
# CONSTRUÇÃO DE UM SISTEMA PARA SIMULAÇÃO DE FLUIDOS COM ÊNFASE NO TRATAMENTO DE BORDA

Carlos Eduardo de Souza

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Antônio Alberto Fernandes de Oliveira

Rio de Janeiro
Junho de 2012

CONSTRUÇÃO DE UM SISTEMA PARA SIMULAÇÃO DE FLUIDOS COM ÊNFASE NO TRATAMENTO DE BORDA

Carlos Eduardo de Souza

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Antônio Alberto Fernandes de Oliveira, D.Sc.

Prof. Abimael Fernando Dourado Loula, D.Sc.

Prof. Cláudio Esperança, Ph.D.

Prof. Marcelo Bernardes Vieira, D.Sc.

Prof. Ricardo Guerra Marroquim, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2012

Souza, Carlos Eduardo de

Construção de um Sistema para Simulação de Fluidos com Ênfase no Tratamento de Borda/Carlos Eduardo de Souza. – Rio de Janeiro: UFRJ/COPPE, 2012.

XVIII, 207 p.: il.; 29, 7cm.

Orientador: Antônio Alberto Fernandes de Oliveira

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2012.

Referências Bibliográficas: p. 199 – 207.

1. Navier-Stokes. 2. Particle Level Set. 3. Fast Marching. 4. Marching Cubes. 5. Superfície Livre. 6. Derivadas Convectivas. 7. Gradiente Biconjugado. I. Oliveira, Antônio Alberto Fernandes de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

*À minha família pela*
*compreensão da minha ausência*
*durante todo esse período de*
*doutorado.*

# Agradecimentos

Ao meu orientador Professor Doutor Antônio Oliveira pelo modo como me orientou nesta investigação: com grande empenho, dedicação, paciência e disponibilidade. Esta forma de orientação foi o que me permitiu superar as dificuldades deste processo. Não só eu, mas a grande maioria dos membros do LCG, se não todos, acreditam que ele possui uma inteligência muito acima da nossa média. É um grande prazer e uma aula conversar com ele sobre qualquer assunto.

Aos meus companheiros de doutorado, André Maximo, Ricardo Marroquim, Yalmar Ponce e Felipe Moura pelas grandes dicas de como programar e usar o linux e seus aplicativos. Ao professor Claudio Esperança pelas aulas informais de programação. Meia hora de conversa com ele sobre programação equivale a um curso avançado.

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo suporte financeiro.

As minhas filhas, Isabela e Ana Júlia, que mesmo tão novas copreenderam a importância dessa investida e por muitas vezes me liberaram do meu compromisso de brincar com elas. E também por serem minha melhor referência de amor.

A minha esposa, Ana Cristina, que sempre procurou proporcionar as melhores condições para minhas longas horas de estudo, e suportou todo meu cansaço e indisponibilidade durante esses longos 4 anos de doutorado. Com ela tenho aprendido a viver a dois.

Aos meus pais, Givaldo e Tuca, e minha irmã, Ana Paula, por tudo que me ensinaram e ensinam até hoje, por todo carinho e amor que me deram e ainda dão. Eles compõe meu alicérce.

As minhas irmãs Carla e Tânia, que mesmo distantes, sempre me elogiaram pelo meu esforço.

Aos amigos Carlão e Robson pela amizade e companheirismo desde os tempos de criança.

Ao companheiro Solano por sua amizade e seu maravilhoso bar, onde pude aliviar o estresse da tese e tomar a melhor cerveja de Realengo.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

# CONSTRUÇÃO DE UM SISTEMA PARA SIMULAÇÃO DE FLUIDOS COM ÊNFASE NO TRATAMENTO DE BORDA

Carlos Eduardo de Souza

Junho/2012

Orientador: Antônio Alberto Fernandes de Oliveira

Programa: Engenharia de Sistemas e Computação

Neste trabalho estudamos o uso de modelos Eulerianos para simulação de fluidos descrevendo em detalhes todos os principais passos, porém com especial ênfase no traçado da interface ar-líquido, empregando um arcabouço baseado na teoria "Level Sets". Nesse contexto, entre outras questões que requerem considerações mais específicas, esse trabalho foca em três que podem ser consideradas as mais fundamentais:

1. Como inicializar velocidades nas células que acabaram de se tornar líquido?
2. Como calcular velocidades artificiais em células ar perto da fronteira líquido?
3. Como evoluir o volume líquido evitando perda ou ganho de volume?

Uma metodologia alternativa para cada uma destas três questões é proposta com o objetivo de eliminar artefatos ou seqüências irrealistas que possam ser observadas em versões menos precisas, bem como para manter o volume sob controle. Estas metodologias não aumentam significativamente nem o esforço computacional, nem os requisitos de armazenamento da abordagem clássica.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

COMPUTER SIMULATION OF INCOMPRESSIBLE FLUID WITH FREE SURFACE

Carlos Eduardo de Souza

June/2012

Advisor: Antônio Alberto Fernandes de Oliveira

Department: Systems Engineering and Computer Science

In this work we study the use of Eulerian models for fluid simulation describing in details all the main steps but focusing specially on the tracking of the air-fluid interface employing a framework based on the Level Set theory. In that context among other questions that require more specific considerations, this work focus on three which can be considered the most fundamental ones:

1. How to initialize velocities at the cells that have just become fluid?
2. How to compute artificial velocities at the air cells close to the fluid border?
3. How to evolve the fluid volume avoiding volume loss or gain?

An alternative methodology for each one of these three issues is proposed with the aim of eliminating artifacts or unrealistic sequences that can be observed in less accurate versions as well as to maintain volume under control. These methodologies do not increase significantly neither the computational effort nor the storage requirements of the classical approach.

# Sumário

# Lista de Figuras

# Lista de Algoritmos

# Lista de Tabelas

# Capítulo 1

# Introdução

## 1.1 O contexto do Trabalho e o Enfoque Adotado

Este trabalho trata da construção de um sistema para efetuar simulações de fluidos com ênfase especial no processo de evolução da borda. O contexto básico de seu desenvolvimento é o de se resolver a equação de Navier-Stokes no formato clássico

$$\rho\frac{\partial \boldsymbol{v}}{\partial t} + \rho\nabla.(\boldsymbol{v}\boldsymbol{v}^T) = -\nabla p + \mu\nabla^2\boldsymbol{v} + \rho\boldsymbol{g} \tag{1.1}$$

em domínios com fronteiras móveis ao logo do tempo, onde $\boldsymbol{v}$ é a velocidade, $p$, a pressão, $\mu$, o coeficiente de viscosidade e $g$ representa a aceleração determinada por forças externas. $\nabla$ e $\nabla^2$ são, respectivamente os operadores gradiente e laplaciano. Essa equação deve ser resolvida sob a condição de incompressibilidade expressa por:

$$\nabla.v = 0 \tag{1.2}$$

onde $\nabla.$ é o operador divergente.

A velocidade e a pressão são discretizadas segundo o modelo das "staggered grids", em uma grade regular cúbica, com as pressões definidas no centro das células e cada componente da velocidade amostrada nas faces que são ortogonais a ela.

Para dar um tratamento especial a evolução da interface ar-liquido (utilizaremos o termo borda para se referir a essa interface), vai se trabalhar com células cuja geometria deixa de ser cúbica e passa a ser determinada pelo corte que a borda efetua nela. Essas células são denominadas células de borda. À estrutura dessas células se acrescenta uma face dita externa - por separar a célula do exterior da

massa fluida - para representar sua interseção com a borda. A representação da geometria dessa face não precisa ser explicitada de forma completa. Para alguns propósitos é mais prático adotar para ela uma representação planar. Para outros ela pode ser representada por suas intersecções com as faces da célula, obtidas por exemplo, por um procedimento de PLIC ("Piecewise Linear Interface Calculation") como o algoritmo de "Marching Cubes".

Para as células de borda a idéia é que a dinâmica do fluido dentro delas seja inteiramente expressa por transportes de fluxo de algumas de suas faces para outras. Identificam-se linhas de corrente computadas dentro da célula numa dada iteração com as trajetórias percorridas por partículas de fluido desde o momento de sua entrada na célula até sairem dela no instante a que se refere a iteração. Isto é, se assume que o fluxo dentro da célula se manteve estacionário durante o período em que as partículas, que estão saindo dela num dado momento, passaram transitando dentro da célula. Essa identificação permite propor um esquema não semi-lagrangeano para simular a advecção que não requer o conhecimento da geometria da borda no interior da célula.

Observamos que há inúmeros trabalhos em que o refinamento no entorno da borda se destina apenas a obtenção de seu traçado. Aqui se pretende que o recorte determinado pela interface em uma célula influencie também a evolução das velocidades em suas faces. Assim os procedimentos relativos à advecção do campo de velocidades e a própria construção do sistema de Poisson, que são utilizados para células regulares precisam ser adaptados para levar em conta o corte feito pela borda.

Visando manter o custo computacional das operações envolvendo células de borda da mesma ordem do despendido para células regulares, se propõe utilizar um pré-cômputo para efetuar determinados cálculos específicos. Definida uma resolução, coeficientes e medidas empregados nesses cálculos, podem num pré-processamento, ser tabelados em função da representação da geometria da borda que é empregada ou da velocidade nas faces de uma célula de borda. A questão é como fazer com que a massa de dados que precisa ser gerada se mantenha de tamanho módico.

Idealmente, tendo uma estrutura flexível para a representação de células de borda, algoritmos de duas das classes principais da área de "Interface Tracking" - "Level Sets"e "Volume of Fluid" - podem ser utilizados ao longo de uma mesma

simulação, passando-se de um a outro se for detectado um evento que indique que esse outro deve ser preferido. Mais ainda, eles podem ser utilizados em porções distintas da borda em uma mesma iteração. Isso, entretanto só se torna cabível porque a borda resultante de qualquer um dos métodos é submetida a um procedimento de ajuste às arestas da grade, operação que será referida aqui por regularização. Essa regularização, além do mais, permite realizar mudanças topológicas sem que sejam necessários procedimentos específicos para detectá-las ou efetuá-las.

Entre os eventos referidos acima, um que é de fácil detecção é a perda de volume. Embora neste trabalho não se elabore acerca de formulações para a transição de uma estratégia de evolução da borda de uma classe para outra, vai se apresentar uma estratégia do tipo "forward"[1] focada na preservação de volume que é intercambiável com um modelo de evolução clássico da metodologia que usa "Level Sets".

Dando um tratamento especial a borda se tem em vista que um grau de realismo satisfatório possa ser produzido, mesmo que a massa fluida e em especial seu interior seja tratado num nível de resolução mais baixo. O tempo gasto pelo procedimento notoriamente mais dispendioso de todo o processo, a resolução do sistema de Poisson, cresce rapidamente com a resolução da grade. Resolvê-lo em resolução menor pode, certamente, evitar que o tempo da simulação se torne excessivo.

Esclarecemos que apesar do trabalho se reportar à simulação de fluidos, nosso interesse se restringe essencialmente ao contexto da área de computação gráfica. De fato, a motivação para o desenvolvimento do trabalho foi o desejo de contar com uma ferramenta que pudesse evitar a criação de artefatos ou o surgimento de dinâmicas não plausíveis. Anomalias desses tipos puderam ser identificadas visualmente em exemplos produzidos por um simulador nos moldes clássicos que foi construído na primeira fase deste trabalho. Algumas dessas deficiências puderam ser detectadas nos próprios vídeos produzidos mas outras precisaram que se analisassem instâncias específicas ou as saídas de uma seqüência de iterações. As causas dessas anomalias são deficiências inerentes às metodologias empregadas como difusão numérica, "aliasing"ou modelos de interpolação inadequados.

[1] Em que os pontos onde a borda corta as arestas da malha são movidos para a frente, isto é no sentido da velocidade. Comparar com a estratégia "backward", em que cada ponto é movido para trás, empregada quando se utiliza um esquema semi-lagrangeano para fazer a advecção da função distância à borda dentro de um procedimento típico do padrão "Level Sets".

Feitas essas considerações que resumem o escopo e os objetivos do trabalho, passamos nas seções seguintes a comentar com maior detalhamento a opção pelo modelo euleriano e os componentes principais da estrutura que vamos utilizar, como células de borda e faces externas.

## 1.2 Modelo Euleriano ou Lagrangeano

A evolução da borda é certamente o aspecto mais relevante de um processo de simulação de fluidos no que diz respeito a se obter visualizações realistas. Além disso, devido a potencial irregularidade da superfície livre de um fluido em movimento, é um tema que ainda apresenta desafios os quais deram origem a uma série de propostas apresentadas recentemente na literatura. Ver, por exemplo, [43].

Modelos lagrangeanos - do tipo SPH - vem sendo utilizados para a geração de vídeos de fluidos em movimento com resultados que um observador mais atento e com conhecimento de causa, poderia identificar como não realistas num ou noutro aspecto. Por exemplo, métodos que seguem o paradigma SPH puro não conservam volume. Mas devido a velocidade em que a evolução do fluido decorre, eles podem se tornar pouco perceptíveis sendo o produto final considerado convincente e mesmo agradável, em especial por um observador leigo. Modelos eulerianos, por outro lado, apesar de conhecidas imperfeições, podem produzir resultados mais próximos da realidade física, até porque não incluem modelos de difusão com parâmetros que não tem qualquer justificativa física.

Além disso, embora a obtenção de um alto grau de realismo em tempo real via esses modelos, ainda seja um desafio, propostas para sua implementação em GPU, entre elas a de [21], permitem acreditar que se caminha nesse sentido.

Para justificar a opção por se trabalhar com uma grade fixa, se acrescenta ainda que se o foco é a evolução da borda, então conta o fato dela ser controlada com muito mais facilidade quando se tem essa grade do que dispondo apenas de partículas com posição aleatória - Técnicas de interface tracking são apresentadas majoritariamente no contexto euleriano. Isso não quer dizer, no entanto, que não se possa recorrer a modelos híbridos, com finalidades diversas, tais como, criar aleatoriedade, produzir efeitos especiais ou simplesmente para tornar mais fino o recorte da borda.

## 1.3 Células de Borda

Assumida uma abordagem euleriana, uma primeira idéia para dar um tratamento diferenciado à borda seria a de se utilizar uma resolução mais alta no seu entorno, o que criaria dois níveis de células - microcélulas cobrindo as proximidades da borda e macrocélulas que agrupam as micro nesse entorno e que se estenderiam por todo o restante do volume fluido. Essa possibilidade, entretanto, apresenta algumas questões que requerem um tratamento adequado:

- O intervalo de tempo entre as iterações, que é determinado pelas condições CFL, depende do tamanho das células e assim, é preciso sincronizar a evolução nos dois níveis.

- Como impedir que a maior resolução das microcélulas acabe se refletindo em maior complexidade computacional no processamento de uma iteração a nível macro. Exemplo, se tivermos que resolver um sistema de Poisson para toda a grade incluindo células nos dois níveis perdemos boa parte da vantagem de resolvê-lo apenas no nível macro.

- Resolver a questão anterior, requer que as células menores sejam agregadas em macrocélulas. Dado que dentro de uma macrocélula, haverá células com e sem fluido, como então aproximar a estrutura determinada pelas microcélulas dentro dessa célula maior?

Por isso não nos fixamos na idéia de simplesmente aumentar a resolução no entorno da borda. É preferível dar às células cortadas por ela uma estrutura mais complexa que possa também, incluir a possibilidade do refinamento da malha dentro dela. Conforme já foi dito, essa estrutura constitui o que se vai chamar de célula de borda. À estrutura dessas células se acrescenta uma face chamada "face externa"para representar sua interseção com a borda.

Há propostas em que se admite a possibilidade da borda cortar uma célula sem atingir suas arestas. Em outras a representação da face externa é dada por uma função polinomial de grau superior. Aqui, como uma das finalidades é aproveitar a informação proveniente do "Marching Cubes", assumimos que a evolução da borda dentro de uma célula pode ser determinada a partir dos pontos de corte que ela

determina nas arestas da célula. Com isso queremos dizer que a representação da geometria da face externa não precisa ser conhecida inteiramente. Dela fazemos uso apenas de estimativas de sua área e orientação de forma que conhecendo o fluxo [2] que a atravessa possamos calcular a velocidade com que ela se desloca para fazer a borda evoluir. Essas estimativas podem ser obtidas considerando o corte que uma aproximação planar local da borda determina na célula. Esse corte será referido aqui como a versão planar da face externa. Alternativamente, embora isso seja mais complexo, pode-se não recorrer a versão planar, computando-se a evolução da borda diretamente a partir de informações sobre a componente de velocidade e o traçado da borda que se tem em cada face regular da célula.

Há certamente situações em que a aproximação linear por partes da face externa que é produzida pelo "Marching Cubes"não é apropriada. Por exemplo, em casos em que a borda corta uma célula sem interceptar suas arestas. Esses casos, que ocorrem quando se pretende uma precisão sub-célula, não serão abordados neste trabalho. Quando a borda é dada explicitamente por uma malha triangulada [78] apresenta uma metodologia interessante para lidar com essas situações.

A definição de célula de borda permite que ela seja introduzida gradualmente na massa fluida evitando que isso só aconteça quando a célula estiver cheia. É que nesse caso, a inicialização das velocidades nas novas células fluidas pode apresentar alguma dificuldade: Como não temos idéia de quanto tempo essa célula levou para ficar cheia e nesse tempo as forças externas atuaram, não temos parâmetros para computar a velocidade advectada para uma dada face, especialmente se essa face não era adjacente à massa fluida no passo anterior. Mais ainda, parte do fluxo advectado para a nova célula pode provir de células que ainda são consideradas AR por não estarem inteiramente cheias. Para elas podem existir apenas velocidades artificiais computadas sem preocupações relativas à incompressibilidade e fora do esquema das "staggered meshes"(Normalmente essas velocidades são computadas nos vértices da malha ainda na porção AR).

Assim se acaba recorrendo a mecanismos de extrapolação ou mesmo à cópia de componentes de velocidade definidas em células próximas que já são fluidas.

[2] Nesse trabalho os termos fluxo e vazão são empregados indistintamente para fazer referência a velocidade x área.

Ocorre que proximidade não significa necessariamente similaridade nos valores das velocidades. Da mesma forma esquemas simples de extrapolação podem efetuá-la empregando dados não correlatos.

Além disso, a alternativa de ao inicializar-se uma célula, já se resolver a condição de incompressibilidade sem deixar isso por conta do sistema de Poisson pode gerar resultados irrealistas.

Um simples exemplo 2D: Duas frentes se deslocando em sentidos opostos com velocidade horizontal $v$ se encontram, cobrindo, então, uma célula $C$. Se inicializarmos $C$ atribuindo a sua face horizontal livre uma velocidade $2v$ para zerar o divergente vamos gerar um "esguicho"vertical em $C$. Na solução obtida pelo Poisson as velocidades horizontais são diminuídas e o fluxo na vertical determinado pelo choque das frentes é distribuído entre C e outras células. Entretanto, deve-se observar que, uma vez feita uma inicialização errada as forças de pressão não tem como corrigi-la.

Retomando a questão de não se saber quanto tempo a célula de borda levou para ficar cheia adotamos a seguinte assertiva, que não corresponde integralmente a realidade física mas permite uma simplificação considerável: Na redistribuição de fluxos dentro de uma célula na etapa de advecção, se assume que o fluxo entre uma face de entrada e outra de saída, ambas já contendo fluido, possa ser computado tomando com o base apenas o valor atual do fluxo de entrada sem levar em conta o tempo de percurso dentro da célula. Essa suposição permite abstrair a forma da borda dentro da célula quando se faz a advecção e obtê-la considerando apenas os cortes que ela determina nas faces. Reiteramos que essa propriedade se aplica apenas a transferência de fluxos entre faces já atingidas pelo volume fluido não estendendo, por si só esse volume.

Uma outra possibilidade, conforme já antecipado na seção 1.1, que o modelo de célula de borda empregado neste trabalho permite, é a de, com base numa estrutura de dados mutável, passar de uma metodologia padrão "Level Sets"para outra incorporando preceitos da classe "Volume of Fluid"numa mesma simulação. Passa-se de um para o outro se for identificada uma condição, por exemplo, a perda de volume que indique que esse outro método deve ser preferido. Mais ainda, eles podem ser utilizados em regiões distintas da borda em uma mesma iteração. Isso se torna plausível porque a borda obtida de qualquer um dos métodos é submetida a um

procedimento de regularização.

Essa possibilidade de se mudar a metodologia e a própria estrutura da célula de borda podem ser usadas para aprimorar a simulação em vários aspectos. Exemplos:

- Evitar o surgimento de dinâmicas inesperadas especialmente onde há contato com os limites do container - Implementadores reclamam, por exemplo, do fato do fluido "colar nas paredes subindo mais do que o plausivelmente esperado". Isso em implementações menos acuradas é decorrente do fato da simulação junto às paredes do container se tornar dependente da resolução. Maior resolução, maiores velocidades, mais o fluido sobe.

- Impedir que filetes de espessura inferior a uma célula tenham sua evolução detida. Esses filetes podem ser produzidos por um "input"reduzido mas também podem ser formados junto às junções de paredes do container qualquer que seja a entrada.

- Evitar a perda de volume decorrente de atrasos na evolução da massa fluida.

