕ devem ser circulares e concêntricas. Entretando, quando a advecção foi feita utilizando o método CUBISTA, o resultado ficou muito distante disso. Nesse trabalho, utilizamos a combinação do método WENO de quinta ordem para discretização espacial descrito em [16] com um Runge-Kutta de terceira ordem para discretização temporal, conforme sugerido em [20], para o transporte da interface.

2.4.1 O Método WENO de Quinta Ordem

Métodos WENO (*Weighted Essentially Non-Oscilatory*) consistem, como o nome sugere, de médias ponderadas de aproximações ENO (*Essentially Non-Oscilatory*) para o cálculo das derivadas espaciais. Através da média ponderada (com pesos apropriados) de aproximações ENO, é possível obter uma aproximação WENO de ordem superior à de cada uma das aproximações ENO. Esta seção será desenvolvida como foi feito em [16].

Utilizando as moléculas computacionais combinadas de três esquemas ENO de terceira ordem (ver Figura 2.5), é possível, nas regiões suaves de ϕ, obter um método que aproxime as derivada espaciais com precisão de quinta ordem. Para simplificar a apresentação do

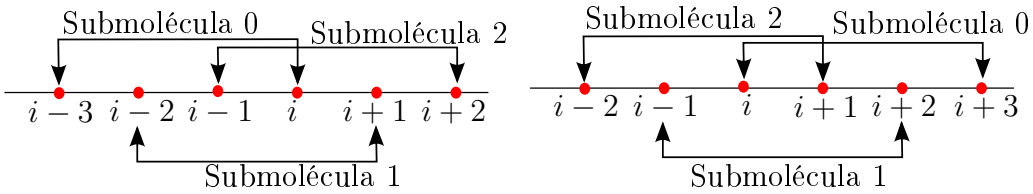


Figura 2.5: **a)** Molécula computacional à esquerda, utilizado para aproximar $\phi^-_{x,i}$; **b)** Molécula computacional à direita, utilizado para aproximar $\phi^+_{x,i}$.

método, consideraremos apenas o caso unidimensional

$$\phi_t + u\phi_x = 0. \tag{2.14}$$

O caso bidimensional pode ser estendido facilmente do unidimensional, pois as derivadas espaciais são calculadas independentemente umas das outras.

Seja ϕ uma função Lipschitz contínua em $D \subset \mathbb{R}^1$, com derivadas suaves por partes. Além disso, assumamos que as descontinuidades de ϕ, se existirem, são isoladas. Discretizemos D através de pontos x_k com espaçamento Δx. Introduzamos a notação

$$\phi_k = \phi(x_k), \quad \Delta^+\phi_k = \phi_{k+1} - \phi_k, \quad \Delta^-\phi_k = \phi_k - \phi_{k-1}. \tag{2.15}$$

Com base nessa notação, podemos obter

$$\Delta^-\Delta^+\phi_i = \Delta^-\left(\phi_{i+1} + \phi_i\right) = \phi_{i+1} - \phi_i - (\phi_i - \phi_{i-1}) = \phi_{i+1} - 2\phi_i + \phi_{i-1} \tag{2.16}$$

$$\Delta^-\Delta^-\Delta^+\phi_i = \Delta^-(\phi_{i+1} - 2\phi_i + \phi_{i-1}) = \phi_{i+1} - 3\phi_i + 3\phi_{i-1} - \phi_{i-2} \tag{2.17}$$

$$\Delta^+\Delta^-\Delta^+\phi_i = \Delta^+(\phi_{i+1} - 2\phi_i + \phi_{i-1}) = \phi_{i+2} - 3\phi_{i+1} + 3\phi_i - \phi_{i-1}. \tag{2.18}$$

Para aproximar $\frac{\partial\phi}{\partial x}(x_i) = \phi_{x,i}$ utilizando uma molécula computacional à esquerda $\{x_k, k = i-3, \ldots, i+2\}$ como mostrado na Figura 2.5, o esquema ENO de terceira ordem analisa as seguintes aproximações

$$\begin{aligned}
\phi^{-,0}_{x,i} &= \frac{1}{3}\frac{\Delta^+\phi_{i-3}}{\Delta x} - \frac{7}{6}\frac{\Delta^+\phi_{i-2}}{\Delta x} + \frac{11}{6}\frac{\Delta^+\phi_{i-1}}{\Delta x} \\
\phi^{-,1}_{x,i} &= -\frac{1}{6}\frac{\Delta^+\phi_{i-2}}{\Delta x} + \frac{5}{6}\frac{\Delta^+\phi_{i-1}}{\Delta x} + \frac{1}{3}\frac{\Delta^+\phi_i}{\Delta x} \\
\phi^{-,2}_{x,i} &= \frac{1}{3}\frac{\Delta^+\phi_{i-1}}{\Delta x} + \frac{5}{6}\frac{\Delta^+\phi_i}{\Delta x} - \frac{1}{6}\frac{\Delta^+\phi_{i+1}}{\Delta x},
\end{aligned} \tag{2.19}$$

sendo $\phi^{-,s}_{x,i}, s \in \{0,1,2\}$, aproximações de terceira ordem para $\phi_x(x_i)$. A escolha dentre uma das três aproximações acima é feita com base na suavidade de ϕ nos pontos da submolécula computacional utilizada por elas. A aproximação ENO de terceira ordem é dada por

$$\phi^-_{x,i} = \begin{cases}
\phi^{-,0}_{x,i}, & \text{se } |\Delta^-\Delta^+\phi_{i-1}| < |\Delta^-\Delta^+\phi_i| \text{ e } |\Delta^-\Delta^-\Delta^+\phi_{i-1}| < |\Delta^+\Delta^-\Delta^+\phi_{i-1}| \\
\phi^{-,1}_{x,i}, & \text{se } |\Delta^-\Delta^+\phi_{i-1}| > |\Delta^-\Delta^+\phi_i| \text{ e } |\Delta^-\Delta^-\Delta^+\phi_{i-1}| > |\Delta^+\Delta^-\Delta^+\phi_{i-1}| \\
\phi^{-,2}_{x,i}, & \text{nos demais casos.}
\end{cases} \tag{2.20}$$

Por exemplo, se tivermos um vetor ϕ de elementos $\phi_i (0 \leq i \leq 4)$ dado por $\phi = (0, 0.1, 0.3, 0.8, 0.5)$, e desejamos calcular $\phi^-_{x,3}$, temos

$$\begin{aligned}
|\Delta^-\Delta^+\phi_2| &= |\phi_3 - 2\phi_2 + \phi_1| = |0.8 - 2\times 0.3 + 0.1| = 0.3 \\
|\Delta^-\Delta^+\phi_3| &= |\phi_4 - 2\phi_3 + \phi_2| = |0.5 - 2\times 0.8 + 0.3| = 0.8 \\
|\Delta^-\Delta^-\Delta^+\phi_2| &= |\phi_3 - 3\phi_2 + 3\phi_1 - \phi_0| = |0.8 - 3\times 0.3 + 3\times 0.1 - 0| = 0.2 \\
|\Delta^+\Delta^-\Delta^+\phi_2| &= |\phi_4 - 3\phi_3 + 3\phi_2 - \phi_1| = |0.5 - 3\times 0.8 + 3\times 0.3 - 0.1| = 1.1.
\end{aligned}$$

Comparando os resultados acima com a equação (2.20), vemos que a aproximação a ser utilizada é $\phi_{x,3}^{-,0}$. Assim, $\phi_{x,3}^{-}$ é aproximada por

$$\begin{aligned}\phi_{x,3}^{-} \approx \phi_{x,3}^{-,0} &= \frac{1}{3}\frac{\Delta^+\phi_0}{\Delta x} - \frac{7}{6}\frac{\Delta^+\phi_1}{\Delta x} + \frac{11}{6}\frac{\Delta^+\phi_2}{\Delta x} \\ &= \frac{1}{3}\frac{\phi_1-\phi_0}{\Delta x} - \frac{7}{6}\frac{\phi_2-\phi_1}{\Delta x} + \frac{11}{6}\frac{\phi_3-\phi_2}{\Delta x} \\ &= \frac{1}{3}\frac{0.1}{\Delta x} - \frac{7}{6}\frac{0.2}{\Delta x} + \frac{11}{6}\frac{0.5}{\Delta x} \\ &= \frac{43}{60\Delta x}.\end{aligned}$$

A aproximação WENO, contudo, é feita através de uma média ponderada de $\phi_{x,i}^{-,s}, s \in \{0,1,2\}$:

$$\phi_{x,i}^{-} = \omega_0\phi_{x,i}^{-,0} + \omega_1\phi_{x,i}^{-,1} + \omega_2\phi_{x,i}^{-,2}. \tag{2.21}$$

Escolhendo $\omega_0 = C_0 = 0.1$, $\omega_1 = C_1 = 0.6$ e $\omega_2 = C_2 = 0.3$, obtemos

$$\phi_{x,i}^{-} = \frac{1}{30}\frac{\Delta^+\phi_{i-3}}{\Delta x} - \frac{13}{60}\frac{\Delta^+\phi_{i-2}}{\Delta x} + \frac{47}{60}\frac{\Delta^+\phi_{i-1}}{\Delta x} + \frac{9}{20}\frac{\Delta^+\phi_i}{\Delta x} - \frac{1}{20}\frac{\Delta^+\phi_{i+1}}{\Delta x}, \tag{2.22}$$

que é uma aproximação de quinta ordem, e é conhecida como a aproximação com menor erro de truncamento com uma molécula computacional de seis pontos. Caso ϕ não seja suave em algum ponto dessa molécula computacional, as sub-moléculas computacionais que contêm esse ponto não devem ser utilizadas na aproximação da derivada. Isso é feito zerando o(s) ω_s correspondente(s) a essa(s) sub-molécula(s) computacional(is), desde que haja ao menos uma sub-molécula dentre as três mostradas na Figura 2.5 que não contenha o ponto em que ϕ não é suave. Caso hajam duas sub-moléculas que em que ϕ é suave, escolhe-se um dos ω_s correspondentes a essas sub-moléculas como 1 e os demais como 0. Desse modo, a aproximação é feita com um esquema ENO de terceira ordem.

A seguir mostraremos como calcular os ω_s analisando a suavidade de ϕ numa molécula computacional de seis pontos. Se ϕ for suave em toda a molécula de seis pontos, o método apresentado produzirá $\omega_s = C_s + O(\Delta x^2)$. Caso contrário, teremos $\omega_s \in \{0,1\}$. Substituindo $\omega_1 = 1 - \omega_0 - \omega_2$ em (2.21), obtemos

$$\phi_{x,i}^{-} = \frac{1}{2}\left(\phi_{x,i}^{-1} + \phi_{x,i}^{-2}\right) + \omega_0\left(\phi_{x,i}^{-0} - \phi_{x,i}^{-1}\right) + \left(\omega_2 - \frac{1}{2}\right)\left(\phi_{x,i}^{-2} - \phi_{x,i}^{-1}\right). \tag{2.23}$$

Agora, substituindo cada $\phi_{x,i}^{-,s}, s \in \{0,1,2\}$ de (2.19) em (2.23), determinamos:

$$\begin{aligned}\phi_{x,i}^{-} &= \frac{1}{12\Delta x}\left(-\Delta^+\phi_{i-2} + 7\Delta^+\phi_{i-1} + 7\Delta^+\phi_i - \Delta^+\phi_{i+1}\right) \\ &\quad -\Phi^{\text{WENO}}\left(\frac{\Delta^-\Delta^+\phi_{i-2}}{\Delta x}, \frac{\Delta^-\Delta^+\phi_{i-1}}{\Delta x}, \frac{\Delta^-\Delta^+\phi_i}{\Delta x}, \frac{\Delta^-\Delta^+\phi_{i+1}}{\Delta x}\right),\end{aligned} \tag{2.24}$$

onde

$$\Phi^{\text{WENO}}(a,b,c,d) = \frac{1}{3}\omega_0(a-2b+c) + \frac{1}{6}(\omega_2 - 0.5)(b-2c+d). \tag{2.25}$$

Os pesos ω_0 e ω_2 são definidos como

$$\omega_0 = \frac{\alpha_0}{\alpha_0+\alpha_1+\alpha_2}, \qquad \omega_2 = \frac{\alpha_2}{\alpha_0+\alpha_1+\alpha_2}$$

$$\alpha_0 = \frac{1}{(\epsilon+IS_0)^2}, \qquad \alpha_1 = \frac{6}{(\epsilon+IS_1)^2}, \qquad \alpha_2 = \frac{3}{(\epsilon+IS_2)^2}$$

$$IS_0 = 13(a-b)^2 + 3(a-3b)^2$$

$$IS_1 = 13(b-c)^2 + 3(b+c)^2$$

$$IS_2 = 13(c-d)^2 + 3(3c-d)^2,$$

onde ϵ é usado para evitar que o denominador se torne nulo, e foi utilizado em [16] como 10^{-6}. Esse procedimento resulta em ω_s que satisfazem a condição de que a soma dos pesos seja 1, e garante que, nas regiões suaves de ϕ, (2.24) se reduza a (2.22). Uma aproximação semelhante para $\phi_x(x_i)$ pode ser obtida utilizando-se uma molécula computacional à direita:

$$\begin{aligned}\phi^+_{x,i} &= \frac{1}{12\Delta x}\left(-\Delta^+\phi_{i-2} + 7\Delta^+\phi_{i-1} + 7\Delta^+\phi_i - \Delta^+\phi_{i+1}\right) \\ &+\Phi^{\text{WENO}}\left(\frac{\Delta^-\Delta^+\phi_{i+2}}{\Delta x}, \frac{\Delta^-\Delta^+\phi_{i+1}}{\Delta x}, \frac{\Delta^-\Delta^+\phi_i}{\Delta x}, \frac{\Delta^-\Delta^+\phi_{i-1}}{\Delta x}\right). \end{aligned} \tag{2.26}$$

A escolha de usar $\phi^+_{x,i}$ ou $\phi^-_{x,i}$ é feita de acordo com o sinal da velocidade u. Se $u > 0$, então informações sobre ϕ estão sendo propagadas da esquerda para a direita. Isso significa que as derivadas devem ser calculadas com diferenças atrasadas, utilizando a molécula computacional mais à esquerda, de modo que $\phi^-_{x,i}$ deve ser utilizado. Por outro lado, se $u < 0$, informações sobre ϕ estão sendo propagadas da direita para a esquerda, o que indica que as derivadas devem ser aproximadas com diferenças avançadas, indicando que $\phi^+_{x,i}$ seja utilizado.

2.4.2 Integração Temporal

Como mostrado em [26], experimentos práticos têm indicado que a equação (2.13) é muito sensível à precisão da solução espacial, de modo que a aproximação espacial de quinta ordem sugerida na seção anterior é desejável. Já a aproximação temporal não possui importância tão grande para o resultado final da advecção, de modo que até uma discretização explícita de primeira ordem pode ser utilizada sem comprometimento da solução. Contudo, há situações em que maior precisão temporal pode ser desejada. Em [32], foi proposto um esquema Runge-Kutta TVD (*total variation diminishing*) para aproximação temporal com o método das linhas.

O método das linhas assume que a discretização temporal e a espacial podem ser consideradas independentes, sem prejuízo à solução. Embora hajam vários métodos Runge-Kutta, apenas alguns deles são TVD. Mais formalmente, um método é TVD se

$$\sum_i |\phi^{n+1}_{i+1} - \phi^{n+1}_i| \leq \sum_i |\phi^n_{i+1} - \phi^n_i|, \tag{2.27}$$

onde n representa o índice de tempo em que a aproximação está sendo calculada, e $n+1 = t + \Delta t$, sendo Δt o passo de tempo utilizado na discretização temporal.

Infelizmente o método WENO utilizado não é TVD, mas experimentos numéricos têm mostrado que ele é, provavelmente, TVB(*Total Variation Bounded*), o que limita as oscilações introduzidas pelas aproximações temporais com diferença avançada. O esquema TVD Runge-Kutta proposto em [32] é apresentado a seguir.

Primeiramente, utiliza-se duas diferenças avançadas no tempo

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + \boldsymbol{u}^n \cdot \nabla\phi^n = 0 \tag{2.28}$$

$$\frac{\phi^{n+2} - \phi^{n+1}}{\Delta t} + \boldsymbol{u}^{n+1} \cdot \nabla\phi^{n+1} = 0. \tag{2.29}$$

A seguir, calcula-se uma aproximação para o tempo $m+0.5\Delta t$, representada por $\phi^{n+\frac{1}{2}}$:

$$\phi^{n+\frac{1}{2}} = \frac{3}{4}\phi^n + \frac{1}{4}\phi^{n+2}. \tag{2.30}$$

Agora, marchando $\phi^{n+\frac{1}{2}}$ um passo no tempo, obtemos

$$\frac{\phi^{n+\frac{3}{2}} - \phi^{n+\frac{1}{2}}}{\Delta t} + \boldsymbol{u}^{n+\frac{1}{2}} \cdot \nabla \phi^{n+\frac{1}{2}} = 0. \tag{2.31}$$

Por fim, calcula-se outra média para a solução final

$$\phi^{n+1} = \frac{1}{3}\phi^{n} + \frac{2}{3}\phi^{n+\frac{3}{2}}. \tag{2.32}$$

Em [26], é dito que métodos de Runge-Kutta de ordem maior que três acabam denegrindo a solução espacial, fazendo o método WENO da seção anterior ter apenas a precisão de um ENO de terceira ordem.

Capítulo 3

Reinicialização da Função Level-Set

Durante o processo de advecção, ϕ pode perder as propriedades de função distância. Isso faz com que, ao longo do tempo, a solução do problema de advecção seja comprometida. Para resolver esse problema, foram desenvolvidos alguns processos de reinicialização, que tentam, principalmente, restaurar a propriedade $|\nabla\phi| = 1$. Estudaremos duas reinicializações distintas: a primeira, proposta em [39], consiste em resolver até o estado estacionário uma EDP. A segunda foi proposta em [4], e utiliza informações geométricas para reinicializar a função level-set. As duas reinicializações precisam de detalhes adicionais para reduzir a perda de massa. Esses detalhes serão discutidos, e sumarizados na forma de algoritmo no Apêndice B.

3.1 Reinicialização por EDP

Nessa seção, estudaremos a reinicialização proposta por Sussman *et al.* [39]. Essa reinicialização consiste de resolver uma equação de advecção, equação essa que "constrói" uma função distância sinalizada a partir do contorno da interface para o restante do domínio. A cada passo de tempo dessa equação, uma vizinhança maior em volta da interface se torna uma função distância sinalizada. Contudo, erros numéricos fazem com que essa reinicialização altere a massa da interface. Por esse motivo, a cada passo de tempo dessa equação, um passo adicional deve ser realizado para melhorar a conservação de massa.

3.1.1 Advecção na Direção Normal à Interface

Durante a reinicialização por EDP, precisamos resolver uma equação de advecção em que as informações são propagadas a partir do contorno da interface para o restante do domínio na direção normal à interface. Por esse motivo, iniciamos os estudos da reinicialização por EDP com o problema de advecção sob um campo de velocidade normal à interface.

Consideremos um campo de velocidade

$$\boldsymbol{u} = u_n\boldsymbol{n} + u_t\boldsymbol{t}, \tag{3.1}$$

sendo $\boldsymbol{n}$ um vetor unitário normal à interface, $\boldsymbol{t}$ um vetor unitário tangencial a ela, u_n a componente normal da velocidade e u_t sua componente tangencial. Substituindo $\boldsymbol{u}$ em

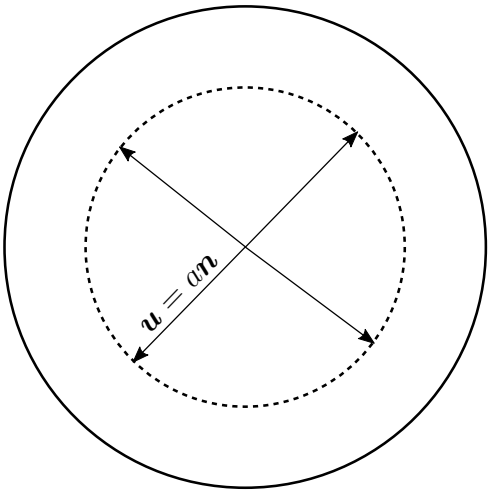


Figura 3.1: Advecção da interface sob um campo de velocidade normal a ela. A linha tracejada representa a interface no instante n, e a linha contínua a representa no instante $n+1$. Os pontos do isocontorno $\phi = 0$ no instante n estão na região interior à interface no instante $n+1$, e a uma distância de $a\Delta t$ do novo isocontorno nulo de ϕ. Logo, $\phi^{(n+1)} = a\Delta t$ nesses pontos.

(2.13),

$$\begin{aligned} \phi_t + (u_n \boldsymbol{n} + u_t \boldsymbol{t}) \cdot \nabla\phi &= 0 \\ \phi_t + u_n \boldsymbol{n} \cdot \nabla\phi &= 0, \end{aligned} \tag{3.2}$$

pois $\boldsymbol{t} \cdot \nabla\phi = 0$, uma vez que $\nabla\phi$ possui a mesma direção de $\boldsymbol{n}$ (e, portanto, é perpendicular a $\boldsymbol{t}$). Além disso, como

$$\boldsymbol{n} \cdot \nabla\phi = -\frac{\nabla\phi}{|\nabla\phi|} \cdot \nabla\phi = -\frac{|\nabla\phi|^2}{|\nabla\phi|} = -|\nabla\phi|, \tag{3.3}$$

a equação (3.2) pode ser reescrita como

$$\begin{aligned} \phi_t - u_n \frac{\nabla\phi}{|\nabla\phi|} \cdot \nabla\phi &= 0 \\ \phi_t - u_n |\nabla\phi| &= 0. \end{aligned} \tag{3.4}$$

Essa equação representa a movimentação da interface na direção normal sob velocidade u_n. Suponha $u_n = a$ constante, de qualquer sinal. Se $a > 0$, a interface se movimenta no sentido da normal, e se $a < 0$, no sentido contrário.

Quando ϕ é uma função distância sinalizada, $|\nabla\phi| = 1$, e a equação (3.4) se reduz a $\phi_t = a$. Discretizando no tempo com diferença avançada, chegamos a $\phi^{n+1} = \phi^n + a\Delta t$. Se $a > 0$, após um passo de tempo, o isocontorno $\phi = 0$ se torna o isocontorno $\phi = a\Delta t$, e o isocontorno $\phi = -a\Delta t$ se torna o isocontorno $\phi = 0$ (Figura 3.1). Logo, a interface está se movimentando na direção normal com velocidade a.

Calculando o gradiente da equação discretizada, $\nabla\phi^{n+1} = \nabla\phi^n + \nabla(a\Delta t)$, e, uma vez que a é constante no espaço, chegamos a $\nabla\phi^{n+1} = \nabla\phi^n$. Portanto, se a for constante, uma discretização temporal com diferença avançada não faz com que uma função distância perca a propriedade $|\nabla\phi| = 1$. Infelizmente, se a não for constante, essa afirmação não pode ser garantida.

Suponha agora que $u_n = a$ não seja constante no espaço, e reescrevamos a equação (3.4) como

$$\phi_t - \frac{a\nabla\phi \cdot \nabla\phi}{|\nabla\phi|} = 0 \tag{3.5}$$

$$\phi_t - \frac{a}{|\nabla\phi|}(\phi_x, \phi_y) \cdot (\phi_x, \phi_y) = 0 \tag{3.6}$$

$$\phi_t - \frac{a}{|\nabla\phi|}(\phi_x^2 + \phi_y^2) = 0. \tag{3.7}$$

Olhando para (3.7) e comparando com (2.13), vemos que o problema de movimentação na direção normal pode ser tratado como um problema de advecção, de modo que podemos aproveitar o método WENO de quinta ordem desenvolvido na seção 2.4.1.

Considerando apenas o primeiro termo espacial de (3.7) $-a\phi_x|\nabla\phi|^{-1}\phi_x$, onde $u = -a\phi_x|\nabla\phi|^{-1}$ é a "velocidade" com que ϕ se movimenta na direção x, façamos a mesma análise já feita na seção 2.4.1 para a escolha entre aproximar ϕ_x com a molécula computacional à esquerda ou à direita. Essa escolha é feita baseada no sinal da velocidade com a qual ϕ está sendo transportada em cada direção. Como $|\nabla\phi|$ é sempre positivo, apenas o sinal de $a\phi_x$ precisa ser utilizado para escolhermos entre $\phi_{x,i}^+$ e $\phi_{x,i}^-$. Contudo, como u também possui um termo com derivada espacial, a análise deve ser feita com mais cuidado.

Supondo $a > 0$, se $\phi_{x,i}^+ > 0$ e $\phi_{x,i}^- > 0$, então $u < 0$ independente de aproximarmos ϕ_x com uma molécula computacional à esquerda ou à direita. Logo, $\phi_{x,i}^+$ deve ser utilizado. Se $\phi_{x,i}^+ < 0$ e $\phi_{x,i}^- < 0$, então $u > 0$, e $\phi_{x,i}^-$ deve ser usado. O problema reside quando $\phi_{x,i}^+$ e $\phi_{x,i}^-$ possuírem sinais diferentes. Ainda supondo $a > 0$, se $\phi_{x,i}^+ < 0$ e $\phi_{x,i}^- > 0$, suponha que utilizemos $\phi_{x,i}^+$ para aproximar ϕ_x. Nesse caso, teremos $u > 0$, e, como visto na seção 2.4.1, deveríamos aproximar ϕ_x por $\phi_{x,i}^-$. Do mesmo modo, se aproximarmos ϕ_x por $\phi_{x,i}^-$, temos $u < 0$, o que indica que deveríamos ter utilizado $\phi_{x,i}^+$ na aproximação. Nesse caso, como mostrado em [26], a melhor opção é fazer a aproximação através de um esquema de Godunov, como descrito a seguir:

- Se $-a\phi_{x,i}^+ > 0$ e $-a\phi_{x,i}^- > 0$, utilize $\phi_x(x_i) \approx \phi_{x,i}^-$;
- Se $-a\phi_{x,i}^+ < 0$ e $-a\phi_{x,i}^- < 0$, utilize $\phi_x(x_i) \approx \phi_{x,i}^+$;
- Se $-a\phi_{x,i}^+ > 0$ e $-a\phi_{x,i}^- < 0$, utilize $\phi_x(x_i) \approx 0$;
- Se $-a\phi_{x,i}^+ < 0$ e $-a\phi_{x,i}^- > 0$, utilize
  - $\phi_x(x_i) \approx \phi_{x,i}^-$ se $|a\phi_{x,i}^+| < |a\phi_{x,i}^-|$;
  - $\phi_x(x_i) \approx \phi_{x,i}^+$ em caso contrário.

3.1.2 A Equação de Reinicialização

A reinicialização utilizada em [39] consiste em evoluir para o estado estacionário a equação

$$\phi_\tau + S(\phi)|\nabla\phi| = S(\phi), \tag{3.8}$$

sendo S a função sinal e τ um pseudotempo, sem conexão com o tempo t da equação de advecção. A condição inicial para essa equação é

$$\phi^{(\tau=0)} = \phi^{(n+1)}, \tag{3.9}$$

sendo $\phi^{(n+1)}$ a função level set após ser advectada por (2.13).

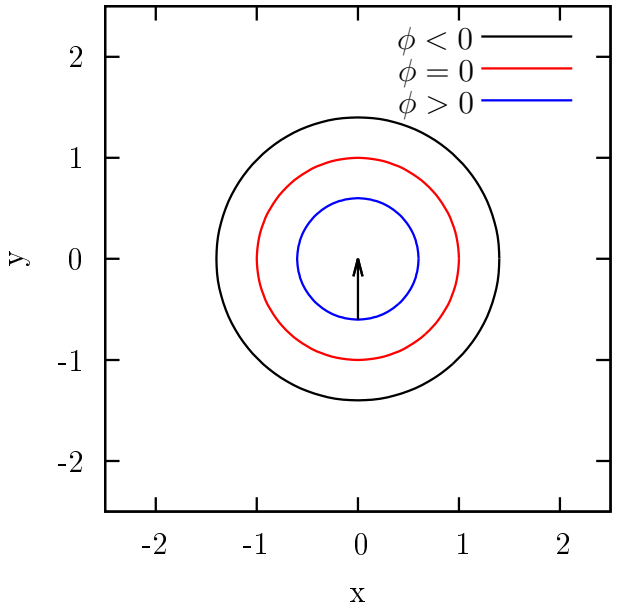


Figura 3.2: Três curvas de nível diferentes de ϕ. Em azul, na região interna à interface, uma curva de nível positiva; em vermelho, a curva de nível zero que representa a interface; em preto, na região externa, uma curva de nível negativa.

Em (3.8), a função sinal precisa ser suavizada para a obtenção de resultados numéricos satisfatórios (veja [26]). Uma possível suavização é

$$S(\phi) = \frac{\phi}{\sqrt{\phi^2 + \epsilon^2}}, \tag{3.10}$$

sendo ϵ um parâmetro de suavização, utilizado em [39] como $\epsilon = \Delta x$, onde Δx é o passo da malha (considerando que o passo seja igual em todas as direções). Podemos ver que a equação (3.8) é semelhante à (3.4), indicando que a reinicialização da interface também propaga informações na direção normal, com $u_n = -S(\phi)$. Em [28], foi sugerido que

$$S(\phi) = \frac{\phi}{\sqrt{\phi^2 + |\nabla\phi|^2\epsilon^2}} \tag{3.11}$$

apresenta uma melhor suavização para S, principalmente quando $|\nabla\phi|$ for muito diferente de 1 próximo à interface. A discretização de (3.8) foi feita utilizando um esquema Godunov como o descrito na seção 3.1.1.

Apesar da solução de (3.8) no estado estacionário ser uma função distância em todo o seu domínio, em problemas de escoamentos multifásicos só necessitamos que ϕ seja uma função distância em algumas células próximas à interface, nas quais calculamos a curvatura e o vetor normal. Por esse motivo, não precisamos resolver (3.8) até o estado estacionário.

A equação (3.8) pode ser reescrita como

$$\frac{\partial\phi}{\partial\tau} + \frac{S(\phi)\nabla\phi}{|\nabla\phi|} \cdot \nabla\phi = S(\phi). \tag{3.12}$$

Podemos ver que trata-se de uma equação de advecção, em que a velocidade é dada por $\boldsymbol{u}_{\text{rein}} = \frac{S(\phi)\nabla\phi}{|\nabla\phi|}$. Vamos mostrar que a equação acima propaga informações a partir da interface, na direção normal a ela e em ambos os sentidos . Primeiramente, é importante observar que a direção da velocidade é determinada unicamente pela direção de $\nabla\phi$, o

sentido depende de $S(\phi)$ e de $|\nabla\phi|$ e o módulo é sempre igual a 1 (a menos, claro, no contorno da interface, onde $\phi = 0$, e, portanto, $S(\phi) = 0$), pois $|\boldsymbol{u}_{\text{rein}}| = |S(\phi)| \left|\frac{\nabla\phi}{|\nabla\phi|}\right| = 1$. De fato, $\frac{\nabla\phi}{|\nabla\phi|}$ é um vetor unitário, e $|S(\phi)| = |\pm 1| = 1$ nos pontos fora do contorno da interface.

Para simplificar a explicação, considere que a interface é um círculo, conforme a Figura 3.2. Como o gradiente de uma função aponta para o sentido de maior crescimento dessa função, o gradiente de ϕ aponta para o centro desse círculo, independente do ponto em que é calculado. Num ponto qualquer da região interna à interface, temos que $\phi > 0$, o que implica em $S(\phi) = 1$. Desse modo, a velocidade possui o mesmo sentido que $\nabla\phi$, apontando para o centro do círculo. Já na região externa, temos $S(\phi) = -1$, o que implica que o sentido da velocidade é oposto ao do gradiente de ϕ (ou seja, a velocidade aponta para fora da interface na direção normal a ela). Por fim, na interface, $S(\phi) = 0$, e não há advecção. Assim, nas regiões internas e externas do domínio, a equação (3.12) propaga informações a partir da interface para o restante do domínio, na direção normal à interface, e com velocidade de módulo 1.

Isso significa que, caso a propriedade de função distância de ϕ seja importante apenas numa vizinhança da interface, a equação de reinicialização não precisa ser resolvida até seu estado estacionário. Para que ϕ seja uma função distância numa região formada por α células em volta da interface, (3.8) precisa ser evoluída somente até o instante em que a correção tenha sido propagada $h\alpha$ unidades de comprimento (ou seja, por $h\alpha$ unidades de tempo), considerando $h = \Delta x = \Delta y$.

Também na seção anterior foi discutido que, durante a solução numérica de (3.4), o sinal da velocidade na direção normal à interface deve ser levado em conta para a escolha da molécula computacional usada na aproximação das derivadas espaciais de ϕ. Contudo, em (3.8), essa velocidade é dada por $-S(\phi)$, e a suavização da função sinal por (3.11) já depende das derivadas espaciais de ϕ, ocasionando uma dependência circular.

Portanto, a estratégia utilizada para avançar um passo do pseudotempo τ foi aproximar, primeiramente, a função sinal S por (3.10), utilizando os valores de $\phi^{(\tau=0)}$. Note que (3.10) não depende de nenhuma das derivadas de ϕ. Com esse valor temporário, são escolhidas as corretas aproximações para as derivadas espaciais de ϕ, com a molécula computacional utilizada de acordo com a primeira aproximação obtida para $S(\phi)$. Por fim, a função sinal é atualizada com a expressão (3.11) utilizando as aproximações para as derivadas recém calculadas. O Apêndice B apresenta um algoritmo detalhado da implementação da reinicialização por EDP utilizando o método WENO de quinta ordem com um Runge-Kutta de terceira ordem.