Esse último caso fornece um exemplo simples de aplicação do esquema. Durante toda a simulação se mantém um valor aproximado da diferença: $\big(\int$ ( Input - Output de fluxo) $dt -$ Volume Corrente da parte fluida$\big)$. Esse valor não precisa ser nulo até por se tratar de uma aproximação. Mas se ele ultrapassar um dado limite, pode-se acionar um procedimento específico para identificar e tratar adequadamente as células onde potencialmente se dá essa perda de volume. Por exemplo células de borda onde a velocidade é transversal a sua face externa, mas que por um problema relativo à resolução ou à alguma idiossincrasia do próprio procedimento de evolução não se deslocam como esperado. A essas células ou a uma vizinhança delas se passaria então a aplicar um procedimento da classe "Volume of Fluid"cuja razão de ser é a preservação do volume. Além dessa possibilidade de se intercambiar metodologias podemos em função de tudo que foi dito até agora sintetizar os objetivos da proposta nos 5 seguintes itens:

1. Aproveitar a informação relativa ao corte de cada face regular, produzido pela borda obtida através do Marching-Cubes, no cômputo da advecção da velocidade numa célula de borda e na montagem do Sistema de Poisson.

2. Utilizar formulações para o cômputo da área da face externa e de sua orientação que dependam apenas da dimensão da área fluida em cada face regular.

3. Na medida do possível fazer com que todas as operações efetuadas num passo da simulação envolvendo uma célula de borda, incluindo as utilizadas para se obter a orientação e a área da face externa, possam ser feitas a partir de dados computados nas faces sem precisar explicitar o traçado da borda no interior da célula.

4. Computar novos pontos de interseção da borda com as arestas ao invés de obtê-los por interpolação a partir de estimativas da distância de seus vértices à borda.

5. Dispor de tabelamentos obtidos a priori para manter a complexidade das operações realizadas nas células de borda comparável a efetuada para células regulares.

### 1.3.1 Representação da Face Externa

Ao obter a face externa empregando-se os dados produzidos pelo "Marching Cubes", conforme já foi mencionado, podem acontecer recortes mais complicados por conta da interseção da borda com uma célula (cúbica) poder ter várias componentes conexas. Vamos agora precisar melhor como todos as possibilidades disso acontecer podem ser tratadas.

Para cada componente conexa da interseção do volume fluido com a célula regular deve ser criada uma célula de borda. Quando, ao contrário, essa interseção é conexa, mas sua fronteira dentro da célula não é, temos o caso em que a face externa tem várias componentes cada uma delas com sua própria velocidade. Os dois casos estão representados na figura 1.1.

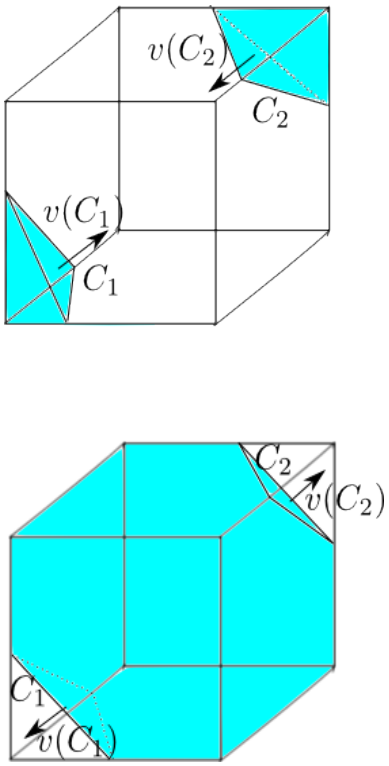



Figura 1.1: Casos em que a interseção da massa fluida com a célula não é conexa e quando ela é conexa mas a interseção da borda não é. Aqui, $C_1$ e $C_2$ representam as componentes da face externa do volume fluido no interior da célula, e $v(C_1)$ e $v(C_2)$ as velocidades nas respectivas faces.

No primeiro caso, as velocidades das faces externas, que são triangulares, podem ser obtidas diretamente a partir das definidas nas faces regulares da célula de borda. No segundo, as várias componentes devem ser consideradas na montagem do sistema de Poisson, mas com respeito a advecção não há diferenças em relação ao tratamento dado no caso usual a menos que uma mesma face regular seja cortada por duas componentes da borda. Esse último caso pode ser resolvido considerando-se uma componente de cada vez e se computando, em cada caso, o complemento em relação ao fluxo que atravessa a célula toda. Tirando desse fluxo a soma dos dois complementos se tem o resultado desejado.

A partir de cada componente dessa representação pode-se obter outra mais simples e independente de opções quanto ao traçado da borda dentro da célula, se encontrando o plano que melhor se ajusta aos vértices da representação entre os que cortam as mesmas arestas da célula que ela. A componente seria então aproximada

pela interseção desse plano com a célula constituindo o que já denominamos antes de versão planar.

A versão planar permite falar em normal à componente. Na realidade ela é obtida por um processo que determina essa normal e simultaneamente uma estimativa da área dessa componente. A partir dela se pode estimar também o volume da massa fluida dentro da célula de borda, que é usado pelo sistema de Poisson.

Tendo a vazão que deve atravessar a face externa (ou cada uma de suas componentes) além de sua área podemos estimar a velocidade com que essa face (ou componente) evolui na direção normal a ela. Essa projeção pode ser empregada para atualizar as distâncias a borda nos vértices ou centros de células da grade, num esquema que visa evitar a perda de volume. Ainda menos suscetível a essa perda ele se torna se as novas interseções com as arestas da grade são calculadas diretamente - e não via interpolações. Isso, entretanto requer uma metodologia que evite que a superfície da borda deslocada tenha buracos, além de, é claro não ser pesada computacionalmente.

Reiteramos que a versão planar de uma face externa, independe de como a borda é dentro célula ou considerando a saída do "Marching Cubes", da triangulação restrita por suas interseções com as faces regulares, empregada por ele para representar a borda dentro da célula. Ela é função apenas do corte da borda nas faces. Sua área e orientação podem ser obtidas computando-se, apenas a partir de dados relativos a própria célula, o gradiente de uma função. Essa função assume no centro de cada face regular o valor da área de sua porção fluida dessa face, conforme será visto no capítulo 7.

## 1.4 Estrutura do Trabalho e Contribuições

Além de introduzir a metodologia descrita nas seções anteriores, pretendemos que esse trabalho possa servir como referência a outros que se pretende produzir dentro dessa mesma linha. Em vista disso fazemos, no capítulo 3 uma revisão indicando como as equações de Navier-Stokes são obtidas e no 4 uma exposição acerca do processo de discretização e aproximação numérica dessas equações, descrevendo alguns métodos tradicionalmente empregados nesse processo.

No capítulo 5 apresentamos alguns conceitos básicos do processo de evolução da interface ar-líquido e fazemos uma descrição do método dos Level Sets, [65], [63], conjugado ao do Fast Marching, focando alguns aspectos teóricos e computacionais. Terminamos o capítulo com uma breve descrição do algoritmo "Marching Cubes"explicitando os pontos que vamos explorar posteriormente.

Num esquema clássico são esses os métodos utilizados na evolução e representação da interface ar-líquido. Esses três capítulos de revisão são precedidos, no capítulo 2, por um histórico da evolução da metodologia empregada nas áreas de simulação de fluidos num contexto euleriano e de interface tracking seguido da descrição de trabalhos recentes nessas áreas.

Nos capítulos 6 e 7, introduzimos duas formas de tratamento das chamadas células de borda. No 6 é apresentado o método que se emprega para redistribuir o fluxo dentro de uma célula de borda mas não se faz ainda uso das informações acerca do traçado da superfície fornecidas pelo Marching-Cubes. Para dispensar essas informações se faz uso de um modelo para representar a parte fluida de uma face regular que se inspira na forma como se representa o volume fluido dentro de uma célula nos primeiros algoritmos da classe Volume-of-Fluid. Se visa essencialmente introduzir células na porção fluida de forma gradual sem a pretensão de gerar elementos que venham a ser empregados para fazer a evolução da borda.

No capítulo 7 se apresenta, então, uma proposta integrando os módulos de resolução de Navier Stokes e evolução da borda com o que determina o seu traçado. Nessa proposta esse traçado interfere na obtenção das velocidades numa célula de borda e por sua vez, é atualizado diretamente a partir das velocidades obtidas para as faces externas. Na metodologia descrita nesse capítulo não há processos correspondentes ao de atualização de Level Sets ou de Fast Marching.

No capítulo 8 fazemos uma introdução ao sistema computacional que foi implementado neste trabalho apresentando sua estrutura em termos gerais e como utilizá-lo. Também se indica um site onde se pode visualizar os vídeos produzidos. Se apresentam ainda algumas tabelas reportando resultados computacionais descrevendo, por exemplo, a relação entre o tempo gasto em cada etapa da simulação, o tamanho da grade e o número de células de borda. Ou ainda, comparando a perda de volume utilizando-se o método semi-lagrangeano, dentro de um esquema "Level

Sets"clássico com a que é produzida quando se emprega um método em que a função distância a borda é cpmputada levando-se em conta seu traçado estabelecido pelo "Marching-Cubes".

Finalmente no capítulo 9 apresentamos conclusões e listamos uma série de trabalhos futuros.

No que se refere a contribuições, para as atividades de pesquisa do Laboratório de Computação Gráfica da COPPE/UFRJ, consideramos relevante a implementação de um sistema para simulação de fluidos passível de ser utilizado em um contexto relativamente diverso de aplicações e apresentando alternativas diferentes para a realização das operações principais. A expectativa é que esse sistema possa servir como ponto de partida para trabalhos futuros e que novas potencialidades sejam acrescentadas a ele.

A análise dos resultados obtidos por essa implementação motivou a apresentação de propostas enfocando, exatamente, a superação das deficiências observadas. Essas propostas incorporam idéias introduzidas em diferentes trabalhos - por exemplo, [37] e [10] - mas que foram elaboradas, ajustadas a um novo contexto e por fim integradas de uma forma que não encontramos na literatura que dispusemos.

Entre o que consideramos elementos de inovação constantes desse trabalho podemos, num contexto mais geral, citar a interdependência entre a forma como evoluem o traçado da borda e o campo de velocidades e a própria proposta de se mudar a metodologia empregada para evoluir a borda diante da constatação ou da possibilidade de um comportamento inadequado. Entre os pontos específicos onde há componentes de inovação, podem ser listados os modelos empregados para fazer a advecção dentro de uma célula de borda, para determinar a área e a orientação de sua face externa, para determinar os novos pontos de corte da borda nas arestas diretamente -isto é, sem efetuar interpolações - e a própria proposta para se introduzir células na massa fluida gradualmente reportada no capítulo 6.

# Capítulo 2

# Trabalhos Relacionados

Fazemos neste capítulo uma revisão da bibliografia que consideramos relevante para o tema tratado aqui, seja pelo valor histórico ou por representarem o estado da arte em simulação de fluidos vinculada a computação gráfica. Esclarecemos que nos limitamos aqui ao contexto euleriano de forma que classes de método bastante populares como SPH (Smoothed Particle Hydrodynamics) e LBM (Lattice Boltzmann methods) não abordados. Há também uma série de artigos, [11], [16], [49], [18], [67], [22], [28], [45], [48], [49], dos quais nos inteiramos durante a realização do trabalho, mas que não serão abordados aqui, porque por concisão demos preferência aos mais antigos e aos de alguma maneira vinculados a metodologia proposta neste trabalho. Isso não significa entretanto que eles não tenham sido importantes para o desenvolvimento da área.

## 2.1 Um Breve Histórico

Um dos primeiros trabalhos publicados sobre simulação computacional de fluido com superfície livre foi o trabalho de Francis Harvey Harlow [27], publicado em 1957, neste trabalho é apresentado o método "Particle-in-Cell" (PIC) [9] [38] [39] [40] , no qual existe um conjunto euleriano de células juntamente com um conjunto lagrangeano de partículas marcadoras que se movem através dessas células. O método "Particle-in-Cell" pode ser dividido em três fases:

**"Phase A: Tentative Cell-Wise Calculations"** As componentes de velocidade em cada célula são avançadas para novos valores de acordo com o gradiente

de pressão:

As equações de movimento são aproximadas por

$$\rho_{i,j}\left(\frac{du}{dt}\right)_{i,j} = -\frac{1}{2h}\left(p_{i+1,j} - p_{i-1,j}\right) \tag{2.1}$$

$$\rho_{i,j}\left(\frac{dv}{dt}\right)_{i,j} = -\frac{1}{2h}\left(p_{i,j+1} - p_{i,j-1}\right), \tag{2.2}$$

onde $\rho$ é a densidade, $u$ e $v$ as componentes de velocidade e $p$ a pressão. As novas componentes de velocidade são, nesse caso, dadas por

$$\tilde{u}_{i,j} = u_{i,j} + \left(\frac{du}{dt}\right)_{i,j}\Delta t \tag{2.3}$$

$$\tilde{v}_{i,j} = v_{i,j} + \left(\frac{dv}{dt}\right)_{i,j}\Delta t \tag{2.4}$$

Para efetuar os cálculos nas equações 2.1 e 2.2 ao longo das fronteiras, são definidos valores de pressão e velocidade em células externas adjacentes ao fluido, de maneira que uma célula vazia adjacente a uma célula contendo fluido tenha a mesma velocidade e valor negativo da pressão da célula para qual os cálculos estão sendo efetuados.

**"Phase B: Particle Movement"** Cada partícula é movida de acordo com a velocidade dada pela interpolação linear entre as velocidades das quatro células mais próximas a partícula;

**"Phase C: Velocity Repartition"** Para cada célula que sofreu entrada e/ou saída de alguma partícula, novas componentes de velocidade são então calculadas. Tomando a média, ponderada pela massa, das velocidades das partículas que permaneceram ou entraram na células em questão. Se nenhuma partícula atravessou a fronteira da célula, então essa fase é ignorada.

Uma restrição necessária para o funcionamento do método é que uma partícula não se mova mais do que uma célula por iteração, $u_{max}\Delta t < h$. Na época de seu desenvolvimento, método foi implementado em máquinas IBM 701 e 704, o que restringia o número de partículas por célula ser algo em torno de 5 partículas.

Em 1965 Francis Harvey Harlow publicou o método "Marker-and-Cell" (MAC) [42] [60] [75], como uma nova técnica para investigação numérica do fluxo de fluido

incompressível com superfície livre, onde as equações de Navier-Stokes são aproximadas por diferenças finitas. Neste método, diferentemente do método PIC, as partículas marcadoras são utilizadas apenas para traçar a evolução da superfície livre. As partículas marcadoras são inicialmente colocadas nas células contendo fluido e são subsequentemente movidas de acordo com uma velocidade local obtida pela interpolação linear das velocidades nos vértices próximos à partícula, da malha euleriana de células.

Também diferentemente do método PIC, no qual as componentes de velocidade são colocadas no centro da célula, no método MAC as componentes de velocidade são colocadas, no caso bidimensional, no centro das arestas de cada célula, de forma que a cada uma das quatro arestas seja associado um valor para a componente de velocidade que lhe é normal. Este arranjo das componentes de velocidade tem como objetivo expressar de forma mais conveniente a restrição de divergência nula do campo de velocidade e passou a ser conhecido como "staggered grid".

Em 1970, Amsdem e Harlow publicaram uma versão simplificada do método MAC, conhecida como "Simplified Marker-and-Cell" - (SMAC) [4] a qual contornou as dificuldades do método original dividindo o cômputo do campo velocidade em duas etapas. Numa primeira etapa se calcula um campo de velocidade provisório. Na segunda se aplica uma função potencial a esse campo para garantir a incompressibilidade do fluido. Nesse artigo Amsdem e Harlow descrevem um programa específico, denominado "ZUNI", que implementa o método SMAC. O programa ZUNI foi usado para fluxos bidimensionais em coordenadas retangulares e cilíndricas. Ele já tratava fluxo com superfície livre e condições "slip" e "no-slip" aplicadas as fronteiras rígidas. Além disso, provisões foram feitas para incluir também "inflows", "outflows" e obstáculos retangulares dentro do campo de fluxo.

No anos 80 foram publicadas várias aplicações dos métodos PIC e MAC: Scilian e Hirt [68] desenvolveram um sistema para predição do nível de contaminantes na atmosfera, "Continmente Atmosphere Predction" (CAP) usando o método PIC para modelar o fluxo de um contaminante. Miyata e Nishimura [62] usaram SMAC para simulação de ondas geradas por navios de configuração arbitrária 3D e Miyata [61] usou o SMAC para simulação de quebra de ondas sobre corpos circulares e elípticos. Ainda nos anos 80, McQueen e Rutter [46] e Markham e Proctor [32],

sob o patrocínio da indústria de fornecimento de eletricidade do Reino Unido, "UK Energy Industry", descreveram modificações para o código original ZUNI, visando melhorar sua perfomance [60]. Essencialmente, eles aplicaram uma rotina para o cálculo do passo no tempo a cada iteração e pré-condicionaram o solver empregando gradientes conjugados para equação de Poisson.

Em 1994 Tomé e McKee[75] extenderam o trabalho de McQueen e Rutter [46] e desenvolveram uma versão melhorada do SMAC, denominada GENSMAC, para tratamento de fluxos de fluido com superfície livre em domínios gerais. A partir daí, Tomé e seu grupo de pesquisadores brasileiros publicaram diversos trabalhos relacionados ao GENSMAC, dentre eles o GENSMAC 3D, em 2001, [76]. Nesse último é apresentada uma extensão do código GENSMAC, onde a superfície do fluido é representada por uma superfície linear por partes composta por quadriláteros e triângulos contendo partículas marcadoras em seus vértices.

## 2.2 Métodos para Interface Tracking

Tratar com precisão a questão do "interface-tracking" (acompanhamento de fronteiras) e as questões de "merging" (junção) e "splitting" (separação) de porções da superfície livre continua sendo, desde os métodos PIC e MAC originais, até hoje, um grande desafio na área de simulação de fluidos. Os principais métodos para acompanhamento de fronteiras podem ser divididos, ainda que com alguma sobreposição, em três categorias [60] [56]: "front-tracking" , "volume of fluid" e "level set", havendo ainda métodos híbridos.

### 2.2.1 Método Front-Tracking

O método front-tracking [77] [35] é um método Lagrangeano para acompanhamento de interfaces, onde a interface é representada usando-se um lista ordenada de partículas marcadoras e funções polinomias sobre essas partículas. A cada iteração, as partículas são devidamente advecionadas e então novos polinômios são definidos. O método front-tracking não suporta mudanças topológicas como merging e splitting [17], e uma vez que tais mudanças topológicas acontecem comumente em simulação de fluidos com superfície live, nós não iremos nos aprofundar em tal método. Além

disso, deve se observar que a própria área de modelagem geométrica baseada em modelos polinomiais perdem vitalidade na era das placas gráficas. Metodologias calçadas nesse tipo de modelagem se tornaram igualmente objeto de interesse menor.

### 2.2.2 Método Volume of Fluid

O método volume of fluid (VOF) [44] é um método para acompanhamento de interfaces, que trabalha com a função de pertinência da parte fluida ($\chi$). Essa função vale 1 em qualquer região da célula ocupada pelo fluido e 0 caso contrário. A fração, $\overline{F}$, do volume $V$ de uma célula cortada/atingida pelo fluido pode ser dada por

$$\overline{F} = \frac{1}{V}\int_V \chi dV. \tag{2.5}$$

Essa função em princípio definida apenas para as células da malha pode ser estendida para qualquer ponto $\boldsymbol{p} = (x, y, z)$ do $R^3$. Assumindo que $\boldsymbol{p}$ é o vértice base - isto é, vértice inferior esquerdo e a frente - de uma célula transladada, se a malha é uniforme, e $h$ é o comprimento de uma aresta da célula, essa função pode ser expressa por

$$\overline{F} = \frac{1}{V}\int_x^{x+h}\int_y^{y+h}\int_z^{z+h} \chi \quad dxdydz \tag{2.6}$$

Agora podemos diferenciar $\overline{F}$ e então pela lei de conservação de um escalar passivo [37], a evolução de $\overline{F}$ com o tempo pode ser traduzida pela expressão:

$$\frac{\partial \overline{F}}{\partial t} + \boldsymbol{v}.\nabla\overline{F} = 0. \tag{2.7}$$

onde $\boldsymbol{v}$ é a velocidade do fluido no ponto em questão.

#### Estimativa do Vetor Normal a Superfície

Assuma que em cada célula $C$ a borda tem uma orientação única. Então a direção normal a superfície nessa célula ($\boldsymbol{m}$), pode ser estimada através da expressão:

$$\boldsymbol{m} = \nabla\overline{F}(\boldsymbol{p}) \tag{2.8}$$

onde $\boldsymbol{p}$ é o vértice base de $C$. Essa igualdade vem da interpretação do gradiente como direção de máximo acréscimo da função. Se $\boldsymbol{p}$ não é um vértice da malha então a orientação $\boldsymbol{m} = (m_1, m_2, m_3)$ da borda dentro da célula transladada que se

deve considerar pode não ser mais a única e a correlação entre $\nabla\overline{F}$ e ela não é mais direta.

### Determinação da Superfície Planar

O plano da borda dentro da célula $C$ pode ser definido por

$$m_1 x + m_2 y + m_3 z = \alpha \tag{2.9}$$

onde $\alpha$ é a distância entre a superfície e a origem que podemos considerar sendo o próprio vértice base da célula. Os pontos nos quais a superfície intersecta os eixos x, y e z são dados respectivamente por $\left(\frac{\alpha}{m_1}, 0, 0\right)$, $\left(0, \frac{\alpha}{m_2}, 0\right)$ e $\left(0, 0, \frac{\alpha}{m_3}\right)$ e conforme visto em [37] o volume, $V$, de fluido na célula em questão pode ser dado por

$$\begin{aligned} V = \; & \tfrac{1}{6m_1m_2m_3}\left[\alpha^3 - \textstyle\sum_1^3 H(\alpha - m_j h)(\alpha - m_j h)^3\right. \\ & \left. + \textstyle\sum_1^3 H(\alpha - \alpha_{max} + m_j h)(\alpha - \alpha_{max} + m_j h)^3\right] \end{aligned} \tag{2.10}$$

onde $\alpha_{max} = h(m_1 + m_2 + m_3)$ e $H$ é a função de Heaviside: 1 para não negativos e 0 em caso contrário.

A equação 2.10 fornece uma relação funcional continua, bijetora e monótona crescente entre $\alpha$ e o volume dentro da célula onde $\langle m, x\rangle < \alpha$. Assim é possível obter $\alpha$ a partir do volume $V$ e vice versa. Dado o volume $V$, o valor de $\alpha$ pode ser encontrado através de simples métodos para determinar raízes de funções, como por exemplo, o método da Bissecção.

Estando a interface definida -dada por uma superfícies planar no interior de cada célula de fronteira- seu movimento pelo fluxo subjacente deve ser modelado. Nas próximas duas subseções apresentamos, respectivamente, um esquema euleriano e um esquema lagrangeano para modelar tal movimento.

### Fluxo através da Fronteira de uma Célula

Para representar a porção fluida de uma célula 2D, no trabalho em que foi introduzida a metodologia Volume of Fluid, são usados retângulos com lados paralelos aos eixos coordenados. Um dos lados é sempre uma aresta da célula em questão, célula $C$, outro tem um comprimento igual a fração do volume da célula que é fluido. A escolha da aresta de $C$ que irá compor o retângulo é feita a partir da estimativa do corte feito pela borda na célula. Pode-se fazer essa escolha, optando pela aresta com

maior fração sendo fluida, ou no caso de empate a mais distante do corte. Dada uma aresta $A$ da célula $C$ temos 3 possíveis situações desta relação ao retângulo que representa a parte fluida de $C$: ou a aresta $A$ contém estritamente um lado do retângulo (figura 2.1 caso (a)), ou é ela mesma um dos lados do retângulo (figura 2.1 caso (b)), ou é disjunta ao retângulo (figura 2.1 caso (c)).

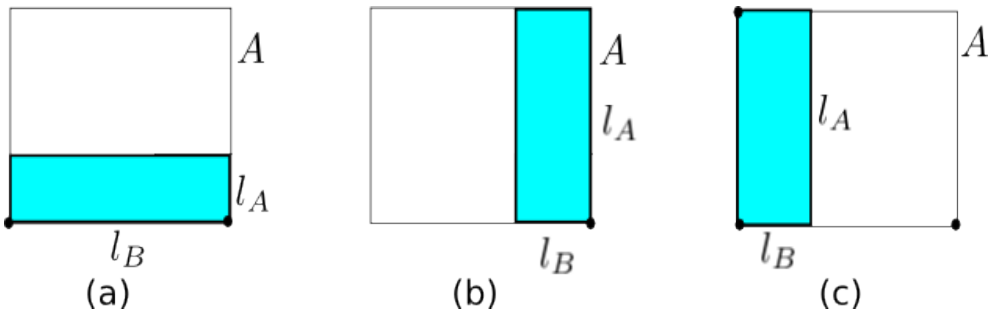



Figura 2.1: Porção fluida de uma célula dentro da abordagem VOF

Suponha agora que o fluxo se locomove de dentro da célula $C$ para a aresta $A$ com uma velocidade ortogonal a $A$ que vale $v$. Então a fração do volume fluido $\Delta V$ de $C$, que é advectada durante um intervalo de tempo $\delta t$ para célula a direita dela através de $A$ é dada em cada caso por: $\Delta V = l_A v \delta t$ no caso (a), $\Delta V = \min\{l_A v \delta t, l_B l_A\}$ no caso (b) e $\Delta V = \max\{0, (v\delta t - (1 - l_B))l_A\}$ no caso (c).

A figura 2.2 representa os 5 casos possíveis para o cômputo da porção fluido advectada através de $A$ no intervalo de tempo $\delta t$. Nela, a linha pontilhada dista $v\delta t$ e a porção fluida transferida através de $A$ é representada com um azul mais escuro.

Todos esses casos podem ser cobertos por uma expressão única proposta por [] que tem a seguinte forma:

$$\Delta V = \min\{lv\delta t + \delta V, l_M l_m\}$$

onde

$$\delta V = \max\{(1 - l)v\delta t - (1 - l_m l_M)\delta V, 0\},$$

$l_M$ é sempre o comprimento do lado maior do retângulo, $l_m$ é sempre o comprimento do lado menor e $l = l_m$ no caso (a) da figura 2.1, $l = 1$ no caso (b), e $l = 0$ no caso (c).

Todo raciocínio acima incluindo essa última fórmula funciona assumindo que $v\delta t < l_M$.

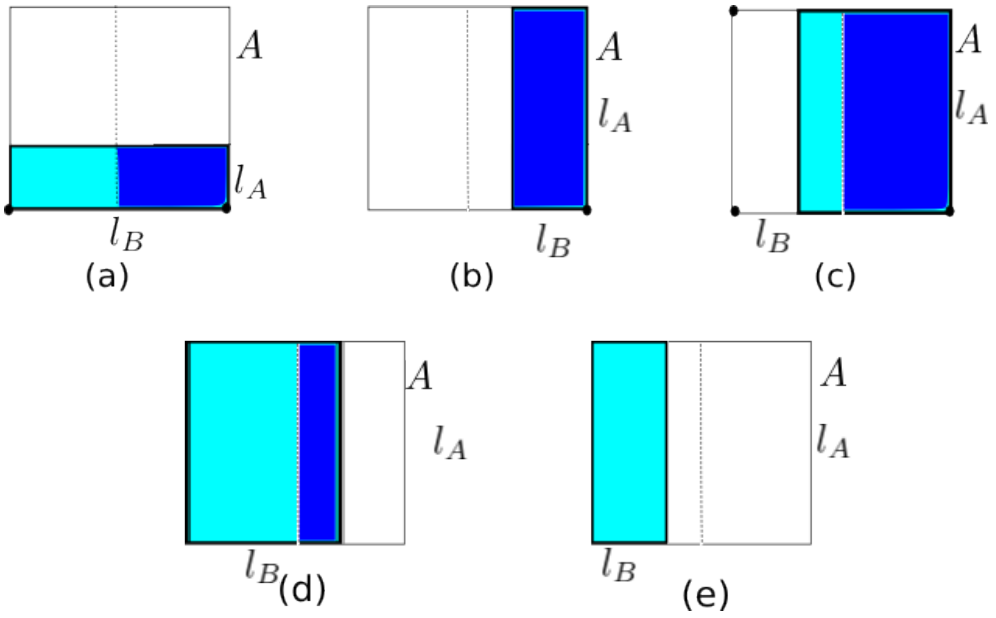



Figura 2.2: Nos casos (a) a (e) temos que a fração advectada para a célula da direita é dada por $l_A v\delta t$, $l_B l_A$, $l_A v\delta t$, $(v\delta t - (1 - l_B))l_A$ e 0 respectivamente

.

**Advecção Lagrangeana**

A abordagem lagrangeana para evolução da superfície livre pode ser descrita considerando o modo como a interface, representada pela equação 2.9, é advectada pelo fluido. Para esse propósito reescrevemos tal equação adcionando um índice temporal a todas as varáveis

$$m_1^n x^n + m_2^n y^n + m_3^n z^n = \alpha^n. \tag{2.11}$$

A advecção da interface pelo fluxo num passo de tempo $\Delta t$ pode ser executada separadamente em cada uma das direções $x, y$ e $z$. Assim, para a direção $x$, aproximamos a velocidade no interior da célula $(i, j, k)$ como uma interpolação linear da forma

$$u(x) = u_{i-\frac{1}{2}} \left(1 - \frac{x}{h}\right) + u_{i+\frac{1}{2}} \frac{x}{h},$$

onde $h$ é o comprimento da célula, $0 \leq x \leq h$, $u_{i-\frac{1}{2}}$ é a velocida da face esquerda da célula e $u_{i+\frac{1}{2}}$ a da face direita.

Para cada ponto inicialmente na interface calculamos sua nova coordenada $x^*$ da seguinte forma:

$$x^* = x^n + u(x^n)\Delta t = \left[1 + \frac{u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k}}{h} \Delta t\right] x^n + u_{i-\frac{1}{2},j,k} \Delta t. \tag{2.12}$$

Durante a advecção ao longo do eixo $x$, as coordenadas $y$ e $z$ permanecem constantes. Para encontrar a equação da superfície planar após essa etapa da advecção, escrevemos $x^n$ como função de $x^*$, de acordo com a equação 2.12,

$$x^n = \frac{x^* - u_{i-\frac{1}{2},j,k}\Delta t}{1 + \left(\left(u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k}\right)/h\right)\Delta t} \tag{2.13}$$

e substituimos o resultado na equação 2.11

$$m_1^n \frac{x^* - u_{i-\frac{1}{2},j,k}\Delta t}{1 + \left(\left(u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k}\right)/h\right)\Delta t} + m_2^n y^n + m_3^n z^n = \alpha^n. \tag{2.14}$$

Podemos reescrever a equação 2.14 numa maneira mais convencional

$$m_1^* x^* + m_2^* y^* + m_3^* z^* = \alpha^*. \tag{2.15}$$

onde

$$m_1^* = \frac{m_1^n}{1 + \left(\left(u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k}\right)/h\right)\Delta t}, \tag{2.16}$$

$$\alpha^* = \alpha^n + \frac{m_1^n u_{i-\frac{1}{2},j,k}\Delta t}{1 + \left(\left(u_{i+\frac{1}{2},j,k} - u_{i-\frac{1}{2},j,k}\right)/h\right)\Delta t}, \tag{2.17}$$

e todos as outras variáveis com superescrito $(*)$ em 2.15 são iguais aos seus valores anteriores.

Após a advecção, precisa-se verificar se a interface passou para o interior da célula vizinha (direita ou esquerda), e se isso ocorreu, calcular o volume movido para tal célula. Por exemplo, se $\alpha^*/m_1^*$ é maior que $h$, uma porção do volume originalmente contido abaixo de 2.11 foi movida para a célula vizinha direita. O volume movido pode ser calculado da seguinte forma:

1. Faz-se a mudança de coordenada

$$x' = h + x^*, \tag{2.18}$$

assim, $x'$ é a distância a partir da face direita da célula, e 2.15 vem a ser

$$m_1^* x' + m_2^* y^* + m_3^* z^* = \alpha'. \tag{2.19}$$

onde

$$\alpha' = \alpha^* - m_1^* h; \tag{2.20}$$

2. Aplica-se os coeficientes da equação 2.19 na fórmula 2.10, obtendo-se

$$\begin{aligned} V = \; & \frac{1}{6m_1^* m_2^* m_3^*} \left[ \alpha'^3 - \sum_1^3 h(\alpha - m_j^* c_j)(\alpha - m_j^* c_j)^3 \right. \\ & \left. + \sum_1^3 h(\alpha' - \alpha'_{max} + m_j^* c_j)(\alpha - \alpha'_{max} + m_j^* c_j)^3 \right] \end{aligned} \tag{2.21}$$

onde $\alpha'_{max} = m_1^* c_1 + m_2^* c_2 + m_3^* c_3$ e $h$ é o comprimento da célula.

Se $u_{i-\frac{1}{2},j,k} < 0$ o volume para célula vizinha esquerda pode ser calculado de maneira similar.

Finalmente, o novo volume da célula $(i, j, k)$ pode ser calculado também aplicando-se os coeficientes da equação 2.15 na equação 2.10, obtendo-se a mesma expressão dada por 2.21.

De maneira similar advectamos a fronteira nas direções $y$ e $z$.

### 2.2.3 Método Level Set

O método Level Set [65] [63], que será discutido em mais detalhes no capítulo 5 é um método semi-Lagrangeano para acompanhamento de interfaces, sendo o método mais difundido para efetuar o acompanhamento de interfaces.

### 2.2.4 Métodos Híbridos

Os métodos híbridos em geral usam combinações de partículas marcadoras, método VOF, malhas, parametrizações e método Level Set, dentre outros conceitos para efetuar o acompanhamento de interfaces. Dentre os mais populares estão:

1. Nesse método, partículas são empregadas para corrigir a evolução da borda determinada advectando-se a função de level set. A idéia é que essa função computada para os vértices da malha e para as partículas, seja advectada por mecanismos independentes. Se os valores obtidos forem discrepantes, então se efetuam correções na função do Level Set nos vértices.

   A cada certo número de iterações são lançadas partículas - 64 partículas por célula próxima a borda é o indicado na literatura - cuja posição dentro da célula é gerada de forma randômica. A distância inicial $\Gamma_p$, entre a borda e uma partícula $p$ é obtida por interpolação trilinear a partir do valor de $\Gamma$ nos

vértices da célula que contem tal partícula. Se cria também uma função de level set específica para a partícula $p$ dada por:

$$\Phi_p(x) = s_p\left(r_p - \|x - x_p\|\right)$$

onde $s_p$ é o sinal de $\Gamma_p$ e $r_p$ está restrito ao intervalo $[h/10, h/2]$ - para evitar que a esfera de raio $r_p$ e centro em $x_p$ seja muito grande ou muito pequena - $h$ é o lado da célula . $\Phi_p(x)$ é a função distância a borda da esfera de raio $r_p$ e centro em $x_p$. A idéia do método é que a borda da porção fluida pode ser bem aproximada pela união de partes de dessas esferas.

A função de level set dos vértices da malha e a das partículas são advectadas por processos distintos. Uma nova função de level set para as partículas, $\Phi'(p)$, é então computada por interpolação. Se $\frac{\Phi'(p)}{\Phi(p)} < -1$ - significando que a partícula mudou de lado em relação a borda e se afastou mais dela do que estava antes - a partícula é colocada num conjunto $\Psi^+$ se seu sinal $s_p$ for positivo ou no conjunto $\Psi^-$ em caso contrário.

$\Psi^+ \cup \Psi^- \neq \phi$ significa que há discrepâncias entre a função de level set computada nos vértices e nas partículas, devendo-se então aplicar uma processo de correção em cada vértice $V$ das células que contém partículas nesses dois conjuntos.

Essa correção é efetuada fazendo-se o valor corrigido de $\Phi(V)$, $\Phi'(V)$, ser dado por:

$$\begin{cases} \Phi'(V) = \Phi^+(V) & \text{se } \Phi^+(V) \geq \Phi^-(V) \\ \Phi'(V) = \Phi^-(V) & \text{caso contrário.} \end{cases}$$

onde

$$\Phi^+(V) = \max\left\{\Phi(V), \max\, \Phi_p(V) \text{ para } p \in \{\Psi^+ \cap U(V)\}\right\}$$
$$\Phi^-(V) = \min\, \left\{\Phi(V), \min\, \Phi_p(V) \text{ para } p \in \{\Psi^- \cap U(V)\}\right\}$$

e $U(V)$ é a união das células adjacentes a $V$.

2. O método desenvolvido por Kim e Lee [71] [72], o qual caracteriza-se como uma variação do método VOF, onde o vetor normal a superfície é dado por

$$\boldsymbol{r} = \frac{\sum_j \left(f_{i,j} V_{i,j}\right) \boldsymbol{n}_j}{\left|\sum_j \left(f_{i,j} V_{i,j}\right) \boldsymbol{n}_j\right|}, \tag{2.22}$$

onde $f_{i,j}$ é a fração de fluido transferida através da face $j$ da célula $i$, dada por

$$f_{i,j} = \frac{\text{volume de fluido transferido através da face } j}{\text{volume total (ar+fluido) transferido através da face } j},$$

$V_{i,j}$ é o volume da célula vizinha adjacente a face $j$ e $\boldsymbol{n}_j$ é o vetor normal a face $j$. E sub-células, denominadas "baby-cells", são definidas na parte fluida de uma célula de borda e utilizadas para calcular o volume atravessando cada uma das faces de uma célula de borda. A contribuição das baby-cells para o fluxo através de uma face começa a ser computada a partir da baby-cell mais distante do plano passando pelo centróide da célula master e normal ao vetor $\boldsymbol{r}$ definido em 2.22.

3. O método desenvolvido por Kohno [50] que acopla o método level ao refinamento adaptativo de malhas. O refinamento é feito de forma a obter uma maior resolução da grade apenas em regiões próximas a fronteira, tais regiões são definidas a partir de um "threshold" função level set, $\Phi$, ou seja, o refinamento acontece apenas nas regiões onde $\Phi \leq \epsilon$.

## 2.3 Trabalhos Recentes

Em 2004 Frank Losasso e outros [54] publicaram um método para simulação de fluido com superfície livre numa estrutura de dados "octree" explorando técnicas de refinamento de malha para capturar detalhes visuais em pequena escala. O método discretiza a equação de Poisson numa octree da grade, de forma que o sistema linear resultante seja simétrico positivo definido, possibilitando o uso do algoritmo do gradiente conjugado pré-condicionado para resolvê-lo.

Em 2005 Guendelman e outros [36] apresentaram um método acoplado ao PLS para o tratamento da interface água/ar capaz de reproduzir detalhes finos dessa interface. Além disso, a metodologia apresentada nesse trabalho também é capaz de tratar iterações do fluido com sólidos infinitesimalmente finos (um lençol, por exemplo) modelados por superfícies trianguladas. Tal método foi implementado em grades uniformes e grades octree adaptativas.

Ainda nesse contexto de estruturas octree, Adam Bargteil e outros [7], em 2006, apresentaram um método semi-lagrangeano para o acompanhamento da fronteira,

onde mantem-se uma malha poligonal explícita que define a fronteira, e uma estrutura de dados octree para representar a massa fluida, a qual oferece um meio eficiente para calcular a função distância com sinal à fronteira. A cada passo no tempo uma nova fronteira é construída extraindo-se o level set zero da função distância com sinal. Um esquema semi-lagrangeano é usado para advectar essa função. Uma outra característica do método é a capacidade de acompanhar características da superfície como cor ou coordenadas de textura, a um custo adicional baixo.

Também em 2006, Geiger e outros publicaram seu trabalho [34] aplicado a produção do filme "Posseidon", onde os mesmos fizeram uso de uma ferramenta desenvolvida em colaboração com a universidade de Stanford, para simulação de fluido utilizando o método PLS. Nesse trabalho foi relatada a dificuldade de se renderizar a saída combinada do método PLS, uma vez que o "ray tracing" pode eficientemente renderizar a superfície definida pelo o level set zero, porém o mesmo não acontece em relação a uma superfície definida por um grande número de partículas. Igualmente difícil é obter uma transição suave entre o level set e as partículas salpicadas.

O método apresentado pelos autores para contornar essas dificuldades, cria uma única malha a partir do level set zero e das partículas salpicadas, e começa subdividindo recursivamente o domínio da simulação na região próxima à superfície do fluido. Uma malha quadrilateral é construída a partir das faces dos voxels, exteriores as fluido, e cada vertex da malha é movido para o ponto mais próximo da superfície implícita, criando uma malha que aproxima a superfície implícita com um algo grau de precisão. Finalmente um simples algoritmo de relaxação é executado sobre a malha para minimizar qualquer artefato remanescente. Tal método efetua uma combinação suave entre as duas partes empregadas para definir a superfície: o level set e das partículas.

Em 2008 Frank Losasso e outros [55] apresentaram uma acoplagem em duas vias entre o método "smoothed particle hydrodynamics" (SPH) e o método PLS, onde assim como no algoritmo particle level set padrão partículas marcadoras são dispostas em ambos os lados da interface ar/fluido e adveccionadas de acordo com o fluxo do fluido. Nas áreas onde a resolução da grade não é suficiente para descrever o comportamento do level set, essas partículas marcadoras poderão passar de um lado da interface para outro, indicando a necessidade da reconstrução local da função le-

vel set a partir das informações características presentes nessas partículas. Quando uma partícula cruza a interface, se afastando além de um limite pré-determinado, a mesma é removida do conjunto de partículas marcadoras e usada para representar gotículas ou bolhas, dependendo se a mesma partiu do fluido ou do ar, respectivamente. Além disso, as partículas, originalmente fluido, removidas são introduzidas num sistema de SPH.

Em simulações com partículas relativamente esparsas ou aquelas onde o número de partículas fluido removidas é pequeno, é suficiente apenas considerar que o fluido exerce força nas partículas, sem que elas afetem o comportamento do fluido. Nesses casos, é feita uma acoplagem de mão única, da grade para o SPH, utilizando a solução obtida a partir da grade como condição de fronteira para se resolver o SPH. Em particular, cada face ao longo da interface level set é definida como uma condição de Neumann com a velocidade fornecida pela solução obtida a partir da grade.

Para simulações com regiões densas em números de partículas, é feita uma acoplagem de mão dupla, onde uma única equação de Poisson é definida para garantir a divergência nula do campo de velocidades sobre a grade e sobre as regiões governadas pelo SPH.

Em 2009 Chris Wotjan e outros [78] publicaram um método para acompanhamento de superfície capaz de acompanhar mudanças topológicas numa grande gama de situações. Essa metodologia foi parte da motivação para o emprego de células com recorte irregular como será proposto neste trabalho. Na verdade, em termos topológicos sua proposta é até mais abrangente do que a que será apresentada aqui. O trabalho é apresentado em detalhe para que se possa avaliar o custo computacional de sua implementação consideravelmente maior do que o da nossa alternativa.

Numa descrição compacta, no método se executa os seguintes passos:

1. Se recebe uma malha $M$ para representar a superfície e um campo de velocidade $V$ definido sobre a grade;

2. Move-se os vértices da malha $M$, de acordo com o campo de velocidade $V$, mantendo-se a conectividade dos triângulos originais. Para isso é usado o método de Runge-Kutta de quarta ordem explícito;

3. Faz-se uma amostragem fina, sobre uma grade regular $G$, da região em torno

da nova posição da malha.