3.1.3 Conservação de Massa

Como dito em [26], o processo de reinicialização descrito por (3.8) tende movimentar a interface, devido a erros numéricos. Em [37], foi sugerido um método para preservar a "massa" de ϕ em cada célula do domínio discretizado. Por "massa", deve-se entender *área* em duas dimensões, e *volume* em três. A conservação da massa foi feita modificando-se a equação (3.8) para

$$\phi_\tau = S(\phi)(1 - |\nabla\phi|) + \lambda\delta(\phi)|\nabla\phi|, \tag{3.13}$$

onde δ é a função Delta de Dirac, λ é função de τ, obtida impondo que

$$\partial_\tau \int_\Omega H(\phi) = \int_\Omega \delta(\phi)\phi_\tau = \int_\Omega \delta(\phi)\left(S(\phi)(1 - |\nabla\phi|) + \lambda\delta(\phi)|\nabla\phi|\right) = 0, \tag{3.14}$$

sendo H a função degrau. Daí, chegamos a

$$\lambda = -\frac{\int_\Omega \delta(\phi) S(\phi)(1-|\nabla\phi|)}{\int_\Omega \delta^2(\phi)|\nabla\phi|}. \tag{3.15}$$

Em (3.14), a integral da primeira expressão é a massa de ϕ ao longo do domínio Ω, e H é a função degrau, cuja derivada é a função δ de Dirac. Embora impor a conservação da massa em Ω faça algum sentido, faz mais sentido ainda impor a conservação da massa em cada célula $\Omega_{i,j}$ do domínio discretizado. Assim, as equações (3.13) e (3.15) se tornam, respectivamente,

$$\phi_\tau = S(\phi)(1-|\nabla\phi|) + \lambda_{i,j}\delta(\phi)|\nabla\phi|, \quad \text{para todo } \boldsymbol{x} \in \Omega_{i,j} \tag{3.16}$$

e

$$\lambda_{i,j} = -\frac{\int_{\Omega_{i,j}} \delta(\phi) S(\phi)(1-|\nabla\phi|)}{\int_{\Omega_{i,j}} \delta^2(\phi)|\nabla\phi|}. \tag{3.17}$$

De (3.8), podemos reescrever (3.17) como

$$\lambda_{i,j} = -\frac{\int_{\Omega_{i,j}} \delta(\phi)\phi_\tau}{\int_{\Omega_{i,j}} \delta^2(\phi)|\nabla\phi|}. \tag{3.18}$$

Diferente de λ, que só depende do tempo τ, $\lambda_{i,j}$ depende do tempo e do espaço, pois seu valor varia de uma célula para a outra. A função δ foi suavizada como em [26] por

$$\delta(\phi) = \begin{cases} 0 & \phi < -\epsilon \\ \frac{1}{2\epsilon}\left(1 + cos\left(\frac{\pi\phi}{\epsilon}\right)\right) & -\epsilon \le \phi \le \epsilon \\ 0 & \epsilon < \phi. \end{cases} \tag{3.19}$$

Em [26], o parâmetro ϵ foi recomendado como $1.5\Delta x$, onde Δx é o espaçamento da malha. Para a solução numérica das integrais, Sussman *et al.* [37] sugeriu a utilização de uma molécula computacional de 9 pontos:

$$\int_{\Omega_{i,j}} g \approx \frac{h^2}{24}\left(16 g_{i,j} + \sum_{m,n=-1;(m,n)\neq(0,0)}^{1} g_{i+m,j+n}\right), \tag{3.20}$$

onde a somatória é feita nos 8 pontos em volta de $g_{i,j}$. Entretanto, Sussman *et al.* não detalhou as condições de contorno que devem ser utilizadas para essa aproximação, razão pela qual tivemos de escolher um tratamento adequado para ser feito próximo às fronteiras do domínio. Primeiramente, note que (3.20) é, nas células distantes das fronteiras, uma média ponderada dos valores de g nas células vizinhas a (i,j), sendo que a célula (i,j) possui peso 16 e cada uma das outras células possui peso 1. Assim, a célula central (i,j) possui peso igual ao dobro da soma dos pesos das células vizinhas.

Nos contornos do domínio, optamos por desconsiderar as células vizinhas a (i,j) que não pertencem a Ω, o que é equivalente a impor condição de contorno $g_{i,j} = 0$, $(i,j) \notin \Omega$. Desse modo, os pontos que se encontram fora do domínio não têm influência no cálculo de (3.20). Contudo, como, nesse caso, não utilizamos mais 8 pontos para compor a média, o peso da célula central não é mais o dobro da soma dos pesos das células vizinha. Optando por manter essa propriedade, primeiramente verificamos quantas das células vizinhas a (i,j) pertencem ao domínio Ω (chamemos de $m(i,j)$ - ver Figura 3.3). Depois, a integral é calculada pela expressão

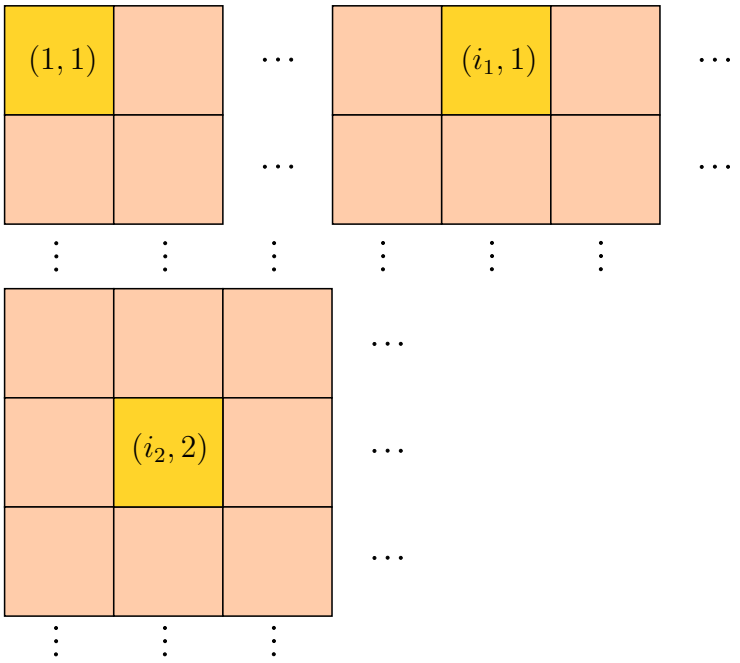


Figura 3.3: Células utilizadas em (3.21) para a aproximação do cálculo da integral. Temos $m(1,1) = 3$ (há três células vizinhas à célula $(1,1)$ no domínio), $m(i_1,1) = 5$ e $m(i_2,2) = 8$.

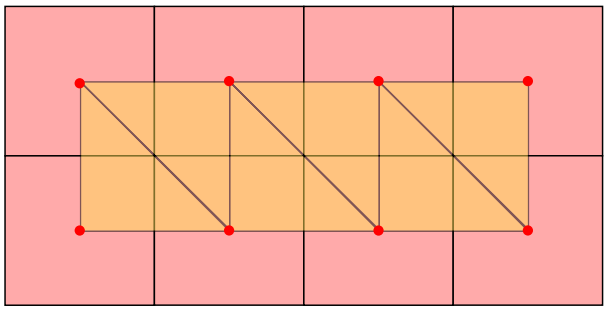


Figura 3.4: Ao fundo, de cor avermelhada, é representada a malha de advecção, e à frente, em cor alaranjada, a malha triangular. Os pontos em vermelho são os centros das células de advecção, nos quais o valor da função level-set é conhecido.

$$\int_{\Omega_{i,j}} g \approx \frac{h^2}{3m(i,j)} \left(2m(i,j)g_{i,j} + \sum_{m,n=-1;(m,n)\neq(0,0)}^{1} g_{i+m,j+n} \right). \tag{3.21}$$

O Apêndice B apresenta um algoritmo detalhado da implementação da advecção da função level-set mais a reinicialização por EDP com conservação de massa.

3.2 Reinicialização Geométrica

Essa seção tem por objetivo descrever uma reinicialização diferente da proposta por Sussman *et al.* [39]. Ao invés de resolvermos uma equação diferencial, a fim de propagar informações a partir da interface para o restante do domínio, a reinicialização é feita de maneira totalmente geométrica. Esse método foi proposto por Ausas *et al.* [4]. para volumes finitos. Não encontramos nenhum artigo que utilizasse essa proposta para diferenças finitas, de modo que foi preciso fazermos algumas adaptações em relação ao artigo original quanto à malha utilizada.

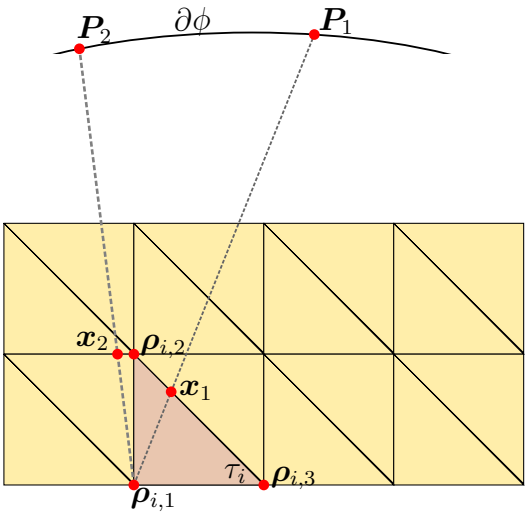


Figura 3.5: $\partial\phi$ é o contorno da interface representada por ϕ. $\boldsymbol{P}_1$ é o ponto da interface mais próximo aos pontos $\boldsymbol{x}_1$ e $\boldsymbol{\rho}_{i,1}$.

Para a reinicialização geométrica, Ausas *et al.* utilizou uma malha de triângulos, conforme mostrada na Figura 3.4. A localização dos vértices dos triângulos em relação à malha de advecção será explicada mais adiante. Primeiramente, explicaremos a ideia básica dessa reinicialização.

Na Figura 3.5, considere que ϕ é uma função distância, de modo que $\phi(\boldsymbol{y})$ seja a distância de $\boldsymbol{y}$ até a interface $\partial\phi$. Além disso, considere que ϕ possui o mesmo sinal em todos os pontos de todos os triângulos que possuem o ponto $\boldsymbol{\rho}_{i,1}$. Assim, como $\boldsymbol{P}_1$ é o ponto da interface mais próximo aos pontos $\boldsymbol{x}_1$ e $\boldsymbol{\rho}_{i,1}$,

$$\phi(\boldsymbol{\rho}_{i,1}) = |\boldsymbol{P}_1 - \boldsymbol{\rho}_{i,1}| = |\boldsymbol{P}_1 - \boldsymbol{x}_1| + |\boldsymbol{x}_1 - \boldsymbol{\rho}_{i,1}| = \phi(\boldsymbol{x}_1) + |\boldsymbol{x}_1 - \boldsymbol{\rho}_{i,1}| \tag{3.22}$$

Portanto, se ϕ for uma função distância em todos os pontos do domínio, com exceção de $\boldsymbol{\rho}_{i,1}$, podemos calcular $\phi(\boldsymbol{\rho}_{i,1})$ de modo que ϕ seja uma função distância também em $\boldsymbol{\rho}_{i,1}$ utilizando (3.22). Contudo, na prática, não é tão simples assim. Se ao invés de $\boldsymbol{P}_1$, o ponto $\boldsymbol{P}_2$ fosse o ponto da interface mais próximo ao ponto $\boldsymbol{\rho}_{i,1}$, em (3.22) deveríamos trocar os pontos $\boldsymbol{P}_1$ e $\boldsymbol{x}_1$ por $\boldsymbol{P}_2$ e $\boldsymbol{x}_2$. Ou seja, devemos utilizar

$$\phi(\boldsymbol{\rho}_{i,1}) = \min_{\boldsymbol{x}\in\mathcal{T}_{\boldsymbol{\rho}_{i,1}}\setminus\bar{\mathcal{T}}} \left(\phi(\boldsymbol{x}) + |\boldsymbol{x} - \boldsymbol{\rho}_{i,1}|\right). \tag{3.23}$$

A notação utilizada até aqui será explicada com maiores detalhes no decorrer do texto, mas adiantamos que $\mathcal{T}_{\boldsymbol{\rho}_{i,1}} \setminus \bar{\mathcal{T}}$ representa o conjunto de todos os triângulos que possuem o vértice $\boldsymbol{\rho}_{i,1}$ e nos quais ϕ possui o mesmo sinal em todos pontos desses triângulos.

Note que, para que utilizemos (3.23), precisamos conhecer ϕ em todos os pontos das arestas dos triângulos de $\mathcal{T}_{\boldsymbol{\rho}_{i,1}} \setminus \bar{\mathcal{T}}$. Como temos infinitos desses pontos, isso não pode ser feito computacionalmente. Desse modo, temos que escolher alguns desses pontos para armazenarmos os valores de ϕ, e utilizar alguma aproximação para estimar ϕ nos demais pontos. A estratégia utilizada por Ausas *et al.* foi conhecer ϕ nos vértices de todos os triângulos, e para aproximar ϕ no ponto $\boldsymbol{x}_1$ da aresta formada pelos vértices $\boldsymbol{\rho}_{i,2}$ e $\boldsymbol{\rho}_{i,3}$, utilizamos interpolação linear, como a seguir:

$$\frac{\phi(\boldsymbol{x}) - \phi(\boldsymbol{\rho}_{i,2})}{\phi(\boldsymbol{\rho}_{i,3}) - \phi(\boldsymbol{\rho}_{i,2})} = \frac{|\boldsymbol{x} - \boldsymbol{\rho}_{i,2}|}{|\boldsymbol{\rho}_{i,3} - \boldsymbol{\rho}_{i,2}|}. \tag{3.24}$$

Como precisamos conhecer ϕ nos vértices dos triângulos, construímos a malha triangular nesse trabalho de modo que os vértices dos triângulos coincidam com o centro das células das malhas de advecção, conforme mostra a Figura 3.4.

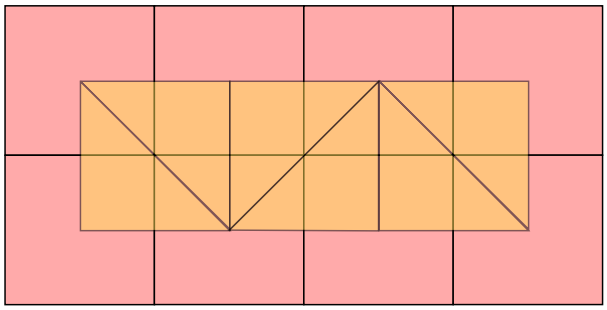


Figura 3.6: Malha triangular em que as hipotenusas dos triângulos não possuem todas as mesmas direções.

Segundo apresentado em [4], as arestas que formam as hipotenusas dos triângulos podem ser geradas sem seguir nenhuma direção particular (ela pode unir o vértice superior esquerdo ao inferior direito ou o inferior esquerdo ao superior direito). Essa direção pode, inclusive, variar dentro da mesma malha, sem que os resultados sejam modificados significativamente. Desse modo, as malhas triangulares das Figuras 3.4 e 3.6 produzem, essencialmente, as mesmas soluções. Nesse trabalho, optamos por gerar todos os triangulos conforme a Figura 3.4. Ou seja, a hipotenusa dos triângulos sempre conecta o ponto superior esquerdo ao inferior direito dos mesmos.

Como faremos interpolações lineares em ϕ, a primeira etapa da reinicialização geométrica consiste da linearização de ϕ nos triângulos em que ϕ mude de sinal (ou seja, há ao menos um vértice de cada um desses triângulos em que $\phi > 0$, e um vértice em que $\phi < 0$). O isocontorno nulo de ϕ após ter sido linearizada será um segmento de reta dentro de cada um dos triângulos, como mostra a Figura 3.7. Uma vez linearizada, é preciso fazer algumas correções para melhorar a conservação de massa do método. Contudo, essa correção será feita de modo a continuar preservando a linearidade da interface. Isso será feito somando-se à função level-set linearizada uma função que também seja linear. Por fim, ϕ é aproximada nos demais triângulos do domínio, utilizando uma expressão bem semelhante à (3.23).

A seguir, faremos algumas definições que serão necessárias para o desenvolvimento do processo de reinicialização geométrica.

3.2.1 O processo de Reinicialização

Denominemos cada um dos triângulos da malha da reinicialização geométrica de τ_i, $i = 1, 2, \ldots, N$, e a malha formada pelos N triângulos de $\mathcal{T}$. O conjunto de todos os pontos que são vértices de algum triângulo será chamado de $\mathcal{V}$, e o conjunto de todos os triângulos que possuem o vértice $\boldsymbol{v}_i \in \mathcal{V}$ de $\mathcal{T}_{\boldsymbol{v}_i}$.

Considere V o espaço das funções contínuas que são lineares dentro de cada τ_i, $\bar{\phi} \in V$ uma aproximação para ϕ e $\partial\bar{\phi}$ o isocontorno nulo de $\bar{\phi}$. Nessa seção, não podemos considerar que ϕ (e muito menos que $\bar{\phi}$) seja uma função distância; estamos assumindo que sua propriedade de função distância foi perdida durante a etapa de advecção. Seja ainda $\hat{d}$ a função distância sinalizada à interface $\partial\bar{\phi}$, dada por

$$\hat{d}(\boldsymbol{x}) = S^*(\phi(\boldsymbol{x})) \min_{\boldsymbol{y} \in \partial\bar{\phi}} |\boldsymbol{x} - \boldsymbol{y}|, \tag{3.25}$$

onde S^* é a função sinal não suavizada. Desejamos, então, obter a função $\tilde{d} \in V$, que é uma aproximação para $\hat{d}$, tal que a área da região interna à interface representada por $\tilde{d}$ seja igual à da interface representada por $\bar{\phi}$.

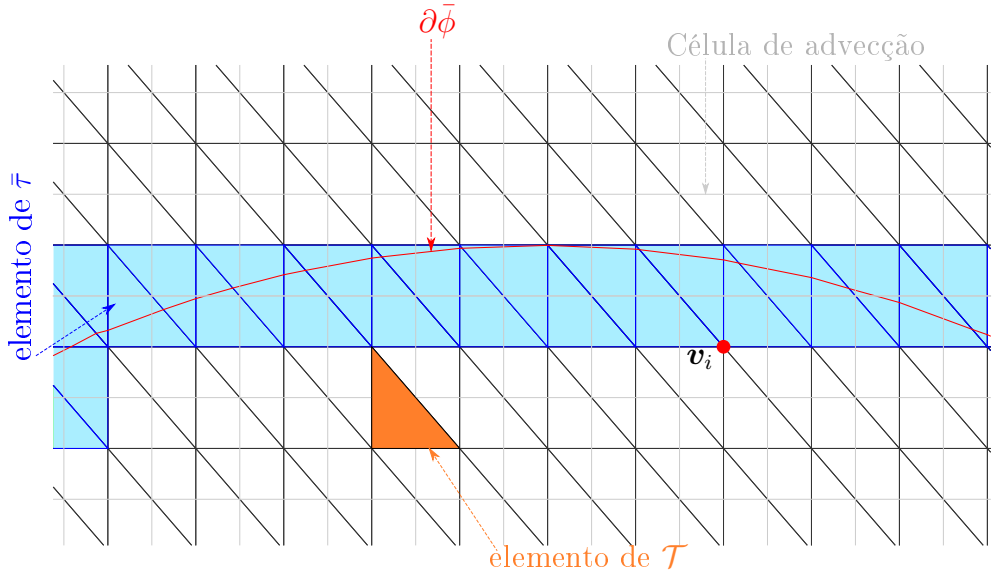


Figura 3.7: Identificação dos elementos utilizados na reinicialização geométrica.

Denominemos, ainda, $\bar{\mathcal{T}}$ como o conjunto dos triângulos nos quais $\bar{\phi}$ muda de sinal, e chamemos seus elementos de $\bar{\tau}$ (Figura 3.7). Se $\bar{\tau}_i \in \bar{\mathcal{T}}$ e $\boldsymbol{\rho}_{i,j}$, $j \in \{1,2,3\}$, são os vértices de $\bar{\tau}_i$, então $\boldsymbol{\rho}_{i,j}$ são vértices adjacentes à interface. Por fim, chamemos o conjunto dos pontos que são vértices de algum triângulo de $\bar{\mathcal{T}}$ como $\bar{\mathcal{V}}$. O conjunto dos triângulos em que ϕ é positivo em todos os seus vértices será denominado $\mathcal{T}^+$, e o conjunto dos triângulos em que ϕ é negativo em todos os seus vértices será denominado $\mathcal{T}^-$.

Feitas essas definições, a área da região interna à interface representada por $\bar{\phi}$ é dada por

$$A(\bar{\phi}) = \int_{\mathcal{T}} H(\bar{\phi}) d\boldsymbol{x} = \int_{\mathcal{T}^+ \cup \bar{\mathcal{T}}} H(\bar{\phi}) d\boldsymbol{x} = \sum_{\tau_i \in \mathcal{T}^+ \cup \bar{\mathcal{T}}} \int_{\tau_i} H(\bar{\phi}) d\boldsymbol{x} = \sum_{\tau_i \in \mathcal{T}^+ \cup \bar{\mathcal{T}}} A_{\tau_i}(\bar{\phi}), \tag{3.26}$$

em que H é a função degrau e $A_{\tau_i}(\bar{\phi})$ é a área da região do triângulo τ_i em que $\bar{\phi}$ é positivo.

O processo de reinicialização pode, então, ser dividido em duas etapas principais, que serão descritas a seguir.

3.2.2 Reinicialização nos Vértices Adjacentes à Interface

Inicialmente, devemos construir $\partial\bar{\phi}$ dentro de cada elemento de $\bar{\mathcal{T}}$, utilizando os valores de ϕ. Para obter os elementos de $\bar{\mathcal{T}}$, a estratégia utilizada foi procurar pelos triângulos que possuam ao menos um vértice em que ϕ seja positivo, e um vértice em que ϕ seja negativo. Para simplificar a construção de $\partial\bar{\phi}$, impomos $|\phi| \geq \text{tol} = 10^{-10}$, garantindo que $\partial\bar{\phi}$ não passe pelos vértices de nenhum triângulo de $\mathcal{T}$.

Para cada $\bar{\tau}_i$, sabemos que $\partial\bar{\phi}$ intersecta exatamente duas de suas arestas. Para obter esses pontos, ϕ é interpolada linearmente sobre cada aresta de $\bar{\tau}_i$. Se $\partial\bar{\phi}$ intersecta no ponto $\boldsymbol{P_1}$ a aresta de $\bar{\tau}_i$ que une os vértices $\boldsymbol{\rho}_{i,j_1}$ e $\boldsymbol{\rho}_{i,j_2}$ (Figura 3.8), obtemos a expressão a seguir:

$$\frac{|\boldsymbol{P_1} - \boldsymbol{\rho}_{i,j_1}|}{|\boldsymbol{\rho}_{i,j_1} - \boldsymbol{\rho}_{i,j_2}|} = \frac{\phi(\boldsymbol{P_1}) - \phi(\boldsymbol{\rho}_{i,j_1})}{\phi(\boldsymbol{\rho}_{i,j_2}) - \phi(\boldsymbol{\rho}_{i,j_1})}. \tag{3.27}$$

Como $\partial\bar{\phi}$ passa por $\boldsymbol{P_1}$, então $\phi(\boldsymbol{P_1}) = 0$. Daí, chegamos a

$$|\boldsymbol{P_1} - \boldsymbol{\rho}_{i,j_1}| = |\boldsymbol{\rho}_{i,j_1} - \boldsymbol{\rho}_{i,j_2}| \frac{\phi(\boldsymbol{\rho}_{i,j_1})}{\phi(\boldsymbol{\rho}_{i,j_1}) - \phi(\boldsymbol{\rho}_{i,j_2})}. \tag{3.28}$$

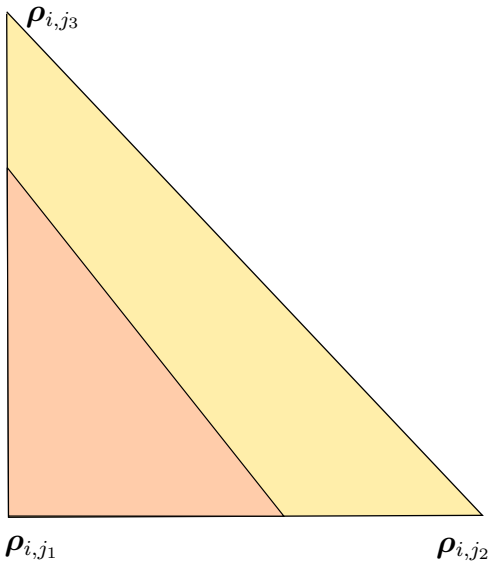


Figura 3.8: $\partial\bar{\phi}$ intersecta no ponto $\boldsymbol{P_1}$ a aresta que une os vértices $\boldsymbol{\rho}_{i,j_1}$ e $\boldsymbol{\rho}_{i,j_2}$. Como $\partial\bar{\phi}$ é o isocontorno nulo de $\bar{\phi}$, ou $\bar{\phi}$ é negativo em todo o polígono abaixo de $\partial\bar{\phi}$ ou em todo o polígono acima dessa reta.

A partir da Figura 3.8, podemos obter a expressão a seguir para as coordenadas do ponto $\boldsymbol{P_1}$:

$$\boldsymbol{P_1} = \boldsymbol{\rho}_{i,j_1} + |\boldsymbol{P_1} - \boldsymbol{\rho}_{i,j_1}| \frac{\boldsymbol{\rho}_{i,j_2} - \boldsymbol{\rho}_{i,j_1}}{|\boldsymbol{\rho}_{i,j_2} - \boldsymbol{\rho}_{i,j_1}|}. \tag{3.29}$$

Essa expressão pode ser compreendida da seguinte forma: para chegarmos ao ponto $\boldsymbol{P_1}$ partindo de $\boldsymbol{\rho}_{i,j_1}$, devemos somar um vetor que tenha a direção e o sentido de $\boldsymbol{\rho}_{i,j_2} - \boldsymbol{\rho}_{i,j_1}$ e magnitude $|\boldsymbol{P_1} - \boldsymbol{\rho}_{i,j_1}|$.

Por fim, substituindo (3.28) em (3.29), chegamos a

$$\boldsymbol{P_1} = \boldsymbol{\rho}_{i,j_1} + \frac{\phi(\boldsymbol{\rho}_{i,j_1})}{\phi(\boldsymbol{\rho}_{i,j_1}) - \phi(\boldsymbol{\rho}_{i,j_2})}(\boldsymbol{\rho}_{i,j_2} - \boldsymbol{\rho}_{i,j_1}). \tag{3.30}$$

Note que quando o contorno de $\partial\bar{\phi}$ é obtido dessa maneira, ele é contínuo ao longo de todo o domínio. Na Figura 3.8, o mesmo ponto $\boldsymbol{P}_1$ será obtido ao analisarmos o triângulo

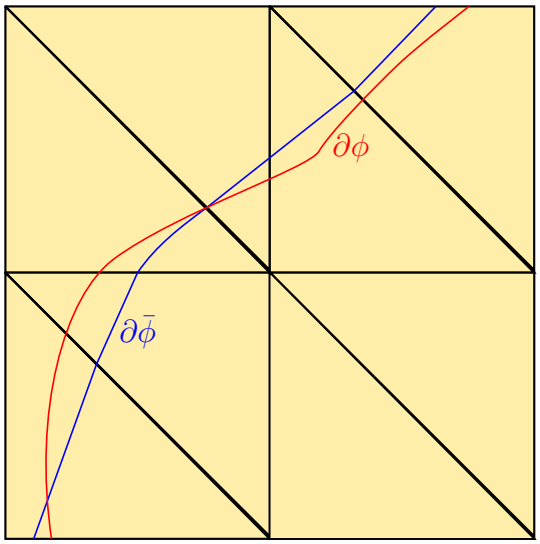


Figura 3.9: $\partial\phi$ é o contorno da interface representada por ϕ, e $\partial\bar{\phi}$ é o contorno da interface representada por $\bar{\phi}$.

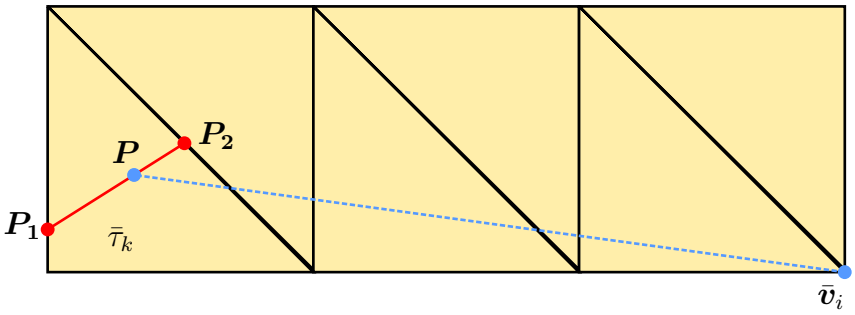


Figura 3.10: $\overline{\boldsymbol{P_1}\ \boldsymbol{P_2}}$ é a parte da interface que se encontra dentro do triângulo $\bar{\tau}_k$.

imediatamente à direita ao que foi representado, de modo que os segmentos de reta de cada triângulo formam uma poligonal contínua. Além disso, é importante ressaltar que $\bar{\phi}$ coincide com ϕ nos vértices dos triângulos, visto que a interpolação linear para o cálculo de $\bar{\phi}$ utiliza justamente os valores de ϕ nesses pontos. Na Figura 3.9 comparamos as interfaces $\partial\phi$ com $\partial\bar{\phi}$.

Uma vez que $\partial\bar{\phi}$ foi obtida em todos os triângulos de $\bar{\mathcal{T}}$, podemos calcular a função

$$\phi^*(\bar{\boldsymbol{v}}_i) = \hat{d}(\bar{\boldsymbol{v}}_i), \quad \forall \bar{\boldsymbol{v}}_i \in \bar{\mathcal{V}} \tag{3.31}$$

onde ϕ^* é uma função que pertence a V e que coincide com $\hat{d}$ nos vértices adjacentes à interface. Nos demais pontos do domínio, os valores de ϕ^* não são conhecidos, e precisam ser aproximados quando isso for necessário. Na prática, só precisamos dos valores de ϕ^* nos pontos que pertencem às arestas dos triângulos. Esses valores são aproximados utilizando-se interpolação linear sempre que necessário.

Para o cálculo de $\hat{d}(\bar{\boldsymbol{v}}_i)$, precisamos obter a menor distância do vértice $\bar{\boldsymbol{v}}_i$ ao contorno da interface $\partial\bar{\phi}$. Esse problema é um pouco menos simples do que parece. Note que a interface é formada por vários segmentos de reta, de modo que precisamos calcular a distância de $\bar{\boldsymbol{v}}_i$ a cada um desses segmentos, e usar a menor dentre essas distâncias.

Considere, por exemplo, o caso da Figura 3.10. A interface é delimitada pelos pontos $\boldsymbol{P_1}$ e $\boldsymbol{P_2}$, e desejamos calcular a menor distância entre $\bar{\boldsymbol{v}}_i$ e o segmento de reta $\overline{\boldsymbol{P_1}\ \boldsymbol{P_2}}$. Para isso, devemos encontrar uma expressão para a distância entre $\bar{\boldsymbol{v}}_i$ e um ponto $\boldsymbol{P}$ qualquer desse segmento de reta. Primeiramente, escrevamos $\boldsymbol{P}$ como

$$\boldsymbol{P} = \boldsymbol{P_1} + \lambda(\boldsymbol{P_2} - \boldsymbol{P_1}), \quad \lambda \in [0,1]. \tag{3.32}$$

Assim, a distância de $\bar{\boldsymbol{v}}_i$ a $\boldsymbol{P}$ pode ser obtida pela expressão a seguir:

$$|\bar{\boldsymbol{v}}_i - \boldsymbol{P}| = |\bar{\boldsymbol{v}}_i - (\boldsymbol{P_1} + \lambda\,(\boldsymbol{P_2} - \boldsymbol{P_1}))\,|. \tag{3.33}$$

Escrevamos as coordenadas dos pontos envolvidos na expressão acima como $\boldsymbol{P_1} = (P_{1_x}, P_{1_y})$, $\boldsymbol{P_2} = (P_{2_x}, P_{2_y})$ e $\bar{\boldsymbol{v}}_i = (\bar{v}_{i_x}, \bar{v}_{i_y})$. Assim, expandindo (3.33), chegamos a

$$|\bar{\boldsymbol{v}}_i - \boldsymbol{P}| = \sqrt{(\bar{v}_{i_x} - (P_{1_x} + \lambda(P_{2_x} - P_{1_x})))^2 + \left(\bar{v}_{i_y} - \left(P_{1_y} + \lambda(P_{2_y} - P_{1_y})\right)\right)^2}. \tag{3.34}$$

Agora, devemos obter o parâmetro λ que nos dê o ponto P mais próximo de $\bar{\boldsymbol{v}}_i$ (ou seja, obter λ' que minimize (3.34)). Derivando (3.34) em relação a λ e igualando a zero, obtemos:

$$\lambda' = \frac{(P_{1_x} - \bar{v}_{i_x})\,(P_{1_x} - P_{2_x}) + \left(P_{1_y} - \bar{v}_{i_y}\right)\left(P_{1_y} - P_{2_y}\right)}{(P_{2_x} - P_{1_x})^2 + (P_{2_y} - P_{1_y})^2}. \tag{3.35}$$

Devemos nos atentar ao fato de que em (3.32) temos a restrição de $\lambda \in [0,1]$, para garantir que o ponto $\boldsymbol{P}$ esteja entre $\boldsymbol{P_1}$ e $\boldsymbol{P_2}$. Se obtivermos λ' fora desse intervalo, o

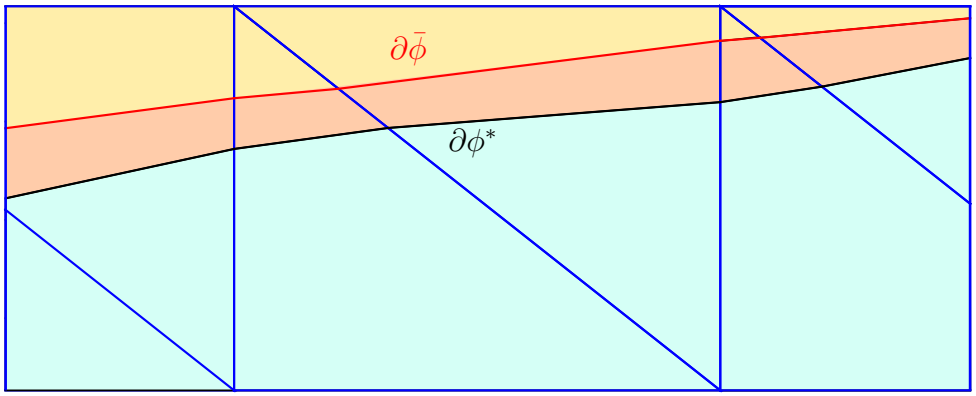


Figura 3.11: Interfaces representadas por $\bar{\phi}$ (vermelho) e ϕ^* (preto). Em laranja, mostramos a área da região interna à interface representada por $\bar{\phi}$, e em azul a área interna às duas interfaces. A diferença entre as duas retas foi intencionalmente exagerada para facilitar a visualização.

ponto $\boldsymbol{P'}$ calculado por (3.32) não estará em $\overline{\boldsymbol{P_1}\,\boldsymbol{P_2}}$. Desse modo, devemos escolher λ' que nos dê o ponto $\boldsymbol{P''}$ de $\overline{\boldsymbol{P_1}\,\boldsymbol{P_2}}$ mais próximo de $\boldsymbol{P'}$. Para $\lambda' > 1$, devemos, então, utilizar $\lambda' = 1$, e para $\lambda' < 0$, devemos utilizar $\lambda' = 0$. Se $\lambda \in [0, 1]$, fazemos simplesmente $\boldsymbol{P''} = \boldsymbol{P'}$.