4. Calcula-se a distância à malha $M$ em cada vértice da grade $G$ próximo a $M$, tomando-se o menor valor em módulo dentre as distâncias com sinal entre o vértice e cada um dos triângulos cortando células vizinhas a esse ponto. O resultado dessa etapa é um campo de distâncias com sinal $D$.

   Ao final desta etapa do método, existem duas representações para a interface. Uma explícita, a malha $M$, e uma definida implicitamente na grade $G$ pelo campo de distância $D$.

5. Identificam-se as "células complexas" na grade $G$, onde:

   - Uma aresta complexa, em $G$, é uma aresta que intercecta $M$ mas de uma vez;
   - Uma face complexa, em $G$, é uma face cuja sua interseção com $M$ forma uma curva fechada.
   - Uma célula complexa, em $G$, é uma célula:
     - Onde seus 8 vértices possuem valores em $D$ com o mesmo sinal e além disso ela tem alguma interseção com $M$, ou
     - Possui uma face ou uma aresta complexa;

6. Encontra-se cada triângulo da malha $M$ que intersecta uma aresta de uma célula complexa. Então, se calcula o ponto de interseção $P1$ entre a aresta e o triângulo, e se subdivide o triângulo em três novos triângulos que compartilham o vértice $P1$. Esse novo vértice $P1$ é classificado como vértice do tipo 1.

7. Determinam-se as arestas de cada triângulo, criado na etapa anterior, que intersectam uma face da célula complexa. Para cada uma dessas arestas, se encontra o ponto $P2$ de interseção dela com a face. Então se subdivide o triângulo em questão em dois novos triângulos que compartilham um vértice no ponto $P2$. Esse novo vértice $P2$ é classificado como vértice do tipo 2.

8. Neste ponto do método, as arestas de triângulos cujos vértices são dos tipos 1 e 2, descrevem uma única curva linear por partes conectando vértices do tipo

1 ao longo de cada face da célula. Então para se transformar esta curva em um segmento de reta entre vértices do tipo 1,se colapsam na malha as arestas entre vértices do tipo 1 e tipo 2. Isto é feito até que todos os vértices do tipo 2 sejam removidos da malha;

9. Após todas essas operações terem sido executadas, nenhum triângulo irá cruzar uma face de qualquer célula complexa. Cada triângulo estará completamente fora ou completamente dentro de uma célula complexa. Então deleta-se todos os triângulos que estiverem completamente dentro de uma célula complexa.

10. Neste ponto, já não existem mais células complexas, então pode-se aplicar o método marching cubes para gerar novas malhas trianguladas, e conectar essas malhas nos vértices do tipo 1.

Em 2010, o mesmo grupo coordenado por Chris Wotjan [79] publicou melhorias no método que reduziram os artefatos de reamostragem através de um algoritmo de costura de malha baseado em subdivisão e um esquema de interpolação de ordem superior para determinar a posição dos novos vértices criados.

# Capítulo 3

# Fundamentos Teóricos

Neste capítulo apresentamos alguns conceitos básicos envolvidos na fundamentação teórica da simulação de fluidos. Dentre eles, o própio conceito de fluido, o conceito de campo de tesnsão, as leis de conservação e as equações de Navier-Stokes. Utilizaremos neste e nos próximos capítulos as notações, $\boldsymbol{x} = (x, y, z)$ para representar as coordenadas do vetor posição e $\boldsymbol{v} = (u, v, w)$ para representar as coordenadas do vetor velocidade. Grandezas vetoriais serão representadas em negrito.

## 3.1 Uma Definição de Fluido

Segundo Munson [13], fluido é uma substância que se deforma continuamente quando submetida a uma tensão de cisalhamento constante qualquer.

Fluidos podem ser divididos em duas grandes classes, líquidos e gases. Os líquidos quando em repouso apresentam uma superfície estacionária delimitada mas não inteiramente determinada pelo recipiente que o contém, denominada superfície livre. Já os gases apresentam a propriedade de se expandirem livremente quando não confinados por um recipiente, não formando portanto uma superfície livre estacionária [13].

Fluidos também podem ser classificados como newtonianos e não newtonianos, [13]. Essa classificação diz respeito à relação entre a tensão exercida sobre o fluido e á derivada espacial da velocidade das partículas do fluido. Antes de formalizar tal classificação, vamos introduzir o conceito de campo de tensão.

### 3.1.1 Campo de Tensão

Conforme em [8] [52], considere um elemento de área $\boldsymbol{\delta A}$ contendo um ponto $\boldsymbol{p}$. Tal elemento de área será representado por um vetor normal a superfície no ponto $\boldsymbol{p}$ com norma igual a área do elemento. Considerando ainda que sobre a área $\boldsymbol{\delta A}$ atua uma força $\boldsymbol{F_p}(\boldsymbol{\delta A})$, a tensão no ponto $\boldsymbol{p}$ é definida por

$$T_p = \lim_{|\boldsymbol{\delta A}|\to 0} \frac{|\boldsymbol{F_p}(\boldsymbol{\delta A})|}{|\boldsymbol{\delta A}|}.$$

O vetor $\boldsymbol{\delta A}$ pode ser escrito como $\boldsymbol{\delta A} = \delta A_1 \boldsymbol{e_1} + \delta A_2 \boldsymbol{e_2} + \delta A_3 \boldsymbol{e_3}$, onde $\delta A_1, \delta A_2$ e $\delta A_3$ são as projeções de $\boldsymbol{\delta A}$ sobre os eixos $x, y$ e $z$, e $\boldsymbol{e_1}, \boldsymbol{e_2}$ e $\boldsymbol{e_3}$ são os vetores canônicos do $R^3$. Analogamente, temos $\boldsymbol{F_p}(\boldsymbol{\delta A}) = \boldsymbol{F_{1p}}(\boldsymbol{\delta A})\, e_1 + \boldsymbol{F_{2p}}(\boldsymbol{\delta A})\, e_2 + \boldsymbol{F_{3p}}(\boldsymbol{\delta A})\, e_3$. Então, podemos definir a tensão $\tau_{ij}$, ou seja, aquela que atua sobre o plano $i$ (plano normal a direção $i$) na direção $j$ como

$$\tau_{ij} = \lim_{|\boldsymbol{\delta A_i}|\to 0} \frac{\boldsymbol{F_j}}{|\boldsymbol{\delta A_i}|}.$$

O campo diferencial de tensão é então definido como

$$\tau = [\tau_{i,j}] = \begin{bmatrix} \tau_{11} & \tau_{12} & \tau_{13} \\ \tau_{21} & \tau_{22} & \tau_{23} \\ \tau_{31} & \tau_{32} & \tau_{33} \end{bmatrix}. \tag{3.1}$$

As componentes $\tau_{ii}$ são denominadas componentes de tensão normal, enquanto as componentes $\tau_{ij,i\neq j}$ são denominadas componentes de tensão de cisalhamento, isto é, aquelas que são paralelas aos planos onde se aplicam.

O campo de tensão 3.1 pode ser expresso como a soma de dois outros campos matriciais de tensão:

$$\tau = S - pI \tag{3.2}$$

onde $p$ é denominado pressão e dado por $p = -\frac{\tau_{11}+\tau_{22}+\tau_{33}}{3}$, $I$ é a matriz identidade 3x3, $S = [s_{i,j}]$ com $s_{i,j} = \tau_{i,j} + p\delta_{i,j}$, $\delta_{i,j}$ é o delta de Kronecker.

A classificação dos fluidos como newtonianos e não newtonianos, é então feita da seguinte forma:

**Fluido Newtoniano**: As componetes do campo $S$ possuem relação linear com

as derivadas espaciais do campo de velocidade. Explicitamente:

$$S = \mu \begin{bmatrix} 2\frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} & \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \\ \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} & 2\frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \\ \frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} & \frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} & 2\frac{\partial w}{\partial z} \end{bmatrix} = \mu \left( \nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^t \right) \tag{3.3}$$

a constante de proporcionalidade $\mu$, é denominada viscosidade cinemática. Veremos mais adiante que em nossas equações aparecerá a expressão $\frac{\mu}{\rho}$, onde $\rho$ é densidade do fluido, tal expressão é denominada viscosidade dinâmica do fluido.

**Fluido Não Newtoniano**: Num fluido não newtoniano as componentes do campo $S$ não possuem a relação linear com as derivadas espaciais do campo de velocidade dada por 3.3.

## 3.2 Equações de Navier-Stokes

As equações de Navier-Stokes [8] descrevem o movimento de um fluido, e a dedução das mesmas, pode ser feita a partir da lei de conservação da quantidade de movimento e da lei de conservação de massa.

### 3.2.1 Leis de Conservação

Em física, uma Lei de conservação [8] é a afirmação que a variação temporal da quantidade de uma determinada grandeza física em um volume arbitrário fixo, é igual a quantidade dessa grandeza que sai (ou entra) pela fronteira do volume mais a quantidade dessa grandeza que é gerada (ou absorvida) no interior do volume. O que matematicamente pode ser traduzido como:

$$\frac{\partial}{dt} \int_V L \ dV = - \int_S L \ \boldsymbol{v}.\boldsymbol{n} \ dS + \int_V Q \ dV \tag{3.4}$$

onde $V$ é o volume em questão, $L$ é a quantidade da grandeza física por unidade de volume, $S$ é a superfície do volume $V$, $\boldsymbol{v} = [u(x,y,z,t), v(x,y,z,t), w(x,y,z,t)]^T$ é o campo vetorial velocidade definido sobre o volume $V$ e a fronteira $S$, $\boldsymbol{n} = [n_1(x,y,z,t), n_2(x,y,z,t), n_3(x,y,z,t)]^T$ é o campo vetorial normal a superfície $S$, e $Q$ é quantidade da grandeza gerada no interior do volume por unidade de tempo e volume.

Utilizando o teorema da divergência (teorema de Gauss), o qual define a relação entre a integral de volume da divergência de um campo vetorial e a integral de superfície do fluxo definido por este campo, temos:

$$\int_V \nabla.(L\boldsymbol{v})\ dV = \int_S L\ \boldsymbol{v}.\boldsymbol{n}\ dS,$$

Assim a equação 3.4 se torna:

$$\frac{\partial}{dt}\int_V L\ dV = -\int_V \nabla.(L\boldsymbol{v})\ dV + \int_V Q\ dV.$$

Trocando, então a ordem dos operadores $\frac{d}{dt}$ e $\int_V dV$ temos

$$\int_V \frac{\partial}{dt}L\ dV = -\int_V \nabla.(L\boldsymbol{v})\ dV + \int_V Q\ dV$$

como essa equação deve valer em qualquer volume por menor que seja, podemos remover o operador $\int_V dV$ obtendo:

$$\frac{\partial}{dt}L\ = -\nabla.(L\boldsymbol{v})\ + Q. \tag{3.5}$$

Se $L = [l_1(x,y,z,t), l_2(x,y,z,t), l_3(x,y,z,t)]^t$, é uma grandeza vetorial, temos que a operação $L\boldsymbol{v}$ representa o produto matricial $L\boldsymbol{v}^t$, onde $L$ é um vetor coluna e $\boldsymbol{v}^t$ é um vetor linha, a saber,

$$L\boldsymbol{v}^t = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} [u,v,w] = \begin{bmatrix} l_1u & l_1v & l_1w \\ l_2u & l_2v & l_2w \\ l_3u & l_3v & l_3w \end{bmatrix},$$

Nesse caso se tem:

$$\nabla.(L\boldsymbol{v}^t) = \nabla.\begin{bmatrix} l_1u & l_1v & l_1w \\ l_2u & l_2v & l_2w \\ l_3u & l_3v & l_3w \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x}(l_1u) + \frac{\partial}{\partial y}(l_1v) + \frac{\partial}{\partial z}(l_1w) \\ \frac{\partial}{\partial x}(l_2u) + \frac{\partial}{\partial y}(l_2v) + \frac{\partial}{\partial z}(l_2w) \\ \frac{\partial}{\partial x}(l_3u) + \frac{\partial}{\partial y}(l_3v) + \frac{\partial}{\partial z}(l_3w) \end{bmatrix}.$$

**Lei de Conservação da Quantidade de Movimento**

Aplicando a equação 3.5 quando a grandeza é a quantidade de movimento, $m\boldsymbol{v}$ onde $m$ é a massa e $\boldsymbol{v}$ a velocidade, temos $L = \rho\boldsymbol{v}$ onde $\rho$ é a densidade do fluido num dado elemento de volume $\Delta V$. Em relação a $Q$ temos

$$Q(\Delta V) = \frac{1}{|\Delta V|}\left(\frac{d}{dt}(m\boldsymbol{v})_{ger}\right) = \frac{1}{|\Delta V|}\left(\left(\frac{dm}{dt}\boldsymbol{v}\right)_{ger} + (m\boldsymbol{a})_{ger}\right)$$

onde $\boldsymbol{a}$ é a aceleração e o subscrito $ger$ indica derivado das forças produzidas no elemento de volume. Assumindo que variações na massa do elemento são resultantes apenas de advecção, a primeira parcela após a segunda igualdade da equação 3.2.1 se anula e assim,

$$Q\left(\Delta V\right) = \frac{1}{|\Delta V|}\left(m\boldsymbol{a}\right)_{ger}$$

ou ainda,

$$Q\left(\Delta V\right) = \frac{1}{|\Delta V|}\boldsymbol{F}_{ger},$$

onde $\boldsymbol{F}_{ger} = \left(m\boldsymbol{a}\right)_{ger}$, é a força gerada em $\Delta V$.

A força $\boldsymbol{F}_{ger}$ só pode ter duas origens:

1. $\boldsymbol{F}_{ger}$ é gerada na superfície, $A$, do elemento $\Delta V$ sendo obtida então por:

$$\boldsymbol{F}_{ger}^{sup} = \int_A \frac{\partial \boldsymbol{F}_{ger}^{sup}}{\partial A} dA.$$

Tomando $\Gamma = \frac{\partial \boldsymbol{F}_{ger}^{sup}}{\partial A}$ temos

$$\boldsymbol{F}_{ger}^{sup} = \int_A \Gamma dA,$$

e pelo teorema da divergência segue que

$$\boldsymbol{F}_{ger}^{sup} = \int_{\Delta V} \nabla.\Gamma dV.$$

2. $\boldsymbol{F}_{ger}$ é gerada pela ação de campo, cuja aceleração produzida é representada por $\boldsymbol{g}$

$$\boldsymbol{F}_{ger}^{campo} = \boldsymbol{g}m = \int_{\Delta V} \rho\boldsymbol{g} dV.$$

Assim temos que

$$Q\left(\Delta V\right) = \frac{1}{|\Delta V|}\left(\boldsymbol{F}_{ger}^{sup} + \boldsymbol{F}_{ger}^{campo}\right) = \frac{1}{|\Delta V|}\int_{\Delta V}\left(\nabla.\Gamma + \rho\boldsymbol{g}\right) dV.$$

Fazendo $\Delta V \to 0$ temos finalmente $\nabla.\Gamma + \rho\boldsymbol{g}$.

Assim, 3.5 toma a forma

$$\frac{\partial(\rho\boldsymbol{v})}{\partial t} + \ \nabla.(\rho\boldsymbol{v}\boldsymbol{v}^T) \ - (\nabla.\tau + \rho\boldsymbol{g}) \ = 0.$$

Reescrevendo o campo de tensão $\tau$ como em 3.2, ficamos com:

$$\frac{\partial(\rho\boldsymbol{v})}{\partial t} + \ \nabla.(\rho\boldsymbol{v}\boldsymbol{v}^T) \ - (\nabla.(S - pI) + \rho\boldsymbol{g}) \ = 0.$$

ou seja,

$$\frac{\partial(\rho \boldsymbol{v})}{\partial t} + \ \nabla.(\rho \boldsymbol{v}\boldsymbol{v}^T) \ - (\nabla.S - \nabla p + \rho \boldsymbol{g}) \ = 0.$$

Observe agora, que para um fluido newtoniano, conforme 3.3, temos

$$\begin{aligned} \nabla.S &= \mu \nabla. \left( \nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^T \right) \\ &= \mu \left( \nabla^2 \boldsymbol{v} + \nabla. (\nabla \boldsymbol{v})^T \right) \\ &= \mu \left( \nabla^2 \boldsymbol{v} + \nabla (\nabla. \boldsymbol{v}) \right) \end{aligned}$$

Logo a lei de conservação da quantidade de movimento para um fluido newtoniano pode ser expressa por:

$$\frac{\partial(\rho \boldsymbol{v})}{\partial t} + \ \nabla.(\rho \boldsymbol{v}\boldsymbol{v}^T) \ - \mu \left( \nabla^2 \boldsymbol{v} + \nabla (\nabla. \boldsymbol{v}) \right) + \nabla p - \rho \boldsymbol{g} = 0. \tag{3.6}$$

**Lei de Conservação da Massa**

Considere a lei de conservação expressa pela equação 3.5 para o caso da massa de um fluido, num dado volume elementar ($\Delta V$). Nesse caso $Q = 0$, já que não existem fontes nem sumidouros de massa dentro do volume, a equação 3.5 toma a forma:

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho \boldsymbol{v}) = 0.$$

Nos casos onde $\rho$ é constante, chamamos o fluido de fluido incompressível, e a equação de conservação da massa se resume a

$$\nabla. \boldsymbol{v} = 0. \tag{3.7}$$

**Equações de Navier-Stokes para Fluidos Incompressíveis**

Considerando a equação 3.7 o termo $\nabla (\nabla. \boldsymbol{v})$ da equação 3.6 desaparece e ela pode ser reescrita na forma

$$\rho \frac{\partial \boldsymbol{v}}{\partial t} + \rho \nabla.(\boldsymbol{v}\boldsymbol{v}^T) = -\nabla p + \mu \nabla^2 \boldsymbol{v} + \rho \boldsymbol{g}$$

ou ainda,

$$\frac{\partial \boldsymbol{v}}{\partial t} + \nabla.(\boldsymbol{v}\boldsymbol{v}^t) = -\frac{1}{\rho} \nabla p + \frac{\mu}{\rho} \nabla^2 \boldsymbol{v} + \boldsymbol{g},$$

que junto com 3.7 forma as denominadas equações de Navier-Stokes para fluidos incompressíveis.

Na forma não vetorial temos

$$\frac{\partial u}{\partial t} + \frac{\partial (uu)}{\partial x} + \frac{\partial (uv)}{\partial y} + \frac{\partial (uw)}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{\mu}{\rho}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) + g_x$$
$$\frac{\partial v}{\partial t} + \frac{\partial (uv)}{\partial x} + \frac{\partial (vv)}{\partial y} + \frac{\partial (vw)}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \frac{\mu}{\rho}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) + g_y$$
$$\frac{\partial w}{\partial t} + \frac{\partial (uw)}{\partial x} + \frac{\partial (vw)}{\partial y} + \frac{\partial (ww)}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial z} + \frac{\mu}{\rho}\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) + g_z$$
$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

## 3.3 Condições de Contorno

Para completar as especificações do modelo empregado para simulação de fluidos falta descrever as condições de contorno empregadas. Genericamente elas são de dois tipos:

1. Condições de Dirichlet - as que são aplicadas à velocidade/pressão em pontos da superfície que delimita a extensão da massa fluida.

2. Condições de Neuman - as que são aplicadas à derivada da velocidade/pressão na direção da normal a essa superfície em alguns ou todos os seus pontos.

As condições de Dirichlet podem especificar inteiramente a velocidade num ponto da superfície limitante. É o caso dos pontos rotulados como "No-slip"onde a velocidade deve se anular. A especificação pode se restringir também, apenas a componente normal da velocidade. Nos pontos com rótulo "slip"só essa componente precisa se anular. A condição de Neuman mais usual é obrigar a derivada normal da pressão a ser nula nos pontos da superfície limitante. Também podem ser consideradas condições de Dirichlet as que definem as áreas dessa superfície por onde o fluido é introduzido no sistema.Para os pontos dessas áreas o valor da componente normal da velocidade ou o da pressão são especificados explicitamente. Esses pontos são referidos como da classe "inflow". As áreas por onde o fluido deixa o sistema - constituída pelos pontos referidos como "outflow- podiam ser definidas de forma inteiramente análoga. Entretanto exigir uma vazão pré-especificada de saída exige, no mínimo, que a área por onde se dará essa vazão esteja coberta de fluido, o que na maior parte das aplicações não se sabe quando vai ocorrer. Por isso os pontos de outflow são aqueles em que não há restrições sobre a velocidade ou a pressão possibilitando o fluido deixar o sistema. Para que as equações de Poisson tenham

solução única, entretanto, é necessário extrapolar a pressão verificada no lado fluido de uma área de outflow para o outro lado.

# Capítulo 4

# Discretização e Aproximação Numérica das Equações de Navier-Stokes

A discretização do espaço sobre o qual as equações de Navier-Stokes estão definidas pode ser feita definindo-se uma grade regular de dimensões $m_1$ x $m_2$ x $m_3$ onde cada voxel da grade é dado por um cubo elementar $h$ x $h$ x $h$, e tomando-se $(x, y, z)$ sobre o paralelepípedo $[0, m_1h]$ x $[0, m_2h]$ x $[0, m_3h] \subset \Re^3$. Daqui por diante, por simplicidade faremos $m_1 = m_2 = m_3 = \frac{1}{h}$. Nesse caso, a região coberta pela grade será o cubo unitário do $\Re^3$. As figuras 4.1 e 4.2 ilustram tal grade indicando-se para cada voxel as coordenadas absolutas e relativas à malha discretizada, de cada um dos seus vértices:

Usaremos os índices $i, j$ e $k$ para indicar, respectivamente, as coordenadas relativas à malha horizontais, verticais e de profundidade. As coordenadas absolutas correspondentes serão representadas por $x, y$ e $z$. Cada voxel será referido pelas coordenadas relativas de seu canto inferior esquerdo frontal. Faces serão identificadas por sua orientação e pelas coordenadas relativas a malha de seu vértice onde elas forem as menores possíveis.

Cada componente da velocidade $u_{i,j,k}, v_{i,j,k}$, e $w_{i,j,k}$ é definida no centro da face

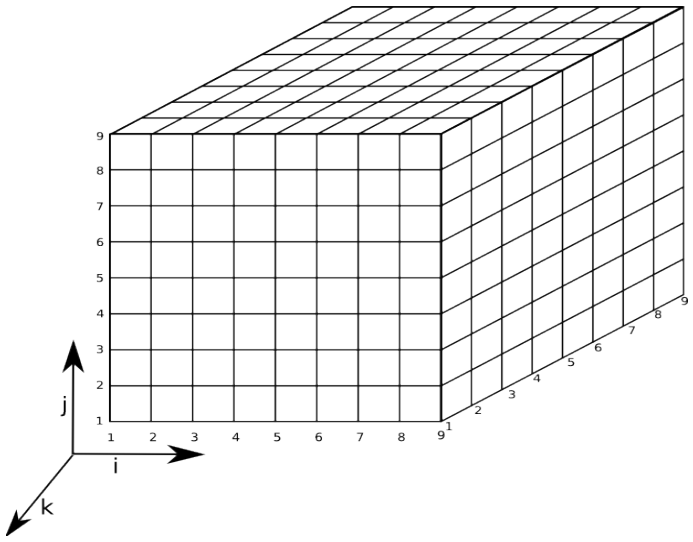



Figura 4.1: Grid com posições realtivas de seus voxels

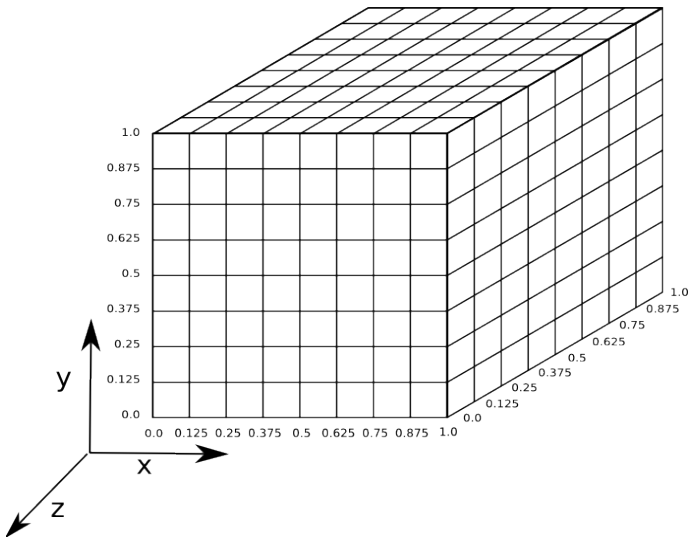



Figura 4.2: Grid com posições absolutas de seus voxels

ortogonal a mesma, referenciada por $(i, j, k)$, ou seja,

$$u_{i,j,k} \text{ está definida em } \left(x_i, y_{j+\frac{1}{2}}, z_{k+\frac{1}{2}}\right),$$
$$v_{i,j,k} \text{ está definida em } \left(x_{i+\frac{1}{2}}, y_j, z_{k+\frac{1}{2}}\right),$$
$$w_{i,j,k} \text{ está definida em } \left(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}}, z_k\right).$$

Ver figura 4.3.

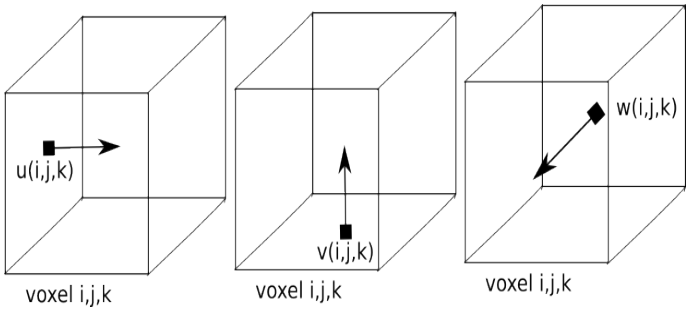



Figura 4.3: Componentes da velocidade definidas nas respectivas faces

Já a pressão $p_{i,j,k}$ é definida no centro da célula $(i, j, k)$, ou seja, $p_{i,j,k}$ está definida em $\left(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}}, z_{k+\frac{1}{2}}\right)$. Ver figura 4.4.

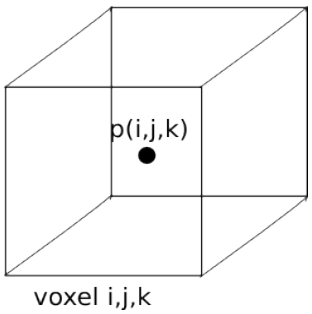



Figura 4.4: Pressão definida no centro de cada célula

Usando a indexação desta discretização espacial, temos que as equações a serem

resolvidas podem ser escritas na forma,

$$\begin{aligned}
\left(\frac{\partial u}{\partial t}\right)_{i,j,k} + \left(\frac{\partial (uu)}{\partial x}\right)_{i,j,k} + \left(\frac{\partial (uv)}{\partial y}\right)_{i,j,k} + \left(\frac{\partial (uw)}{\partial z}\right)_{i,j,k} &= -\tfrac{1}{\rho}\left(\frac{\partial p}{\partial x}\right)_{i,j,k} + \\
-\frac{\mu}{\rho}\left(\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j,k} + \left(\frac{\partial^2 u}{\partial y^2}\right)_{i,j,k} + \left(\frac{\partial^2 u}{\partial z^2}\right)_{i,j,k}\right) &+ (g_x)_{i,j,k} \\
\left(\frac{\partial v}{\partial t}\right)_{i,j,k} + \left(\frac{\partial (uv)}{\partial x}\right)_{i,j,k} + \left(\frac{\partial (vv)}{\partial y}\right)_{i,j,k} + \left(\frac{\partial (vw)}{\partial z}\right)_{i,j,k} &= -\tfrac{1}{\rho}\left(\frac{\partial p}{\partial y}\right)_{i,j,k} + \\
-\frac{\mu}{\rho}\left(\left(\frac{\partial^2 v}{\partial x^2}\right)_{i,j,k} + \left(\frac{\partial^2 v}{\partial y^2}\right)_{i,j,k} + \left(\frac{\partial^2 v}{\partial z^2}\right)_{i,j,k}\right) &+ (g_y)_{i,j,k} \\
\left(\frac{\partial w}{\partial t}\right)_{i,j,k} + \left(\frac{\partial (uw)}{\partial x}\right)_{i,j,k} + \left(\frac{\partial (vw)}{\partial y}\right)_{i,j,k} + \left(\frac{\partial (ww)}{\partial z}\right)_{i,j,k} &= -\tfrac{1}{\rho}\left(\frac{\partial p}{\partial z}\right)_{i,j,k} + \\
-\frac{\mu}{\rho}\left(\left(\frac{\partial^2 w}{\partial x^2}\right)_{i,j,k} + \left(\frac{\partial^2 w}{\partial y^2}\right)_{i,j,k} + \left(\frac{\partial^2 w}{\partial z^2}\right)_{i,j,k}\right) &+ (g_z)_{i,j,k} \\
\left(\frac{\partial u}{\partial x}\right)_{i,j,k} + \left(\frac{\partial v}{\partial y}\right)_{i,j,k} + \left(\frac{\partial w}{\partial z}\right)_{i,j,k} &= 0 \qquad \text{para } 1 \le i,j,k \le m.
\end{aligned} \tag{4.1}$$

A discretização temporal é feita tomando $t_n = \sum_0^n \Delta t_n$ onde $n = 0, 1, 2, \ldots$ e $\Delta t_n$ um valor real calculado a cada iteração. Assim, é assumido que $(u, v, w)^n_{i,j,k} = (u_{i,j,k}(t_n), v_{i,j,k}(t_n), w_{i,j,k}(t_n))$ e $p^n_{i,j,k} = p_{i,j,k}(t_n)$.

Desta forma, o problema consiste em: Dado o campo de velocidades $(u, v, w)^n_{i,j,k}$ que satisfaz a condição de divergência nula, $\nabla.(u, v, w)^n = 0$, no interior do fluido, o campo de pressão $p^n_{i,j,k}$ para $1 \le i, j, k \le m$, e as condições de contorno a serem consideradas, encontrar o campo de velocidades $(u, v, w)^{n+1}_{i,j,k}$ no interior do fluido que satisfaça o sistema de equações 4.1.

## 4.1 Método de Projeção do tipo Chorin

Na aproximação mumérica da solução das equações de Navier-Stokes através do método de projeção do tipo Chorin [20], inicialmente computamos um campo auxiliar $(u, v, w)^{aux}$,

$$\begin{aligned}
u^{aux} &= u + \Delta t\left[-\left(\frac{\partial (uu)}{\partial x} + \frac{\partial (uv)}{\partial y} + \frac{\partial (uw)}{\partial z}\right) + \frac{\mu}{\rho}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) + g_x\right] \\
v^{aux} &= v + \Delta t\left[-\left(\frac{\partial (uv)}{\partial x} + \frac{\partial (vv)}{\partial y} + \frac{\partial (vw)}{\partial z}\right) + \frac{\mu}{\rho}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) + g_y\right] \\
w^{aux} &= w + \Delta t\left[-\left(\frac{\partial (uw)}{\partial x} + \frac{\partial (vw)}{\partial y} + \frac{\partial (ww)}{\partial z}\right) + \frac{\mu}{\rho}\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) + g_z\right]
\end{aligned} \tag{4.2}$$

Observe que em tal campo auxiliar, não é levado em consideração o termo relativo a pressão $-\frac{\Delta t}{\rho}\nabla p$, o qual é inserido num segundo passo de forma a garantir a condição de divergência nula, $\nabla.(u, v, w)^{new} = 0$, conforme indicado pelas equações abaixo:

$$(u, v, w)^{new} = (u, v, w)^{aux} - \frac{\Delta t}{\rho}\nabla p \tag{4.3}$$

A condição de divergência nula determina então que,

$$\nabla.(u,v,w)^{new} = \nabla.\left((u,v,w)^{aux} - \frac{\Delta t}{\rho}\nabla p\right) = 0$$

de onde, assumindo que $\rho$ é constante, se tira que:

$$\Delta p = \frac{\rho}{\Delta t}\nabla.(u,v,w)^{aux} \tag{4.4}$$

onde $\Delta p$ representa o Laplaciano da pressão. A pressão $p$ obtida resolvendo-se a equação acima é aplicada em 4.3 para se obter, finalmente $(u,v,w)^{new}$.

## 4.2 Aproximação dos Termos Convectivos

Uma primeira abordagem para a aproximação dos termos convectivos da equação 4.2,

$$\begin{bmatrix} \frac{\partial(uu)}{\partial x} + \frac{\partial(uv)}{\partial y} + \frac{\partial(uw)}{\partial z} \\ \frac{\partial(uv)}{\partial x} + \frac{\partial(vv)}{\partial y} + \frac{\partial(vw)}{\partial z} \\ \frac{\partial(uw)}{\partial x} + \frac{\partial(vw)}{\partial y} + \frac{\partial(ww)}{\partial z} \end{bmatrix},$$

fica mais simples de explicar se os considerarmos na forma dada por $\boldsymbol{v}.\nabla(\boldsymbol{v})$, ou seja,

$$\begin{bmatrix} u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} \\ u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} \\ u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} \end{bmatrix}.$$

Cada parcela das somas que constituem as coordenadas do vetor acima é, então, tratada separadamente pelo Método de Lax-Wendroff que se aplica a equações diferenciais que tenham a forma:

$$\frac{\partial v(p,t)}{\partial t} + A(x)\frac{\partial v(p,t)}{\partial p} = 0, \tag{4.5}$$

onde, $p$ e $t$ são variáveis unidimensionais e $v$ e $A$ são funções também 1-D.

Por exemplo, a contribuição da primeira parcela da segunda linha será tratada aplicando-se Lax-Wendroff a equação

$$\frac{\partial v(x,y_0,z_0,t)}{\partial t} + u(x,y_0,z_0,t_0)\frac{\partial v(x,y_0,z_0,t)}{\partial x} = 0. \tag{4.6}$$

Fazendo $v = v(.,y_0,z_0,.)$, $A = u(.,y_0,z_0,t_0)$ e $p = x$, tem-se que 4.6 se enquadra no modelo dado pela equação 4.5.

A maneira como o procedimento de Lax-Wendroff trata a equação 4.5 pode ser explicada a partir da expressão de $v(x, t+\Delta t)$ dada pela série de Taylor até segunda ordem desenvolvida em torno de $v(x,t)$:

$$v(p,t+\Delta t) = v(p,t) + \frac{\partial v}{\partial t}(p,t)\Delta t + \frac{\partial^2 v}{\partial t^2}(p,t)\frac{\Delta t^2}{2} + O(\Delta t^3),$$

Usando 4.5 para substituir as derivadas no tempo por derivadas espaciais, temos

$$v(p,t+\Delta t) = v(p,t) - A(p)\frac{\partial v}{\partial p}(p,t)\Delta t - \frac{\partial}{\partial t}\left(A(p)\frac{\partial v}{\partial p}\right)(p,t)\frac{\Delta t^2}{2} + O(\Delta t^3).$$

Uma vez que $A$ é uma matriz que não varia com o tempo, substituindo a ordem de derivação do terceiro termo do lado esquerdo da igualdade, temos

$$v(p,t+\Delta t) = v(p,t) - A(p)\frac{\partial v}{\partial p}(p,t)\Delta t - A(p)\frac{\partial}{\partial p}\left(\frac{\partial v}{\partial t}\right)(p,t)\frac{\Delta t^2}{2} + O(\Delta t^3).$$

Usando novamente 4.5 para trocar a derivada no tempo pela derivada espacial, temos

$$v(p,t+\Delta t) = v(p,t) - A(p)\frac{\partial v}{\partial p}(p,t)\Delta t + A(p)\frac{\partial}{\partial p}\left(A(p)\frac{\partial v}{\partial p}\right)(p,t)\frac{\Delta t^2}{2} + O(\Delta t^3).$$

Aproximando as derivadas espaciais por diferenças centradas com espaçamento de $2\Delta p$ no segundo termo e de $\Delta p$ no terceiro termo, temos

$$\begin{aligned} v(p,t+\Delta t) = \quad & v(p,t) - A(p)\tfrac{v(p+\Delta p,t)-v(p-\Delta p,t)}{2\Delta p}\Delta t + \\ & A(p)\tfrac{\frac{\partial}{\partial p}(A(p)v)(p+\frac{\Delta p}{2},t)-\frac{\partial}{\partial p}(A(p)v)(p-\frac{\Delta p}{2},t)}{\Delta p}\tfrac{\Delta t^2}{2} + \\ & O(\Delta t^3). \end{aligned}$$

Novamente, aproximando as derivadas restantes por diferenças centradas e reorganizando os termos, obtemos

$$\begin{gathered} v(p,t+\Delta t) = v(p,t) - A(p)\tfrac{\Delta t}{2\Delta p}\left(v(p+\Delta p,t) - v(p-\Delta p,t)\right) + \\ A(p)\left\{\left(\tfrac{\Delta t}{\Delta p}\right)^2 A(p+\tfrac{\Delta p}{2}+\tfrac{\Delta p}{2})\left[v(p+\tfrac{\Delta p}{2}+\tfrac{\Delta p}{2},t)A(p+\tfrac{\Delta p}{2}-\tfrac{\Delta p}{2})v(p+\tfrac{\Delta p}{2}-\tfrac{\Delta p}{2},t)\right] - \right. \\ \left. \tfrac{1}{2}\tfrac{\Delta t^2}{\Delta p^2}\left[A(p-\tfrac{\Delta p}{2}+\tfrac{\Delta p}{2})v(p-\tfrac{\Delta p}{2}+\tfrac{\Delta p}{2},t) - A(p-\tfrac{\Delta p}{2}-\tfrac{\Delta p}{2},t)v(p-\tfrac{\Delta p}{2}-\tfrac{\Delta p}{2},t)\right]\right\} + \\ O(\Delta t^3). \end{gathered}$$

Daí tiramos que:

$$\begin{gathered} v(p,t+\Delta t) = v(p,t) - \tfrac{\Delta t}{2\Delta p}A(p)\left[v(p+\Delta p,t) - v(p-\Delta p,t)\right] + \\ \tfrac{\Delta t^2}{2\Delta p^2}A(p)\left[A(p+\Delta p)v(p+\Delta p,t) - A(p)v(p,t) - \right. \\ \left. (A(p)v(p,t)v(p,t) - A(p-\Delta p)v(p-\Delta p,t))\right] + \\ O(\Delta t^3). \end{gathered}$$

Assuma agora que $p$ e $t$ correspondem, respectivamente, aos índices $i$ da discretização em $p$ e $n$ da realizada em $t$. Considere também que os intervalos de amostragem nessas variáveis são $\Delta p$ e $\Delta t$ respectivamente. Nesse caso a expressão acima pode ser reescrita como

$$\begin{gathered} v_i^{n+1} = v_i^n - \frac{\Delta t}{2\Delta p} A_i \left(v_{i+1}^n - v_{i-1}^n\right) + \\ \frac{\Delta t^2}{\Delta p^2} A_i \left[\left(A_{i+1} v_{i+1}^n - A_i v_i^n\right) - \left(A_i v_i - A_{i-1} v_{i-1}^n\right)\right] + O(\Delta t^3). \end{gathered} \tag{4.7}$$

Substituindo

$$\frac{(v_{i+1} - v_{i-1})}{2}$$

por

$$\frac{(v_{i+1} + v_i)}{2} - \frac{(v_i - v_{i-1})}{2}$$

a expressão 4.7 se torna

$$\begin{gathered} v_i^{n+1} = v_i^n - \frac{\Delta t}{\Delta p} A_i \left[\left(\frac{v_{i+1}^n + v_i^n}{2} - \frac{\Delta t}{\Delta p} \frac{A_{i+1} v_{i+1}^n - A_i v_i^n}{2}\right) - \right. \\ \left. \left(\frac{v_{i-1}^n - v_i^n}{2} - \frac{\Delta t}{\Delta p} \frac{A_i v_i - A_{i-1} v_{i-1}^n}{2}\right)\right] + O(\Delta t^3). \end{gathered}$$

Defina agora $v_{i+\frac{1}{2}}^{n+\frac{1}{2}}$ como sendo a primeira expressão entre colchetes acima. Observe então que a segunda expressão entre colchetes será $v_{i-1+\frac{1}{2}=i-\frac{1}{2}}^{n+\frac{1}{2}}$. Feito isso podemos finalmente escrever a atualização de $v$ na forma característica do método de Lax-Wendroff

$$v_i^{n+1} = v_i^n - \frac{\Delta t}{\Delta p} A_i \left(v_{i+\frac{1}{2}}^{n+\frac{1}{2}} - v_{i-\frac{1}{2}}^{n+\frac{1}{2}}\right) \tag{4.8}$$

onde

$$v_{i+\frac{1}{2}}^{n+\frac{1}{2}} = \frac{1}{2}\left(v_{i+1}^n + v_i^n\right) - \frac{1}{2}\frac{\Delta t}{\Delta p}\left(A_{i+1} v_{i+1}^n - A_i v_i^n\right). \tag{4.9}$$

Definindo-se $A_{i+\frac{1}{2}}$ pela igualdade

$$\left(A_{i+1} v_{i+1}^n - A_i v_i^n\right) = A_{i+\frac{1}{2}}\left(v_{i+1}^n - v_i^n\right)$$

e aplicando-se essa mesma expressão para $i = i - \frac{1}{2}$

$$\left(A_{i+\frac{1}{2}} v_{i+\frac{1}{2}}^n - A_{i-\frac{1}{2}} v_{i-\frac{1}{2}}^n\right) = A_{i+\frac{1}{2}}\left(v_{i+\frac{1}{2}}^n - v_{i-\frac{1}{2}}^n\right)$$

as expressões dadas em 4.8 e 4.9 se tornam respectivamente

$$v_i^{n+1} = v_i^n - \frac{\Delta t}{\Delta p} A_{i+\frac{1}{2}} \left(v_{i+\frac{1}{2}}^{n+\frac{1}{2}} - v_{i-\frac{1}{2}}^{n+\frac{1}{2}}\right) \tag{4.10}$$

e

$$v_{i+\frac{1}{2}}^{n+\frac{1}{2}} = \frac{1}{2}\left(v_{i+1}^{n} + v_{i}^{n}\right) - \frac{1}{2}\frac{\Delta t}{\Delta p}A_{i+\frac{1}{2}}\left(v_{i+1}^{n} - v_{i}^{n}\right). \tag{4.11}$$

Um segundo ajuste na aproximação dos termos convectivos, busca tornar o método, da classe TVD (Total Variation Diminishing), na tentativa de diminuir possíveis ocilações de origem não física em regiões de gradientes pronunciados [29]. Isso é de suma importância para se obter soluções numéricas fisicamente aceitáveis.