Uma vez que $\boldsymbol{P''}$ é conhecido, a distância de $\bar{\boldsymbol{v}}_i$ a $\boldsymbol{P''}$ pode ser facilmente encontrada. Essa é a menor distância de $\bar{\boldsymbol{v}}_i$ ao segmento de reta que representa a interface em $\bar{\tau}_k$. Devemos realizar o mesmo processo para encontrar essa distância à interface em cada um dos outros triângulos de $\bar{\mathcal{T}}$ e, então, $\phi^*(\bar{\boldsymbol{v}}_i) = \hat{d}(\bar{\boldsymbol{v}}_i)$ será a menor dessas distâncias.

Como ϕ^* possui os mesmos valores que $\hat{d}$ nos vértices dos triângulos que são adjacentes à interface, ϕ^* pode ser considerada como uma função distância. Contudo, não há garantias de que a área interna à interface representada por ϕ^* seja igual à da interface representada por $\bar{\phi}$. Veja na Figura 3.11 um exemplo que ilustra a diferença entre essas duas áreas.

Assim, vamos em busca de uma função $\tilde{d}$ que também seja linear, e cuja área interna seja igual à de $\bar{\phi}$. Primeiramente, vamos representar a variação de área entre as interfaces representadas por duas funções level-set por

$$\Delta A(\bar{\phi}, \phi^*) = A(\bar{\phi}) - A(\phi^*) = \sum_{\tau_i \in \mathcal{T}^+ \cup \bar{\mathcal{T}}} \left(A_{\tau_i}\bar{\phi} - A_{\tau_i}(\phi^*)\right) = \sum_{\bar{\tau}_i \in \bar{\mathcal{T}}} \left(A_{\bar{\tau}_i}\bar{\phi} - A_{\bar{\tau}_i}(\phi^*)\right). \quad (3.36)$$

A última passagem se deve ao fato de que, se $\tau_i \in \mathcal{T}^+$, $A_{\tau_i}(\phi^*) = A_{\tau_i}(\bar{\phi}) = \Delta x \Delta y/2$, sendo Δx e Δy o espaçamento da malha nas direções x e y, respectivamente.

O cálculo de $A_{\bar{\tau}_i}(\bar{\phi})$ (e, de maneira análoga, o de $A_{\bar{\tau}_i}(\phi^*)$) foi feito de maneira totalmente geométrica, utilizando apenas as coordenadas dos vértices que delimitam a região interna à interface em $\bar{\tau}_i$ (ou seja, a parte do triângulo em que $\bar{\phi}$ é positivo). Note que trata-se de um polígono de três ou quatro vértices. Conhecidos os vértices, a área foi calculada pela expressão

$$A_{\bar{\tau}_i}(\bar{\phi}) = \frac{1}{2}\left|\sum_{i=0}^{i_{\max}} \begin{vmatrix} x_i & x_{i+1} \\ y_i & y_{i+1} \end{vmatrix}\right|, \quad (3.37)$$

onde $i_{\max} \in \{2, 3\}$ e (x_i, y_i) é o i-ésimo ponto que delimita o polígono cuja área está sendo calculada. Esses pontos devem estar ordenados de maneira que o polígono seja construído unindo-se o ponto (x_i, y_i) ao ponto (x_{i+1}, y_{i+1}) por um segmento de reta. Além disso, como o polígono é fechado, devemos ter $x_0 = x_{i_{\max}+1}$ e $y_0 = y_{i_{\max}+1}$. Dois desses pontos

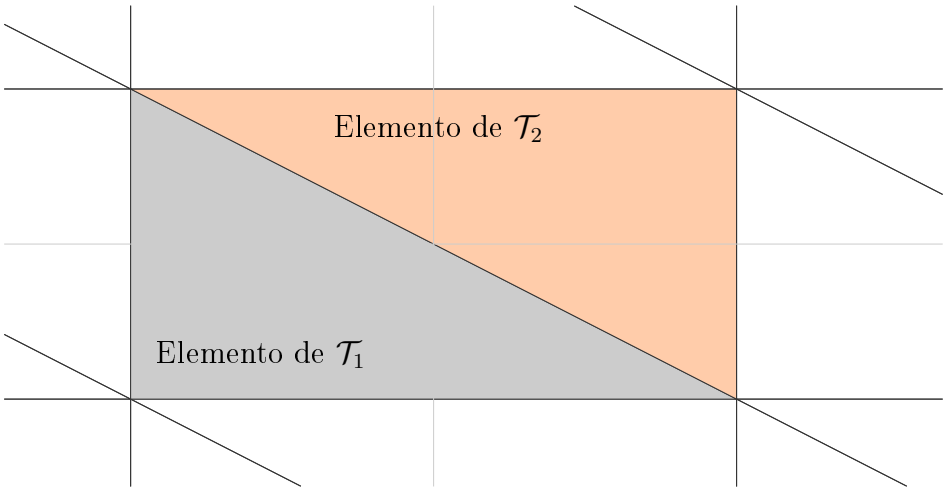


Figura 3.12: Tipos de triângulos utilizados nesse trabalho.

foram obtidos na construção de $\partial\bar{\phi}$ (pontos $\boldsymbol{P_1}$ e $\boldsymbol{P_2}$ da Figura 3.8), mas ainda precisamos obter o(s) outro(s) ponto(s) que forma(m) o polígono.

Para simplificar a explicação, consideraremos apenas o cálculo dessa área nos triângulos $\bar{\tau}_i \in \mathcal{T}_1$, que são os triângulos nos quais o ângulo de 90^o está no vértice inferior esquerdo (Figura 3.12). O cálculo nos triângulos que pertencem a $\mathcal{T}_2$ pode ser feito de maneira análoga.

Para a obtenção dos vértices do polígono cuja área desejamos calcular, note que, se ϕ for positivo apenas no vértice $\boldsymbol{\rho}_{i,j}$, para um j específico, o polígono é formado pelas intersecções de $\partial\bar{\phi}$ com as arestas de $\bar{\tau}_i$ mais o ponto $\boldsymbol{\rho}_{i,j}$ (veja Figura 3.13a), e esses pontos podem ser utilizados em (3.37) em qualquer ordem. Já se ϕ for positivo em dois vértices do triângulo, o polígono possui dois pontos além das intersecções do contorno da interface com as arestas do triângulo.

Caso o polígono possua quatro vértices, teremos sete possíveis casos, que também são apresentados na Figura 3.13. Apesar da grande quantidade de casos a serem verificados, isso não é algo complexo de ser feito. Por exemplo, se $\partial\bar{\phi}$ intersectar a aresta vertical do triângulo mas $\bar{\phi}$ não for positivo no vértice superior, então ocorre um dos casos das Figuras 3.13b, 3.13c ou 3.13d. Se $\partial\bar{\phi}$ intersectar a aresta vertical do triângulo, e $\bar{\phi}$ for positivo no vértice superior esquerdo do triângulo, então o vértice inferior direito também pertence ao polígono (Figura 3.13h). De forma análoga, podemos analisar o caso em que $\partial\bar{\phi}$ intersecta a aresta horizontal do triângulo, para chegar aos casos das Figuras 3.13e, 3.13f ou 3.13g.

Agora que sabemos calcular a área da região interna à interface, vamos, então, em busca de uma função η, constante em cada $\bar{\tau}_i$, tal que

$$\Delta A_{\bar{\tau}_i}(\bar{\phi}, \phi^* + \eta) = A_{\bar{\tau}_i}(\bar{\phi}) - A_{\bar{\tau}_i}(\phi^* + \eta) = 0. \tag{3.38}$$

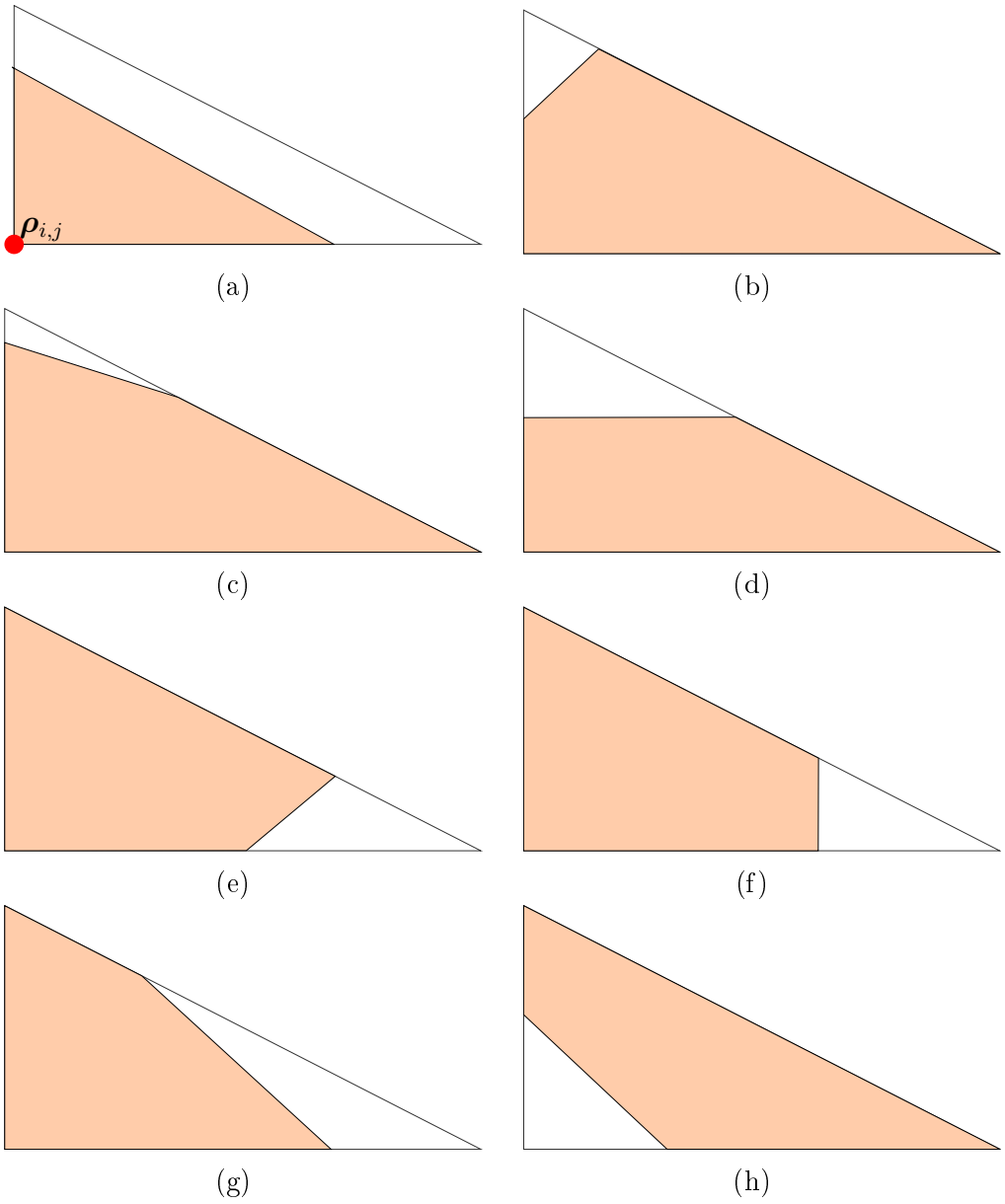


Figura 3.13: Possíveis casos ao calcular a área da região interna à interface (região alaranjada). $\bar{\phi}$ é positivo nos vértices do triângulo que estão dentro da região alaranjada.

Veja a Figura 3.14, e considere $P_i = (x_i, y_i)$, $i = 1, 2, \ldots, 7$. Aplicando (3.29) ao ponto $\boldsymbol{P_4}$,

$$\begin{aligned}
\boldsymbol{P_4} &= \boldsymbol{P_3} + |\boldsymbol{P_4} - \boldsymbol{P_3}| \frac{\boldsymbol{P_1} - \boldsymbol{P_3}}{|\boldsymbol{P_1} - \boldsymbol{P_3}|} \\
&= \boldsymbol{P_3} + |\boldsymbol{P_3} - \boldsymbol{P_1}| \frac{\bar{\phi}(\boldsymbol{P_3})}{\bar{\phi}(\boldsymbol{P_3}) - \bar{\phi}(\boldsymbol{P_1})} \frac{\boldsymbol{P_1} - \boldsymbol{P_3}}{|\boldsymbol{P_1} - \boldsymbol{P_3}|} \qquad (3.39) \\
&= \boldsymbol{P_3} \left(1 - \frac{\bar{\phi}(\boldsymbol{P_3})}{\bar{\phi}(\boldsymbol{P_3}) - \bar{\phi}(\boldsymbol{P_1})}\right) + \frac{\bar{\phi}(\boldsymbol{P_3})}{\bar{\phi}(\boldsymbol{P_3}) - \bar{\phi}(\boldsymbol{P_1})} \boldsymbol{P_1}. \qquad (3.40)
\end{aligned}$$

Para chegar a (3.39), basta utilizar (3.28) para obter $|\boldsymbol{P_4} - \boldsymbol{P_3}|$. De maneira análoga, deduzimos que

$$\boldsymbol{P_5} = \boldsymbol{P_3} \left(1 - \frac{\bar{\phi}(\boldsymbol{P_3})}{\bar{\phi}(\boldsymbol{P_3}) - \bar{\phi}(\boldsymbol{P_2})}\right) + \frac{\bar{\phi}(\boldsymbol{P_3})}{\bar{\phi}(\boldsymbol{P_3}) - \bar{\phi}(\boldsymbol{P_2})} \boldsymbol{P_2}.$$

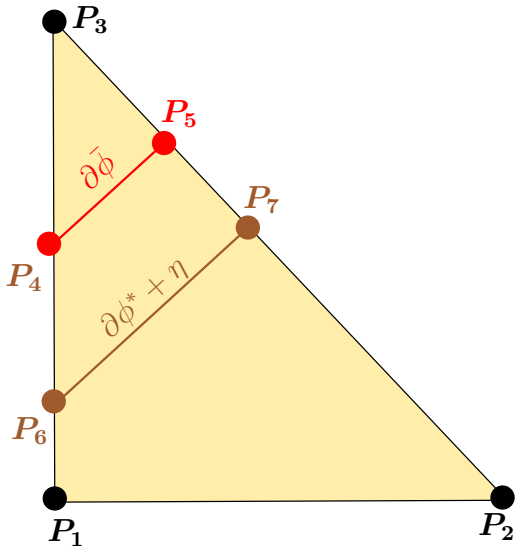


Figura 3.14: Interfaces representadas por $\bar{\phi}$ (vermelho) e $\phi^* + \eta$ (marrom).

Então, podemos escrever

$$\begin{aligned}
x_4 &= x_3\left(1 - \frac{\bar{\phi}(\boldsymbol{P_3})}{\bar{\phi}(\boldsymbol{P_3}) - \bar{\phi}(\boldsymbol{P_1})}\right) + \frac{\bar{\phi}(\boldsymbol{P_3})}{\bar{\phi}(\boldsymbol{P_3}) - \bar{\phi}(\boldsymbol{P_1})}x_1 \\
y_4 &= y_3\left(1 - \frac{\bar{\phi}(\boldsymbol{P_3})}{\bar{\phi}(\boldsymbol{P_3}) - \bar{\phi}(\boldsymbol{P_1})}\right) + \frac{\bar{\phi}(\boldsymbol{P_3})}{\bar{\phi}(\boldsymbol{P_3}) - \bar{\phi}(\boldsymbol{P_1})}y_1 \\
x_5 &= x_3\left(1 - \frac{\bar{\phi}(\boldsymbol{P_3})}{\bar{\phi}(\boldsymbol{P_3}) - \bar{\phi}(\boldsymbol{P_2})}\right) + \frac{\bar{\phi}(\boldsymbol{P_3})}{\bar{\phi}(\boldsymbol{P_3}) - \bar{\phi}(\boldsymbol{P_2})}x_2 \\
y_5 &= y_3\left(1 - \frac{\bar{\phi}(\boldsymbol{P_3})}{\bar{\phi}(\boldsymbol{P_3}) - \bar{\phi}(\boldsymbol{P_2})}\right) + \frac{\bar{\phi}(\boldsymbol{P_3})}{\bar{\phi}(\boldsymbol{P_3}) - \bar{\phi}(\boldsymbol{P_2})}y_2.
\end{aligned} \tag{3.41}$$

Podemos obter expressões semelhantes para x_6, y_6, x_7, y_7 utilizando $\phi^* + \eta$ ao invés de $\bar{\phi}$ em (3.41). Usando (3.37) e (3.38), e considerando que ϕ seja positivo no ponto $\boldsymbol{P_3}$ e negativo em $\boldsymbol{P_2}$ e $\boldsymbol{P_1}$, para que as áreas da interface representada por $\phi^* + \eta$ e a da interface representada por $\bar{\phi}$ no triângulo da Figura 3.14 sejam iguais, devemos ter

$$\left|\begin{vmatrix} x_4 & x_5 \\ y_4 & y_5 \end{vmatrix} + \begin{vmatrix} x_5 & x_3 \\ y_5 & y_3 \end{vmatrix} + \begin{vmatrix} x_3 & x_4 \\ y_3 & y_4 \end{vmatrix}\right| - \left|\begin{vmatrix} x_6 & x_7 \\ y_6 & y_7 \end{vmatrix} + \begin{vmatrix} x_7 & x_3 \\ y_7 & y_3 \end{vmatrix} + \begin{vmatrix} x_3 & x_6 \\ y_3 & y_6 \end{vmatrix}\right| = 0 \tag{3.42}$$

Note que a expressão acima nos dá uma equação não linear para η para cada $\bar{\tau}_i$.

Cada uma dessas equações foram resolvidas utilizando o método da posição falsa. Esse método é bastante semelhante ao método da secante, mas possui a vantagem de que, dado o intervalo inicial $\left(\eta^{(0)}, \eta^{(1)}\right)$, com $\eta^{(0)}\eta^{(1)} < 0$, $\eta^{(k)}$ estará no intervalo $\left(\eta^{(0)}, \eta^{(1)}\right)$ na k-ésima iteração para todo $k > 0$.

O método da secante tradicional para resolver a equação $f(x) = 0$ possui a forma

$$x^{(k+2)} = x^{(k+1)} - f\left(x^{(k+1)}\right)\frac{x^{(k+1)} - x^{(k)}}{f\left(x^{(k+1)}\right) - f\left(x^{(k)}\right)}. \tag{3.43}$$

Já o método da posição falsa possui a iteração

$$x^{(k)} = x^{(a)} - f\left(x^{(a)}\right)\frac{x^{(a)} - x^{(b)}}{f\left(x^{(a)}\right) - f\left(x^{(b)}\right)}, \quad a, b < k,\ k > 2, \tag{3.44}$$

onde a é o maior número menor que k tal que $f(x^{(a)}) < 0$, e b é o maior número menor que k tal que $f(x^{(b)}) > 0$. Assim, dados a e b que satisfaçam as condições já ditas, se $f(x^{(k)}) > 0$, deve-se fazer a atualização $b = k$. Caso contrário, faz-se $a = k$.

Assim, dado um triângulo $\bar{\tau}_i$, se escolhermos chutes iniciais para o método da posição falsa tais que a raiz da equação gerada por (3.38) para $\bar{\tau}_i$ esteja em $\left(\eta^{(0)}, \eta^{(1)}\right)$, o método iterativo convergirá para a raíz dessa equação. Isso significa que devemos ter $\Delta A_{\bar{\tau}_i}(\bar{\phi}, \phi^* + \eta^{(0)}) < 0$ e $\Delta A_{\bar{\tau}_i}(\bar{\phi}, \phi^* + \eta^{(1)}) > 0$. Como impusemos $|\phi| \geq 10^{-10}$, esses chutes iniciais são facilmente encontrados:

$$\begin{aligned} \eta^{(0)} &= -\left(\min_{j\in\{1,2,3\}} \phi^*(\boldsymbol{\rho}_{i,j}) + 10^{-11}\right). \\ \eta^{(1)} &= -\left(\max_{j\in\{1,2,3\}} \phi^*(\boldsymbol{\rho}_{i,j}) - 10^{-11}\right) \end{aligned} \tag{3.45}$$

Para a obtenção das expressões acima, note que devemos ter

$$\begin{aligned} &\Delta A_{\bar{\tau}_i}(\bar{\phi}, \phi^* + \eta^{(0)}) < 0 \\ \implies\ & A_{\bar{\tau}_i}(\bar{\phi}) - A_{\bar{\tau}_i}(\phi^* + \eta^{(0)}) < 0 \\ \implies\ & A_{\bar{\tau}_i}(\bar{\phi}) < A_{\bar{\tau}_i}(\phi^* + \eta^{(0)}). \end{aligned} \tag{3.46}$$

Isso significa que a região do triângulo $\bar{\tau}_i$ em que $\phi^* + \eta^{(0)}$ é positivo deve ser maior que a região desse mesmo triângulo em que $\bar{\phi}$ é positivo. Isso pode ser obtido escolhendo-se um $\eta^{(0)}$ que seja positivo tal que $|\eta^{(0)}|$ seja ligeiramente menor que o módulo do menor valor de ϕ^* nos vértices desse triângulo. Desse modo, $\phi^* + \eta^{(0)}$ será negativo em apenas um dos vértices, e a região em que essa função é positiva será grande (quando comparada à área do triângulo). Para garantir que ela seja maior do que a região em que $\bar{\phi}$ é positivo, como impusemos $|\phi| \geq 10^{-10}$, a escolha para $\eta^{(0)}$ mostrada em (3.45) é adequada. Observe que $\eta^{(0)}$ obtida dessa forma será sempre positivo, pois o menor dos valores de ϕ^* é negativo, e em módulo, menor que 10^{-11}. Raciocínio análogo pode ser usado para obter $\eta^{(1)}$, impondo $\Delta A_{\bar{\tau}_i}(\bar{\phi}, \phi^* + \eta^{(1)}) > 0$. A Figura 3.15 ilustra uma possível configuração após as escolhas de $\eta^{(0)}$ e $\eta^{(1)}$.

Como η possui um valor diferente para cada triângulo, e um mesmo vértice pode estar contido em mais de um triângulo, não podemos adicionar η diretamente a ϕ^* na tentativa de obtermos a função $\tilde{d}$. Contudo, para aproveitar os valores de η já calculados em cada um dos triângulos de $\bar{\mathcal{T}}$, introduziremos a função ξ, dada por

$$\xi(\boldsymbol{v}_i) = \frac{\displaystyle\sum_{\tau\in\mathcal{T}_{\boldsymbol{v}_i}\cap\bar{\mathcal{T}}} \eta(\tau)}{\mathrm{tam}(\mathcal{T}_{\boldsymbol{v}_i} \cap \bar{\mathcal{T}})}, \quad \boldsymbol{v}_i \in \bar{\mathcal{V}} \tag{3.47}$$

onde $\mathrm{tam}(\mathcal{T}_{\boldsymbol{v}_i} \cap \bar{\mathcal{T}})$ corresponde à quantidade de triângulos que contém o ponto $\boldsymbol{v}_i$ e nos quais a função $\bar{\phi}$ muda de sinal. Na Figura 3.7, $\mathrm{tam}(\mathcal{T}_{\boldsymbol{v}_i} \cap \bar{\mathcal{T}}) = 3$, pois há três triângulos em azul que contêm o vértice $\boldsymbol{v}_i$. Assim, ξ é uma função definida em todos os vértices adjacentes à interface (ou seja, vértices dos triângulos de $\bar{\mathcal{T}}$) tal que $\xi(\boldsymbol{v}_i)$ é a média dos valores de η em todos os triângulos de $\bar{\mathcal{T}}$ que contêm o vértice $\boldsymbol{v}_i$.

Agora, vamos em busca de uma constante global C tal que

$$\Delta A(\bar{\phi}, \phi^* + \xi C) = \sum_{\bar{\tau}_i\in\bar{\mathcal{T}}} \Delta A(\bar{\phi}, \xi C) = 0. \tag{3.48}$$

Temos, novamente, uma equação não-linear para resolver. É uma equação bastante semelhante a (3.42), com exceção de que as coordenadas dos pontos $\boldsymbol{P_6}$ e $\boldsymbol{P_7}$ são obtidas substituindo em (3.41) $\bar{\phi}$ por $\phi^* + \xi C$. Essa equação foi resolvida pelo método da secante convencional. Os chutes iniciais utilizados foram $C^{(0)} = 0$ e $C^{(1)} = \Delta A(\bar{\phi}, \phi^* + \xi C^{(0)})$.

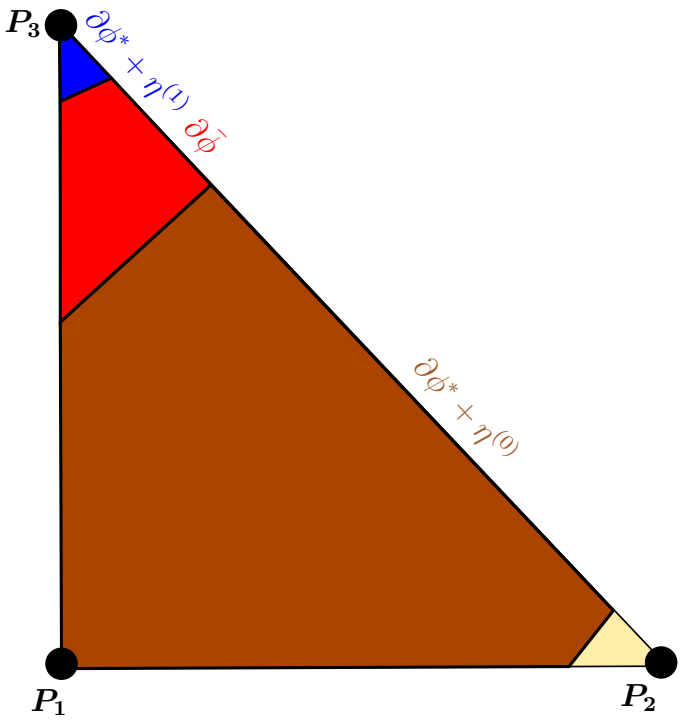


Figura 3.15: Na Figura, $\bar{\phi}(\boldsymbol{P_2}) < \bar{\phi}(\boldsymbol{P_1}) < 0 < \bar{\phi}(\boldsymbol{P_3})$. Em marrom, área interna da interface representada por $\phi^* + \eta^{(0)}$. Em vermelho, área da região interna às interfaces representadas por $\phi^* + \eta^{(0)}$ e por $\bar{\phi}$. Em azul, área da região interna às três interfaces.

Por fim, concluímos a reinicialização de ϕ nos vértices adjacentes à interface através da expressão

$$\tilde{d} = \phi^* + \xi C. \tag{3.49}$$

Note que, calculada dessa maneira, $\tilde{d}$ é linear, pois ϕ^* e ξC também o são.

3.2.3 Reinicialização nos demais Vértices da Malha

Uma vez que os valores de $\tilde{d}$ são conhecidos nos vértices adjacentes à interface, devemos construir uma função distância no restante do domínio. Essa função distância pode ser obtida somente em algumas células vizinhas à interface ou em todo o domínio, conforme a aplicação exigir. Um detalhe interessante é que essa etapa não leva em consideração os valores de ϕ em nenhum dos pontos do domínio distantes da interface: apenas o sinal de ϕ e os valores de $\tilde{d}$ nos vértices adjacentes à interface são utilizados. Para simplificar a explicação, vamos considerar apenas a região Ω^+ do domínio em que $\phi > 0$.

Definindo o conjunto $\mathcal{V}^+$ como o conjunto dos vértices $\boldsymbol{r} \in \mathcal{V} \cup \Omega^+$ e que não são adjacentes à interface, a primeira estimativa para a função distância em $\mathcal{V}^+$ é a única função [4] que satisfaz

$$\tilde{d}(\boldsymbol{r}_i) = \min_{\boldsymbol{v}_i \in \text{adj}(\boldsymbol{r}_i)} (\tilde{d}(\boldsymbol{v}_i) + |\boldsymbol{r}_i - \boldsymbol{v}_i|), \tag{3.50}$$

onde adj$(\boldsymbol{r}_i)$ representa todos os vértices de $\mathcal{V}^+$ que estão conectados a $\boldsymbol{r}_i$ através de alguma aresta de algum triângulo.

Note que a expressão acima ainda não nos dá uma função distância. Para cada ponto $\boldsymbol{r}_i \in \mathcal{V}^+$, ela nos dá o menor caminho ao longo das arestas dos triângulos partindo desse ponto até a interface, conforme mostrado na Figura 3.16). A exceção é o segmento de reta que vai de $\boldsymbol{r}_1$ a P, que pode não seguir as arestas de nenhum triângulo.

O isocontorno $\partial\tilde{d}$ é calculado interpolando-se linearmente $\tilde{d}$ utilizando seus valores nos vértices adjacentes à interface. Portanto, para $\boldsymbol{r}_1$, esse valor já foi calculado por

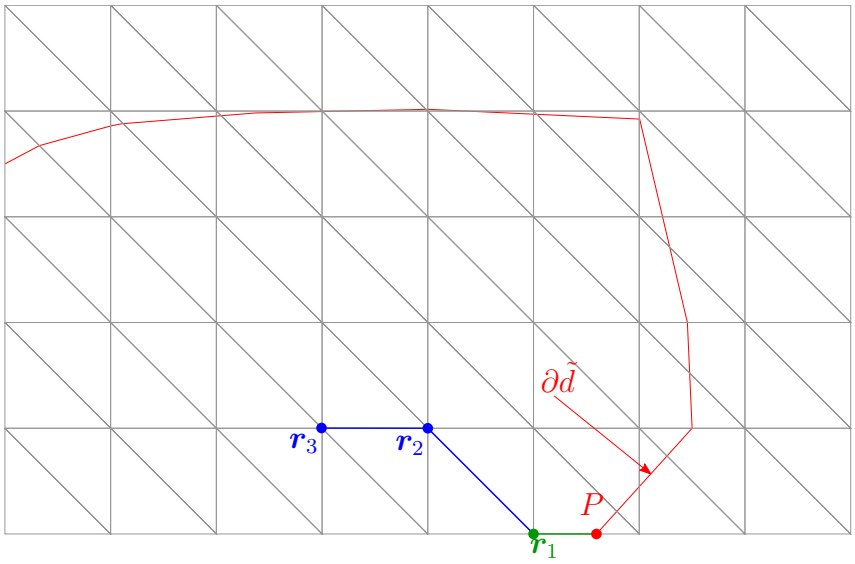


Figura 3.16: $\partial\tilde{d}$ é o isocontorno nulo de $\tilde{d}$, após ser calculada por (3.49). $\boldsymbol{r}_1$ é um vértice adjacente à interface (pois pertence a um triângulo intersectado por $\partial\tilde{d}$), de modo que sua distância até $\partial\tilde{d}$ já foi obtida usando (3.49), e $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ e $\boldsymbol{r}_3$ são vértices de $\mathcal{V}^+$. Em verde é mostrado o segmento de reta de menor comprimento que vai de $\boldsymbol{r}_1$ à interface. O comprimento desse segmento de reta, complementado pelo comprimento do caminho em azul, é o valor de $\tilde{d}(\boldsymbol{r}_3)$ que será obtido por (3.50) quando $\tilde{d}$ deixar de variar.

(3.49). Os valores de $\tilde{d}$ em $\boldsymbol{r}_2$ e $\boldsymbol{r}_3$ que satisfazem (3.50) são $\tilde{d}(\boldsymbol{r}_2) = \tilde{d}(\boldsymbol{r}_1) + |\boldsymbol{r}_2 - \boldsymbol{r}_1|$ e $\tilde{d}(\boldsymbol{r}_3) = \tilde{d}(\boldsymbol{r}_2) + |\boldsymbol{r}_3 - \boldsymbol{r}_2|$, respectivamente. Contudo, $\tilde{d}(\boldsymbol{r}_3)$ não é a ainda a menor distância de $\boldsymbol{r}_3$ a $\partial\tilde{d}$, de modo que (3.50) ainda não nos dá uma função distância. Essa função distância ainda precisará ser obtida a seguir.

Há várias formas de se obter a função que satisfaz (3.50), mas optamos por escolher o mesmo método iterativo utilizado por Ausas *et al.*. Primeiramente, $\tilde{d}$ é inicializada em todos os vértices de $\mathcal{V}^+$ que não sejam adjacentes à interface. O valor com o qual a função será inicializada deve ser ajustado de acordo com o "tamanho" da vizinhança da interface em que desejamos que a reinicialização seja aplicada. Por exemplo, se desejamos reinicializar a função level-set até uma distância de $10\Delta x$ ao redor da interface, inicializamos $\tilde{d}$ com o valor $10\Delta x$ em $\mathcal{V}^+$ e $-10\Delta x$ em $\mathcal{V}^-$. Representaremos esse valor inicial como $\tilde{d}^{(0)}$. Nos vértices adjacentes à interface, a função deve ser inicializada com os valores já calculados por (3.49).

Após a inicialização, para cada vértice $\boldsymbol{r}_i$ de $\mathcal{V}^+$, calculamos

$$\tilde{d}^{(k+1)}(\boldsymbol{r}_i) = \min\left\{\tilde{d}^{(k)}(\boldsymbol{r}_i), \min_{\boldsymbol{r}_j \in \mathcal{V}^+ \cap \mathrm{adj}(\boldsymbol{r}_i)} \left\{\tilde{d}^{(k)}(\boldsymbol{r}_j) + |\boldsymbol{r}_i - \boldsymbol{r}_j|\right\}\right\}. \tag{3.51}$$

A equação acima deve ser aplicada para atualizar $\tilde{d}$ até que seu valor não seja modificado em nenhum dos vértices de $\mathcal{V}^+$. Note que, escolhendo $\tilde{d}^{(0)} = 10\Delta x$ a função $\tilde{d}$ não terá seu valor modificado em nenhum dos vértices cuja distância à interface for maior que $10\Delta x$. Obviamente, quanto maior o valor de $\tilde{d}^{(0)}$, mais vezes (3.50) precisará ser utilizada.

Agora, descrevemos a última etapa da reinicialização, que melhorará a aproximação para $\tilde{d}$ obtida até então. Para cada vértice $\boldsymbol{r}_i$ de $\mathcal{V}^+$, corrigiremos $\tilde{d}$ de modo que $\tilde{d}(\boldsymbol{r}_i)$ será o comprimento do menor caminho partindo de $\boldsymbol{r}_i$ até a interface.

Para cada triângulo em que ϕ não mude de sinal, e para cada vértice $\boldsymbol{r}_i$ desse triângulo (considerando ainda somente os vértices de $\mathcal{V}^+$), definamos $\boldsymbol{F}_i$ como a aresta desse

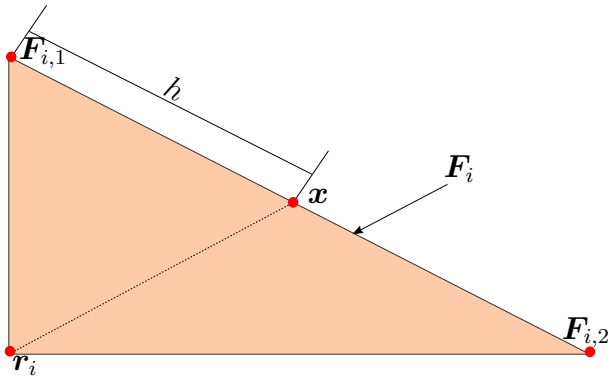


Figura 3.17: $\boldsymbol{F}_i$ é a face oposta ao vértice $\boldsymbol{r}_i$, delimitada pelos vértices $\boldsymbol{F}_{i,1}$ e $\boldsymbol{F}_{i,2}$. $\boldsymbol{x}$ é um ponto qualquer dessa face, distante h do ponto $\boldsymbol{F}_{i,1}$.

triângulo oposta a $\boldsymbol{r}_i$, conforme a Figura 3.17. Então, atualizamos $\tilde{d}$ como

$$\tilde{d}(\boldsymbol{r}_i) = \min\left\{\tilde{d}(\boldsymbol{r}_i), \min_{\boldsymbol{x}\in\boldsymbol{F}_i}\{\tilde{d}(\boldsymbol{x}) + |\boldsymbol{r}_i - \boldsymbol{x}|\}\right\} = \min\{\tilde{d}(\boldsymbol{r}_i), \min_{\boldsymbol{x}\in\boldsymbol{F}_i}\psi(\boldsymbol{x})\}, \tag{3.52}$$

onde $\tilde{d}(\boldsymbol{x})$ é obtida interpolando $\tilde{d}$ linearmente com os valores nos vértices de $\boldsymbol{F}_i$. Essa expressão é muito semelhante a (3.23). De maneira semelhante ao que foi feito em (3.50), essa equação deve ser sucessivamente aplicada até que o valor de $\tilde{d}$ não seja modificado em nenhum dos vértices dos triângulos da malha. Ou seja, na prática, devemos calcular

$$\tilde{d}^{(k+1)}(\boldsymbol{r}_i) = \min\left\{\tilde{d}^{(k)}(\boldsymbol{r}_i), \min_{\boldsymbol{x}\in\boldsymbol{F}_i}\{\tilde{d}^{(k)}(\boldsymbol{x}) + |\boldsymbol{r}_i - \boldsymbol{x}|\}\right\} = \min\{\tilde{d}^{(k)}(\boldsymbol{r}_i), \min_{\boldsymbol{x}\in\boldsymbol{F}_i}\psi^{(k)}(\boldsymbol{x})\}, \tag{3.53}$$

onde $\tilde{d}^{(0)}$ deve ser inicializada com os valores que foram obtidos na última iteração de (3.51).

Para o cálculo de (3.53), veja a Figura 3.17. Interpolando $\tilde{d}^{(k)}$ linearmente usando os valores em $\boldsymbol{F}_{i,1}$ e $\boldsymbol{F}_{i,2}$, obtemos

$$\frac{\tilde{d}^{(k)}(\boldsymbol{F}_{i,2}) - \tilde{d}^{(k)}(\boldsymbol{F}_{i,1})}{|\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}|} = \frac{\tilde{d}^{(k)}(\boldsymbol{x}) - \tilde{d}^{(k)}(\boldsymbol{F}_{i,1})}{h}. \tag{3.54}$$

Novamente, trata-se apenas de uma regra de três simples. Desejamos, então, obter o parâmetro h que minimize $\psi^{(k)}$. Fazendo uma mudança de variáveis de $\tilde{d}^{(k)}(\boldsymbol{x})$ para $\tilde{d}^{(k)}(h)$ na equação acima, chegamos a

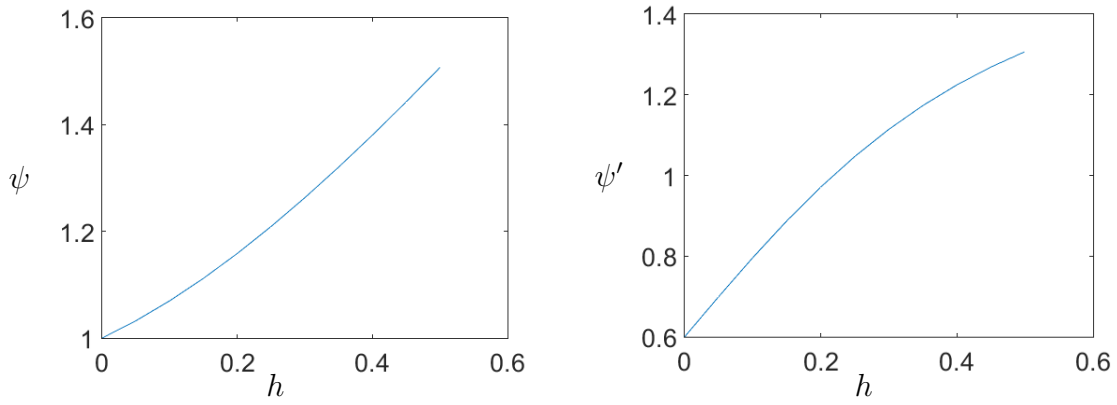
$$\tilde{d}^{(k)}(h) = \tilde{d}^{(k)}(\boldsymbol{F}_{i,1}) + h\frac{\tilde{d}^{(k)}(\boldsymbol{F}_{i,2}) - \tilde{d}^{(k)}(\boldsymbol{F}_{i,1})}{|\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}|}. \tag{3.55}$$

Podemos também escrever

$$\boldsymbol{x} = h\frac{\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}}{|\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}|} + \boldsymbol{F}_{i,1}. \tag{3.56}$$

A equação acima nos diz que, para chegar ao ponto $\boldsymbol{x}$ partindo de $\boldsymbol{F}_{i,1}$, devemos somar um vetor de comprimento h com a direção e o sentido de $\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}$. De (3.56), chegamos a

$$|\boldsymbol{r}_i - \boldsymbol{x}| = \left|h\frac{\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}}{|\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}|} + \boldsymbol{F}_{i,1} - \boldsymbol{r}_i\right|. \tag{3.57}$$



(a) (b)

Figura 3.18: ψ e ψ' em função de h. Nesses gráficos, foram considerados $\tilde{d}(\boldsymbol{F}_{i,2}) = 0.8$, $\tilde{d}(\boldsymbol{F}_{i,1}) = 0.5$, $\boldsymbol{F}_{i,2} = (0, \Delta x)$, $\boldsymbol{F}_{i,1} = (0,0)$ e $\boldsymbol{r}_i = (\Delta x, 0)$, para $\Delta x = 0.5$. Note que ψ' não possui raíz nesse intervalo, de modo que o máximo de ψ' ocorre em um dos extremos do intervalo (nesse caso, no extremo direito).

Assim, podemos reescrever ψ em (3.52) como a função que satisfaz

$$\psi^{(k)}(h) = \tilde{d}^{(k)}(\boldsymbol{F}_{i,1}) + h\frac{\tilde{d}^{(k)}(\boldsymbol{F}_{i,2}) - \tilde{d}^{(k)}(\boldsymbol{F}_{i,1})}{|\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}|} + \left| h\frac{\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}}{|\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}|} + \boldsymbol{F}_{i,1} - \boldsymbol{r}_i \right|. \tag{3.58}$$

Agora, derivemos $\psi^{(k)}$ em relação a h:

$$\psi'^{(k)}(h) = \frac{\tilde{d}^{(k)}(\boldsymbol{F}_{i,2}) - \tilde{d}^{(k)}(\boldsymbol{F}_{i,1})}{|\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}|} + \frac{1}{|\boldsymbol{r}_i - \boldsymbol{x}|}\frac{\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}}{|\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}|} \cdot \left(h\frac{\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}}{|\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}|} + \boldsymbol{F}_{i,1} - \boldsymbol{r}_i \right), \tag{3.59}$$

Note que o termo $|\boldsymbol{r}_i - \boldsymbol{x}|$ depende da incógnita h por (3.57), e foi escrito dessa forma apenas para deixar a notação mais compacta. A fim de encontrar h que minimize (3.58), devemos achar h que anule (3.59). Contudo, veja na Figura 3.17 que h deve estar no intervalo $I = [0, |\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}|]$, para garantir que o ponto $\boldsymbol{x}$ esteja na face $\boldsymbol{F}_i$, e não há garantias de que ψ' se anule nesse intervalo.

De fato, veja na Figura 3.18 um caso em que ψ' não possui raiz no intervalo desejado. Nesse caso, assumimos que a função ψ possui valor mínimo em um dos extremos de I, e os possíveis valores de h que minimizam ψ dentro do triângulo são 0 e $|\boldsymbol{F}_{i,2} - \boldsymbol{F}_{i,1}|$. Para obter o mínimo de ψ, então, basta que esses dois valores sejam testados.

Uma vez que obtivemos h que minimize ψ, podemos calcular $\tilde{d}^{(k)}$ por (3.53). Quando $\tilde{d}^{(k+1)} = \tilde{d}^{(k)}$ para todos os vértices dos triângulos de $\mathcal{T}$, a reinicialização geométrica terminou, e podemos atualizar $\phi(\boldsymbol{v}_i) = \tilde{d}^{(k+1)}(\boldsymbol{v}_i), \forall \boldsymbol{v}_i \in \mathcal{V}$.

3.3 Testes Numéricos

Essa seção tem por objetivo comparar ambas as técnicas de reinicialização apresentadas nesse Capítulo em testes com campo de velocidade prescrito. Aplicações em escoamentos bifásicos são apresentadas na seção 5.

Todas as simulações foram realizadas com uma malha deslocada e quadrada ($h = \Delta x = \Delta y$), como mostra a Figura 2.3. Apesar de as velocidades serem armazenadas nas faces, o processo de advecção mostrado no item 2.4.1 requer que as velocidades e a função ϕ sejam conhecidas no mesmo ponto. Desse modo, as componentes da velocidade

$\mathbf{u} = (u, v)$ foram aproximadas no meio da célula por

$$u_{i,j} = 0.5\left(u_{i+\frac{1}{2},j} + u_{i-\frac{1}{2},j}\right) \quad v_{i,j} = 0.5\left(v_{i,j+\frac{1}{2}} + v_{i,j-\frac{1}{2}}\right). \tag{3.60}$$

Vamos adotar a notação EDP para a reinicialização da Seção 3.1 e GEO para a reinicialização da Seção 3.2.

O passo de tempo utilizado na rotina de reinicialização por EPD foi $\Delta\tau = h/2$, conforme recomendado em [37]. Em todos os testes, optamos por reinicializar a função level-set a cada dez passos de tempo da equação de advecção, em ambas as técnicas de reinicialização.

Um dos parâmetros de verificação que utilizaremos é a perda de massa ocorrida durante a simulação. A área interna da interface é calculada de maneira geométrica, como descrito na seção 3.2.2, para ambas as técnicas de reinicialização. O erro de massa foi calculado através da expressão

$$e_m(\phi^{(0)}, \phi^{(T)}) = \frac{\Delta A(\phi^{(0)}, \phi^{(T)})}{A(\phi^{(0)})}, \tag{3.61}$$

sendo T o instante final da simulação numérica. O erro de ϕ foi calculado como feito em [31]:

$$e_p(\phi^{(0)}, \phi^{(T)}) = \frac{1}{L}\sum_{i=0}^{n}\sum_{j=0}^{m}\left|H(\phi^{(0)}(\mathbf{x}_{i,j})) - H(\phi^{(T)}(\mathbf{x}_{i,j}))\right| h^2, \tag{3.62}$$

onde L é o perímetro esperado da interface. Como nos testes que faremos, devemos ter $\phi^{(T)} = \phi^{(0)}$, a interface esperada no instante $t = T$ coincide com a interface no instante $t = 0$, de modo que podemos usar o perímetro inicial da interface para calcular o erro por (3.62).

3.3.1 Advecção sob um Campo de Velocidade Cisalhante Reversível

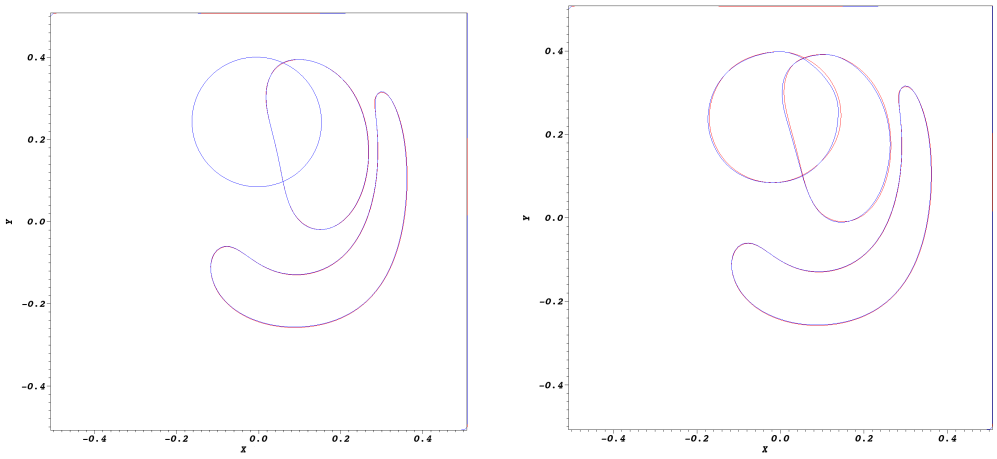
Esse teste tem por objetivo verificar o comportamento da interface quando ela sofre uma grande deformação. O campo de velocidade é dado por

$$\begin{aligned} u &= -sin^2(\pi x)sin(2\pi y)cos(\pi t/T) \\ v &= sin(2\pi x)sin^2(\pi y)cos(\pi t/T) \end{aligned}, \tag{3.63}$$

onde T é o tempo final da simulação, e o domínio de interesse é $\Omega = [0, 1] \times [0, 1]$. A interface é um círculo de raio 0.15π, com centro inicialmente posicionado em $(0.5, 0.75)$.

Nesse teste, o campo de velocidade produz uma severa deformação na interface por um tempo $T/2$, e depois tenta reconstruí-la através de um campo de velocidade com sentido oposto ao original em cada ponto do domínio (Figura 3.19). Analiticamente, a interface deveria retornar exatamente à posição inicial, sem variação de área e com erro 0. Na Figura 3.20, comparamos as reinicializações por EDP e geométrica com o estado inicial do problema. A Tabela 3.1 apresenta uma comparação mais detalhada.

Podemos perceber que a reinicialização geométrica não se sai muito bem quando o passo de tempo é muito pequeno. Esse comportamento da reinicialização geométrica foi ressaltado em [4], e deve ser destacado: ela não pode ser executada com uma frequência muito alta, ou ela trará uma contribuição negativa para a solução do problema. Quanto menor o número CFL utilizado, maior deve ser a quantidade de passos de tempo entre duas aplicações sucessivas da reinicialização geométrica. Não observamos esse comportamento na reinicialização por EDP. Nos casos com número CFL não muito pequeno, ambas as soluções se mostraram bastante semelhantes.



(a) (b)

Figura 3.19: Representação do teste de advecção sob um campo de velocidade cisalhante reversível. A malha utilizada foi de 128×128. Em azul, utilizamos a reinicialização por EDP, e em vermelho a reinicialização geométrica. Em **a)** apresentamos a simulação nos instantes $t = 0$ (círculo perfeito), $t = T/10$ e $T = T/2$ (círculo mais deformado). Em **b)**, apresentamos as simulações nos instantes $t = T/2$, $t = 9T/10$ e $t = T$ (círculo quase perfeito).

		$e_m(\%)$		e_p	
Malha	CFL	EDP	GEO	EDP	GEO
16×16	1.0	40.00	58.41	$3.94E^{-2}$	$4.84E^{-2}$
	0.1	46.61	69.88	$3.38E^{-2}$	$5.05E^{-2}$
	0.01	35.02	80.30	$3.58E^{-2}$	$5.84E^{-2}$
32×32	1.0	17.18	15.93	$2.44E^{-2}$	$2.24E^{-2}$
	0.1	16.16	20.34	$1.44E^{-2}$	$1.54E^{-2}$
	0.01	12.99	29.87	$1.03E^{-2}$	$2.33E^{-2}$
64×64	1.0	5.80	3.47	$1.36E^{-2}$	$9.80E^{-3}$
	0.1	4.80	6.02	$6.29E^{-3}$	$4.88E^{-3}$
	0.01	2.72	11.06	$2.71E^{-3}$	$8.92E^{-3}$
128×128	1.0	2.28	1.02	$7.36E^{-3}$	$4.73E^{-3}$
	0.1	1.69	2.34	$2.76E^{-3}$	$1.80E^{-3}$
	0.01	0.70	4.96	$9.60E^{-4}$	$3.98E^{-3}$
256×256	1.0	1.08	0.38	$3.79E^{-3}$	$2.33E^{-3}$
	0.1	0.74	1.06	$1.35E^{-3}$	$8.15E^{-4}$
	0.01	0.24	2.31	$4.18E^{-4}$	$1.86E^{-3}$

Tabela 3.1: Variação de área e erro após a advecção de ϕ por um campo de velocidade cisalhante reversível.

Além disso, também comparamos o tempo de execução das duas técnicas de reinicialização. As simulações foram executadas num processador Intel Xeon E5-2690 de 2.90GHz, com 32GB de memória RAM disponível, e os resultados são apresentados na Tabela 3.2. Podemos ver que a reinicialização geométrica permitiu reduzir consideravelmente o tempo gasto na reinicialização da função level-set, fazendo inclusive com que a reinicialização se torne computacionalmente menos cara do que o transporte da interface.

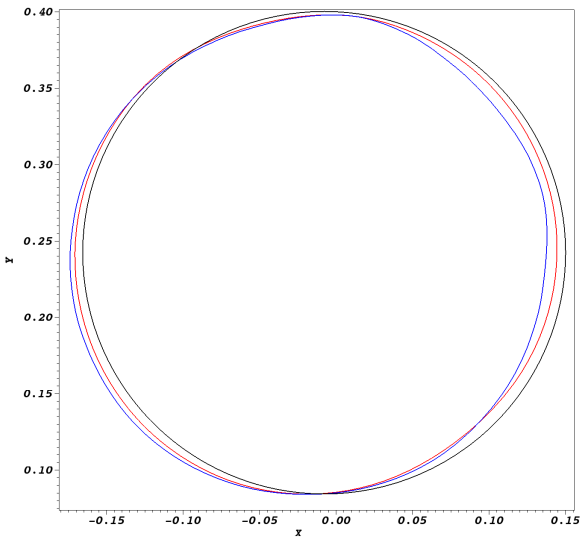


Figura 3.20: Comparação entre os estados iniciais e finais da interface no teste do campo de velocidade cisalhante reversível. A malha utilizada foi de 128×128, em azul, mostramos o resultado final da simulação utilizando a reinicialização por EDP, em vermelho utilizando a reinicialização geométrica, e em preto o estado inicial da simulação.

Malha	Advecção	EPD	GEO
16×16	0.18	0.39	0.22
32×32	1.47	3.26	1.67
64×64	11.7	27.44	11.67
128×128	93.6	232.60	87.48
256×256	730.0	1851.11	649.58

Tabela 3.2: Tempo de execução (em segundos) das rotinas de advecção, reinicialização por EDP e reinicialização geométrica, no teste de advecção por um campo de velocidade cisalhante reversíel. O número CFL utilizado foi 1.0.

3.3.2 Círculo de Zalesak

Esse teste avalia a rotação de um corpo rígido com algumas "quinas", para verificar o quanto da solução original é perdida nessas quinas durante o processo de advecção. O domínio possui dimensões 1×1, e o círculo inicial é centrado no ponto $(0.5, 0.8)$. O campo de velocidade é dado por

$$\begin{aligned} u(x,y) &= \frac{\pi}{3.14}(50 - y) \\ v(x,y) &= \frac{\pi}{3.14}(x - 50), \end{aligned} \tag{3.64}$$

de modo que o círculo completa uma revolução a cada 628 unidades de tempo. A Figura 3.22 exibe o comportamento da interface ao longo do tempo nesse teste. A Figura 3.21 compara o estado final da simulação com o inicial, utilizando as reinicializações por EDP e geométrica. Por fim, a Tabela 3.3 apresenta resultados quantitativos variando a quantidade de células nas malhas.

Podemos constatar nas Figuras 3.22 e 3.21 que a reinicialização geométrica provoca uma maior suavizaçao da solução próximo às "quinas" do círculo do que a reinicialização por EDP. A Tabela 3.3 reforça os resultados observados nas Figuras, pois a reinicialização por EDP apresentou menor erro e menor variação de área.

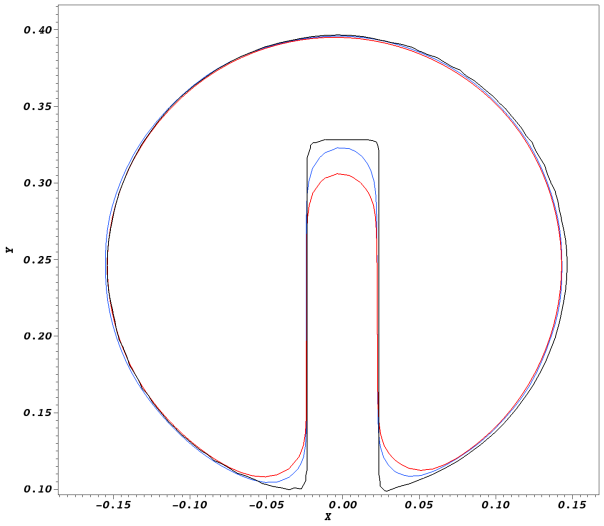


Figura 3.21: Comparação entre os estados iniciais e finais da interface no teste do círculo de Zalesak. A malha utilizada foi de 128×128, em azul, mostramos o resultado final da simulação utilizando a reinicialização por EDP, em vermelho utilizando a reinicialização geométrica, e em preto o estado inicial da simulação.

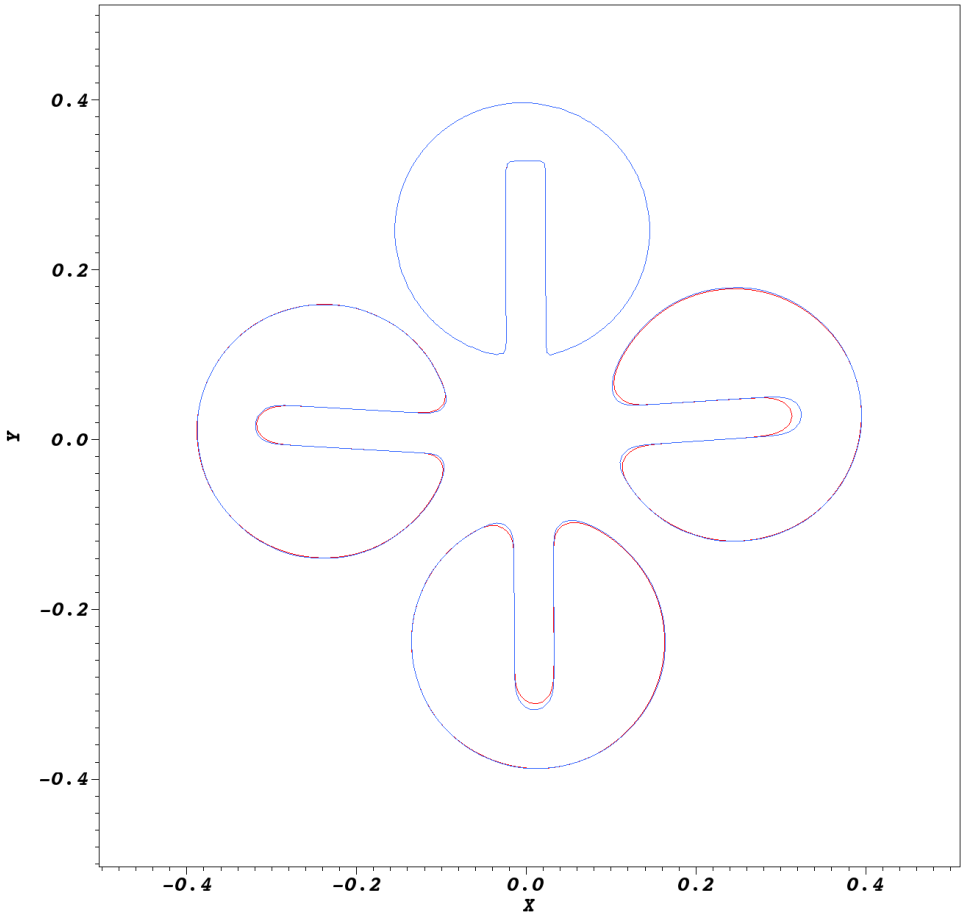


Figura 3.22: Representação do teste do círculo de Zalesak. A malha utilizada foi de 128×128. Em azul, representamos a solução utilizando reinicialização por EDP, e em vermelho utilizando a reinicialização geométrica. A simulação no tempo $t = 0$ é a que se encontra mais acima do domínio. A partir daí, exibimos no sentido antihorário os estados das simulações nos instantes $t = 13T/50$,, $t = 26T/50$ e $t = 39T/50$, respectivamente.

	$e_m(\%)$		e_p	
Malha	EDP	GEO	EDP	GEO
64×64	$8.21E^{-3}$	$8.31E^{-1}$	$7.05E^{-3}$	$1.14E^{-2}$
128×128	$4.76E^{-1}$	1.14	$2.16E^{-3}$	$4.10E^{-3}$
256×256	$2.70E^{-1}$	$8.61E^{-1}$	$1.05E^{-3}$	$1.89E^{-3}$
512×512	$8.56E^{-2}$	$3.77E^{-1}$	$3.28E^{-4}$	$7.67E^{-4}$
1024×1024	$3.80E^{-2}$	$1.81E^{-1}$	$1.99E^{-4}$	$4.14E^{-4}$

Tabela 3.3: Variação de área e erro após a advecção de ϕ por um campo de velocidade cisalhante reversível. O número CFL utilizado foi de 0.67, de modo que, nas malhas mais grossas, o passo temporal foi de 0.010467.

Capítulo

4

Conceitos Básicos em Solução Numérica de Escoamentos Bifásicos

Esse Capítulo tem por objetivo apresentar a modelagem matemática necessária para a simulação numérica de escoamentos multifásicos. Inicialmente, deduzimos as equações de Navier-Stokes, que modelam o comportamento dos fluidos. A seguir, essas equações são ligeiramente modificadas para adicionar o termo referente à tensão superficial nas regiões de interface. Por fim, são apresentados um método de projeção e um método implícito para a solução numérica dessas equações.

4.1 Equações de Navier-Stokes

A derivação das equações de Navier-Stokes baseia-se em três princípios gerais:

- *Hipótese da continuidade* - as propriedades físicas (massa específica e viscosidade) de cada fluido podem ser modeladas por uma função contínua. Essa é uma boa aproximação, desde que não estejamos preocupados com escalas de comprimento muito pequenas;
- *Hipótese da descontinuidade da interface* - na região de interface entre diferentes fluidos, as propriedades físicas são consideradas descontínuas. A interface é, em geral, muito fina, quando comparada com as dimensões envolvidas nos problemas. Sendo assim, a transição entre as propriedades físicas ocorre de maneira muito abrupta, justificando a utilização dessa hipótese;
- *Forças de ação a distância são desconsideradas* - forças como as de origem eletromagnéticas ou gravitacionais não são consideradas. Forças intermoleculares são modeladas através de seu principal efeito: a força devido à tensão superficial, uma força aplicada na região de interface entre os fluidos.

Embora este trabalho lidará apenas com fluidos incompressíveis e newtonianos, as equações de Navier-Stokes serão deduzidas em sua forma geral, e depois particularizadas. Essa seção segue, principalmente, a abordagem feita em [42].

4.1.1 Equação da Continuidade

Para a derivação da equação da continuidade, assume-se que, dentro de um volume V fixo no espaço, não se pode criar ou destruir massa. Dessa forma, a massa m de V só pode variar se houver um fluxo de massa através da fronteira de V.

Chamando de $\boldsymbol{n}$ a normal que aponta para fora de V, o fluxo de massa que deixa V através de sua fronteira é dado por $\rho\boldsymbol{u}\cdot\boldsymbol{n}$, onde ρ é a massa específica de V e $\boldsymbol{u}$ é a velocidade do fluido que passa através da fronteira. Assim, o princípio da conservação de massa pode ser escrito matematicamente como

$$\frac{d}{dt}\int_V \rho\, dv = -\oint_S \rho\boldsymbol{u}\cdot\boldsymbol{n}\, ds, \tag{4.1}$$

sendo S a fronteira de V. A integral de volume à esquerda de 4.1 representa a massa total de V, e a integral à direita representa o fluxo total de massa que deixa V através de suas fronteiras. Assim, a equação acima deve ser interpretada como: a variação da massa em V ao longo do tempo é, em módulo, igual ao fluxo de massa que deixa V através de sua fronteira. Se há massa deixando V na direção de $\boldsymbol{n}$, a derivada do primeiro membro é negativa, enquanto a integral do segundo membro é positiva (o fluxo de massa está deixando V no mesmo sentido da normal $\boldsymbol{n}$, que aponta para fora de V), daí a necessidade do sinal negativo no segundo membro.