Um método é dito TVD quando, sua aplicação reiterada produz uma sequência de aproximações $v_i^n$ da solução exata $v_i,\ i = 1, 2, ..., N$, com a seguinte propriedade:

$$\sum_i |v_{i+1}^{n+1} - v_i^{n+1}| \leq \sum_i |v_{i+1}^{n} - v_i^{n}|.$$

Nesse segundo ajuste se substitui a expressão dada em 4.11, por

$$v_{i+\frac{1}{2}}^{n+\frac{1}{2}} = v_i + \frac{1}{2}\left(1 - \frac{\Delta t}{\Delta x}a_{i+\frac{1}{2}}\right)\ g(r_i)\left(v_{i+1} - v_i\right). \tag{4.12}$$

Observe que 4.11 é um caso particular de 4.12, obtido quando fazemos $g(r_i) = 1$. Por simplicidade vamos fazer também $A_{i+\frac{1}{2}} = \left(A_i + A_{i+1}\right)/2$. $g$ é a denominada função limitadora, seu parâmetro $r$ é dado pela razão entre dois gradientes consecutivos:

$$r_i = \begin{cases} \frac{v_i - v_{i-1}}{v_{i+1} - v_i} & \text{se } A_{i+\frac{1}{2}} > 0 \\ \frac{v_{i+1} - v_i}{v_i - v_{i-1}} & \text{se } A_{i+\frac{1}{2}} < 0 \end{cases} \tag{4.13}$$

e $g$ satisfaz a seguinte condição

$$\begin{cases} g(r) = 0 & \text{se } r \leq 0 \\ 0 \leq g(r) \leq \min\{2, 2r\} & \text{se } r > 0 \end{cases}, \tag{4.14}$$

esta condição, chamada condição de Sweby [74], garante que tal método sob a condição CFL (Courant-Friedrichs-Lewy), $\left|A\frac{\Delta t}{\Delta x}\right| \leq 1$, é TVD.

A condição de Sweby, interpretada graficamente, informa que o gráfico de $g$ deve estar contido na região rachurada da figura 4.5.

A formulação dada pela expressão 4.12 pode ser reescrita em termos das varáveis normalizadas introduzido por Gaskell e Lau[33], onde o valor de $v_{i+\frac{1}{2}}^{n+\frac{1}{2}}$ é determinado por dois nós $(v_D^n, v_R^n)$ ilustrados na figura 4.6.

A normalização de $v$ é dada por

$$\hat{v}(x,t) = \frac{v(x,t) - v_R^n}{v_D^n - v_R^n}, \tag{4.15}$$

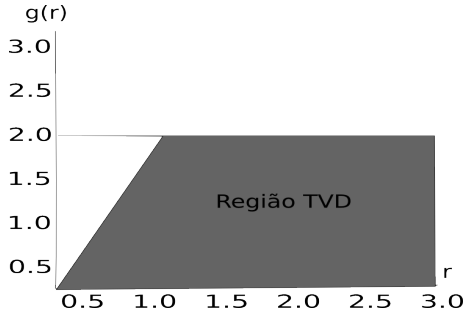



Figura 4.5: Diagrama de Sweby

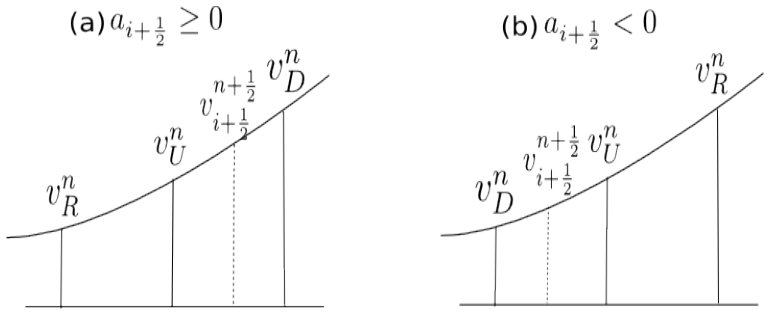



Figura 4.6: Localização de $v_R^n, v_U^n, v_D^n$ e $v_{i+\frac{1}{2}}^{n+\frac{1}{2}}$

ou seja,

Substituindo $v_{i+1}$ por $v_D$ na equação 4.12 temos que a mesma e a equação 4.13 em variáveis normalizadas passam a ser dadas, repectivamente, por

$$\hat{v}_{i+\frac{1}{2}}^{n+\frac{1}{2}} = \hat{v}_U + \frac{1}{2}\left(1 - \frac{\Delta t}{\Delta x} a_{i+\frac{1}{2}}\right) g(r_i)\,(1 - \hat{v}_U) \tag{4.16}$$

e

$$r_i = \frac{\hat{v}_U}{1 - \hat{v}_U}. \tag{4.17}$$

Por simplicidade a partir de agora vamos representar $\hat{v}_{i+\frac{1}{2}}^{n+\frac{1}{2}}$ por $\hat{v}_f$. A partir das equações 4.16 e 4.17, e da condição de Sweby (equações 4.14) temos que:

1. Por 4.17, $r_i \leq 0$ se, e somente se, $\frac{\hat{v}_U}{1-\hat{v}_U} \leq 0$, ou seja, $r \leq 0$ se, e somente se, $\hat{v}_U \leq 0$
   ou $\hat{v}_U \geq 1$;

2. Pela primeira equação de 4.14, $r \leq 0$ implica $g(r) = 0$;

3. Por 4.16, $g(r) = 0$ implica $\hat{v}_f = \hat{v}_U$;

4. Por 1, 2 e 3 podemos concluir que $\hat{v}_f = \hat{v}_U$ para $\hat{v}_U \leq 0$ ou $\hat{v}_U \geq 1$ ;

5. Por 4.17 $r > 0$ se, e somente se, $\frac{\hat{v}_U}{1-\hat{v}_U} > 0$, ou seja, $r > 0$ se, e somente se, $0 < \hat{v}_U < 1$;

6. Pela segunda equação de 4.14, $r > 0$ implica $0 < g(r) < \min\{2, 2r\}$;

7. Por 4.16, $0 < g(r) < \min\{2, 2r\}$ implica $\hat{v}_U \leq \hat{v}_f \leq \min\{1 - \frac{\Delta t}{\Delta x}a_{i+\frac{1}{2}} + \frac{\Delta t}{\Delta x}a_{i+\frac{1}{2}}\hat{v}_U, (2 - \frac{\Delta t}{\Delta x}a_{i+\frac{1}{2}})\hat{v}_U\}$;

8. Por (v), (vi) e (vii) podemos concluir que $\hat{v}_U \leq \hat{v}_f \leq \min\{1 - \frac{\Delta t}{\Delta x}a_{i+\frac{1}{2}} + \frac{\Delta t}{\Delta x}a_{i+\frac{1}{2}}\hat{v}_U, \left(2 - \frac{\Delta t}{\Delta x}a_{i+\frac{1}{2}}\right)\hat{v}_U\}$; para $0 < \hat{v}_U < 1$

Ou seja, por 4 e 8, a relação entre $\hat{v}_f$ e $\hat{v}_U$ que satisfaz a condição de Sweby, pode ser ser expressa por

$$\begin{cases} \hat{v}_U \leq \hat{v}_f \leq \min\{1 - \frac{\Delta t}{\Delta x}a_{i+\frac{1}{2}} + \frac{\Delta t}{\Delta x}a_{i+\frac{1}{2}}\hat{v}_U, (2 - \frac{\Delta t}{\Delta x}a_{i+\frac{1}{2}})\hat{v}_U\} & \text{para } 0 < \hat{v}_U < 1 \\ \hat{v}_f = \hat{v}_U & \text{para } \hat{v}_U \leq 0 \text{ ou } \hat{v}_U \geq 1; \end{cases} \tag{4.18}$$

Graficamente, temos que para cada valor de $\nu = \frac{\Delta t}{\Delta x}a_{i+\frac{1}{2}} \in [0, 1]$, a região correspondente a este valor de $\nu$ deve estar contida na região TVD global apresentada na figura 4.7 definida pelas equações 4.18.

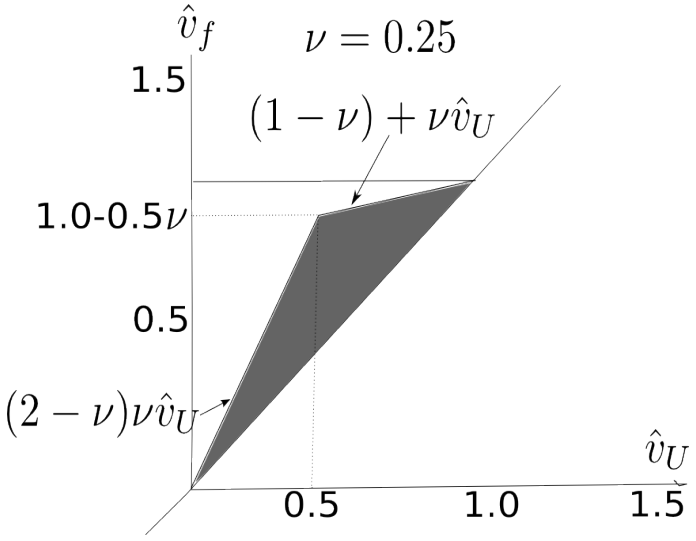



Figura 4.7: Região TVD em variáveis normalizadas

A região TVD é definida como sendo aquela em que o esquema é TVD para algum valor de $v$. Ela é dada pelo triângulo $[(0, 0), (0.5, 1), (1, 1)]$. Esse triângulo é

obtido tomando-se as retas limitadoras $\hat{v}_f = \hat{v}_U, \hat{v}_f = (2-\nu)\hat{v}_U e \hat{v}_f = (1-\nu) + \nu\hat{v}_U$ indicadas na figura 4.7 e fazendo $\nu = 0$.

Entre os esquemas mais conhecidos onde o gráfico de $\hat{v}_f$ encontra-se inteiramente na região TVD temos:

**WACEB:** Desenvolvido por Song et al. [70], o esquema WACEB (Weight-Average Coefficient Ensuring Boundedness) é dado por

$$\hat{v}_f = \begin{cases} \hat{v}_U & \text{se } \hat{v}_U \notin [0,1], \\ 2\hat{v}_U & \text{se } 0 \leq \hat{v}_U < 3/10, \\ \frac{1}{8}(3 + 6\hat{v}_U) & \text{se } 3/10 \leq \hat{v}_U \leq 5/6, \\ 1 & \text{se } 5/6 < \hat{v}_U < 1. \end{cases}$$

**CUBISTA:** O esquema CUBISTA (Convergent and Universally Bounded Interpolation Scheme for Treatment of Advection) foi desenvolvido por Alves et al [3], sendo dado por

$$\hat{v}_f = \begin{cases} \hat{v}_U & \text{se } \hat{v}_U \notin [0,1], \\ \frac{1}{4}(7\hat{v}_U - 3\hat{v}_R) & \text{se } 0 < \hat{v}_U < 3/8, \\ \frac{1}{8}(3\hat{v}_D + 6\hat{v}_U - \hat{v}_R) & \text{se } 3/8 \leq \hat{v}_U \leq 3/4, \\ \frac{1}{4}(3\hat{v}_D + \hat{v}_U) & \text{se } 3/4 < \hat{v}_U < 1. \end{cases}$$

Este esquema é TVD para $\nu \leq \frac{1}{4}$.

**ADBQUICKEST:** O esquema ADBQUICKEST (Adaptative Bounded Quadratic Upstream for Convective Kinematics with Estimated Streaming Terms), desenvolvido por Ferreira at al. [30], se ajusta ao valor de $\nu$ para manter a propriedade de ser TVD e é dado por

$$\hat{v}_f = \begin{cases} \hat{v}_U & \text{se } \hat{v}_U \notin [0,1], \\ (2-\nu)\hat{v}_U & \text{se } 0 < \hat{v}_U < a, \\ \alpha_D + \alpha_U\hat{v}_U & \text{se } a \leq \hat{v}_U \leq b, \\ (1-\nu) + \theta\hat{v}_U & \text{se } b < \hat{v}_U < 1. \end{cases}$$

onde

$$\alpha_D = \tfrac{1}{6}(2 - 3|\nu| + \nu^2),$$
$$\alpha_U = \tfrac{1}{6}(5 + 3|\nu| - 2\nu^2)$$

e

$$a = \frac{2-\|\nu\|+\nu^2}{7-6\nu-3\|\nu\|+2\nu^2},$$
$$b = \frac{-4+6\nu-3\|\nu\|+\nu^2}{-5+6\nu-3\|\nu\|+2\nu^2}.$$

Outros esquemas bastantes populares não atendem a condição de TVD, mas apenas garantem que a solução obtida será limitada. Essa propriedade é decorrente desses esquemas satisfazerem o chamado Conection Bounded Criterium (CBC) que é atendido se restringimos o gráfico de $\{\hat{v}_U \in [0,1]\} \to \hat{v}$ a parte acima da diagonal do quadrado unitário, como mostra a figura 4.8. Entre esses métodos podemos listar:

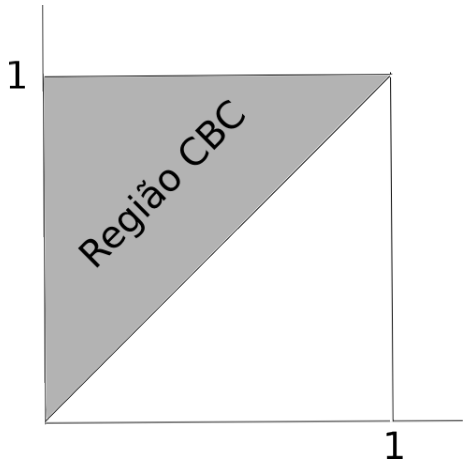



Figura 4.8: Região CBC

1. Interpolação Ajustada Quadratica para Cinemática Convectiva (Quadratic upwind interpolation for convective kinematics-QUICK)

$$\hat{v}_f = \begin{cases} \hat{v}_U & \text{se } \hat{v}_U \notin [0,1], \\ \frac{3}{4}\hat{v}_U + \frac{3}{8} & \text{se } \hat{v}_U \in [0,1] \end{cases}$$

2. Aproximação Parabólica Híbrida Linear (Hybrid-Linear Parabolic Aproximation-HLPA)

$$\hat{v}_f = \begin{cases} \hat{v}_U & \text{se } \hat{v}_U \notin [0,1], \\ (2-\hat{v}_U)\,\hat{v}_U & \text{se } \hat{v}_U \in [0,1] \end{cases}$$

3. Algoritmo sobre Funções Monotônicas e Acentuadas para Transporte Realis-

tico (Sharp and Monotonic Algorithm for Realistic Transport-SMART)

$$\hat{v}_f = \begin{cases} \hat{v}_U & \text{se } \hat{v}_U \notin [0,1], \\ 10\hat{v}_U & \text{se } \hat{v}_U \in [0, \frac{3}{74}), \\ \frac{3}{8}\left(1 + 2\hat{v}_U\right) & \text{se } \hat{v}_U \in [\frac{3}{74}, \frac{5}{6}), \\ 1 & \text{se } \hat{v}_U \in [\frac{5}{6}, 1]. \end{cases}$$

4. Esquema não Oscilatório de Ordem Variável (Variable-Order Non-Oscillatory Scheme-VONOS)

$$\hat{v}_f = \begin{cases} \hat{v}_U & \text{se } \hat{v}_U \notin [0,1], \\ 10\hat{v}_U & \text{se } \hat{v}_U \in [0, \frac{3}{74}), \\ \frac{3}{8}\left(1 + 2\hat{v}_U\right) & \text{se } \hat{v}_U \in [\frac{3}{74}, \frac{1}{2}), \\ 1,5\hat{v}_U & \text{se } \hat{v}_U \in [\frac{1}{2}, \frac{2}{3}), \\ 1 & \text{se } \hat{v}_U \in [\frac{2}{3}, 1]. \end{cases} \tag{4.19}$$

Para expressar os esquemas acima em variáveis não normalizadas basta substituir $\hat{v}_U$ por $\frac{v_U - v_R}{v_D - v_R}$ como indicado na equação 4.15.

Tomando a segunda linha da definição do método de VONOS, 4.19, como exemplo temos:

$$\frac{v_f - v_R}{v_D - v_R} = 10\frac{v_U - v_R}{v_D - v_R},$$

de onde tiramos que $v_f = 10v_U - 9v_R$.

Fazendo essa substituição para as demais linhas do método VONOS, 4.19, temos ao final:

$$v_f = \begin{cases} v_U & \text{se } \hat{v}_U \notin [0,1], \\ 10v_U - 9v_R & \text{se } \hat{v}_U \in [0, \frac{3}{74}), \\ \frac{1}{8}\left(3v_D + 6v_U - v_R\right) & \text{se } \hat{v}_U \in [\frac{3}{74}, \frac{1}{2}), \\ 1,5v_U - 0.5v_R & \text{se } \hat{v}_U \in [\frac{1}{2}, \frac{2}{3}), \\ v_D & \text{se } \hat{v}_U \in [\frac{2}{3}, 1]. \end{cases} \tag{4.20}$$

Assim, ao final do processo, a discretização dos termos convectivos baseada no método de Lax-Wendroff, é dada, por exemplo para o termo $\frac{\partial(vv)}{\partial x}$ por

$$\frac{\partial(vv)}{\partial x} = \frac{a^n_{i+\frac{1}{2}} v^{n+\frac{1}{2}}_{i+\frac{1}{2}} - a^n_{i-\frac{1}{2}} v^{n-\frac{1}{2}}_{i-\frac{1}{2}}}{\Delta x} \tag{4.21}$$

onde $v^{n+\frac{1}{2}}_{i+\frac{1}{2}}$ é dado por uma das relações apresentadas para $v_f$, e $a_{i+\frac{1}{2}} = v^n_{i+\frac{1}{2}}$ é aproximado por $\frac{v^n_{i+1} + v^n_i}{2}$.

## 4.3 Aproximação dos Termos Difusos

Para a aproximação dos termos difusos usamos o esquema de diferenças centradas. Para coordenada $u$ temos

$$\begin{aligned}\frac{\partial^2 u}{\partial x^2} &= \frac{1}{h^2}\left(u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}\right)\\ \frac{\partial^2 u}{\partial y^2} &= \frac{1}{h^2}\left(u_{i,j+1,k} - 2u_{i,j,k} + u_{i,j-1,k}\right)\\ \frac{\partial^2 u}{\partial z^2} &= \frac{1}{h^2}\left(u_{i,j,k+1} - 2u_{i,j,k} + u_{i,j,k-1}\right)\end{aligned} .$$

$\nabla^2 v$ e $\nabla^2 w$ são aproximados de forma análoga.

Em relação ao tratamento da viscosidade devemos fazer as seguintes considerações:

Não pode existir modelo de fluido discretizado em uma malha, que seja mais viscoso que aquele em que o efeito da viscosidade transforma a velocidade $\boldsymbol{v_p}$ num vértice da malha $\boldsymbol{p}$ no valor médio das velocidades nos vértices vizinhos ($\boldsymbol{v_m}$). Escolhas inadequadas do par (coeficiente de viscosidade, time-step) podem, entretanto, fazer com que esse valor médio seja ultrapassado, e a velocidade no vértice se tornar $\boldsymbol{v_p} + \lambda(\boldsymbol{v_m} - \boldsymbol{v_p})$ com $\lambda > 1$. Para evitar que isso aconteça, pode-se pensar em reduzir o time-step tornando a simulação consideravelmente mais lenta ou se aplica o chamado "tratamento implícito", em que a desaceleração de uma componente de $\boldsymbol{v_p}$, determinada pela viscosidade passa a ser proporcional a derivada segunda da diferença entre o valor atual da componente no vértice $\boldsymbol{p}$ e a média dos vizinhos no passo seguinte, ao invés de na iteração atual. Isso determina que mais um sistema, similar ao de Poisson tenha que ser resolvido, o que também torna o procedimento mais pesado. Em relação a esse ponto, no sentido de continuar a computar a influência da viscosidade de uma forma inteiramente explícita, se pode alternativamente obtê-la através da expressão:

$$\boldsymbol{v_p} = \boldsymbol{v_m} + e^{-\frac{6k}{(\Delta x)^2}\Delta t}(\boldsymbol{v_p} - \boldsymbol{v_m})$$

onde $k$ é o coeficiente de viscosidade. Essa expressão aproxima em $\boldsymbol{v} = \boldsymbol{v}_p$ a solução de

$$\frac{d\left(\boldsymbol{v} - \boldsymbol{v}_m\right)}{dt} = -\frac{6k}{(\Delta x)^2}(\boldsymbol{v} - \boldsymbol{v}_m).$$

Para $(\boldsymbol{v} = \boldsymbol{v}_p)$ o lado direito dessa equação se torna $\Delta^2(\boldsymbol{v}_p)$. Assim, garantims que o novo valor de $\boldsymbol{v}_p$ ficará restrito ao segmento $[\boldsymbol{v}_p, \boldsymbol{v}_m)$. Se o decaimento determinado

pela exponencial for considerado rápido pode-se diminuir o valor do coeficiente de viscosidade $k$.