Pelo Teorema da divergência, podemos escrever

$$\oint_S \rho\boldsymbol{u}\cdot\boldsymbol{n}ds = \int_V \nabla\cdot(\rho\boldsymbol{u})\, dv. \tag{4.2}$$

Além disso, como o volume V é fixo, podemos passar a derivada no primeiro membro de (4.1) para dentro da integral, chegando a

$$\int_V \left[\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\boldsymbol{u})\right] dv = 0. \tag{4.3}$$

Tratando V como um volume infinitesimal, a integral acima só pode ser nula se todo o integrando também o for. Logo, deduzimos que

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\boldsymbol{u}) = 0. \tag{4.4}$$

Definindo a derivada material como

$$\frac{D()}{Dt} = \frac{\partial()}{\partial t} + \boldsymbol{u}\cdot\nabla() \tag{4.5}$$

e expandindo $\nabla\cdot(\rho\boldsymbol{u}) = \boldsymbol{u}\cdot\nabla\rho + \rho\nabla\cdot\boldsymbol{u}$, chegamos, de (4.4), à equação da continuidade em sua forma mais geral:

$$\frac{D\rho}{Dt} = -\rho\nabla\cdot\boldsymbol{u}. \tag{4.6}$$

4.1.2 Conservação da Quantidade de Movimento

A equação da quantidade de movimento é derivada assumindo que a taxa de variação da quantidade de movimento do fluido num volume fixo V é a diferença entre o fluxo de

quantidade de movimento que deixa V pela sua fronteira S e a força resultante que age em V. Matematicamente,

$$\frac{d}{dt}\int_V \rho\boldsymbol{u}\ dv = -\oint_S \rho\boldsymbol{u}(\boldsymbol{u}\cdot\boldsymbol{n})\ dS + \int_V \boldsymbol{f}\ dv + \oint_S \boldsymbol{n}\cdot\boldsymbol{T}\ dS. \tag{4.7}$$

Na equação acima, a integral do primeiro membro representa a quantidade de movimento total do fluido em V, a primeira integral de superfície do segundo membro é o fluxo de quantidade de movimento que deixa V através de S (o sinal de negativo é justificado da mesma forma que em (4.1)), seguido da força resultante aplicada em V ($\mathbf{f}$) e da força total devido à tensão de cisalhamento aplicada em V. $\boldsymbol{T}$ é um tensor simétrico construído de tal forma que $\boldsymbol{n}\cdot\boldsymbol{T}$ é a força devido à tensão de cisalhamento aplicada no elemento dS que tenha normal $\boldsymbol{n}$.

Pelo Teorema da Divergência, e usando o mesmo argumento que na dedução de (4.4), chegamos a

$$\frac{\partial\boldsymbol{u}}{\partial t} = -\nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}) + \boldsymbol{f} + \nabla\cdot\boldsymbol{T}, \tag{4.8}$$

sendo $\boldsymbol{u}\boldsymbol{u}$ um tensor tal que $(\boldsymbol{u}\boldsymbol{u})_{i,j} = u_i u_j$. Usando o fato de que $\nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}) = \rho\boldsymbol{u}\cdot\nabla\boldsymbol{u} + \boldsymbol{u}\cdot\nabla\rho\boldsymbol{u}$ e a equação da continuidade (4.6), chegamos a

$$\rho\frac{D\boldsymbol{u}}{Dt} = \boldsymbol{f} + \nabla\cdot\boldsymbol{T}. \tag{4.9}$$

4.1.3 Fluidos Newtonianos

Se considerarmos que o fluido é newtoniano, o tensor $\boldsymbol{T}$ pode ser escrito como

$$\boldsymbol{T} = (-p + \lambda\nabla\cdot\boldsymbol{u})\boldsymbol{I} + 2\mu\boldsymbol{S}, \tag{4.10}$$

sendo $\boldsymbol{I}$ o tensor unitário, p a pressão, μ a viscosidade, λ o segundo coeficiente de viscosidade e $\boldsymbol{S} = \frac{1}{2}(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)$ o tensor deformação, cujas componentes são

$$S_{i,j} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right). \tag{4.11}$$

Se assumirmos a hipótese de Stokes, podemos considerar $\lambda = -(2/3)\mu$. Substituindo essas expressões em (4.9), chegamos a

$$\rho\frac{D\boldsymbol{u}}{Dt} = \boldsymbol{f} - \nabla p + \nabla\cdot\left(\mu\left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\right)\right) + \nabla\left(-\frac{2}{3}\mu\nabla\cdot\boldsymbol{u}\right), \tag{4.12}$$

que é a equação de conservação da quantidade de movimento para fluidos newtonianos.

4.1.4 Escoamentos Incompressíveis

Escoamentos incompressíveis são aqueles em que a massa específica de cada uma de suas partículas permanece constante conforme elas se movimentam. Matematicamente,

$$\frac{D\rho}{Dt} = 0. \tag{4.13}$$

Assim, a equação da continuidade (4.6) pode ser simplificada para

$$\nabla\cdot\boldsymbol{u} = 0. \tag{4.14}$$

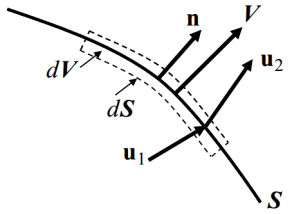


Figura 4.1: Um fino elemento de volume $d\boldsymbol{V}$, com fronteiras dS contendo uma interface S. Se $d\boldsymbol{V}$ for suficientemente fino, não pode haver acúmulo de massa em seu interior. Imagem retirada de [42].

Observe que não estamos, ainda, impondo que a massa específica seja constante ao longo de todo o domínio (e nem constante numa mesma célula com o passar do tempo): a condição é que a massa específica de uma mesma partícula não seja alterada ao longo do tempo, conforme ela se movimenta ao longo do escoamento.

Em escoamentos incompressíveis, a equação da conservação da quantidade de movimento pode ser simplificada para

$$\frac{D\boldsymbol{u}}{Dt} = \frac{1}{\rho}\left(-\nabla p + \boldsymbol{f} + \nabla \cdot \left(\mu\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right)\right)\right). \tag{4.15}$$

Note que o termo $\frac{2}{3}\mu\nabla \cdot \boldsymbol{u}$ de (4.12) é zero devido à equação da continuidade.

4.2 Escoamentos Multifásicos

Para que possamos utilizar as equações de Navier-Stokes ao longo de todo o domínio, precisamos estabelecer condições de contorno na interface para a equação da continudade, e adicionar uma força extra na equação da quantidade de movimento: a força devido à tensão superficial.

A fim de melhorar os resultados numéricos, optamos por suavizar as propriedades físicas (massa específica e viscosidade) nos pontos próximos à interface. Dessa forma, para todo ponto $\boldsymbol{x}$ do domínio, essas propriedades devem ser calculadas atravéz das expressões a seguir:

$$\rho(\boldsymbol{x}) = \rho_0 + (\rho_1 - \rho_0)H(\phi(\boldsymbol{x})) \tag{4.16}$$
$$\mu(\boldsymbol{x}) = \mu_0 + (\mu_1 - \mu_0)H(\phi(\boldsymbol{x})). \tag{4.17}$$

Na equação acima, o índice 1 corresponde às propriedades físicas do fluido que se encontra na região interior à interface, enquanto que o índice 0 corresponde à região exterior a ela. H é a função degrau, que foi suavizada nesse trabalho como a seguir

$$H(\phi(\boldsymbol{x})) = \begin{cases} 1 & \phi(\boldsymbol{x}) > \alpha \\ 0 & \phi(\boldsymbol{x}) < -\alpha \\ 0.5\left(1 + \frac{\phi(\boldsymbol{x})}{\alpha} + \sin\left(\frac{\pi\phi(\boldsymbol{x})}{\alpha}\right)\right) & |\phi(\boldsymbol{x})| \leq \alpha, \end{cases} \tag{4.18}$$

sendo que α representa a espessura da interface. Utilizamos $\alpha = 0.75\,(\Delta x + \Delta y)$, conforme feito em [26].

4.2.1 Condição de Contorno para a Equação da Continuidade

Veja a Figura 4.1. Se as velocidades nas duas fases do fluido, próximas à interface, são $\boldsymbol{u}_1$ e $\boldsymbol{u}_2$, e a velocidade da interface na direção normal é $\boldsymbol{V}$, os fluxos de massa que

entram e saem de $d\boldsymbol{V}$ são, respectivamente, $\rho_1(\boldsymbol{u}_1 \cdot \boldsymbol{n} - \boldsymbol{V})$ e $\rho_2(\boldsymbol{u}_2 \cdot \boldsymbol{n} - \boldsymbol{V})$. Como o elemento de volume tem espessura infinitesimal, não pode haver acúmulo de massa em $d\boldsymbol{V}$, de modo que devemos ter

$$\rho_1(\boldsymbol{u}_1 \cdot \boldsymbol{n} - \boldsymbol{V}) = \rho_2(\boldsymbol{u}_2 \cdot \boldsymbol{n} - \boldsymbol{V}).$$

Pensando num caso geral de escoamento multifásico, a condição acima deve ser válida para qualquer par de massa específicas ρ_1, ρ_2, razão pela qual podemos escrever

$$\boldsymbol{u}_1 \cdot \boldsymbol{n} = \boldsymbol{u}_2 \cdot \boldsymbol{n}.$$

Experimentalmente, observa-se que, na maioria dos casos, deve-se utilizar condições de contorno do tipo *no-slip* para a componente tangencial da velocidade. Portanto, a condição de contorno para a velocidade na interface é

$$\boldsymbol{u}_1 = \boldsymbol{u}_2. \tag{4.19}$$

4.2.2 A Tensão Superficial

Para a resolução da equação da conservação da quantidade de movimento na interface, devemos levar em conta os efeitos da força devido à tensão superficial, denotada nesse trabalho por $\boldsymbol{f}_\sigma$. Há dois métodos principais para se modelar essa força: a SSF (*Sharp Surface Tension Force*) e a CSF(*Continuous Surface Tension Force*). Detalhamos esses dois métodos a seguir.

4.2.2.1 *Continuous Surface Tension Force* - CSF

O método CSF para o cálculo da força devido à tensão superficial foi introduzido em 1992 por Brackbill *et al.*, e foi utilizado por muitos trabalhos desde então [25, 39, 41]. Essa seção foi escrita tendo como base o artigo de Brackbill *et al.*, junto com o livro de Tryggvason *et al.* [42].

Através de uma visão puramente mecânica, a tensão superficial é uma força por unidade de área que atua perpendicularmente a cada ponto da interface. Sendo assim, se $\boldsymbol{p}$ é um vetor perpendicular a um segmento de reta da interface, conforme a Figura 4.2, temos que a força aplicada nesse segmento é $\sigma\boldsymbol{p}$ por unidade de comprimento. O coeficiente σ é comumente chamado de *coeficiente de tensão superficial*.

Considerando um pequeno elemento de superfície S, com contornos ∂S, usemos a relação $\sigma\boldsymbol{p} = \sigma\boldsymbol{I}_s \cdot \boldsymbol{p}$, onde $\boldsymbol{I}_s = \boldsymbol{I} - \boldsymbol{nn}$ é a tangente interface. Daí, podemos obter a força devido à tensão superficial total em S através da expressão a seguir:

$$\boldsymbol{f}_\sigma = \oint_{\partial S} \sigma\boldsymbol{I}_s \cdot \boldsymbol{p} dl = \int_S \nabla_S \cdot \sigma\boldsymbol{I}_s dS, \tag{4.20}$$

onde ∇_s é o gradiente de superfície (componente do gradiente tangencial à superfície) e dl é um elemento de ∂S. Na equação acima, a última igualdade é derivada do Teorema da Divergência. No limite, quando a interface tende a um ponto, a expessão acima se torna

$$\boldsymbol{f}_\sigma = \nabla_S \cdot \sigma\boldsymbol{I}_s = \sigma\nabla_S \cdot \boldsymbol{I}_s + \boldsymbol{I}_s \cdot \nabla_S\sigma.$$

O termo $\boldsymbol{I}_s \cdot \nabla_s\sigma$ se anula para um coeficiente de tensão superficial constante, e pode-se mostrar que $\sigma\nabla_S \cdot \boldsymbol{I}_s = \sigma\kappa\boldsymbol{n}$ [42]. Portanto,

$$\boldsymbol{f}_\sigma = \sigma\kappa\boldsymbol{n}. \tag{4.21}$$

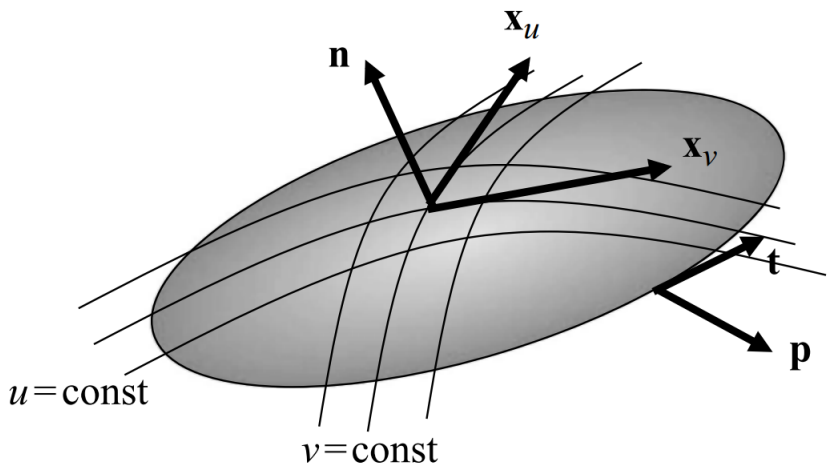


Figura 4.2: O vetor $\boldsymbol{p}$ é perpendicular ao contorno da interface, mas é diferente do vetor $\boldsymbol{n}$ [42].

Considerando um elemento de volume V, que contenha uma interface S, a tensão superficial $\boldsymbol{f}_{\sigma,V}$ dentro desse elemento de volume é calculada somando-se a força $\boldsymbol{f}_\sigma$ ao longo de S. Matematicamente,

$$\boldsymbol{f}_{\sigma,V} = \int_S \sigma\kappa\boldsymbol{n}ds. \tag{4.22}$$

Para que essa força possa ser adicionada à equação da conservação da quantidade de movimento, precisamos que a integral seja feita sobre V, e não sobre S. Isso pode ser feito utilizando-se a função $\delta_S(\boldsymbol{x})$, definida como

$$\delta_S(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} \text{ pertence à interface S} \\ 0, & \boldsymbol{x} \text{ não percente à interface S} \end{cases}. \tag{4.23}$$

Assim, (4.22) se torna

$$\boldsymbol{f}_{\sigma,V} = \int_V \sigma\kappa\boldsymbol{n}\delta_S dv. \tag{4.24}$$

Seja $\tilde{c}(\boldsymbol{x})$ uma função tal que

$$\tilde{c}(\boldsymbol{x}) = \begin{cases} c_0, & \boldsymbol{x} \in \Omega_1 \\ c_1, & \boldsymbol{x} \in \Omega_2 \end{cases}, \tag{4.25}$$

com c_0 e c_1 reais. Assumimos que $\tilde{c}$ apresenta uma transição suave entre c_0 e c_1 na interface. Brackbill *et al.* [7] mostrou que

$$\sigma\kappa\boldsymbol{n}\delta_S = \sigma\kappa\frac{\nabla\tilde{c}}{[\tilde{c}]}, \tag{4.26}$$

onde $[\tilde{c}] = c_0 - c_1$. Se utilizamos $\tilde{c} = H(\phi)$ [1], note que $c_0 = 1$ e $c_1 = 0$, logo $[\tilde{c}] = 1$.

Agora, adicionando a força $\boldsymbol{f}_{\sigma,V}$ à equação da conservação da quantidade de movimento (4.7), e utilizando os mesmos argumentos detalhados na seção 4.1.2, a equação da conservação da quantidade de movimento para fluidos newtonianos incompressíveis se torna

$$\frac{D\boldsymbol{u}}{Dt} = \frac{1}{\rho}\left(-\nabla p + \boldsymbol{f}_e + \sigma\kappa\boldsymbol{n}\nabla H(\phi) + \nabla\cdot\mu(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)\right), \tag{4.27}$$

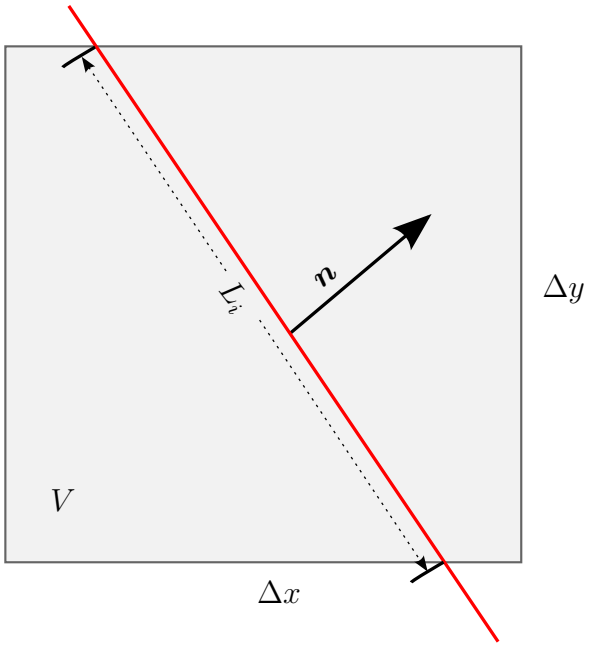


Figura 4.3: Volume de controle V, com dimensões $\Delta x \times \Delta y$. A interface é representada pela linha vermelha. O comprimento da parte da interface interna a V é L_i, e $\boldsymbol{n}$ é um vetor unitário normal à interface.

onde f_e representam as forças externas que atuam no escoamento, como a gravidade. Note que num elemento de volume que não contenha nenhuma parte da interface, $H(\phi)$ é constante, de modo que $\nabla H(\phi)$ é zero, fazendo com que essa equação se reduza novamente a (4.15).

4.2.2.2 *Sharp Surface Tension Force* - SSF

O método SSF recebe esse nome porque, diferentemente do CSF, a força devido à tensão superficial é tratada de maneira descontínua, sendo aplicada exatamente sobre o contorno da interface. Detalhes sobre sua implementação podem ser encontrados nos trabalhos de Francois *et al.* [12] e Pourmousa *et al.* [29].

Nesse método, a força $\boldsymbol{f}_\sigma$ aplicada num volume de controle V é calculada através da expressão

$$\boldsymbol{f}_\sigma = \frac{\sigma \kappa L_i \boldsymbol{n}}{\Delta x \Delta y}, \tag{4.28}$$

onde κ é a curvatura da interface no volume de controle, $\boldsymbol{n} = (n_x, n_y)$ é a normal unitária, L_i é o comprimento da interface que passa por V (ver Figura 4.3) e $\Delta x, \Delta y$, são o comprimento de V nas direções x e y, respectivamente.

É comum aproximar o produto $L_i n_x$ por Δy, e $L_i n_y$ por Δx, como feito em [29]. Através dessas aproximações, podemos simplificar (4.28) da forma a seguir

$$\begin{aligned} f_{\sigma_x} &= \frac{\sigma \kappa}{\Delta x} \\ f_{\sigma_y} &= \frac{\sigma \kappa}{\Delta y}. \end{aligned} \tag{4.29}$$

Francois *et al.* [12] detalhou como adicionar a tensão superficial pelo método SSF à equação da conservação da quantidade de movimento. Diferentemente do método CSF, no método SSF essa adição só deve ser feita nas células em que ϕ muda de sinal. Como utilizamos uma malha deslocada (ver Figura 2.3b), a equação da conservação da quantidade

de movimento é discretizada nas faces das células. Isso significa que a tensão superficial também deve ser calculada nas faces das células. Assim, adicionamos (4.29) à equação da conservação da quantidade de movimento na face $(i-\frac{1}{2},j)$ se $\phi_{i,j}\phi_{i-1,j}<0$ (fazemos analogamente na direção y).

Além disso, a curvatura também precisa ser aproximada nas faces da célula. Essa aproximação é feita através de uma média ponderada das curvaturas nas células vizinhas a essa face, dando um peso maior para a célula cujo centro estiver mais próximo do contorno da interface. Isso é feito pelas expressões a seguir:

$$\kappa_{i-\frac{1}{2},j}=\frac{\kappa_{i,j}|\phi_{i-1,j}|+\kappa_{i-1,j}|\phi_{i,j}|}{|\phi_{i,j}|+|\phi_{i-1,j}|} \tag{4.30}$$

$$\kappa_{i,j-\frac{1}{2}}=\frac{\kappa_{i,j}|\phi_{i,j-1}|+\kappa_{i,j-1}|\phi_{i,j}|}{|\phi_{i,j}|+|\phi_{i,j-1}|}. \tag{4.31}$$

4.3 Solução Numérica das Equações de Navier-Stokes

4.3.1 Discretização Temporal e Espacial

Para a integração temporal, foi utilizado um método implícito, conforme abaixo

$$\rho\left(\frac{\boldsymbol{u}^{n+1}-\boldsymbol{u}^{n}}{\Delta t}+\boldsymbol{A}^{n}\right)=\boldsymbol{D}^{n+1}+\boldsymbol{f}^{n}-\nabla p^{n+1}. \tag{4.32}$$

Na equação acima, $\boldsymbol{A}=\boldsymbol{u}\cdot\nabla\boldsymbol{u}$, $\boldsymbol{D}=\nabla\cdot\left(\mu\left(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^{T}\right)\right)$ e $\boldsymbol{f}$ representa as forças como as geradas pela gravidade e pela tensão superficial. No caso bidimensional, essa equação vetorial pode ser dividida em duas outras equações, para as componentes x e y, como a seguir:

$$\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right)=-\frac{\partial p}{\partial x}+f_x+\frac{\partial}{\partial x}\left(\mu\left(\frac{\partial u}{\partial x}+\frac{\partial u}{\partial y}\right)\right) \tag{4.33}$$

$$\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right)=-\frac{\partial p}{\partial y}+f_y+\frac{\partial}{\partial y}\left(\mu\left(\frac{\partial v}{\partial x}+\frac{\partial v}{\partial y}\right)\right), \tag{4.34}$$

sendo u e v as componentes do vetor velocidade $\boldsymbol{u}=(u,v)$ e f_x e f_y as componentes do vetor $\boldsymbol{f}$.

O termo convectivo $\boldsymbol{A}$ foi discretizado com um esquema CUBISTA [2], e para a discretização dos demais termos com derivadas utilizamos diferenças centradas.

Pelas expressões (4.16) e (4.17), μ e ρ dependem de ϕ. Contudo, como pode ser visto na Figura 2.3, os valores de ϕ estão armazenados no centro das células. Assim, para as expressões (4.16) e (4.17) apenas nos permitem calcular os valores de μ e ρ nos centros das células. Para obtermos seus valores nas faces das células, utilizamos as expressões

$$\mu_{i+\frac{1}{2},j}=\frac{\mu_{i+1,j}+\mu_{i,j}}{2} \tag{4.35}$$

$$\rho_{i+\frac{1}{2},j}=\frac{\rho_{i+1,j}+\rho_{i,j}}{2}. \tag{4.36}$$

4.3.2 Método de Projeção

Para a resolução das equações da conservação da quantidade de movimento (4.33) e (4.34), juntamente com a equação da continuidade (4.14) em variáveis primitivas (u, v e p), há duas maneiras distintas. Ou calculamos essas três variáveis de maneira simultânea,

resolvendo um único sistema não-linear [30], ou de maneira segregada [13]. Nesse trabalho, utilizaremos um método de projeção, que está dentro da classe dos métodos segregados.

Métodos de projeção, como o introduzido por Chorin [9], são formas de resolver numericamente as equações de Navier-Stokes em variáveis primitivas. Primeiramente, definimos $\tilde{\boldsymbol{u}}^{n+1}$ de forma que satisfaça a equação da conservação da quantidade de movimento (mas não necessariamente satisfaça a conservação de massa)

$$\rho\left(\frac{\tilde{\boldsymbol{u}}^{n+1}-\boldsymbol{u}^{n}}{\Delta t}+(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n}\right)=-\nabla p^{n}+\nabla\cdot\left(\mu\left(\nabla\tilde{\boldsymbol{u}}^{n+1}+(\nabla\tilde{\boldsymbol{u}}^{n+1})^{T}\right)\right)+\boldsymbol{f}^{n}. \tag{4.37}$$

Pelo Teorema da Decomposição de Helmholtz [6], podemos escrever

$$\tilde{\boldsymbol{u}}^{n+1}=\boldsymbol{u}^{n+1}+\frac{1}{\rho}\nabla\psi^{n+1}\implies\boldsymbol{u}^{n+1}=\tilde{\boldsymbol{u}}^{n+1}-\frac{1}{\rho}\nabla\psi^{n+1}. \tag{4.38}$$

onde ψ é uma função potencial escalar. Aplicando o operador divergente em (4.38) e usando (4.14),

$$\nabla\cdot\left(\frac{1}{\rho}\nabla\psi^{n+1}\right)=\nabla\cdot\tilde{\boldsymbol{u}}^{n+1}. \tag{4.39}$$

Subtraindo (4.37) de (4.32),

$$\rho\left(\frac{\boldsymbol{u}^{n+1}-\tilde{\boldsymbol{u}}^{n+1}}{\Delta t}\right)=-\nabla(p^{n+1}-p^{n})+\nabla\cdot\left(\mu\left(\nabla(\boldsymbol{u}^{n+1}-\tilde{\boldsymbol{u}}^{n+1})+\left(\nabla(\boldsymbol{u}^{n+1}-\tilde{\boldsymbol{u}}^{n+1})\right)^{T}\right)\right). \tag{4.40}$$

Substituindo (4.38) em (4.40),

$$-\frac{\nabla\psi^{n+1}}{\Delta t}=-\nabla(p^{n+1}-p^{n})+\nabla\cdot\left(\mu\left(\nabla\left(-\frac{1}{\rho}\nabla\psi^{n+1}\right)+\left(\nabla\left(-\frac{1}{\rho}\nabla\psi^{n+1}\right)\right)^{T}\right)\right).. \tag{4.41}$$

Agora, simplificamos a equação acima assumindo que μ e ρ são constantes [44], chegando a

$$-\frac{\nabla\psi^{n+1}}{\Delta t}=-\nabla(p^{n+1}-p^{n})+\frac{\mu}{\rho}\left(\nabla\left(\nabla^{2}\psi^{n+1}\right)+\nabla^{T}\left(\nabla^{2}\psi^{n+1}\right)\right). \tag{4.42}$$

Como ∇ e ∇^{T} operam sobre escalares, ambos os operadores são iguais. Logo, chegamos a

$$\nabla\left(-\frac{\psi^{n+1}}{\Delta t}+p^{n+1}-p^{n}+2\frac{\mu}{\rho}\nabla^{2}\psi^{n+1}\right)=0. \tag{4.43}$$

Para o gradiente de uma função se anular, essa função precisa ser constante. Logo,

$$-\frac{\psi^{n+1}}{\Delta t}+p^{n+1}-p^{n}+2\frac{\mu}{\rho}\nabla^{2}\psi^{n+1}=p_{0} \tag{4.44}$$

$$p^{n+1}=p^{n}+\frac{\psi^{n+1}}{\Delta t}-2\frac{\mu}{\rho}\nabla^{2}\psi^{n+1}+p_{0}. \tag{4.45}$$

Como a pressão é definida em função de uma pressão de referência, podemos assumir $p_0=0$. Além disso, para simplificar o cálculo da pressão, ignoramos o termo $\nabla^2\psi^{n+1}$, o que introduz um erro de ordem $O(\Delta t)$ no cálculo da pressão.

Assim, o método de projeção que utilizamos consiste dos passos a seguir:

1. Calcular $\tilde{\boldsymbol{u}}^{n+1}$ através de (4.37);
2. Calcular ψ^{n+1} através de (4.39)
3. Calcular p^{n+1} através de (4.45)

Um fluxograma de uma simulação completa, desde a inicialização de ϕ até a solução das Equações de Navier-Stokes, é apresentado na Figura B.1.

4.3.3 Condições de Contorno

Todos os testes que faremos nesse trabalho são de escoamentos confinados, sem a presença de *inflow* e de *outflow*, de modo que utilizamos condições *no-slip* em todas as fronteiras do domínio. Além disso, as paredes do domínio são consideradas impermeáveis. Isso implica nas condições

$$\begin{array}{rl} \boldsymbol{u} = \mathbf{0}, & \text{em } (4.37) \\ \dfrac{\partial \psi}{\partial \boldsymbol{n}} = 0 & \text{em } (4.39) \end{array}.$$

Capítulo 5

Aplicações em Escoamentos Bifásicos

Esse Capítulo visa aplicar as técnicas de reinicialização descritas no Capítulo 3 a problemas de escoamentos bifásicos. Utilizamos a mesma malha e a mesma frequência de reinicialização que foram descritos na Seção 3.3.

Com exceção do teste das correntes parasitas, o passo de tempo Δt utilizado é calculado dinamicamente. Isso é necessário, visto que no transporte da interface, a discretização temporal é feita sempre de maneira explícita. Desse modo, o passo de tempo precisa ser recalculado de forma a respeitar a condição CFL. O número CFL que utilizamos foi de 0.35.

5.1 Correntes Parasitas

Esse teste consiste de uma bolha imersa num fluido de mesma massa específica e viscosidade. Quando os efeitos da gravidade são desprezados, caso o fluido se encontre em repouso no início da simulação, ele deverá continuar em repouso até o término da mesma. Entretanto, aproximações ruins da curvatura e inconsistências na discretização da força devido à tensão superficial e do gradiente de pressão podem levar ao surgimento de um campo de velocidade falso (as chamadas *correntes parasitas*) [22] (ver Figura 5.1).

O domínio utilizado possui dimensões 1×1 (metros), a bolha possui diâmetro inicial de 0.5m e é posicionada no centro do domínio, e o coeficiente de tensão superficial utilizado é $\sigma = 0.357N/m$. Nas duas fases do fluido, a viscosidade utilizada foi de $\mu = 1Pa.s$, e a massa específica $\rho = 4kg/m^3$. O passo de tempo é $\Delta t = 10^{-5}s$, e a simulação foi realizada até o tempo $T = 200\Delta t$. O número adimensional relevante para esse teste é o número de Ohnesorge, dado por

$$Oh = \mu/\sqrt{\sigma\rho r} = 1.6737,$$

em que r é o raio da bolha.

A validação do teste foi feita comparando a norma $l_{2,1}$ da velocidade com as apresentadas em [17]. Essa norma foi calculada como

$$\|\boldsymbol{u}\|_{2,1} = h\sum_{i=1}^{N}\left(\sum_{j=1}^{N}\|\boldsymbol{u}_{i,j}\|_2^2\right)^{1/2}, \tag{5.1}$$

onde N é a quantidade de células do domínio em cada dimensão (o domínio é quadrado) e $\boldsymbol{u}_{i,j}$ é o vetor velocidade na célula (i,j).

Podemos, também, considerar um vetor $\bar{\boldsymbol{u}}$ em que cada componente $\bar{u}_i$ corresponde à norma da velocidade em uma determinada célula do domínio. Nesse caso, podemos

utilizar a norma do máximo no vetor $\bar{\boldsymbol{u}}$. Como esse teste apresenta pouca (idealmente nenhuma) movimentação da interface, não foi feita reinicialização da função level-set.

Malha	Curvatura	Tensão Superficial	$\|\boldsymbol{u}\|_{2,1}$	Ordem	$\|\bar{\boldsymbol{u}}\|_{\infty}$	Ordem
32×32	LS1	CSF	$4.66E-05$	-	$6.14E-05$	-
64×64	LS1	CSF	$1.86E-05$	1.32	$2.28E-05$	1.43
128×128	LS1	CSF	$6.86E-06$	1.44	$6.51E-06$	1.81
256×256	LS1	CSF	$2.46E-06$	1.48	$1.71E-06$	1.93
32×32	LS2	CSF	$6.49E-05$	-	$4.36E-05$	-
64×64	LS2	CSF	$2.54E-05$	1.36	$1.84E-05$	1.24
128×128	LS2	CSF	$9.40E-06$	1.43	$5.61E-06$	1.71
256×256	LS2	CSF	$3.46E-06$	1.43	$1.61E-06$	1.80
32×32	LS3	CSF	$6.53E-05$	-	$4.37E-05$	-
64×64	LS3	CSF	$2.54E-05$	1.36	$1.84E-05$	1.43
128×128	LS3	CSF	$9.40E-06$	1.43	$5.61E-06$	1.72
256×256	LS3	CSF	$3.49E-06$	1.43	$1.61E-06$	1.80
32×32	LS1	SSF	$1.70E-05$	-	$1.63E-05$	-
64×64	LS1	SSF	$6.50E-06$	1.39	$4.40E-06$	1.89
128×128	LS1	SSF	$2.16E-06$	1.59	$1.06E-06$	2.05
256×256	LS1	SSF	$8.20E-07$	1.40	$2.36E-07$	2.17
32×32	LS2	SSF	$7.08E-05$	-	$5.00E-05$	-
64×64	LS2	SSF	$2.40E-05$	1.56	$1.18E-05$	2.08
128×128	LS2	SSF	$8.55E-06$	1.49	$2.95E-06$	2.00
256×256	LS2	SSF	$2.95E-06$	1.53	$7.13E-07$	2.05
32×32	LS3	SSF	$7.13E-05$	-	$5.02E-05$	-
64×64	LS3	SSF	$2.41E-05$	1.57	$1.18E-05$	2.09
128×128	LS3	SSF	$8.55E-06$	1.49	$2.95E-06$	2.00
256×256	LS3	SSF	$2.95E-06$	1.53	$7.13E-07$	2.05

Tabela 5.1: Simulação para o teste das correntes parasitas, para diversas combinações de malhas e métodos para o cálculo da curvatura e da tensão superficial. Nesses testes não foi feita reinicialização da função level-set.

Na Figura 5.2a, apresentamos a variação de $\|\bar{\boldsymbol{u}}\|_{\infty}$ em função do tempo. Podemos ver que com as combinações SSF e LS1, CSF e LS2, e CSF e LS3, o máximo da velocidade tornou-se praticamente constante depois de algum tempo. A combinação SSF e LS1 foi a que estacionou com menor valor. Na Figura 5.2b, exibimos a variação da norma $\|\boldsymbol{u}\|_{2,1}$ em função do tempo. Nesse gráfico podemos ver que, em todas as simulações, o campo de velocidade se torna mais intenso com o passar do tempo, utilizando essa norma.

Podemos, ainda, ver na Tabela 5.1 que as curvaturas LS2 e LS3 apresentam resultados muito parecidos, e que a LS1 foi superior em todos os testes. Além disso, o método SSF produziu correntes parasitas de menor magnitude em todos os testes, quando comparado com o CSF.

5.2 Oscilação da bolha

Esse é um teste bastante semelhante ao anterior, com a diferença de que os fluidos possuem viscosidades e massas específicas diferentes, além do fato de a bolha ser inicializada com o formato de elipse. Com o passar do tempo, a bolha tem sua forma modificada de maneira periódica, conforme mostra a Figura 5.3.

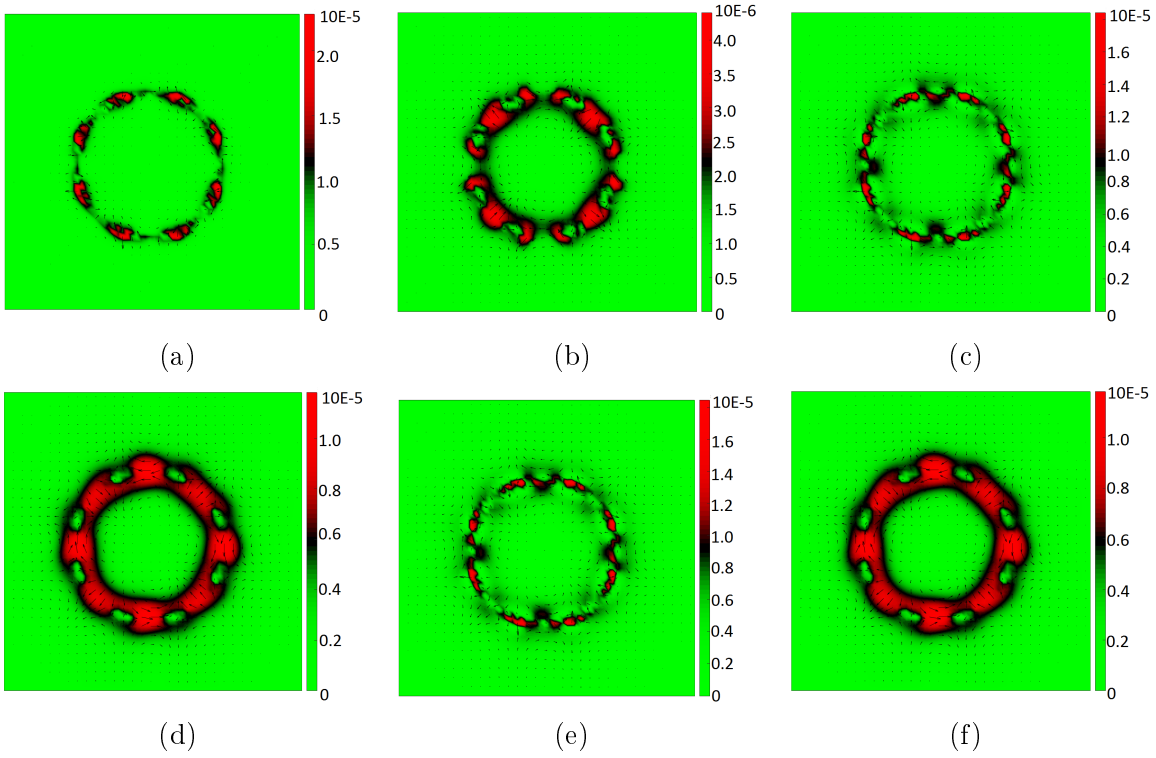


Figura 5.1: Campo de velocidade (em m/s) do teste da gota estática, após duzentos passos de tempo, com malha de 64×64. **a)** LS1 e CSF; **b)** LS1 e SSF; **c)** LS2 e CSF; **d)** LS2 e SSF; **e)** LS3 e CSF; **f)** LS3 e SSF;

A bolha é representada pela elipse de equação

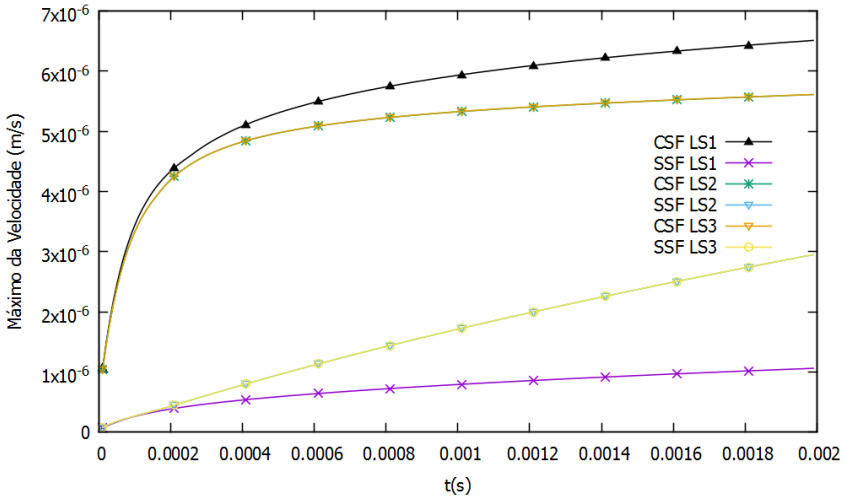
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1,$$

com $a = 0.02856$ e $b = 0.01782$, conforme utilizado em [14]. A bolha é de etanol, enquanto que o restande do domínio está cheio de ar. Os parâmetros utilizados são: $\mu_1 = 1.2E10^{-3} Pa.s, \rho_1 = 787.88 kg/m^3$ para o etanol, e $\mu_0 = 1E10^{-5}, \rho_0 = 1.1768 kg/m^3$ para o ar. Além disso, temos $\sigma = 0.02361 N/m$.

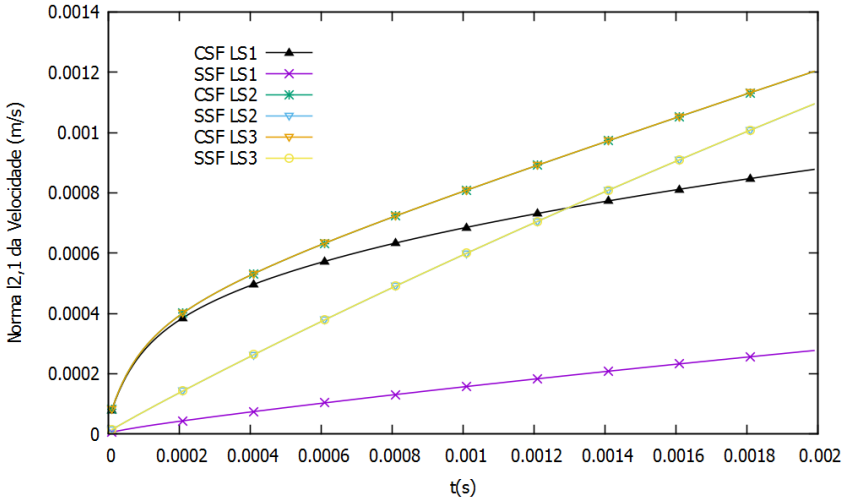
Utilizamos apenas a curvatura LS1, pois foi a que apresentou melhores resultados no teste das correntes parasitas. Os resultados são apresentados na Tabela 5.2, onde podemos ver que os resultados obtidos nesse trabalho ficaram em boa concordância com a solução de referência. O período $T_i, i = 0, 1, \ldots, 6$, corresponde à i-ésima oscilação da bolha numa mesma simulação numérica. Esses períodos foram obtidos numericamente analisando os instantes em que a componente x da velocidade da célula de interface mais à direita tem seu sinal invertido. Em [14] o valor pseudoanalítico para o período de oscilação da gota é de 1.588.

Na Figura 5.4, pode-se ver a variação da coordenada x dessa célula ao longo do tempo, e na Figura 5.5, a variação da velocidade da célula computacional de interface mais à direita do domínio (essa velocidade é ligeiramente diferente da velocidade com que a interface está se movimentando).

Pode-se ver que, em geral, os resultados estão próximos à solução pseudoanalítica, mesmo nos casos em que a reinicialização de ϕ não é feita. Ambas as técnicas de reinicialização trouxeram resultados bastante próximos. Também não observamos diferenças significativas entre as duas técnicas para o cálculo da tensão superficial.



(a)



(b)

Figura 5.2: Variação das normas $\|\bar{\boldsymbol{u}}\|_\infty$ (**a**) e $\|\boldsymbol{u}\|_{2,1}$ (**b**) em função do tempo, no teste das correntes parasitas. Utilizamos uma malha de 128×128, sem reinicialização da função level-set.

5.3 Ascensão da Bolha

Nesse teste, o objetivo é estudar o comportamento de uma bolha imersa em um fluido mais denso. O domínio possui dimensões 1×2 (metros), e a simulação é realizada até o tempo $T = 3s$. Inicialmente, a bolha possui raio $r_0 = 0.25m$ e é centrada no ponto $(0.5, 0.5)$. Denotaremos como fluido 1 a bolha, e como 0 o fluido que a envolve. Os dados utilizados nos dois testes simulados podem ser vistos na Tabela 5.3, e a Figura 5.6 exibe a evolução da interface ao longo do tempo.

Nessa tabela, utilizamos duas constantes adimensionais, que auxiliam a caracterizar as simulações numéricas. As adimensionalizações foram feitas considerando-se como comprimento característico do domínio $L = 2r$, sendo r o raio da bolha. O tempo foi adimensionalizado utilizando-se L/U_g, onde $U_g = \sqrt{2gr_0}$ é a velocidade gravitacional. Dados esses parâmetros, os números adimensionais podem ser calculados através das expressões abaixo [15].

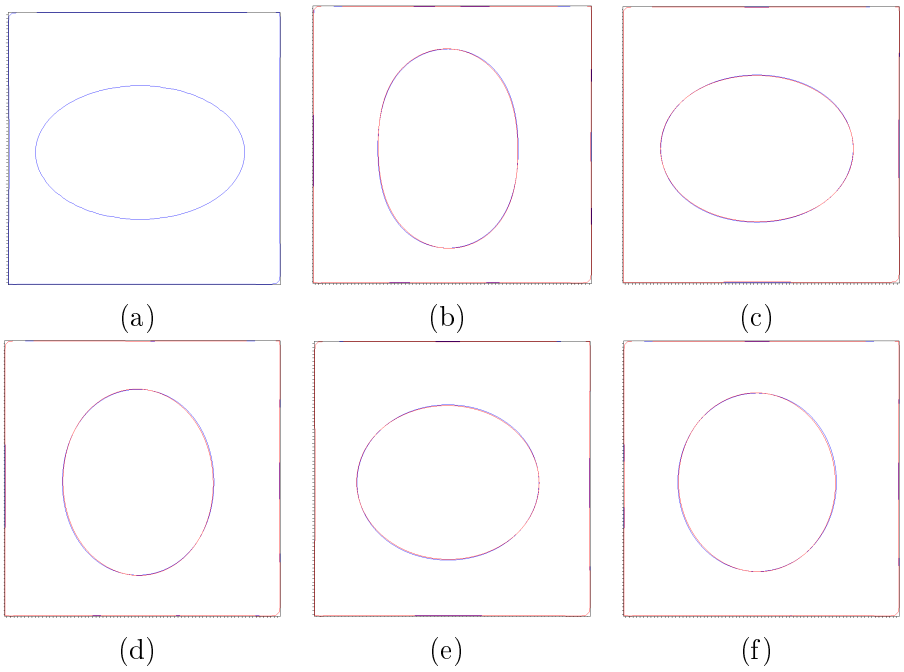
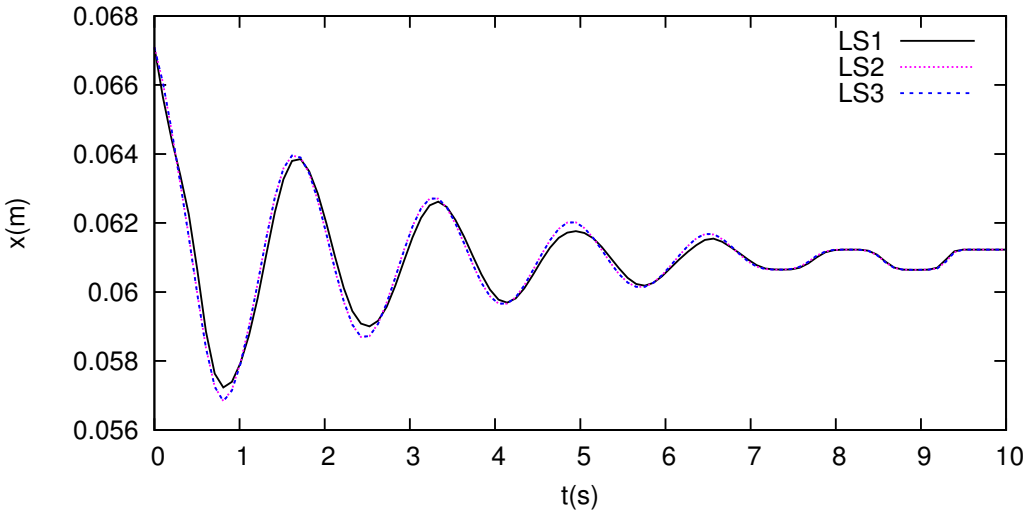


Figura 5.3: Evolução do teste da oscilação da gota, numa malha de 64×64, com curvatura LS1. Em azul está a solução com reinicialização por EDP, e em vermelho com reinicialização geométrica. De cima para baixo, da esquerda para a direita, as imagens são referentes aos instantes $t = iT/50$, $i = 0, \ldots, 5$, com $T = 10s$.

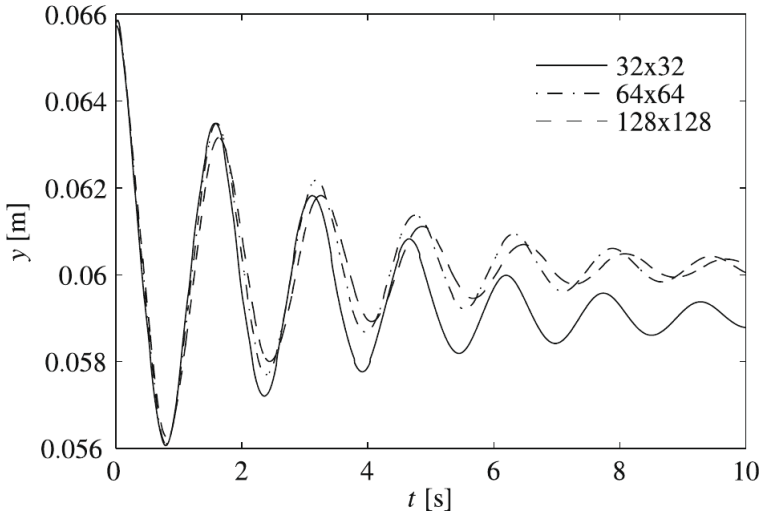
Malha	Tensão Sup.	Reinicialização	T1	T2	T3	T4	T5	T6
64×64	CSF	-	1.685	1.644	1.603	1.591	1.574	1.579
64×64	CSF	EDP	1.657	1.629	1.609	1.604	1.606	1.603
64×64	CSF	GEO	1.603	1.588	1.562	1.548	1.564	1.597
128×128	CSF	-	1.671	1.656	1.621	1.606	1.592	-
128×128	CSF	EDP	1.661	1.635	1.610	1.606	1.603	1.606
128×128	CSF	GEO	1.604	1.574	1.559	1.560	1.541	1.604
64×64	SSF	-	1.663	1.657	1.614	1.585	-	-
64×64	SSF	EDP	1.641	1.629	1.612	1.613	1.607	1.609
64×64	SSF	GEO	1.617	1.600	1.583	1.574	1.566	1.556
128×128	SSF	-	1.656	1.661	1.622	1.599	1.547	-
128×128	SSF	EDP	1.642	1.631	1.612	1.608	1.604	1.607
128×128	SSF	GEO	1.614	1.606	1.581	1.573	1.578	1.584
Solução de referência [14]								
64×64	-	-	-	-	-	-	-	1.56
128×128	-	-	-	-	-	-	-	1.60

Tabela 5.2: Resultados para o teste da oscilação da bolha, para as malhas de 64×64 e 128×128, diferentes tipos de curvatura, variando a técnica de reinicialização utilizada e calculando a tensão superficial pelo método SSF. Períodos representados por - não puderam ser detectados porque a oscilação se tornou muito pequena.

$$Re = \frac{\rho_0 U_g L}{\mu_0} \quad Eo = \frac{\rho_0 U_g^2 L}{\sigma}$$



(a)



(b)

Figura 5.4: Nesse gráfico é exibida a coordenada x da célula de interface mais à direita do domínio (a que possui a coordenada x com maior valor), a cada passo de tempo da simulação; **a)** Simulação feita numa malha 128×128 e sem reinicialização da função level-set; **b)** Solução de referência de [40] para três malhas diferentes.

Re é o número de Reynolds, que descreve a razão entre as forças inerciais e as viscosas, enquanto *Eo* é o número de Eötvös, que mede a razão entre as forças gravitacionais e a as forças provocadas pela tensão superficial.

Testes	$\rho_0(Kg/m^3)$	$\rho_1(kg/m^3)$	$\nu_0(Pa.s)$	$\nu_1(Pa.s)$	$g(m/s^2)$	$\sigma(N/m)$	Re	Eo
Teste 1	1000	100	10	1	0.98	24.5	35	10
Teste 2	1000	1	10	0.1	0.98	1.96	35	125

Tabela 5.3: Parâmetros de entrada para o teste da ascensão da bolha.

Várias simulações foram feitas para esse caso, variando o espaçamento da malha h e a técnica de reinicialização da função level-set. Como parâmetros quantitativos, compararemos com o artigo de referência [15] a circularidade da interface, a velocidade média da região interior da interface e a posição de seu centroide. Em todas elas, utilizamos a

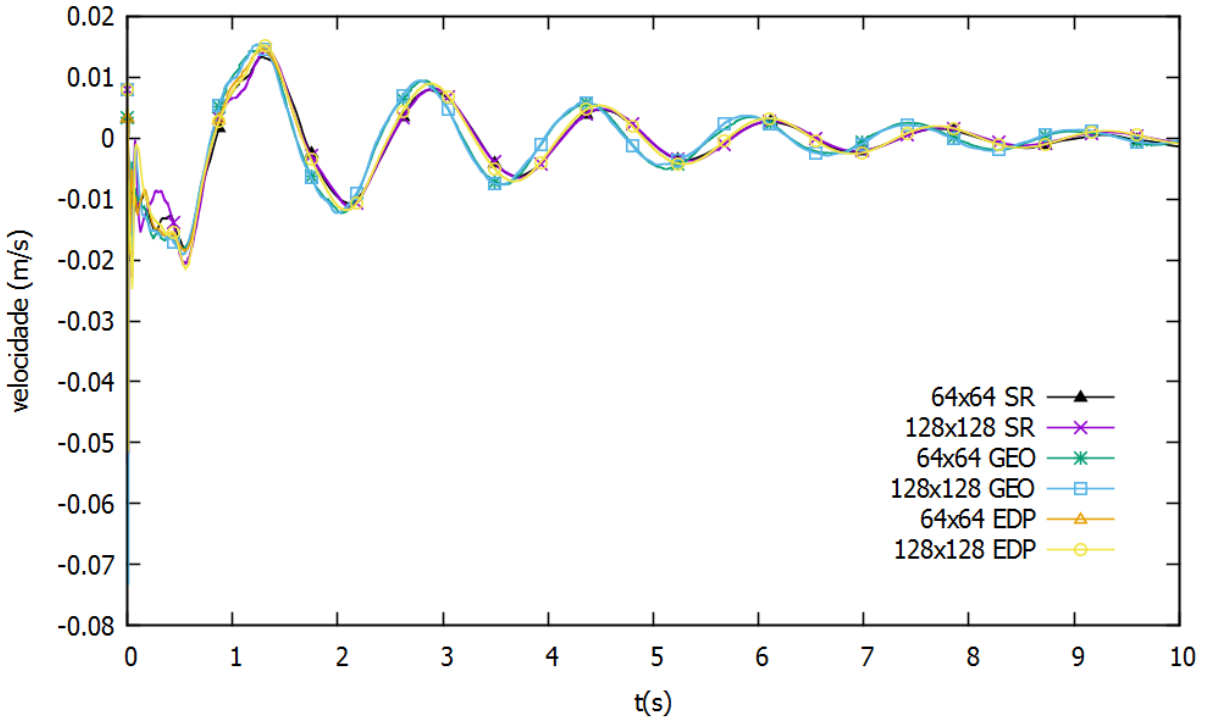


Figura 5.5: Componente x da velocidade da célula computacional de interface mais à direita do domínio para diferentes malhas, e variando a técnica de reinicialização empregada. SR significa Sem Reinicialização.

curvatura LS1, e o cálculo da tensão superficial foi feito com o método SSF, pois foram as escolhas que apresentaram melhores resultados nos testes anteriores.

A circularidade é a razão entre o perímetro de um círculo que possui a mesma área que a interface e o perímetro da interface. O perímetro da interface foi calculado usando

$$P_\Omega \approx \sum_i \sum_j \delta(\phi_{i,j}) |\nabla \phi_{i,j}| h^2, \tag{5.2}$$

sendo δ a função Delta de Dirac, H a função degrau e Ω a região ocupada pela bolha. A área interna à interface foi calculada de maneira geométrica, conforme descrito na Seção 3.2, e será denotada por S_Ω. Se considerarmos um círculo que possua área $S_c = S_\Omega$ e perímetro P_c, podemos dizer que

$$S_\Omega = S_c = \pi r^2 \implies r = \pm\sqrt{\frac{S_\Omega}{\pi}}. \tag{5.3}$$

Daí, adotando o valor positivo de r, temos

$$P_c = 2\pi r = 2\pi\sqrt{\frac{S_\Omega}{\pi}} = 2\sqrt{S_\Omega \pi}. \tag{5.4}$$

Por fim, a circularidade C_Ω da interface pode ser obtida como

$$C_\Omega = \frac{P_c}{P_\Omega} = \frac{2\sqrt{S_\Omega \pi}}{P_\Omega}. \tag{5.5}$$

A velocidade média da bolha foi calculada como em [15], através da expressão

$$\boldsymbol{u}_\Omega = (u_\Omega, v_\Omega) = \frac{\int_\Omega \boldsymbol{u} d\boldsymbol{x}}{\int_\Omega 1 d\boldsymbol{x}} \approx \frac{\sum_i \sum_j \boldsymbol{u}_{i,j} H(\phi_{i,j})}{\sum_i \sum_j H(\phi_{i,j})}. \tag{5.6}$$

Também em [15] foi sugerida a expressão a seguir para o cálculo da centroide da região ocupada pela bolha:

$$\boldsymbol{x}_\Omega = (x_\Omega, y_\Omega) = \frac{\int_\Omega \boldsymbol{x} d\boldsymbol{x}}{\int_\Omega 1 d\boldsymbol{x}} \approx \frac{\sum_i \sum_j \boldsymbol{x}_{i,j} H(\phi_{i,j})}{\sum_i \sum_j H(\phi_{i,j})}. \tag{5.7}$$

As Figuras 5.7 e 5.8 exibes a evolução dessas grandezas ao longo do tempo os testes 1 e 2, respectivamente. Note que, no Teste 2, a circularidade da bolha foi reduzida consideravelmente, devido à grande deformação sofrida por ela. As tabelas 5.4 e 5.5 comparam quantitativamente os resultados obtidos nesse trabalho com os de [15].

Em ambos os testes, pode-se ver que o ganho com a utilização da reinicialização é significativo para o cálculo correto da circularidade. Além disso, pouca diferença foi observada nos resultados obtidos comparando as duas técnicas de reinicialização.

$1/h$	Rein.	$C_{\Omega_{\min}}$	t_1	$\boldsymbol{u}_{\Omega_{\max}}$	t_2	$y_\Omega^{(t=3)}$
40	-	0.7389	2.3814	0.2466	1.0042	1.0517
40	EDP	0.8964	1.9008	0.2447	0.9683	1.0805
40	GEO	0.9052	1.8972	0.2445	0.9683	1.0873
40	[15]	**0.9060**	**1.8375**	**0.2427**	**0.9000**	**1.0715**
80	-	0.6565	2.9151	0.2452	0.9332	1.0536
80	EDP	0.8963	1.9147	0.2434	0.9383	1.0774
80	GEO	0.9032	1.9489	0.2441	0.9383	1.0849
80	[15]	**0.9021**	**1.9125**	**0.2410**	**0.9375**	**1.0817**
160	-	0.6675	2.8010	0.2447	0.9280	1.0657
160	EDP	0.8959	1.9142	0.2425	0.9235	1.0763
160	GEO	0.9030	1.8766	0.2434	0.9298	1.0851
160	[15]	**0.9011**	**1.8750**	**0.2421**	**0.9313**	**1.0799**

Tabela 5.4: Análise quantitativa do teste ascensão da bolha (teste 1), comparadas com a solução de referência de [15] (em negrito). t_1 se refere ao tempo em que a circularidade atinge seu valor mínimo, e t_2 é o tempo em que a velocidade média da bolha atinge seu valor máximo.

Também comparamos os tempos de execução das duas técnicas de reinicialização com o tempo gasto na resolução das equações de Navier-Stokes. O tempo gasto na solução das equações de Navier-Stokes considera apenas a implementação computacional do método de projeção que foi descrito neste trabalho. Essa comparação é importante para sabermos a contribuição das rotinas de reinicialização no tempo total da simulação, quando comparado com o método de projeção.

As simulações foram executadas num processador Intel Xeon E5-2690 de 2.90GHz, com 32GB de memória RAM disponível, e os resultados são apresentados na Tabela 5.6. Nesse teste, ambas as reinicializações apresentaram tempos muito próximos, diferente do que aconteceu no teste da Seção 3.3.1. Essa diferença pode ser explicada pelo fato de que o tempo de execução da reinicialização geométrica é totalmente dependente da taxa de convergência dos métodos iterativos que foram empregados. Essa convergência pode ser mais ou menos rápida, dependendo do quanto ϕ deixa de ser uma função distância durante o seu transporte. Isso faz com que seja difícil prever qual das duas técnicas de reinicialização terá o menor custo computacional sem que isso seja testado para cada aplicação em particular.

$1/h$	Rein.	$C_{\Omega_{\min}}$	t_1	$\boldsymbol{u}_{\Omega_{\max}}$	t_2	$y_{\Omega}^{(t=3)}$
40	-	0.5067	2.6261	0.2529	0.7591	1.1122
40	EDP	0.5708	2.9609	0.2539	0.7591	1.0991
40	GEO	0.5946	2.9537	0.2531	0.7535	1.1163
40	[15]	**0.4868**	**2.7500**	**0.2563**	**0.7750**	**1.0843**
80	-	0.4609	2.7785	0.2520	0.7396	1.1186
80	EDP	0.5093	2.9993	0.2522	0.7368	1.1095
80	GEO	0.5232	2.9951	0.2524	0.7410	1.1196
80	[15]	**0.5071**	**2.8438**	**0.2518**	**0.7188**	**1.1099**
160	-	0.4216	2.3948	0.2512	0.7337	1.1299
160	EDP	0.4983	2.9998	0.2513	0.7348	1.1199
160	GEO	0.5047	2.9998	0.2517	0.7369	1.1258
160	[15]	**0.4647**	**3.0000**	**0.2514**	**0.7281**	**1.1249**

Tabela 5.5: Análise quantitativa do teste ascensão da bolha (teste 2), comparadas com a solução de referência de [15] (em negrito). t_1 se refere ao tempo em que a circularidade atinge seu valor mínimo, e t_2 é o tempo em que a velocidade média da bolha atinge seu valor máximo.

# Teste	$1/h$	Advecção	EDP	GEO	Navier-Stokes
1	40	19	49	44	209
1	80	180	559	486	4768
1	160	2262	6676	6578	118328
2	40	10	32	23	140
2	80	162	496	436	5552
2	160	3368	9492	10079	296214

Tabela 5.6: Tempo de execução (em segundos) das técnicas de reinicialização por EDP e geométrica, da advecção da interface e do método de projeção para a solução das equações de Navier-Stokes.

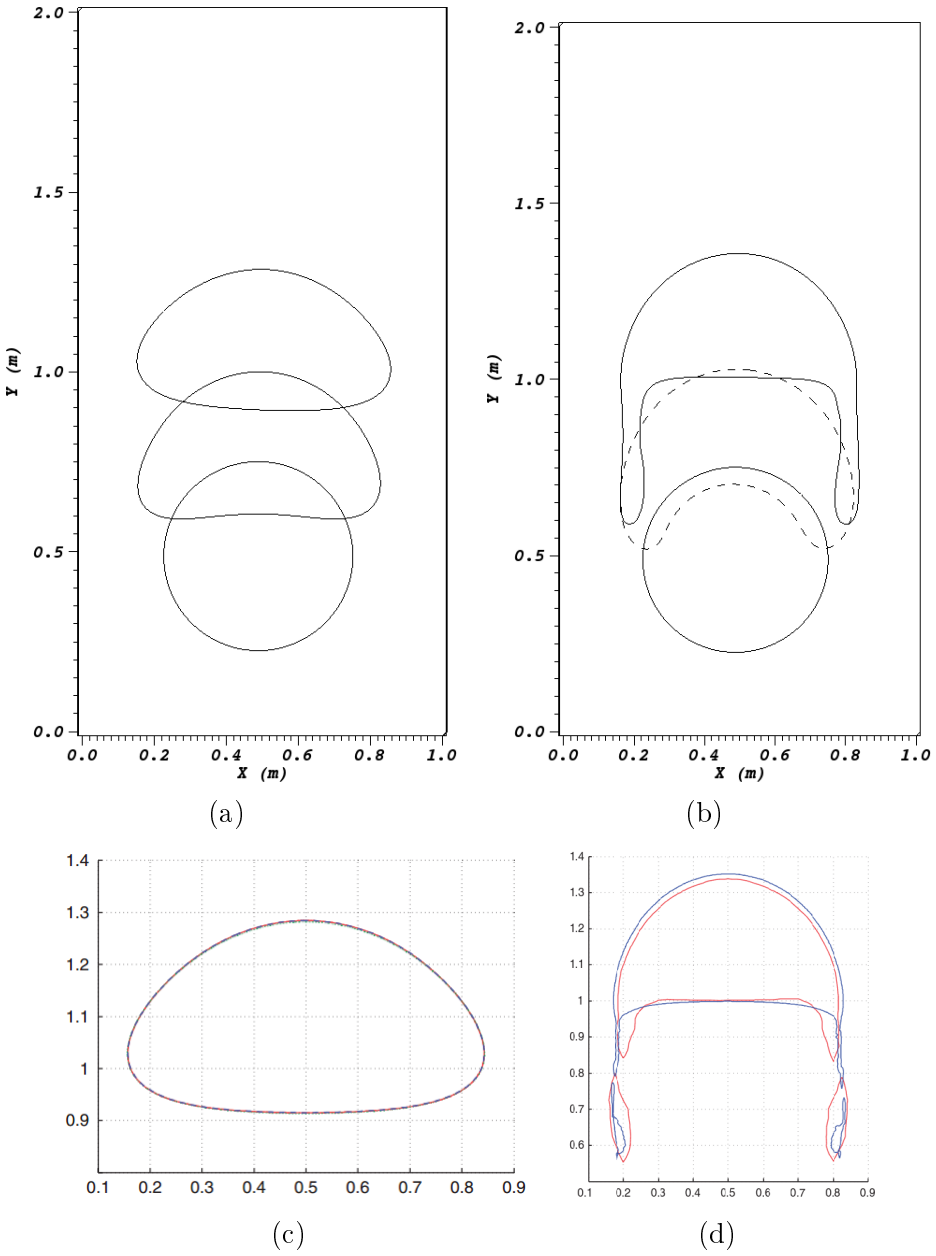
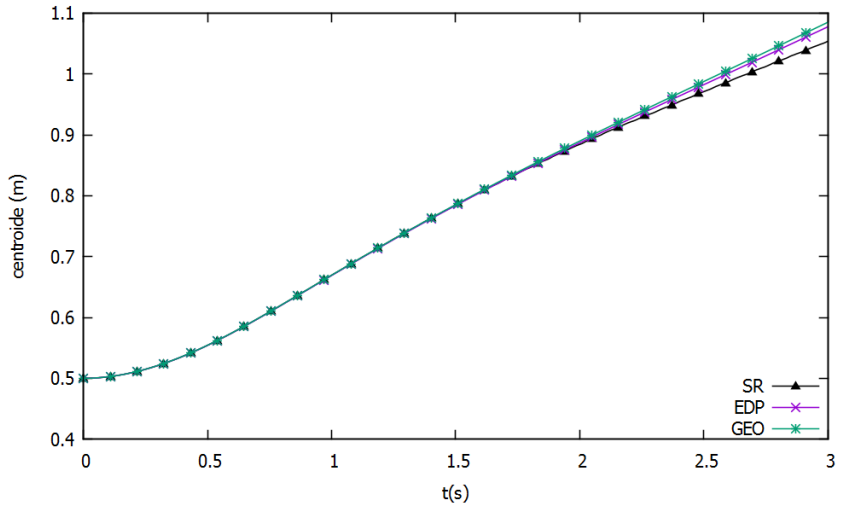
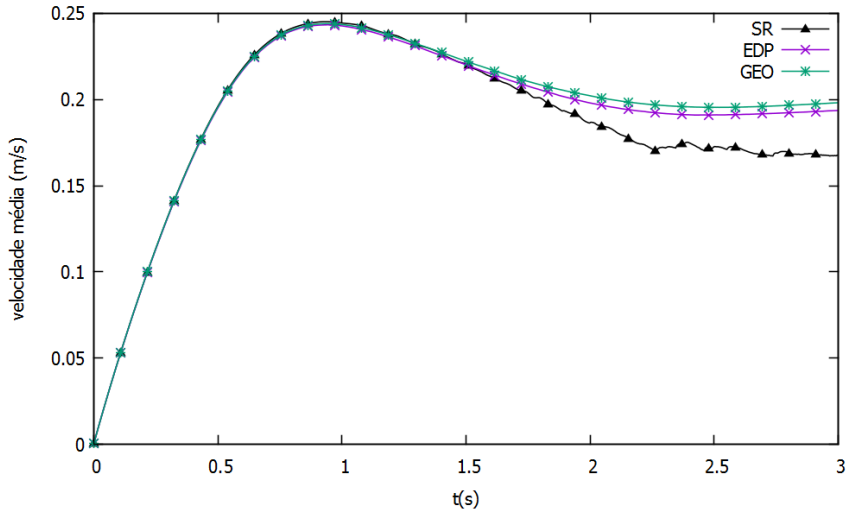


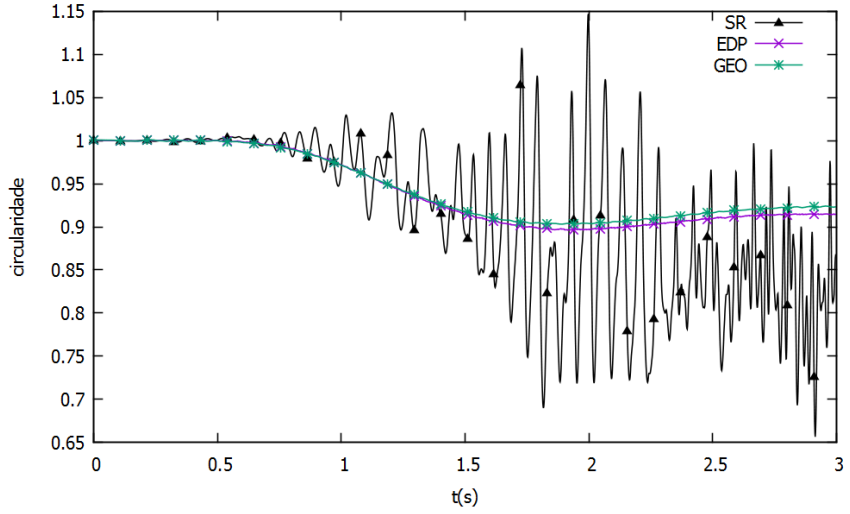
Figura 5.6: Simulação do teste da ascensão da bolha, com espaçamento $1/h = 80$, curvatura LS1 e utilizando a reinicialização de ϕ em todos os passos de tempo. A tensão superficial foi calculada pelo método SSF. Em **a)** foi simulado o teste 1, e em **b)** o teste 2, com os parâmetros da tabela 5.3. Os resultados foram impressos nos instantes $t = 0$, $t = 1.5s$ e $t = 3.0s$. A linha tracejada na figura **b)** foi utilizada apenas para facilitar a visualização. Em **c)** e **d)** são exibidas as soluções de referência de [15] em $t = 3.0s$ para os testes 1 e 2, respectivamente.