## 4.4 Aproximação do Laplaciano da Pressão

A aproximação do Laplaciano da pressão, é feita utilizando-se um aninhamento do esquema de diferenças centradas, ou seja, aplicando-se o esquema de diferenças centradas para a divergência do gradiente de pressão, e em seguida, aplicando-se novamente o esquema de diferenças centradas para o gradiente de pressão:

$$\begin{aligned}
\Delta p &= \nabla.(\nabla p) \\
&= \frac{\left(\frac{\partial p}{\partial x}\right)_{i+\frac{1}{2},j,k} - \left(\frac{\partial p}{\partial x}\right)_{i-\frac{1}{2},j,k}}{\Delta x} + \frac{\left(\frac{\partial p}{\partial y}\right)_{i,j+\frac{1}{2},k} - \left(\frac{\partial p}{\partial x}\right)_{i,j-\frac{1}{2},k}}{\Delta y} + \frac{\left(\frac{\partial p}{\partial z}\right)_{i,j,k+\frac{1}{2}} - \left(\frac{\partial p}{\partial x}\right)_{i,j,k-\frac{1}{2}}}{\Delta z} \\
&= \frac{\frac{p_{i+1,j,k}-p_{i,j,k}}{\Delta x} - \frac{p_{i,j,k}-p_{i-1,j,k}}{\Delta x}}{\Delta x} + \frac{\frac{p_{i,j+1,k}-p_{i,j,k}}{\Delta y} - \frac{p_{i,j,k}-p_{i,j-1,k}}{\Delta y}}{\Delta y} + \frac{\frac{p_{i,j,k+1}-p_{i,j,k}}{\Delta z} - \frac{p_{i,j,k}-p_{i,j,k-1}}{\Delta z}}{\Delta z} \\
&= \frac{p_{i+1,j,k}-2p_{i,j,k}+p_{i-1,j,k}}{(\Delta x)^2} + \frac{p_{i,j+1,k}-2p_{i,j,k}+p_{i,j-1,k}}{(\Delta y)^2} + \frac{p_{i,j,k+1}-2p_{i,j,k}+p_{i,j,k-1}}{(\Delta z)^2}
\end{aligned} \tag{4.22}$$

A aproximação do lado direito da equação 4.4, aqui denominado de $LD$, é feita através do esquema de diferenças avançadas:

$$LD_{i,j,k} = \frac{u_{i,j,k} - u_{i-1,j,k}}{\Delta x} + \frac{v_{i,j,k} - v_{i,j-1,k}}{\Delta y} + \frac{w_{i,j,k} - w_{i,j,k-1}}{\Delta z} \tag{4.23}$$

As aproximações 4.22 e 4.23 dos respectivos termos da equação 4.4, recai num sistema linear do tipo $A\boldsymbol{x} = \boldsymbol{b}$ onde a matriz dos coeficientes $A$ é uma matriz heptagonal, simétrica e positiva definida, $\boldsymbol{x}$ representa a pressão $p$ e $\boldsymbol{b}$ o divergente da velocidade.

Podemos observar que para simulções em grades com $n > 2$ elementos em cada uma das três direções, a matriz $A$, é uma matriz com $n^6$ elementos sendo desse total, apenas $7n^3 - 6n^2$ elementos diferentes de zero, o que em geral, a torna uma matriz esparsa e muito extensa para um armazenamento convecional.

No apêndice A, apresentamos uma breve descrição e os respectivos algoritmos, de alguns dos principais métodos para solução de sistemas $Ax = b$ onde a matriz $A$ é uma matriz extensa (mais de 1.000.000), simétrica e positiva definida. Dentre os métodos apresentados estão o método do gradiente conjugado e o método do gradiente biconjugado estabilizado precondicionado [64]. Observamos que esses métodos são preferidos aos que empregam pivoteamentos dado que eles permitem preservar a esparsidade.

O estudo desssas metodologias é relevante considerando-se que numa implementação sequencial resolver o sistema de Poisson pode representar mais de 80% do tempo gasto em cada iteração.

# Capítulo 5

# Evolução da Interface Ar-Líquido: A abordagem via Level Sets/Fast Marching

No estado da arte atual, na simulação de escoamento de fluidos com superfície livre, a metodologia mais difundida para representar e evoluir a interface Ar-Liquido, superfície livre do fluido, consiste em aplicar a o método Level Set Method conjugado ao método Fast Marching Method [65] [63].

Neste capítulo apresentamos alguns conceitos preliminares ao processo de evolução de borda e fazemos uma descrição desses dois métodos focando alguns aspectos teóricos e computacionais da área de interface tracking.

## 5.1 Preliminares

Por questões de simplicidade, iremos iniciar nossa apresentação tratando domínios bidimensionais e depois estenderemos o exposto para o $R^3$.

Nas suseções seguintes, apresentamos alguns conceitos preliminares utilizados pelo método dos level set.

### 5.1.1 Fronteira

Como vamos trabalhar apenas com conjuntos regulares contidos num subespaço compacto, o conceito fronteira vai se referir sempre a uma curva -no caso 2D- ou

suprefície fechada [58]. Fronteira livre é a que pode evoluir sem estar restrita pelos limites do domínio ou obstáculos.

Muitas vezes, no processo de evolução das fronteiras, estamos interessados apenas no que ocorre em sua direção normal, por exemplo, quando consideramos a fronteira como ente geométrico e não material. Em particular, este é o caso de nossa simulação.

Assim sendo, considere uma fronteira evoluindo em sua direção normal, com uma velocidade conhecida $V$, conforme exibido na figura 5.1. A meta é traçar o movimento desta fronteira à medida que a mesma evolui. A velocidade V é,

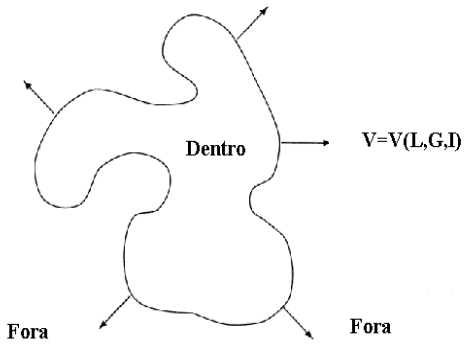



Figura 5.1: Curva se propagando com velocidade V na direção normal.

tipicamente uma função de 3 tipos de propriedades, V=V(L,G,I), onde L, G e I representam, respectivamente:

- Propriedades Locais (L), isto é, que dependem de informações geométricas locais, por exemplo, curvatura e direção normal;
- Propriedades Globais (G), ou seja, propriedades que dependem da forma e posição da fronteira, por exemplo, integrais de linha ao longo da fronteira;
- Propriedades Independentes (I), aquelas que não dependem da fronteira, por exemplo, a velocidade de um fluido subjacente que de modo passivo transporta a fronteira.

### 5.1.2 Curvas Planas

Seja $\gamma(t), t \in [0, \infty)$ a família de curvas gerada pelo movimento de uma curva inicial $\gamma$. E seja

$$\boldsymbol{x} = \boldsymbol{x}(s,t) = (x(s,t), y(s,t)), \qquad 0 \leq s \leq S,$$

uma parametrização da curva $\gamma(t)$. $S$ é uma constante real. Por ser fechada, devemos ter $\boldsymbol{x}(0,t) = \boldsymbol{x}(S,t)$, uma condição de periodicidade na fronteira.

Consideraremos que a parte interior à fronteira fique à esquerda da direção crescente de $s$, conforme ilustrado na figura 5.2. A tangente à curva $\gamma(t)$ em $\boldsymbol{x}$ será

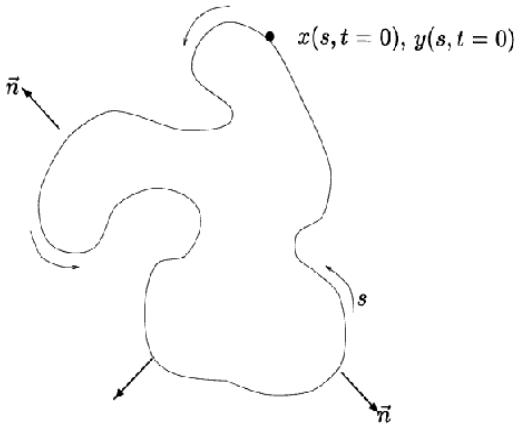



Figura 5.2: Curva parametrizada

notada por $(x_s, y_s)$, onde $x_s = \frac{\partial x(s,t)}{\partial s}$ e $y_s = \frac{\partial y(s,t)}{\partial s}$. O vetor unitário normal externo a $\gamma(t)$ no ponto $\boldsymbol{x}$ é dado por

$$\boldsymbol{n}(s,t) = \frac{(y_s, -x_s)}{(x_s^2 + y_s^2)^{\frac{1}{2}}}. \tag{5.1}$$

A velocidade normal de propagação, $V$, é dada por

$$V = \boldsymbol{x_t}.\boldsymbol{n} \tag{5.2}$$

onde $\boldsymbol{x_t} = (x_t, y_t)$ é o vetor velocidade da curva.

Temos também que o ângulo de inclinação do vetor normal (ângulo entre o vetor normal e o eixo x) pode ser dado por

$$\theta = -arctan\left(\frac{x_s}{y_s}\right).$$

Além disso, a curvatura local da curva, denotada por $k = k(s,t)$, que representa o quociente entre a taxa de variação do ângulo de inclinação do vetor normal à curva em relação ao comprimento de arco da curva, $S^*$, é dada por

$$k = k(s,t) = \theta_s \frac{ds}{ds^*} = \frac{\theta_s}{(x_s^2 + y_s^2)^{\frac{1}{2}}} = \frac{y_{ss}x_s - x_{ss}y_s}{(x_s^2 + y_s^2)^{\frac{3}{2}}} \tag{5.3}$$

Uma vez que estamos interessados apenas na evolução da fronteira em sua direção normal, podemos assumir que $\boldsymbol{x_t}$ é normal a $\boldsymbol{x_s}$. Neste caso, o sistema de equações para o movimento da curva em evolução pode ser dado por

$$\begin{cases} x_t = V(L,G,I)\frac{y_s}{(x_s^2+y_s^2)^{\frac{1}{2}}} \\ y_t = V(L,G,I)\frac{-x_s}{(x_s^2+y_s^2)^{\frac{1}{2}}} \end{cases} \tag{5.4}$$

### 5.1.3 Discretização do sistema de equações em 2D

Entre as múltiplas possibilidades que expressões em diferenças finitas nos permite, escolhemos empregar diferenças avançadas no tempo, e centradas no espaço. Além disso, utilizamos o mesmo espaçamento de malha em $x$ e $y$. Como resultado temos:

$$\begin{aligned} x_i^{n+1} &= x_i^n + \Delta t \quad V(L,G,I) \quad \frac{y_{i+1}^n - y_{i-1}^n}{\left(\left(x_{i+1}^n - x_{i-1}^n\right)^2 + \left(y_{i+1}^n - y_{i-1}^n\right)^2\right)^{\frac{1}{2}}} \\ y_i^{n+1} &= y_i^n - \Delta t \quad V(L,G,I) \quad \frac{x_{i+1}^n - x_{i-1}^n}{\left(\left(x_{i+1}^n - x_{i-1}^n\right)^2 + \left(y_{i+1}^n - y_{i-1}^n\right)^2\right)^{\frac{1}{2}}}. \end{aligned} \tag{5.5}$$

### 5.1.4 Função distância a uma curva

Seja $\gamma$ uma curva fechada em $R^2$ e $\Phi : R^2 \to R$ uma função tal que $\Phi(\boldsymbol{x}) = 0$ para todo $\boldsymbol{x} \in \gamma$. $\Phi$ é dita uma função distância com sinal a curva $\gamma$ se:

$$\Phi(\boldsymbol{x}) = \begin{cases} d(\boldsymbol{x},\gamma) & \text{para } \boldsymbol{x} \text{ externo a } \gamma \\ -d(\boldsymbol{x},\gamma) & \text{para } \boldsymbol{x} \text{ interno a } \gamma \end{cases}$$

onde $d(\boldsymbol{x},\gamma) = inf_{\boldsymbol{y}\in\gamma}\|\boldsymbol{x} - \boldsymbol{y}\|$ para alguma norma no $R^2$. Em particular tomamos $\|\boldsymbol{x}\|$ como a norma euclidiana de $\boldsymbol{x}$.

Uma importante propriedade da função distância, a ser utilizada mais adiante, cujo esboço da demonstração encontra-se em [65] e [2], é que $\|\nabla\Phi\| = 1$, onde ela for diferenciável.

## 5.2 Método Level Set

A idéia fundamental do método Level Set, consiste na mudança do modo de observação de uma fronteira em evolução. Originalmente tínhamos uma perspectiva paramétrica onde faz-se evoluir os parâmetros que definem a curva, obtida de forma explícita a partir deles. Passamos a uma perspectiva implícita onde a fronteira em movimento é representada como uma curva de nível de uma função que é feita evoluir.

Dada uma fronteira inicial $\gamma$, e uma função distância com sinal a ela $\Phi_0$ assumimos que $\gamma$ é a curva de nível zero da função $\Phi_0$.

O método Level Set é constituído por uma equação de evolução para uma função $\Phi = \Phi(\boldsymbol{x}, t)$, com condição inicial $\Phi(\boldsymbol{x}, t = 0) = \Phi_0(\boldsymbol{x})$, que captura o movimento de $\gamma(t)$ como curva de nível zero da função $\Phi$, isto é,

$$\gamma(t) = \{\boldsymbol{x} \mid \Phi(\boldsymbol{x}, t) = 0\}.$$

De forma simplificada, a estratégia deste método é, em vez de evoluir apenas os pontos da fronteira, evoluir a função $\Phi$ mantendo a fronteira sempre no nível zero da função $\Phi$ evoluída.

A exigência de que a curva de nível zero da função $\Phi$ seja sempre igual à fronteira se propagando, implica em $\Phi(\boldsymbol{x}, t) = 0$ sempre que $\boldsymbol{x}$ for um ponto da fronteira. Conseqüentemente, considerando $\boldsymbol{x}(t)$ os pontos da fronteira no instante $t$, pela regra da cadeia, temos que

$$\Phi_t + \nabla\Phi(\boldsymbol{x}(t), t) \cdot \boldsymbol{x'(t)} = 0. \tag{5.6}$$

No caso em que a velocidade $\boldsymbol{v} = (u, v, w)$ sobre a fronteira é dada, podemos reescrever esta equação como

$$\Phi_t + \boldsymbol{v} \cdot \nabla\Phi(\boldsymbol{x}(t), t) = 0. \tag{5.7}$$

A equação 5.7 é também conhecida como *level set equation*, captura o movimento da fronteira caracterizada por $\Phi(x) = 0$ e foi introduzida por Osher e Sethian em [66].

### 5.2.1 Extensão para o Caso Tridimensional

Dentre os principais aspectos favoráveis a esta formulação implicita, método Level Set, em relação à formulação paramétrica, está a facilidade para extendermos a formulação para o caso tridimensional. Para traçarmos a evolução de fronteiras em espaços tridimensionais (em nosso caso, superfícies de nínveis) basta estendermos o vetor posição e o operador gradiente.

Outros aspectos favoráveis que já podemos citar são:

- **Mudanças Topológicas**: Para qualquer função velocidade, $V$, suave, a função $\Phi$ permanecerá sendo uma função continua, mantendo seu grau de diferenciabilidade fora do nível zero a medida que evoluir. Contudo, a curva se propagando, $\gamma(t)$, poderá mudar de topologia, quebrar, fundir-se e/ou formar quinas, sem gerar nehuma dificuldade [65]. Já na formulação paramétrica essas mudanças topológicas precisam ser dectadas e tratadas de forma específica.

- **Propriedades Geométricas**: Algumas propriedades geométricas intrínsecas da fronteira são facilmente determinadas a partir da função curva de nível $\Phi$.

  **Vetor normal**: Em qualquer ponto da fronteira, o vetor unitário normal externo é dado por

  $$\boldsymbol{n} = \frac{\nabla\Phi}{|\nabla\Phi|} \tag{5.8}$$

  **Curvatura**: A curvatura da fronteira pode ser obtida através da divergência de seu vetor unitário normal, $k = -\nabla.\left(\frac{\nabla\Phi}{|\nabla\Phi|}\right)$. Para provar esse fato, inicialmente notamos que o vetor normal é dado por $\nabla\Phi = (\Phi_x, \Phi_y)$, donde temos que $\theta$, o ângulo formado entre o vetor normal $\boldsymbol{n}$ e o eixo $x$, é dado por $\theta = arctan\left(\frac{\Phi_y}{\Phi_x}\right)$.

  Assuma que a curva é parametrizada por $y = f(x)$. Assim, $\Phi(x, f(x)) = 0$ e, derivando em relação a $x$, tem-se que

  $$\Phi_x + \Phi_y y_x = 0$$

  donde

  $$y_x = \frac{-\Phi_x}{\Phi_y}.$$

Seja $\Theta(x) = \theta(x, f(x)) = arctan\left(\frac{\Phi_y(x,f(x))}{\Phi_x(x,f(x))}\right)$, calculando $\Theta_x$ obtemos:

$$\begin{aligned}
\Theta_x &= \frac{1}{1+\left(\frac{\Phi_y}{\Phi_x}\right)^2}\left(\frac{\Phi_y}{\Phi_x}\right)_x \\
&= \frac{1}{1+\left(\frac{\Phi_y}{\Phi_x}\right)^2} \quad \frac{\left(\Phi_{yx}+\Phi_{yy}y_x\right)\Phi_x - \left(\Phi_{xx}+\Phi_{xy}y_x\right)\Phi_y}{\Phi_x^2} \\
&= \frac{\Phi_x^2}{\Phi_x^2+\Phi_y^2} \quad \frac{\Phi_{yx}\Phi_x+\Phi_{yy}\Phi_x y_x - \Phi_{xx}\Phi_y - \Phi_{xy}\Phi_y y_x}{\Phi_x^2} \\
&= \frac{\Phi_{yx}\Phi_x+\Phi_{yy}\Phi_x y_x - \Phi_{xx}\Phi_y - \Phi_{xy}\Phi_y y_x}{\Phi_x^2+\Phi_y^2} \\
&= \frac{\Phi_{yx}\Phi_x+\Phi_{yy}\Phi_x\left(-\frac{\Phi_x}{\Phi_y}\right) - \Phi_{xx}\Phi_y - \Phi_{xy}\Phi_y\left(-\frac{\Phi_x}{\Phi_y}\right)}{\Phi_x^2+\Phi_y^2}
\end{aligned}$$

e uma vez que a curvatura de uma curva representa o quociente entre a taxa de variação do ângulo de inclinação do vetor normal à curva em relação ao comprimento de arco da curva, temos que

$$\begin{aligned}
k &= \frac{\Theta_x}{\left(\left(\frac{dx}{dx}\right)^2+\left(\frac{dy}{dx}\right)^2\right)^{\frac{1}{2}}} \\
&= \frac{\frac{\Phi_{yx}\Phi_x+\Phi_{yy}\Phi_x\left(-\frac{\Phi_x}{\Phi_y}\right)-\Phi_{xx}\Phi_y-\Phi_{xy}\Phi_y\left(-\frac{\Phi_x}{\Phi_y}\right)}{\Phi_x^2+\Phi_y^2}}{\left(1+\left(\frac{\Phi_x}{\Phi_y}\right)^2\right)^{\frac{1}{2}}} \\
&= -\frac{\Phi_{xx}\Phi_y^2 - 2\Phi_{xy}\Phi_x\Phi_y + \Phi_{yy}\Phi_x^2}{\left(\Phi_x^2+\Phi_y^2\right)^{\frac{3}{2}}} \\
&= \nabla\left(\frac{(\Phi_x,\Phi_y)}{\sqrt{\Phi_x^2+\Phi_y^2}}\right)
\end{aligned}$$

## 5.3 Evolução da Função Superfície de Nível

A evolução da função superfície de nível se dá através da solução da equação 5.7

$$\Phi_t + \boldsymbol{v}\cdot\nabla\Phi\left(\boldsymbol{x}\left(t\right),t\right) = 0$$

sujeita à condição inicial

$$\Phi\left(\boldsymbol{x}, t=0\right) = \Phi_0(x).$$

Inicialmente iremos considerar o campo de velocidade $\boldsymbol{V}(\boldsymbol{x},t) = \boldsymbol{x'(t)}$ definido sobre todo o domínio computacional. No entanto, em muitos casos, como por exemplo

num fluido que não chega a encher o recipiente , isso não acontece. Nesse caso, como veremos mais adiante, será necessário extender o campo de velocidade por uma determinada faixa contendo a fronteira.

A discretização da função $\Phi$ é feita sobre uma grade regular e igualmente espaçada, tomando-se seus valores sobre o centro de cada célula.

Uma vez que $\Phi$ e $\boldsymbol{V}$ estão definidas sobre toda a grade do domínio computacional (ou pelo menos numa faixa em torno da fronteira), podemos aplicar métodos numéricos para evoluir $\Phi$ sobre a grade no decorrer do tempo.

Seja $t^n$ o tempo atual e $\Phi^n = \Phi(t^n)$ o valor de $\Phi$ em cada ponto da grade no tempo $n$. Então evoluir $\Phi$ sobre a grade no decorrer do tempo, significa calcular o valor de $\Phi$ em cada ponto da grade após um incremento no tempo $\Delta t$, ou seja, calcular $\Phi^{n+1} = \Phi(t^{n+1})$ onde $t^{n+1} = t^n + \Delta t$.

Uma maneira relativamente simples de se aproximar a derivada temporal da equação 5.7, é através do método *forward Euler*. Ele se torna então

$$\frac{\Phi^{n+1} - \Phi^n}{\Delta t} + \boldsymbol{v^n} \cdot \nabla \Phi^n = 0 \tag{5.9}$$

Nas subseções seguintes apresentamos alguns métodos para aproximar as derivadas espaciais, no processo de evolução da $\Phi$.