(a)

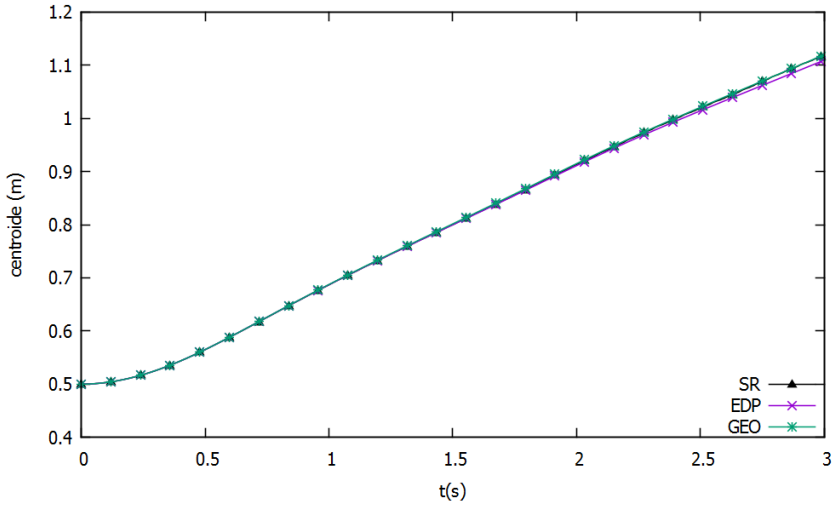


(b)

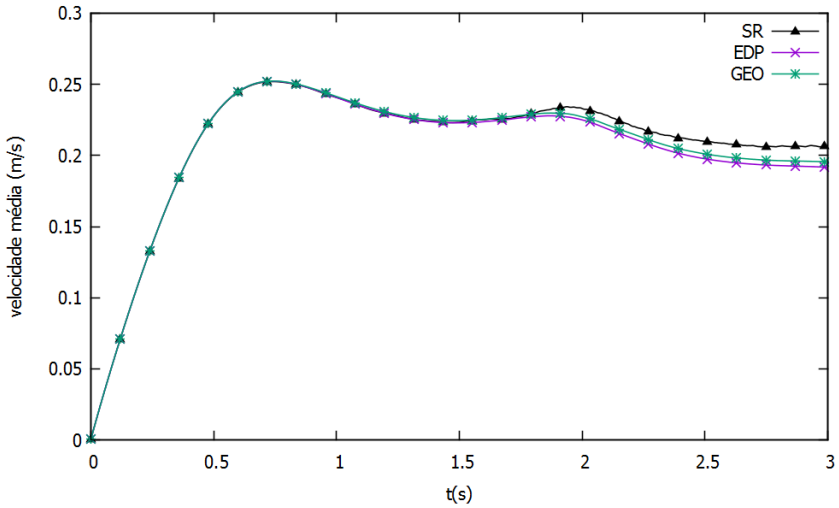


(c)

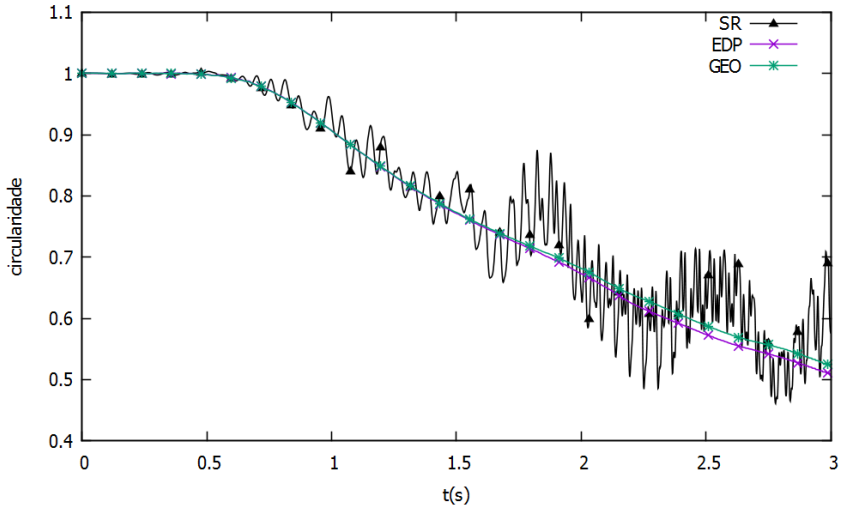
Figura 5.7: Simulação do teste 1 da ascensão da bolha, com espaçamento $1/h = 80$, curvatura LS1 e tensão superficial calculada pelo método SSF. **a)** Centroide; **b)** Velocidade média; **c)** Circularidade.



(a)



(b)



(c)

Figura 5.8: Simulação do teste 2 da ascensão da bolha, com espaçamento $1/h = 80$, curvatura LS1 e tensão superficial calculada pelo método SSF. **a)** Centroide; **b)** Velocidade média; **c)** Circularidade.

Capítulo

6

Conclusão

Neste trabalho, foi desenvolvida a teoria clássica para a utilização de métodos level-set para a representação implícita de interfaces, e suas aplicações para escoamentos multifásicos. Além disso, foi mostrado como combinar o método WENO de quinta ordem com um Runge-Kutta de terceira ordem, para que a advecção da função level-set seja feita de maneira acurada. Utilizamos malha uniforme no contexto de diferenças finitas, e estudamos apenas o caso bidimensional.

Apresentamos, ainda, duas técnicas de reinicialização para restaurar a propriedade de função distância da função level-set: uma uma baseada na resolução de uma equação diferencia parcial, e uma totalmente geométrica. A reinicialização por EDP já foi estudada por muitos autores, enquanto que não há artigos publicados até o momento que utilizem a reinicialização geométrica com diferenças finitas. Desta forma, a reinicialização geométrica mostrou-se eficiente, e pode ser utilizada como uma alternativa em códigos que lidam com escoamentos multifásicos. Observamos que ambas as reinicializações trouxeram soluções resultados semelhantes em todos os problemas que resolvemos.

Também foram propostas três discretizações diferentes para o cálculo da curvatura utilizando a função level-set. Todas elas mostraram resultados próximos, mas a LS1 se mostrou mais precisa, principalmente no teste das correntes espúrias (seção 5.1).

Resolvemos as equações de Navier-Stokes com um método de projeção implícito, e calculamos a tensão superficial usando duas técnicas distintas: CSF e SSF. As duas técnicas trouxeram soluções semelhantes no teste da oscilação da bolha (seção 5.2), mas a SSF se mostrou melhor no teste das correntes espúrias.

Como estudos futuros, fica a generalização dos conceitos estudados para problemas tridimensionais. A reinicialização por EDP precisa de pouquíssimas adaptações para ser aplicada em problemas 3D. Já a geométrica precisa de alterações mais profundas. Como exemplo, a análise feita na Figura 3.13 para o cálculo geométrico da área interna à interface precisa ser totalmente refeita.

Outra possível generalização a ser feita é sobre a malha utilizada. Caso desejemos utilizar uma malha não-uniforme, como uma malha com *stretching*, o método WENO utilizado no transporte da interface, e na reinicialização por EDP, precisa ser melhor estudado. A reinicialização geométrica que foi apresentada nesse trabalho não precisa de adaptações para ser utilizada com stretching.

Apêndice

A

Discretizações LS2 e LS3 para o cálculo da curvatura

Nesse Apêndice estão detalhadas as discretizações LS2 e LS3 para o cálculo numérico da curvatura. O método LS1 é apresentado na seção 2.3.3.

A.1 Método LS2

No método LS2, seguimos a proposta de Sussman [35], que consiste em aproximar todas as derivadas da equação (2.7) por diferenças centradas, utilizando os valores da normal nas faces das células:

$$\kappa_{i,j} = -(\nabla \cdot \boldsymbol{n})_{i,j} = \left(\frac{\partial n_x}{\partial x}\right)_{i,j} + \left(\frac{\partial n_y}{\partial y}\right)_{i,j},$$

onde n_x e n_y são as componentes do vetor $\boldsymbol{n} = (n_x, n_y)$.

Para que a curvatura seja calculada no centro da célula, devemos obter o divergente de $\boldsymbol{n}$, também, em seu centro. As derivadas do vetor normal no centro da célula são aproximadas com diferenças centradas utilizando os valores do vetor normal nas faces da célula

$$\left(\frac{\partial n_x}{\partial x}\right)_{i,j} \approx \frac{1}{\Delta x}\left(\left(\frac{\partial n_x}{\partial x}\right)_{i+\frac{1}{2},j} - \left(\frac{\partial n_x}{\partial x}\right)_{i-\frac{1}{2},j}\right)$$
$$\left(\frac{\partial n_y}{\partial y}\right)_{i,j} \approx \frac{1}{\Delta y}\left(\left(\frac{\partial n_y}{\partial y}\right)_{i,j+\frac{1}{2}} - \left(\frac{\partial n_y}{\partial y}\right)_{i,j-\frac{1}{2}}\right).$$

Os valores de n_x e n_y nas faces das células são calculadas através de médias dos valores nas quinas da celula, como a seguir:

$$(n_x)_{i+\frac{1}{2},j} = \frac{(n_x)_{i+\frac{1}{2},j+\frac{1}{2}} + (n_x)_{i+\frac{1}{2},j-\frac{1}{2}}}{2} \tag{A.1}$$

$$(n_x)_{i-\frac{1}{2},j} = \frac{(n_x)_{i-\frac{1}{2},j+\frac{1}{2}} + (n_x)_{i-\frac{1}{2},j-\frac{1}{2}}}{2} \tag{A.2}$$

$$(n_y)_{i,j+\frac{1}{2}} = \frac{(n_y)_{i+\frac{1}{2},j+\frac{1}{2}} + (n_y)_{i-\frac{1}{2},j+\frac{1}{2}}}{2} \tag{A.3}$$

$$(n_y)_{i,j-\frac{1}{2}} = \frac{(n_y)_{i+\frac{1}{2},j-\frac{1}{2}} + (n_y)_{i-\frac{1}{2},j-\frac{1}{2}}}{2}. \tag{A.4}$$

Por fim, os valores do vetor normal nas quinas das células são aproximados utilizando os valores da função ϕ, como a seguir,

$$\boldsymbol{n}_{i+\frac{1}{2},j+\frac{1}{2}} = -\frac{(\nabla\phi)_{i+\frac{1}{2},j+\frac{1}{2}}}{|\nabla\phi|_{i+\frac{1}{2},j+\frac{1}{2}}} \tag{A.5}$$

$$\boldsymbol{n}_{i-\frac{1}{2},j+\frac{1}{2}} = -\frac{(\nabla\phi)_{i-\frac{1}{2},j+\frac{1}{2}}}{|\nabla\phi|_{i-\frac{1}{2},j+\frac{1}{2}}} \tag{A.6}$$

$$\boldsymbol{n}_{i+\frac{1}{2},j-\frac{1}{2}} = -\frac{(\nabla\phi)_{i+\frac{1}{2},j-\frac{1}{2}}}{|\nabla\phi|_{i+\frac{1}{2},j-\frac{1}{2}}} \tag{A.7}$$

$$\boldsymbol{n}_{i-\frac{1}{2},j-\frac{1}{2}} = -\frac{(\nabla\phi)_{i-\frac{1}{2},j-\frac{1}{2}}}{|\nabla\phi|_{i-\frac{1}{2},j-\frac{1}{2}}}, \tag{A.8}$$

sendo que o gradiente da função level set nas quinas das células são obtidos através das expressões abaixo:

$$(\nabla\phi)_{i+\frac{1}{2},j+\frac{1}{2}} = \left(\frac{\phi_{i+1,j} - \phi_{i,j} + \phi_{i+1,j+1} - \phi_{i,j+1}}{2\Delta x}, \frac{\phi_{i,j+1} - \phi_{i,j} + \phi_{i+1,j+1} - \phi_{i+1,j}}{2\Delta y}\right) \tag{A.9}$$

$$(\nabla\phi)_{i-\frac{1}{2},j+\frac{1}{2}} = \left(\frac{\phi_{i,j} - \phi_{i-1,j} + \phi_{i,j+1} - \phi_{i-1,j+1}}{2\Delta x}, \frac{\phi_{i,j+1} - \phi_{i,j} + \phi_{i-1,j+1} - \phi_{i-1,j}}{2\Delta y}\right) \tag{A.10}$$

$$(\nabla\phi)_{i+\frac{1}{2},j-\frac{1}{2}} = \left(\frac{\phi_{i+1,j-1} - \phi_{i,j-1} + \phi_{i+1,j} - \phi_{i,j}}{2\Delta x}, \frac{\phi_{i+1,j} - \phi_{i+1,j-1} + \phi_{i,j} - \phi_{i,j-1}}{2\Delta y}\right) \tag{A.11}$$

$$(\nabla\phi)_{i-\frac{1}{2},j-\frac{1}{2}} = \left(\frac{\phi_{i,j-1} - \phi_{i-1,j-1} + \phi_{i,j} - \phi_{i-1,j}}{2\Delta x}, \frac{\phi_{i-1,j} - \phi_{i-1,j-1} + \phi_{i,j} - \phi_{i,j-1}}{2\Delta y}\right). \tag{A.12}$$

A.2 Método LS3

Esse método foi proposto em [7] e, assim como o método anterior, aproxima as derivadas do vetor normal nas quinas das células. Primeiramente, reescrevamos a expressão (2.6) como

$$\kappa = \frac{1}{|\tilde{\boldsymbol{n}}|}\left(\left(\frac{\tilde{\boldsymbol{n}}}{|\tilde{\boldsymbol{n}}|}\cdot\nabla\right)|\tilde{\boldsymbol{n}}| - \nabla\cdot\boldsymbol{n}\right).$$

O vetor $\tilde{\boldsymbol{n}}$ é um vetor normal não unitário, dado por $\tilde{\boldsymbol{n}} = -\nabla\phi$. O termo $\nabla\cdot\boldsymbol{n}$ é discretizado como no método LS2. Contudo, agora temos um termo adicional para discretizar:

$$\left(\frac{\tilde{\boldsymbol{n}}_{i,j}}{|\tilde{\boldsymbol{n}}_{i,j}|}\cdot\nabla\right)|\tilde{\boldsymbol{n}}| = \left(\frac{\tilde{n}_x}{|\tilde{\boldsymbol{n}}|}\right)_{i,j}\left(\frac{\partial|\tilde{\boldsymbol{n}}|}{\partial x}\right)_{i,j} + \left(\frac{\tilde{n}_y}{|\tilde{\boldsymbol{n}}|}\right)_{i,j}\left(\frac{\partial|\tilde{\boldsymbol{n}}|}{\partial y}\right)_{i,j}.$$

O vetor normal $\tilde{\boldsymbol{n}}_{i,j}$ é calculado como a média aritmética dos vetores normais calculados nas quinas da célula

$$\tilde{\boldsymbol{n}}_{i,j} = \frac{1}{4}\left(\tilde{\boldsymbol{n}}_{i+\frac{1}{2},j+\frac{1}{2}} + \tilde{\boldsymbol{n}}_{i+\frac{1}{2},j-\frac{1}{2}} + \tilde{\boldsymbol{n}}_{i-\frac{1}{2},j+\frac{1}{2}} + \tilde{\boldsymbol{n}}_{i-\frac{1}{2},j-\frac{1}{2}}\right),$$

onde esses vetores são calculados de maneira similar às equações A.5 a A.8, com a diferença de que, como $\tilde{\boldsymbol{n}}$ não é uma normal unitária, a divisão pelo módulo do gradiente de ϕ não

deve ser feita quando as equações (A.5) a (A.8) forem utilizadas. As derivadas do módulo do vetor normal foram calculadas como a seguir:

$$\left(\frac{\partial|\tilde{\boldsymbol{n}}|}{\partial x}\right)_{i,j} = \frac{1}{2\Delta x}\left(|\tilde{\boldsymbol{n}}|_{i+\frac{1}{2},j+\frac{1}{2}} - |\tilde{\boldsymbol{n}}|_{i-\frac{1}{2},j+\frac{1}{2}} + |\tilde{\boldsymbol{n}}|_{i+\frac{1}{2},j-\frac{1}{2}} - |\tilde{\boldsymbol{n}}|_{i-\frac{1}{2},j-\frac{1}{2}}\right) \tag{A.13}$$

$$\left(\frac{\partial|\tilde{\boldsymbol{n}}|}{\partial y}\right)_{i,j} = \frac{1}{2\Delta y}\left(|\tilde{\boldsymbol{n}}|_{i+\frac{1}{2},j+\frac{1}{2}} - |\tilde{\boldsymbol{n}}|_{i+\frac{1}{2},j-\frac{1}{2}} + |\tilde{\boldsymbol{n}}|_{i-\frac{1}{2},j+\frac{1}{2}} - |\tilde{\boldsymbol{n}}|_{i-\frac{1}{2},j-\frac{1}{2}}\right). \tag{A.14}$$

O módulo de $\tilde{\boldsymbol{n}}$ nas quinas da célula é calculado da maneira trivial.

Apêndice

B

Algoritmos Computacionais

Esse Apêndice tem por objetivo ser um guia mais detalhado do que o texto principal para a implementação do método level-set estudado nesse trabalho. A Figura B.1 mosta uma visão geral da aplicação que implementamos. Em testes com campo de velocidade prescrito, obviamente o quadro "Solução das Equações de Navier-Stokes" deve ser ignorado.

Por ser a parte da aplicação que mais nos dedicamos nesse trabalho, detalharemos nesse Apêndice apenas as duas reinicializações: a por EDP e a geométrica.

B.1 Reinicialização por EDP

A reinicialização de ϕ é significativamente mais complexa do que a advecção no que diz respeito à implementação. Por isso, essa seção visa ser um passo-a-passo da implementação conforme foi feita durante os estudos realizados. Informações adicionais sobre a implementação podem ser encontradas em [26] e [37]. A figura $B.2$ exibe uma visão geral da reinicialização de ϕ por EDP.

Para simplificar a notação, nessa seção ϕ será considerada como função de, somente, x, y e τ, pois a variável t não possui importância para o processo de reinicialização (exceto na inicialização de $\phi(x, y, \tau)$ pela equação (3.9)). Assim, $\phi_{i,j}^{(n)}$ é equivalente a $\phi(x_i, y_j, \tau = n)$.

Primeiramente, reescrevamos as equações (3.8) e (3.16), respectivamente, como

$$\phi_\tau = S(\phi)(1 - |\nabla\phi|) = L(\phi) \tag{B.1}$$

$$\phi_\tau = L(\phi) + \lambda_{i,j}\delta(\phi)|\nabla\phi|. \tag{B.2}$$

A seguinte notação será utilizada:

$$L_{i,j}(\phi) = S(\phi_{i,j})(1 - |\nabla\phi_{i,j}|). \tag{B.3}$$

Daí, a reinicialização com as derivadas espaciais aproximadas com o método WENO de quinta ordem, e a temporal com um método Runge-Kutta de terceira ordem, pode ser feita segundo o procedimento abaixo:

1. Inicialize $n = 0$, $= \tau = 0$ e $\phi^{(0)}$ com o valor vindo da última etapa de advecção.
2. Enquanto $\tau < \tau_{\max}$
   (a) Inicialize $\bar{\phi}^{(0)} = \phi^{(\tau = n\Delta\tau)}$.

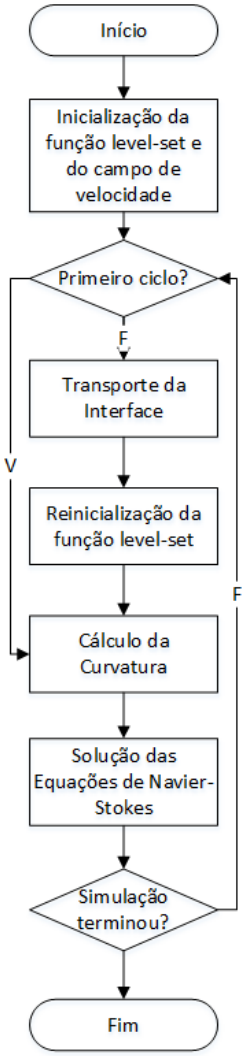


Figura B.1: Fluxograma geral de uma simulação completa resolvendo as equações de Navier-Stokes.

(b) Para $k = 1 \ldots 3$

i. Para cada célula $\Omega_{i,j}$ do domínio, calcule

$$\bar{\phi}_{i,j}^{(k)} = \sum_{z=0}^{k-1} \alpha_{kz} \bar{\phi}_{i,j}^{(z)} + \beta_{kz} \Delta t L_{i,j} \left(\phi^{(0)}, \bar{\phi}^{(z)} \right), \tag{B.4}$$

onde

$$\begin{array}{lllllll} \alpha_{10} = 1 & \alpha_{20} = 3/4 & \alpha_{21} = 1/4 & \alpha_{30} = 1/3 & \alpha_{31} = 0 & \alpha_{32} = 2/3 \\ \beta_{10} = 1 & \beta_{20} = 0 & \beta_{21} = 1/4 & \beta_{30} = 0 & \beta_{31} = 0 & \beta_{32} = 2/3. \end{array}$$

O cálculo de L com o método WENO de quinta ordem será detalhado mais abaixo.

(c) $\tilde{\phi}_{i,j}^{(n+1)} = \bar{\phi}_{i,j}^{(3)}$, $(i,j) \in \omega$.

(d) Aqui começa a conservação da massa. Para cada célula $\Omega_{i,j}$ do domínio, calcule

$$g_{1,i,j} = \delta \left(\phi_{i,j}^{(0)} \right) \frac{\tilde{\phi}_{i,j}^{(n+1)} - \phi_{i,j}^{(0)}}{\Delta \tau} \tag{B.5}$$

$$g_{2,i,j} = \left[\delta \left(\phi_{i,j}^{(0)} \right) \right]^2 \left| \nabla \phi_{i,j}^{(0)} \right|, \tag{B.6}$$

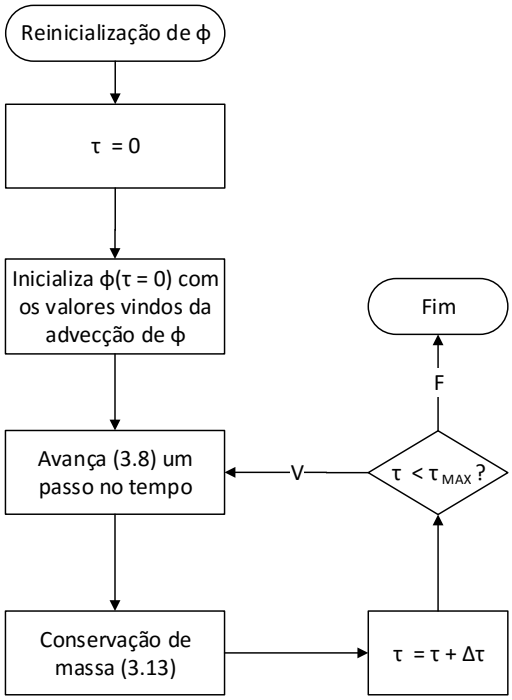


Figura B.2: Visão geral da reinicialização de ϕ por EDP. Note que o pseudotempo τ do processo de reinicialização não possui relação direta com o tempo t da rotina de advecção.

onde g_1 e g_2 são, respectivamente, o numerador e o denominador de (3.18), calculados em cada célula computacional.

Obs: Para o cálculo de g_2, o gradiente não precisa ser recalculado, uma vez que ele já foi obtido no cálculo de L.

(e) Para cada célula $\Omega_{i,j}$ do domínio, calcule

$$\tilde{g}_{1,i,j} = \frac{\Delta x^2}{3m(i,j)} \left(2m(i,j) g_{1,i,j} + \sum_{m,n=-1;(m,n)\neq(0,0)}^{1} g_{1,i+m,j+n} \right) \tag{B.7}$$

$$\tilde{g}_{2,i,j} = \frac{\Delta x^2}{3m(i,j)} \left(2m(i,j) g_{2,i,j} + \sum_{m,n=-1;(m,n)\neq(0,0)}^{1} g_{2,i+m,j+n} \right) \tag{B.8}$$

$$\lambda_{i,j} = \begin{cases} 0, & \text{se } \tilde{g}_{2,i,j} = 0 \\ -\frac{\tilde{g}_{1,i,j}}{\tilde{g}_{2,i,j}}, & \text{se } \tilde{g}_{2,i,j} \neq 0 \end{cases}, \tag{B.9}$$

onde $m(i,j)$ é a quantidade de células vizinhas à célula (i,j) (ver Figura 3.3) e $g_{1,i,j} = g_{2,i,j} = 0, (i,j) \notin \Omega$, e δ é calculada por (3.19).

(f) Para cada célula $\Omega_{i,j}$ do domínio, faça $\phi_{i,j}^{(n+1)} = \tilde{\phi}_{i,j}^{(n+1)} + \Delta\tau\lambda_{i,j}\delta\left(\phi_{i,j}^{(0)}\right)\left|\nabla\phi_{i,j}^{(0)}\right|$.
Obs: o cálculo do gradiente pode ser aproveitado daquele já feito na obtenção de L.

(g) Atualize $n = n + 1$ e $\tau = \tau + \Delta_\tau$.

O cálculo de $L\left(\phi^{(0)}, \phi^{(\tau=n)}\right)$ pode ser feito como a seguir.

1. Para cada célula $\Omega_{i,j}$ do domínio

   (a) Calcule as aproximações para as derivadas de $\phi^{(n)}$, sendo elas $\phi^{(n),+}_{x,i,j}$, $\phi^{(n),-}_{x,i,j}$, $\phi^{(n),+}_{y,i,j}$ e $\phi^{(n),-}_{y,i,j}$, conforme as equações (2.24) e (2.26). Nos contornos do domínio, foi utilizada condição $\frac{\partial \phi}{\partial x} = \frac{\partial \phi}{\partial y} = 0$. Calcule as mesmas aproximações para as derivadas de $\phi^{(0)}$.

   (b) Calcule $\tilde{s}_{i,j} = S(\phi^{(0)}_{i,j})$ com a expressão (3.10).

   (c) i. Se $\tilde{s}_{i,j}\phi^{(n),+}_{x,i,j} \geq 0$ e $\tilde{s}_{i,j}\phi^{(n),-}_{x,i,j} \geq 0$,

$$\phi^{(n)}_{x,i,j} = \phi^{(n),-}_{x,i,j} \tag{B.10}$$
$$\phi^{(0)}_{x,i,j} = \phi^{(0),-}_{x,i,j} \tag{B.11}$$

   ii. Senão, se $\tilde{s}_{i,j}\phi^{(n),+}_{x,i,j} \leq 0$ e $\tilde{s}_{i,j}\phi^{(n),-}_{x,i,j} \leq 0$,

$$\phi^{(n)}_{x,i,j} = \phi^{(n),+}_{x,i,j} \tag{B.12}$$
$$\phi^{(0)}_{x,i,j} = \phi^{(0),+}_{x,i,j} \tag{B.13}$$

   iii. Senão, se $\tilde{s}_{i,j}\phi^{(n),+}_{x,i,j} \geq 0$ e $\tilde{s}_{i,j}\phi^{(n),-}_{x,i,j} \leq 0$

$$\phi^{(n)}_{x,i,j} = 0 \tag{B.14}$$
$$\phi^{(0)}_{x,i,j} = 0 \tag{B.15}$$

   iv. Senão, se $\left|\tilde{s}_{i,j}\phi^{(n),+}_{x,i,j}\right| < \left|\tilde{s}_{i,j}\phi^{(n),-}_{x,i,j}\right|$

$$\phi^{(n)}_{x,i,j} = \phi^{(n),-}_{x,i,j} \tag{B.16}$$
$$\phi^{(0)}_{x,i,j} = \phi^{(0),-}_{x,i,j} \tag{B.17}$$

   v. Senão,

$$\phi^{(n)}_{x,i,j} = \phi^{(n),+}_{x,i,j} \tag{B.18}$$
$$\phi^{(0)}_{x,i,j} = \phi^{(0),+}_{x,i,j} \tag{B.19}$$

   Calcule as derivadas na direção y de forma análoga.

   (d) Calcule os gradientes

$$|\nabla \phi^{(n)}(\boldsymbol{x}_{i,j})| = \sqrt{\left(\phi^{(n)}_{x,i,j}\right)^2 + \left(\phi^{(n)}_{y,i,j}\right)^2} \tag{B.20}$$
$$|\nabla \phi^{(0)}(\boldsymbol{x}_{i,j})| = \sqrt{\left(\phi^{(0)}_{x,i,j}\right)^2 + \left(\phi^{(0)}_{y,i,j}\right)^2} \tag{B.21}$$

   (e) Atualize $s_{i,j} = S\left(\phi^{(n)}_{i,j}\right)$ com a expressão (3.11).

   (f) Calcule $L_{i,j}\left(\phi^{(0)}_{i,j}, \phi^{(n)}_{i,j}\right) = s_{i,j}\left(1 - \left|\nabla \phi^{(n)}_{i,j}\right|\right)$.

B.2 Reinicialização Geométrica

Essa seção tratará dos detalhes referentes à implementação da reinicialização geométrica de ϕ. Como a implementação dessa reinicialização é consideravelmente mais complexa que a da reinicialização por EDP, optamos por utilizar uma linguagem mais próxima à da implementação, como a mostrada na função abaixo.

Function InterfaceIntersectaTriangulo(ϕ, τ_i)

```
1  qtdPositivos ← 0 ;
2  qtdNegativos ← 0 ;
3  foreach j ∈ {1, 2, 3} do
4  |  if φ(ρ_{i,j}) < 0 then
5  |  |  qtdNegativos ← qtdNegativos + 1 ;
6  |  end
7  |  else
8  |  |  qtdPositivos ← qtdPositivos + 1 ;
9  |  end
10 end
11 τ_i→intersectaInterface ← qtdNegativos > 0 and qtdPositivos > 0 ;
12 return τ_i→intersectaInterface
```

Essa função é responsável por determinar se a interface definida por ϕ passa por um determinado triângulo. Note que a mesma função InterfaceIntersectaTriangulo pode ser usada para as interfaces definidas pelo isocontorno nulo de ϕ, ϕ^* ou $\bar{\phi}$, bastando que a função level-set desejada seja passada como parâmetro para InterfaceIntersectaTriangulo. Consideramos que o contorno da interface passa pelo triângulo se houver ao menos um vértice de τ_i em que ϕ seja positiva e um vértice em que ela seja negativa.

Na linha 11, estamos salvando dentro de τ_i a informação de ele ser intersectado pelo contorno da interface ou não. Desse modo, essa informação pode ser facilmente verificada em outros trechos do programa, sem que seja necessário chamar novamente a função InterfaceIntersectaTriangulo. Como o armazenamento dessa informação é feito depende da estratégia utilizada na implementação.

Agora que sabemos determinar se um triângulo é intersectado pelo contorno da interface, podemos construir uma função que retorne todos os triângulos que são intersectados por esse contorno.

Function TriangulosIntersectadosPelaInterface(ϕ, $\mathcal{T}$)

```
1  T̄ ← ∅ ;
2  foreach τ_i ∈ T do
3  |  if InterfaceIntersectaTriangulo (τ_i) then
4  |  |  T̄→PushBack(τ_i) ;
5  |  end
6  end
7  return T̄
```

Na função acima, consideramos que $\bar{\mathcal{T}}$ é uma lista. Assim, na linha 4, estamos adicionando τ_i ao final da lista.

Agora, precisamos de uma função que obtenha todos os vértices adjacentes à interface (ou seja, todos os vértices que pertencem a algum triângulo de $\bar{\mathcal{T}}$). Para isso, assumiremos que, dado o vértice $\boldsymbol{v}_i$, o conjunto adj($\boldsymbol{v}_i$) definido em (3.50) é conhecido. Na implementação feita nesse trabalho, cada vértice $\boldsymbol{v}_i$ possui uma estrutura de dados do tipo lista, com todos os triângulos que contêm esse vértice.

Function VerticesAdjacentesAInterface(ϕ, $\mathcal{V}$)

```
1  V̄ ← ∅ ;
2  foreach v_i ∈ V do
3      foreach τ_i ∈ adj(v_i) do
4          if τ_i→intersectaInterface then
5              V̄→PushBack(v_i) ;
6              break ;
7          end
8      end
9  end
10 return V̄
```

Conhecidos $\bar{\mathcal{T}}$ e $\bar{\mathcal{V}}$, podemos criar uma função que, dada uma função level-set ϕ, obtenha a interface representada por $\bar{\phi}$, onde $\bar{\phi}$ é obtida interpolando linearmente ϕ nos triângulos de $\bar{\mathcal{T}}$.

Function ReconstroiInterface(ϕ, $\bar{\mathcal{T}}$, SalvaPontosDaInterface)

```
1  k ← 0 ;
2  foreach τ̄_i ∈ T̄ do
3      foreach j_1, j_2 ∈ {1,2,3}, j_1 ≠ j_2 do
4          if φ(ρ_{i,j_1})φ(ρ_{i,j_2}) < 0 then
5              |P − ρ_{i,j_1}| ← |ρ_{i,j_2} − ρ_{i,j_1}| φ(ρ_{i,j_1}) / (φ(ρ_{i,j_2}) − φ(ρ_{i,j_1})) ;
6              P ← ρ_{i,j_1} + (ρ_{i,j_2} − ρ_{i,j_1}) / |ρ_{i,j_2} − ρ_{i,j_1}| |P − ρ_{i,j_1}| ;
7              PontosDaInterface[k] ← P ;
8              k ← k + 1 ;
9          end
10     end
11 end
12 SalvaPontosDaInterface(τ̄_i, PontosDaInterface) ;
```

A função acima recebe como parâmetro uma outra função, chamada SalvaPontosDaInterface, que é responsável por salvar os pontos de intersecção do contorno da interface com cada um dos triângulos de $\bar{\mathcal{T}}$. Isso é necessário, visto que precisamos reconstruir a interface para várias funções diferentes ($\bar{\phi}$, $\phi^* + \eta$, etc). SalvaPontosDaInterface é, então, responsável por salvar esses pontos na variável correta, dependendo de qual é a função cuja interface está sendo reconstruída (linha 12).

Agora que já temos uma função para obter o contorno da interface dentro de cada triângulo, podemos calcular ϕ^* em cada um dos vértices adjacentes à interface. Veja que, nesses pontos, ϕ^* é igual a $\hat{d}$ por (3.31). Assim, pela definição de $\hat{d}$ em (3.25), precisamos calcular a distância de cada um pontos vértices à interface $\partial\bar{\phi}$.

Para isso, devemos implementar uma função que, dado um vértice $\boldsymbol{\rho}_{i,j}$ e um triângulo $\bar{\tau}_k$, calcule a menor distância entre $\boldsymbol{\rho}_{i,j} = (\bar{\boldsymbol{v}}_x, \bar{\boldsymbol{v}}_y)$ e o segmento de reta da interface que está em $\bar{\tau}_k$ (equações 3.32–3.35).

Function DistanciaVerticeInterface($\boldsymbol{\rho}_{i,j}$, $\bar{\tau}_k$, ObtemPontosDaInterface)

$(\boldsymbol{P_1}, \boldsymbol{P_2}) = \texttt{ObtemPontosDaInterface}(\bar{\tau}_k)$;
$\lambda = \frac{(P_{1x}-\bar{\boldsymbol{v}}_x)(P_{1x}-P_{2x})+\left(P_{1y}-\bar{\boldsymbol{v}}_y\right)\left(P_{1y}-P_{2y}\right)}{(P_{2x}-P_{1x})^2+(P_{2y}-P_{1y})^2}$;
$\lambda = \min(\lambda, 1)$;
$\lambda = \max(\lambda, 0)$;
$\mathsf{P} = \boldsymbol{P_1} + \lambda(\boldsymbol{P_2} - \boldsymbol{P_1})$;
return $|\boldsymbol{\rho}_{i,j} - \mathsf{P}|$;

A função ObtemPontosDaInterface foi utilizada para manter a compatibilidade com a função SalvaPontosDaInterface de ReconstroiInterface. Agora, podemos calcular ϕ^* em todos os vértices adjacentes à interface, conforme abaixo.

Function CalculaFiAsterisco($\bar{\mathcal{V}}$, $\bar{\mathcal{T}}$, ObtemPontosDaInerface)

foreach $\bar{\boldsymbol{v}}_i \in \bar{\mathcal{V}}$ **do**
    $\mathsf{d} \leftarrow \infty$;
    foreach $\tau_i \in adj(\bar{\boldsymbol{v}}_i)$ **do**
        $\mathsf{d} \leftarrow \min(\mathsf{d}, \texttt{DistanciaVerticeInterface}(\bar{\boldsymbol{v}}_i, \tau_i, \texttt{ObtemPontosDaInterface}))$;
    end
    $\phi^*(\bar{\boldsymbol{v}}_i) = \mathsf{d}$;
end

Para simplificar a construção do algoritmo completo, dividiremos ele em etapas. Todo o procedimento até o cálculo de ϕ^* será chamado de Passo 1. Assim, podemos criar uma função chamada Passo 1 que, basicamente, chamará as funções anteriores na ordem correta.

Function Passo1(ϕ, $\mathcal{T}$, $\mathcal{V}$)

$\bar{\mathcal{T}} \leftarrow \texttt{TriangulosIntersectadosPelaInterface}(\phi, \mathcal{T})$;
$\bar{\mathcal{V}} \leftarrow \texttt{VerticesAdjacentesAInterface}(\phi, \mathcal{V})$;
$\texttt{ReconstroiInterface}(\phi, \bar{\mathcal{T}}, \texttt{SalvaInterfaceFi})$;
$\texttt{CalculaFiAsterisco}(\bar{\mathcal{T}}, \bar{\mathcal{V}}, \texttt{ObtemInterfaceFi})$;
return $(\phi^*, \bar{\mathcal{T}}, \bar{\mathcal{V}})$;

As funções SalvaInterfaceFi e ObtemInterfaceFi devem ser implementadas de forma a lidarem com a interface $\partial\bar{\phi}$. A partir daqui, ϕ^* já é uma função distância, mas devemos realizar alguns passos adicionais para melhorar a conservação de massa do método. O primeiro deles consiste de resolver (3.38). Visto que a teoria já foi desenvolvida por completo na seção 3.2.2, focaremos apenas na implementação.

Primeiramente, precisamos de uma função que receba um triângulo e retorne a área interna à interface nesse triângulo. Isso pode ser feito com o auxílio da equação (3.37), de modo que precisamos, apenas, encontrar os pontos nos quais a somatória dessa equação

será aplicada. Ou seja, precisamos detectar qual dos casos da Figura 3.13 acontece dentro desse triângulo. Para isso, precisaremos identificar cada um dos vértices $\boldsymbol{\rho}_{i,j}$ que pertencem ao triângulo $\bar{\tau}_i$. Conforme mostrado na Figura B.3, nos triângulos de $\mathcal{T}_1$, o vértice inferior esquerdo será identificado por $\boldsymbol{\rho}_{i,1}$, seguido pelos vértices $\boldsymbol{\rho}_{i,2}$ e $\boldsymbol{\rho}_{i,3}$ no sentido horário. Nos triângulos de $\mathcal{T}_2$, o nó $\boldsymbol{\rho}_{i,1}$ é o do canto superior esquerdo, e os vértices $\boldsymbol{\rho}_{i,2}$ e $\boldsymbol{\rho}_{i,3}$ também seguem no sentido horário.

A função a seguir é responsável por calcular a área interna à interface no triângulo $\bar{\tau}_i$.

Function AreaInterfaceTriangulo(ϕ, τ_i, ObtemPontosDaInterface)

```
1  qtdPositivos ← 0 ;
2  qtdNegativos ← 0 ;
3  foreach j ∈ {1,2,3} do
4  |  if φ(ρ_{i,j}) < 0 then
5  |  |  qtdNegativos ← qtdNegativos + 1 ;
6  |  end
7  |  else
8  |  |  qtdPositivos ← qtdPositivos + 1 ;
9  |  end
10 end
11 if qtdPositivos = 3 then
12 |  return 0.5 × Δx × Δy ;
13 end
14 else if qtdNegativos = 3 then
15 |  return 0 ;
16 end
17 (P_1, P_2) ← ObtemPontosDaInterface(τ_i) ;
18 if qtdPositivos = 1 then
19 |  foreach j ∈ {1,2,3} do
20 |  |  if φ(ρ_{i,j}) > 0 then
21 |  |  |  P_3 ← ρ_{i,j} ;
22 |  |  |  return CalculaArea(P_1, P_2, P_3) ;
23 |  |  end
24 |  end
25 end
26 else
27 |  if τ_i ∈ T_1 then
28 |  |  if P_{1_x} = ρ_{i,0} then
29 |  |  |  if φ(ρ_{i,1_x}) > 0 then
30 |  |  |  |  P_3 = ρ_{i,3} ;
31 |  |  |  |  P_4 = ρ_{i,2} ;
32 |  |  |  end
33 |  |  |  else
34 |  |  |  |  P_3 = ρ_{i,3} ;
35 |  |  |  |  P_4 = ρ_{i,1} ;
36 |  |  |  end
37 |  |  end
38 |  |  else
39 |  |  |  P_3 = ρ_{i,1} ;
40 |  |  |  P_4 = ρ_{i,2} ;
41 |  |  end
42 |  end
43 |  else
44 |  |  if P_{1,y} = ρ_{i,1_y} then
45 |  |  |  if φ(ρ_{i,1}) > 0 then
46 |  |  |  |  P_3 = ρ_{i,3} ;
47 |  |  |  |  P_4 = ρ_{i,1} ;
48 |  |  |  end
49 |  |  |  else
50 |  |  |  |  P_3 = ρ_{i,3} ;
51 |  |  |  |  P_4 = ρ_{i,2} ;
52 |  |  |  end
53 |  |  end
54 |  |  else
55 |  |  |  P_3 = ρ_{i,2} ;
56 |  |  |  P_4 = ρ_{i,1} ;
57 |  |  end
58 |  end
59 |  return CalculaArea(P_1, P_2, P_3, P_4) ;
60 end
```

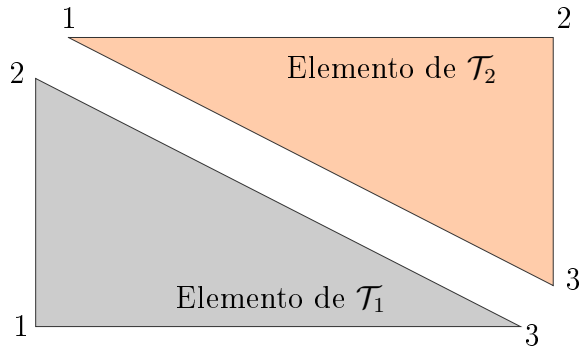


Figura B.3: Identificação de cada vértice de cada um dos dois tipos de triângulos.

Na linha 11, a função retorna toda a área de τ_i, visto que, se ϕ é positiva em seus três vértices, então todo o triângulo pertence à região interna de ϕ. Do mesmo modo, ϕ for negativa nos três vértices, a função retorna zero. Na linha 18, se ϕ for positiva em apenas um dos vértices de τ_i, acontece o caso da Figura 3.13a. Caso contrário, o restante da função determina quais dos demais casos da Figura 3.13 acontecem. Finalmente, a função CalculaArea deve implementar a expressão (3.37).

Agora, a função a seguir calcula a área interna à interface definida por uma função level-set em todo o domínio.

Function AreaInterfaceTotal(ϕ, $\mathcal{T}$, ObtemPontosDaInterface)

```
1 area ← 0 ;
2 foreach τ ∈ T do
3 |   area ← area + CalculaArea(φ, τ, ObtemPontosDaInterface) ;
4 end
5 return area ;
```

Com a função acima, podemos escrever o método da posição falsa que calculará a constante η dentro do triângulo $\bar{\tau}_i$.

Function CalculaEtaTriangulo(ϕ, ϕ^*, $\bar{\tau}_i$, $\mathcal{T}$)

```
   // Obter η^(0) e η^(1) seguindo (3.45), usando os valores de φ
1  ReconstroiInterface(φ* + η^(0), T, SalvaInterfaceEta) ;
2  areaEta^(0) ← AreaInterfaceTriangulo(φ* + η^(0), T, ObtemInterfaceEta) ;
3  areaFi ← AreaInterfaceTriangulo(φ, T, ObtemInterfaceFi) ;
4  dArea^(0) ← areaEta^(0) − areaFi ;
5  if |dArea^(0)| < 10^−15 then
6  |  return η^(0) ;
7  end
8  ReconstroiInterface(φ* + η^(1), T, SalvaInterfaceEta) ;
9  areaEta^(1) ← AreaInterfaceTriangulo(φ* + η^(1), T, ObtemInterfaceEta) ;
10 dArea^(1) ← areaEta^(1) − areaFi ;
11 if |dArea^(1)| < 10^−15 then
12 |  return η^(1) ;
13 end
14 else
15 |  (k, a, b) ← (2, 0, 1) ;
16 |  while k < 16000 do
17 |  |  η^(k) ← η^(b) − dArea^(b) (η^(b) − η^(a)) / (dArea^(b) − dArea^(a)) ;
18 |  |  ReconstroiInterface(φ* + η^(k), T, SalvaInterfaceEta) ;
19 |  |  areaEta^(k) ← AreaInterfaceTriangulo(φ* + η^(k), T, ObtemInterfaceEta) ;
20 |  |  dArea^(k) ← areaEta^(k) − areaFi ;
21 |  |  if |dArea^(k)| < 10^−15 then
22 |  |  |  return η^(k) ;
23 |  |  end
24 |  |  if dArea^(k) dArea^(b) < 0 then
25 |  |  |  a ← b ;
26 |  |  end
27 |  |  b ← k ;
28 |  |  k ← k + 1 ;
29 |  end
30 |  return η^(k) ;
31 end
```

O segundo passo da reinicialização geométrica é, então, calcular η para todos os triângulos de $\bar{\mathcal{T}}$.

Function Passo2(ϕ, ϕ^*, $\bar{\mathcal{T}}$, $\mathcal{T}$)

```
1  foreach τ̄_i ∈ T̄ do
2  |  τ̄_i→η ← CalculaEtaTriangulo(φ, φ*, τ̄_i, T) ;
3  end
```

Finalmente, o terceiro e último passo da reinicialização dos vértices adjacentes à interface consiste em resolver (3.48).

Function Passo3(ϕ, ϕ^*, $\mathcal{T}$, $\bar{\mathcal{T}}$, $\bar{\mathcal{V}}$)

```
   // Para cada triângulo de 𝒯̄, calcular ξ usando (3.47)
1  C^(0) ← 0 ;
2  ReconstroiInterface(ξC^(0), 𝒯, SalvaInterfaceXi) ;
3  areaXi^(0) ← AreaInterfaceTotal(φ* + ξC^(0), 𝒯, ObtemInterfaceXi) ;
4  areaFi ← AreaInterfaceTotal(φ, 𝒯, ObtemInterfaceFi) ;
5  dArea^(0) ← areaXi^(0) − areaFi ;
6  if |dArea^(0)| < 10^−15 then
7  |  C ← C^(1) ;
8  end
9  else
10 |  C^(1) ← dArea^(0) ;
11 |  ReconstroiInterface(φ* + η^(1), 𝒯, SalvaInterfaceXi) ;
12 |  areaXi^(1) ← AreaInterfaceTotal(φ* + ξC^(1), 𝒯, ObtemInterfaceXi) ;
13 |  dArea^(1) ← areaXi^(1) − areaFi ;
14 |  if |dArea^(1)| < 10^−15 then
15 |  |  C ← C^(1) ;
16 |  end
17 |  else
18 |  |  k ← 2 ;
19 |  |  while k < 16000 do
20 |  |  |  C^(k) ← η^(k−1) − dArea^(k−1) (η^(k−1) − η^(k−2)) / (dArea^(k−1) − dArea^(k−2)) ;
21 |  |  |  ReconstroiInterface(ξC^(k), 𝒯, SalvaInterfaceXi) ;
22 |  |  |  areaXi^(k) ← AreaInterfaceTotal(φ* + ξC^(k), 𝒯, ObtemInterfaceXi) ;
23 |  |  |  dArea^(k) ← areaXi^(k) − areaFi ;
24 |  |  |  if |dArea^(k)| < 10^−15 then
25 |  |  |  |  C ← C^(k) ;
26 |  |  |  |  break ;
27 |  |  |  end
28 |  |  |  k ← k + 1 ;
29 |  |  end
30 |  end
31 end
   // Atualiza φ* em cada vértice de 𝒱̄ usando (3.49). Apesar de (3.49) se referir a
      d̃, podemos utilizar φ* = φ* + ξC, visto que não precisamos mais dos valores
      antigos de φ*.
32 return φ*
```

A função acima apenas implementa o método da secante convencional.Após a execução do terceiro passo, a reinicialização geométrica já nos dá uma função distância com área interna bem próxima à da função que tínhamos no início da reinicialização. Agora, devemos calcular os valores dessa função nos demais triângulos do domínio, conforme detalhado na seção 3.2.3.

Function Passo4(ϕ, $\mathcal{T}$, $\bar{\mathcal{T}}$, $\bar{\mathcal{V}}$)

```
1  foreach τ_i ∈ T do
2      if τ_i ∉ T̄ then
3          foreach j ∈ {1,2,3} do
4              φ(ρ_{i,j}) = S*(φ(ρ_{i,j}))∞ ;
5          end
6      end
7  end
8  mudou ← 1 ;
9  while mudou = 1 do
10     mudou ← 0
11     foreach τ_i ∈ T \ T̄ do
12         foreach j ∈ {1,2,3} do
13             if ρ_{i,j} ∉ V̄ then
14                 (ii,jj) = coord(ρ_{i,j}) ;
15                 foreach m ∈ {−1,0,1} do
16                     foreach n ∈ {−1,0,1} do
17                         vizinho ← ρ(ii + m, jj + n) ;
18                         if (m = 0 and n = 0) or (|m| + |n| = 0) or (vizinho ∉ Ω) then
19                             continue ;
20                         end
21                         dist ← |ρ_{i,j} − vizinho| ;
22                         if |φ(vizinho) + S*(φ(ρ_{i,j})) × dist| − |φ(ρ_{i,j})| < −10^{−15} then
23                             φ(ρ_{i,j}) ← φ(vizinho) + S*(φ(ρ_{i,j})) × dist ;
24                             mudou ← 1 ;
25                         end
26                     end
27                 end
28             end
29         end
30     end
31 end
```

Utilizamos a notação $\boldsymbol{\rho}(i,j)$ para representar o vértice de $\mathcal{V}$ que corresponde à célula de índices (i,j) da malha de advecção. Na linha 14, a função *coord* é responsável por obter os índices nas direções x e y da célula da malha de advecção que está centrada no vértice $\boldsymbol{\rho}_{i,j}$. Os laços das linhas 15 e 16 percorrem todos os vértices vizinhos ao vértice $\boldsymbol{\rho}_{i,j}$.

Na linha 18, verificamos três condições: a primeira garante que analisaremos apenas os vizinhos de $\boldsymbol{\rho}_{i,j}$, e não o próprio $\boldsymbol{\rho}_{i,j}$. A segunda existe porque, da forma com que construímos os triângulos de $\mathcal{T}$, o vértice que correspondente à célula (i,j) da malha de advecção não está conectado aos vértices correspondentes às células $(i+1, j+1)$ e $(i-1, j-1)$ pelas arestas de nenhum triângulo. A terceira serve apenas para não processar pontos que estejam fora das fronteiras do domínio.

Na linha 4, outros valores podem ser utilizados para a inicialização de ϕ. Se $\phi(\boldsymbol{\rho}_{i,j}) = S^*(\phi(\boldsymbol{\rho}_{i,j}))\phi^{(0)}$, o algoritmo reinicializará ϕ em todos os pontos que estejam até uma distância $\phi^{(0)}$ da interface.

Por fim, o último passo da reinicialização geométrica consiste de calcular (3.53) até o estado estacionário, o que é feito na função a seguir.

Function Passo5(ϕ, $\mathcal{T}$, $\bar{\mathcal{T}}$, $\bar{\mathcal{V}}$)

```
1  mudou ← 1 ;
2  while mudou = 1 do
3      mudou ← 0
4      foreach τ_i ∈ T \ T̄ do
5          foreach j ∈ {1,2,3} do
6              if ρ_{i,j} ∉ V̄ then
7                  vizinho_1 ← ρ_{i,j%3+1} ;
8                  vizinho_2 ← ρ_{i,(j+1)%3+1} ;
9                  h^(0) = 0 ;
10                 h^(1) = |vizinho_2 − vizinho_1|;
11                 psilinha^(0) = ψ'(φ, h^(0), τ_i) ;
12                 if |psilinha^(0)| < 10^−15 then
13                     h = h^(0) ;
14                 end
15                 else
16                     psilinha^(1) = ψ'(φ, h^(1), τ_i) ;
17                     if |psilinha^(1)| < 10^−15| then
18                         h = h^(1) ;
19                     end
20                     else if psilinha^(0) psilinha^(1) < 0 then
21                         psi^(0) = ψ(φ, h^(0), τ_i) ;
22                         psi^(1) = ψ(φ, h^(0), τ_i) ;
23                         psi = min(psi^(0), psi^(1)) ;
24                         goto linha 47 ;
25                     end
26                     else
27                         (k, a, b) ← (2, 0, 1) ;
28                         while k < 100 do
29                             h^(k) = (h^(a) + h^(b))/2 ;
30                             psilinha^(k) = ψ(φ, h^(k), τ_i);
31                             if |psilinha^(k)| < 10^−15| then
32                                 h = h^(k) ;
33                                 break ;
34                             end
35                             if psilinha^(k) psilinha^(a) < 0 then
36                                 b ← k ;
37                             end
38                             else
39                                 a ← k ;
40                             end
41                             k ← k + 1;
42                         end
43                     end
44                 end
45             end
46             psi = ψ(h);
47             if psi − |φ(ρ_{i,j})| < −10^−15 then
48                 mudou ← 1 ;
49                 φ(ρ_{i,j}) ← S*(φ(ρ_{i,j}))psi ;
50             end
51         end
52     end
53 end
```

A função acima é a última etapa da reinicialização geométrica. Após sua execução, ϕ é novamente uma função distância, e podemos calcular propriedades como curvatura da interface e força devido à tensão superficial com maior precisão do que antes de a reinicialização ser executada. Por fim, sintetizamos a reinicialização geométrica em uma última função, que encerra esse apêndice.

Function ReinicializacaoGeometrica(ϕ, $\mathcal{T}$, $\mathcal{V}$)

1 $(\phi^*, \bar{\mathcal{T}}, \mathcal{V}) \leftarrow$ `Passo1`$(\phi, \mathcal{T}, \mathcal{V})$;
2 `Passo2`$(\phi, \phi^*, \bar{\mathcal{T}}, \mathcal{V})$;
3 $\phi \leftarrow$ `Passo3`$(\phi, \phi^*, \mathcal{T}, \bar{\mathcal{T}}, \bar{\mathcal{V}})$;
4 $\phi \leftarrow$ `Passo4`$(\phi, \mathcal{T}, \bar{\mathcal{T}}, \mathcal{V})$;
5 $\phi \leftarrow$ `Passo5`$(\phi, \mathcal{T}, \bar{\mathcal{T}}, \bar{\mathcal{V}})$;
6 **return** ϕ

Referências

[1] T. Abadie; J. Aubin; D. Legendre. On the combined effects of surface tension force calculation and interface advection on spurious currents within Volume of Fluid and Level Set frameworks. *Journal of Computational Physics*, 297:611–636, 2015.

[2] M. A. Alves; P. J. Oliveira; F. T. Pinho. A convergent and universally bounded interpolation scheme for the treatment of advection. *International Journal for Numerical Methods in Fluids*, 41:47–75, 2003.

[3] R. F. Ausas; G. C. Buscaglia; S. R. Idelsohn. A new enrichment space for the treatment of discontinuous pressures in multi-fluid flows. *International Journal for Numerical Methods in Fluids*, 70(7):829–850, 2012.

[4] R. F. Ausas; E. A. Dari; G. C. Buscaglia. A geometric mass-preserving redistancing scheme for the level set function. *International Journal for Numerical Methods in Fluids*, 64:989–1010, 2009.

[5] N. Balcàzar; O. Jofre, L. Lehmkuhl; J. Castro; J. Rigola. A finite-volume/level-set method for simulating two-phase flows on unstructured grids. *International Journal of Multiphase Flow*, 64:55–72, 2014.

[6] H. Bhatia; G. Norgard; V. Pascucci; P. Bremer. The helmholtz-hodge decomposition - a survey. *IEEE Transactions on Visualization and Computer Graphics*, 19(8):1386–1404, August 2013.

[7] J. U. Brackbill; D. B. Kothe; C. Zemach. A continuum method for modeling surface tension. *Journal of Computational Physics*, 100:335–354, 1991.

[8] G. C. Buscaglia; R. F. Ausas. Variational formulations for surface tension, capillarity and wetting. *Computer Methods in Applied Mechanics and Engineering*, 200(45):3011–3025, 2011.

[9] A. Chorin. Numerical solution of the navier-stokes equations. *Mathematics of Computation*, 22:745–762, 1968.

[10] R. Croce; M. Griebel; M. A. Schweitzer. Numerical simulation of bubble and droplet deformation by a level set approach with surface tension in three dimensions. *International Journal for Numerical Methods in Fluids*, 2010:963–993, 2010.

[11] P. Esser; J. Grande. An accurate and robust finite element level set redistancing method. *IMA Journal of Numerical Analysis*, pages 1913–1933, 2014.

[12] M. Francois; S. Cummins; E. Dendy; D. Kothe; J. Sicilian; M. Williams. A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework. *Journal of Computational Physics*, 213:141–173, 2006.

[13] K. Goda. A multistep technique with implicit difference schemes for calculating two or three-dimensional cavity flow. *Journal of Computational Physics*, 30:76–95, 1979.

[14] X. Hou; J. Rigola; O. Lehmkuhl; A. Oliva. Simulation of the two-fluid model on incompressible fow with fractional step method for both resolved and unresolved scale interfaces. *International Journal of Heat and Fluid Flow*, 52:15–27, 2015.

[15] S. Hysing; S. Turek; D. Kuzmin; E. Parolini, N. Burman; S. Ganesan; L. Tobiska. Quantitative benchmark computation of two-dimensional bubble dynamics. *International Journal for Numerical Methods in Fluids*, 60:1259–1288, 2008.

[16] G. S. Jiang; D. Peng. Weighted ENO schemes for hamilton jacobi equations. *Journal of Scientific Computing*, 21:2126–2143, 2000.

[17] J. Kim. A continuous surface tension force formulation for diffuse-interface models. *Journal of Computational Physics*, 204:784–804, 2005.

[18] B. Lalanne; L. R. Villegas; S. Tanguy; F. Risso. On the computation of viscous terms for incompressible two-phase flows with level set/ghost fluid method. *Journal of Computational Physics*, 301:289–307, 2015.

[19] A. Lefebvre. *Atomization and Sprays*. Combustion (Hemisphere Publishing Corporation). Taylor & Francis, 1988.

[20] F. Losasso; R. Fedkiw; S. Osher. Spatially adaptive techniques for level set methods and incompressible flow. *Computers and Fluids*, 35:995–1010, 2005.

[21] K. Luo; C. Shao; Y. Yang; J. Fan. A mass conserving level set method for detailed numerical simulation of liquid atomization. *Journal of Computational Physics*, 298:495–519, 2015.

[22] R. Meland; I. R. Gran; R. Olsen; S. T. Munkejord. Reduction of parasitic currents in level-set calculations with a consistent discretization of the surface-tension force for the CSF model. In *16th Australasian Fluid Mechanics Conference*, Gold Coast, Australia, 2007.

[23] W. Mulder; S. Osher. Computing interface motion in compressible gas dynamic. *Journal of Computational Physics*, 100:209–228, 1992.

[24] R.R. Nourgaliev; S. Wiri; N.T. Dinh; T.G. Theofanous. On improving mass conservation of level set by reducing spatial discretization errors. *International Journal of Multiphase Flow*, 31(12):1329 – 1336, 2005.

[25] Elin Olsson; Gunilla Kreiss. A conservative level set method for two phase flow. *Journal of Computational Physics*, 210(1):225 – 246, 2005.

[26] S. Osher; R. Fedkiw. *Level set methods and dynamic implicit surfaces*. Springer, New York, 2003.

[27] S. Osher; J. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[28] D. Peng; B. Merriman; S. Osher; H. K. Zhao; M. Kang. PDE-based fast local level set method. *Journal of Computational Physics*, 155:410–438, 1999.

[29] A. Pourmousa; H. Montazeri; J. Mostaghimi. Sharp Surface Tension Force for Level-Set Method in Multiphase Flow Modeling. 11th International Annual Conference on Liquid Atomization and Spray Systems, 2009.

[30] Sousa F. S.; C. M. Oishi; G. C. Buscaglia. Spurious transients of projection methods in microflow simulations. *Computer Methods in Applied Mechanics and Engineering*, 285:659–693, 2015.

[31] A. Salih; S. G. Moulic. Some numerical studies of interface advection properties of level set method. *Sadhana*, 34:271–298, 2009.

[32] C. W. Shu; S. Osher. Efficient implementation of essentially non-oscilatory shock capturing schemes ii. *Journal of Computational Physics*, 83:439–471, 1988.

[33] P. Smereka. Semi-implicit level set methods for curvature and surface diffusion motion. *Journal of Scientific Computing*, 19:439–456, 2003.

[34] S. L. Sun; W. Q. Tao. A coupled volume-of-fluid and level set (VOSET) method for computing incompressible two-phase flows. *International Journal of Heat and Mass Transfer*, 53:645–655, 2010.

[35] M. Sussman. *A level set approach for computing solutions to incompressible two-phase flow*. PhD thesis, University of California, 1994.

[36] M. Sussman; A. S. Almgren; J. B. Bell; P. Colella; L. H. Howell; M. L. Welcome. An adaptive level set approach for incompressible two-phase flows. *Journal of Computational Physics*, 148(1):81 – 124, 1999.

[37] M. Sussman; E. Fatemi. An efficient, interface preserving level set re-distancing algorithm and its application to interfacial incompressible fluid flow. *Journal of Scientific Computing*, 20:1165–1191, 1999.

[38] M. Sussman; E. G. Puckett. A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows. *Journal of Computational Physics*, 162(2):301 – 337, 2000.

[39] M. Sussman; P. Smereka; S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 114:146–159, 1994.

[40] L. $\tilde{S}$trubelj; I. Tiselj; B. Mavko. Simulations of free surface flows with implementation of surface tension and interface sharpening in the two-fluid model. *International Journal of Heat and Fluid Flow*, 30:741–750, 2009.

[41] G. Tryggvason; B. Bunner; A. Esmaeeli; D. Juric; N. Al-Rawahi; W. Tauber; J. Han; S. Nas; Y.-J. Jan. A front-tracking method for the computations of multiphase flow. *Journal of Computational Physics*, 169(2):708 – 759, 2001.

[42] G. Tryggvason; R. Scardovelli; S. Zaleski. *Direct numerical simulation of gas-liquid multiphasic flows*. Cambridge University Press, Cambridge, 2011.

[43] S. O. Unverdi; G. Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *J. Comput. Phys.*, 100(1):25–37, May 1992.

[44] M. M. Villar. *Análise numérica detalhada de escoamentos multifásicos bidimensionais*. Universidade Federal de Uberlândia, 2007.

[45] Y. F. Yap; J. C. Chai; T. N. Wong; K. C. Toh; H. Y. Zhang. A global mass correction scheme for the level-set method. *Numerical Heat Transfer, Part B: Fundamentals*, 50:455–472, 2006.

[46] P. Zaspel; M. Griebel. Solving incompressible two-phase flows on multi-gpu clusters. *Computer and Fluids*, 80:356–364, 2013.