### 5.3.1 Esquema *Upwind* de Primeira Ordem

Reescrevendo o produto escalar da equação 5.9 numa forma extendida temos, já num contexto 3D, que

$$\frac{\Phi^{n+1} - \Phi^n}{\Delta t} + u^n \Phi_x^n + v^n \Phi_y^n + w^n \Phi_z^n = 0. \tag{5.10}$$

No esquema *upwind* de primeira ordem as derivadas espaciais $\Phi_x^n$, $\Phi_y^n$ e $\Phi_z^n$ podem ser tratadas independentemente uma das outras, então por simplicidade, consideremos o caso unidimensional da equação 5.10

$$\frac{\Phi^{n+1} - \Phi^n}{\Delta t} + u^n \Phi_x^n = 0. \tag{5.11}$$

O sinal de $u^n$ nos diz se $\Phi$ está se movendo da esquerda para direita ($u^n < 0$) ou da direita para esquerda ($u^n > 0$). O esquema *upwind* de primeira ordem faz uso dessa informação para definir qual esquema de diferença utilizar. Conforme o

sinal de $(u^n)_i$ os esquemas de diferença recuada e avançada são empregados para aproximar as derivadas espaciais.

$$(\Phi_x^n)_i = \begin{cases} \frac{(\Phi^n)_i - (\Phi^n)_{i-1}}{\Delta x} & \text{se} \quad (u^n)_i > 0 \\ \frac{(\Phi^n)_{i+1} - (\Phi^n)_i}{\Delta x} & \text{se} \quad (u^n)_i < 0 \end{cases} . \tag{5.12}$$

A combinação do método *forward Euler* e do esquema *upwind* de primeira ordem forma um esquema de aproximação para equação 5.7 considerado *consistente*, uma vez que o erro da aproximação tende a zero quando $\Delta t \to 0$ e $\Delta x \to 0$, e *estável* desde que satisfaça a condição Courant-Friedreichs-Lewy (condição CFL) $\Delta t < \frac{\Delta x}{\max\{\|u\|\}}$ [69].

Em computação gráfica, muitas vezes estamos menos concentrados em precisão do que em eficiência computacional, desde que tenhamos resultados visualmente coerentes. Por esta razão, em nossa simulação utilizamos uma grade relativamente "grossa" e damos preferência por métodos que permitam passos no tempo relativamente grandes, como é o caso do método semi-lagrangeano apresentado a seguir.

### Advecção Semi-Lagrangiana

Neste método semi-lagrangeano para cada ponto da grade, $(i, j, k)$ com velocidade $\boldsymbol{v}$, fazemos a "advecção contrária", isto é, calculamos a posição $(r, s, t) = (i, j, k) - \Delta t \boldsymbol{v}$. Em seguida interpolamos na posição $(r, s, t)$ o valor da propriedade sendo adveccionada e atribuimos tal valor ao ponto $(i, j, k)$, conforme equação 5.13 e figura 5.3 .

$$\begin{aligned} \Phi_{i,j,k} = \; & \alpha\beta\gamma\ \Phi_{r,s,t} && + && (1-\alpha)\beta\gamma\ \Phi_{r+1,s,t} && + \\ & \alpha(1-\beta)\gamma\ \Phi_{r,s+1,t} && + && \alpha\beta(1-\gamma)\ \Phi_{r,s,t+1} && + \\ & (1-\alpha)(\beta-1)\gamma\ \Phi_{r+1,s+1,t} && + && (1-\alpha)\beta(1-\gamma)\ \Phi_{r+1,s,t+1} && + \\ & \alpha(1-\beta)(1-\gamma)\ \Phi_{r,s+1,t+1} && + && (1-\alpha)(1-\beta)(1-\gamma)\ \Phi_{r+1,s+1,t+1} \end{aligned} \tag{5.13}$$

onde:

$$\begin{aligned} r &= i - \lceil u_{i,j,k} \tfrac{\Delta t}{\Delta x} \rceil, \\ s &= j - \lceil v_{i,j,k} \tfrac{\Delta t}{\Delta y} \rceil, \\ t &= k - \lceil w_{i,j,k} \tfrac{\Delta t}{\Delta z} \rceil, \end{aligned}$$

e

$$\alpha = u_{i,j,k}\frac{\Delta t}{\Delta x} - \lfloor u_{i,j,k}\frac{\Delta t}{\Delta x}\rfloor,$$
$$\beta = v_{i,j,k}\frac{\Delta t}{\Delta y} - \lfloor v_{i,j,k}\frac{\Delta t}{\Delta y}\rfloor,$$
$$\gamma = w_{i,j,k}\frac{\Delta t}{\Delta z} - \lfloor w_{i,j,k}\frac{\Delta t}{\Delta z}\rfloor.$$

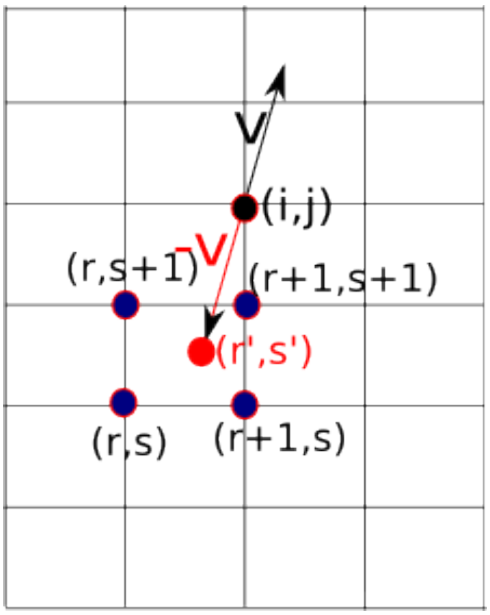



Figura 5.3: Advecção pelo método Semi-Lagrangeano em 2D

Apesar de sua praticidade o esquema semi-lagrangeano com interpolação trilinear é conhecido por introduzir difusão numérica o que para ser evitado requer mecanismos de interpolação de grau mais alto. Outra questão diz respeito ao caso em que a regressão $\boldsymbol{q}$ de um vértice $p = (i, j, k)$ cai fora do recipiente. Nesse caso, uma alternativa para obter o valor de $\Phi$ advectado para $\boldsymbol{q}$ consiste em:

1. Encontrar $\boldsymbol{w}$ o ponto do segmento $[\boldsymbol{p}, \boldsymbol{p} - \Delta t\boldsymbol{v}(\boldsymbol{p})]$ mais distante de $\boldsymbol{p}$ que ainda está dentro do recipiente.

2. Determinar $\Phi(\boldsymbol{w})$ por interpolação bilinear empregando os valores de $\Phi$ nos vértices da face a que $\boldsymbol{w}$ pertence.

3. Determinar o novo valor de $\Phi(\boldsymbol{p})$ computando no ponto $\boldsymbol{p} - \boldsymbol{v}(\boldsymbol{p})\Delta t$ a função linear definida na reta suporte cujo gráfico passa por $(\boldsymbol{p}, \Phi(\boldsymbol{p}))$ e $(\boldsymbol{w}, \Phi(\boldsymbol{w}))$.

Explicitamente:

$$\Phi(\boldsymbol{p}) = \Phi(\boldsymbol{p}) + \frac{\Phi(\boldsymbol{w}) - \Phi(\boldsymbol{p})}{\|\boldsymbol{w} - \boldsymbol{p}\|}\Delta t\boldsymbol{v}(\boldsymbol{p})$$

Esuqemas semi-lagrangeanos clássicos para promover a evolução da massa fluida podem também determinar a perda de volume. Essa perda é caracterizada pelo fato de o volume de massa fluida num dado momento da simulação ser menor que a diferença entre o volume introduzido no sistema e o retirado dele desde o início da simulação. Para evitar essa impropriedade física, outras alternativas devem ser aplicadas.

## 5.4 Método Fast Marching

Após evoluirmos a função curva de nível, a mesma pode deixar de ser uma função distância com sinal, então precisamos reinicia-la de forma a garantir que a mesma permanecerá uma função distância. Tal reinicialização pode ser feita através do método Fast Marching, o qual consiste em uma técnica para calcular o tempo de chegada de uma fronteira nos pontos de uma grade, a partir da velocidade definida sobre tal fronteira.

O método Fast Marching se aplica em casos onde a fronteira se expande (ou se contrai) de forma monotônica na direção dos pontos onde pretendemos calcular o tempo de chegada da fronteira.

O calculo do tempo de chegada $T(\boldsymbol{x})$ de uma fronteira $S$ nos pontos $\boldsymbol{x} \in R^3$ dos quais a fronteira se aproxima com velocidade monotônica $F$ em sua direção normal, pode ser formulado da seguinte maneira (conforme esboçado em [12]):

$$|\nabla T| = \frac{1}{F}. \tag{5.14}$$

Em nosso caso, iremos considerar que a fronteira se movimenta apenas em sua direção normal com velocidade $F = 1$, ver figura 5.4. Assim calcular o tempo de chegada $T(\boldsymbol{x})$ de uma fronteira nos pontos $\boldsymbol{x}$ de uma grade, a partir da velocidade definida sobre tal fronteira significa calcular a distância $\Phi$ dos pontos $\boldsymbol{x}$ da grade a fronteira.

Em síntese, o método Fast Marching [65] faz uso do método de Dijkstra [23] para busca de caminhos ótimos para definir a ordem de atualização do tempo de chegada $T$ nos vérices da grade. Operadores de diferenças upwind são usados para aproximar o gradiente de $T$. Em nosso caso, conforme visto acima, a função $T$

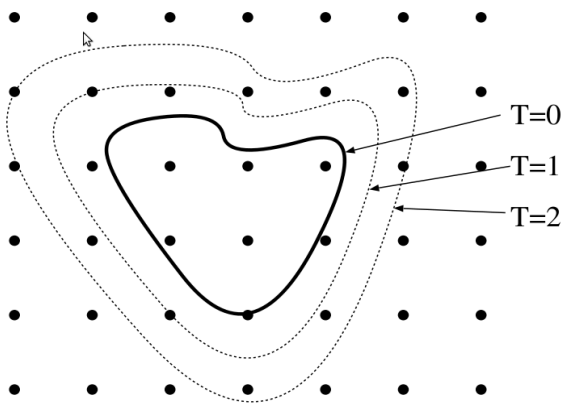



Figura 5.4: Curva se propagando com velocidade $F = 1$ em sua direção normal.

é substituida por $\Phi$. Abaixo indicamos, passo a passo, os procedimento de uma possível implementação do método Fast Marching:

1. Inserimos em uma lista, denominada $ListBorderAir$ todos os vértices da grade que são vizinhos externos a superfície ou pertencentes a superfície (vértices com $\Phi \geq 0$ e que possuem pelo menos um vizinho com $\Phi < 0$).

2. Ajustamos o valor de $\Phi$ para cada vértice $\boldsymbol{p}$ inserido em $ListBorderAir$. Esse ajuste visa fazer com que o valor de $\Phi$ em $p$ e em qualquer de seus vizinhos da massa fluida não difira mais que $h$ e será explecitado na seção seguinte.

3. Inserimos em uma estrutura do tipo "heap"os vértices pertencentes a $ListBorderAir$ (em ordem crescente de $\Phi$).

4. Retiramos da heap o vértice com o menor valor de $\Phi$ e fazemos o seguinte:

   (a) Atribuímos a ele o rótulo **computado**;

   (b) Atualizamos o valor de cada um dos seus vizinhos que ainda não esse rótulo. Para cada vizinho $(i, j, k)$ o valor de $\Phi$ é atualizado de acordo com a equação

$$\begin{aligned} &\max\left(\Phi_{i,j,k} - \Phi_{i-1,j,k}, 0\right)^2 + \min\left(\Phi_{i+1,j,k} - \Phi_{i,j,k}, 0\right)^2 + \\ &\max\left(\Phi_{i,j,k} - \Phi_{i,j-1,k}, 0\right)^2 + \min\left(\Phi_{i,j+1,k} - \Phi_{i,j,k}, 0\right)^2 + \\ &\max\left(\Phi_{i,j,k} - \Phi_{i,j,k-1}, 0\right)^2 + \min\left(\Phi_{i,j,k+1} - \Phi_{i,j,k}, 0\right)^2 = h^2. \end{aligned} \tag{5.15}$$

Cada uma das diferenças da equação 5.15 só é considerada se o respectivo vizinho de $\Phi_{i,j,k}$ já for um ponto rotulado como **computado**. Além disso, se existirem duas soluções para $\Phi$ na equação quadrática resultante da equação 5.15 utilizaremos a maior delas. Se ao contrário, não houver solução possível, vamos retirar a componente relativa ao vizinho com maior valor de $\Phi$. Se ainda assim não houver solução, repetimos esse procedimento. Se apenas um vizinho for considerado é claro que há solução.

(c) Atualizamos o valor da velocidade artificial que é definida nesse vértice para posteriormente se fazer a advecção de $\Phi$.

5. Inserimos na heap (preservando a ordem crescente da heap) os pontos cuja $\Phi$ foi atualizada no passo anterior, desde que o ponto em questão esteja dentro de uma faixa limite e ainda não pertença a heap.

6. Repetir os passos 4 e 5 até que a heap esteja vazia.

O significado geométrico da equação 5.15 é o seguinte:

Suponha que consideramos três vizinhos de $(i, j, k)$ cada um segundo uma direção coordenada. Sejam $\Phi_i^*, \Phi_j^*$ e $\Phi_k^*$ os valores de $\Phi$ nesses vizinhos indexados conforme a direção do vetor que os liga a $(i, j, k)$. Para cada um desses vizinhos $(\boldsymbol{v}_l, l = i, j, k)$ considere a esfera $S_l$ centrada em $\boldsymbol{v}_l$ e de raio $\Phi_l^*$. Se $\Phi_l^*$ é a distância de $\boldsymbol{v}_l$ a borda então a borda deve ser tangente a $S_l$. Suponha, então que usamos como aproximação local da borda, um único plano $P$, que nesse caso deve ser tangente às três esferas $S_i, S_j$ e $S_k$ e fazer com que $v_i, v_j$ e $v_k$ estejam no mesmo subespaço determinado por $P$ e mais próximo dele do que $(i, j, k)$. Essas especificações definem precisamente $P$ se os valores de $\Phi_i^*, \Phi_j^*$ e $\Phi_k^*$ propiciarem que ele exista.

O valor que se pretende obter empregando a expressão acima é a distância de $(i, j, k)$ a $P$. Para ver que definindo $\Phi(i, j, k)$ dessa forma a expressão é satisfeita, seja $\boldsymbol{n}$ o vetor normal a $P$. Como a distância de $\boldsymbol{v}_l$ a $P$ é $\Phi_l^*$, teremos então que

$$\Phi_{i,j,k} - \Phi_l^* = \|(i, j, k) - v_l\| \langle \boldsymbol{n}, e_l \rangle = h \langle \boldsymbol{n}, e_l \rangle$$

e portanto

$$(\Phi_{i,j,k} - \Phi_i^*)^2 + \left(\Phi_{i,j,k} - \Phi_j^*\right)^2 + (\Phi_{i,j,k} - \Phi_k^*)^2 = h^2 \left( \langle \boldsymbol{n}, e_i \rangle^2 + \langle \boldsymbol{n}, e_j \rangle^2 + \langle \boldsymbol{n}, e_k \rangle^2 \right) = h^2.$$

A figura 5.5 ilustra o significado da equação 5.15 num contexto 2D.

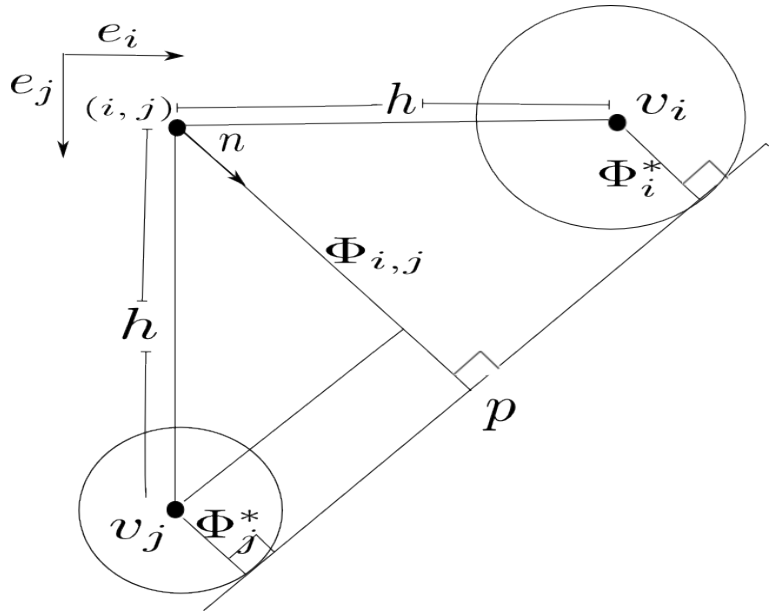



Figura 5.5: Ilustração da equação 5.15 no caso 2D mostrando que segundo essa equação $\Phi_{i,j}$ se refere a distância de $(i,j)$ a uma reta tangente externa aos círculos de centros $v_i$ e $v_j$ e raios $\Phi_i^*$ e $\Phi_j^*$, respectivamente. Entre as duas tangentes externas se escolhe a que deixa $(i,j)$, $v_i$ e $v_j$ do mesmo lado.

Com respeito a atualização de $\Phi$ experimentamos neste trabalho uma variação com bom resultado, cuja razão de ser explicamos abaixo.

Um problema notoriamente conhecido, relativo a implementação do mecanismo de evolução das superfícies de nível, acontece quando o processo de advecção leva um ponto no "ar" a outro que está do outro lado do eixo medial (ou esqueleto externo) do volume ocupado pelo fluido. Ocorre que a advecção é computada por um processo inverso, a saber, ao invés de se movimentar a borda do fluido e medir quanto ela se aproxima de um ponto $\boldsymbol{p}$, se movimenta $\boldsymbol{p}$ em sentido contrário até um ponto $\boldsymbol{p'}$.

Usa-se para dimensionar esse deslocamento, uma estimativa de quanto deve ser a velocidade no ponto $\boldsymbol{q}$ mais próximo de $\boldsymbol{p}$ na borda. Avalia-se, então a distância de $\boldsymbol{p'}$ à borda, ou seja, ao ponto $\boldsymbol{q'}$ mais próximo dele na borda. Entretanto, quando o segmento $[\boldsymbol{p},\boldsymbol{p'}]$ corta o eixo medial, $\boldsymbol{q}$ e $\boldsymbol{q'}$, podem ser bastante diferentes e se $\boldsymbol{q}$ se afasta de $\boldsymbol{p}$ com uma velocidade grande enquanto $\boldsymbol{q'}$ se aproxima dele mais devagar, pode acontecer que se venha considerar $\boldsymbol{p}$ como tendo sido atingido pela

massa fluida, erroneamente. Ver figura 5.6.

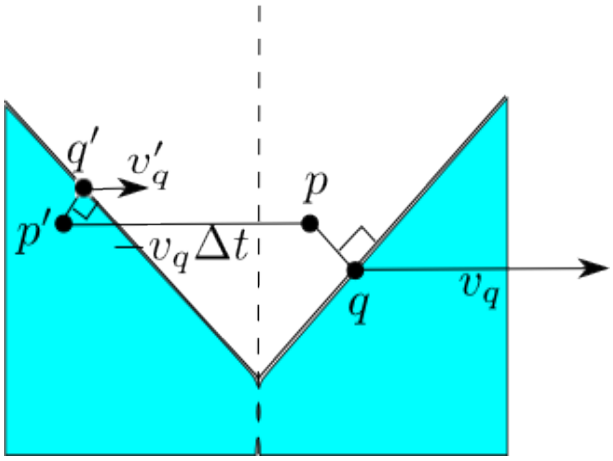



Figura 5.6: Ponto $p$ seria considerado erradamente como coberto pela massa na próxima iteração. Isso ocorre porque $v_q$ é bem maior que $v_{q'}$.

Esse problema pode ser evitado substituindo-se o ponto $\boldsymbol{q}$, se necessário, pelo ponto da borda mais próximo de $\boldsymbol{p}$ entre os que estão se aproximando dele. Para determinar, é claro que de forma aproximada, a distância de $\boldsymbol{p}$ a esse novo ponto podemos em proceder da seguinte maneira:

1. Fazemos uma avaliação da velocidade $\boldsymbol{v_p}$ do ponto da borda mais próximo de um vértice $\boldsymbol{p}$, em função dos valores dessa velocidade já estimados em um conjunto $S$ de até 3 vizinhos de $\boldsymbol{p}$, todos atingíveis a partir dele segundo arestas de direções diferentes.

2. Seja ainda $\boldsymbol{CG}(S)$ o centro de massa dos vértices de $S$. Se

$$\langle \boldsymbol{v_p} \ , \ \boldsymbol{p} - \boldsymbol{CG}(S) \rangle \geq 0 \tag{5.16}$$

   retorne a distância $\Phi(\boldsymbol{p})$ e $\boldsymbol{v_p}$ .

3. Em caso contrário, troque cada vértice $\boldsymbol{w}$ de $S$ por seu simétrico em relação a $\boldsymbol{p}$, $\boldsymbol{w'}$, mas um apenas de cada vez. Se $\Phi(\boldsymbol{w'}) < \Phi(\boldsymbol{p})$, recompute $\boldsymbol{v_p}$ e $\Phi(\boldsymbol{p})$, substituindo no cômputo deles a contribuição de $\boldsymbol{w}$ pela de $\boldsymbol{w'}$ e repita o passo 2.

4. Se com apenas uma troca dos vértices vizinhos não for possível satisfazer a equação 5.16 tente duas. Se ainda assim não for possível experimente três.

5. Se isso ainda não for suficiente, para cada vértice $\boldsymbol{w}$ de $S$, repita os passos 2,3,4 com $S - \{\boldsymbol{w}\}$ no lugar de $S$. Caso para mais de um $\boldsymbol{w}$, 5.16 for satisfeita escolha a que gerou o menor valor de $\Phi(\boldsymbol{p})$.

6. Se retirando um só elemento de $S$ não conseguirmos atender a equação 5.16, então:

   Enquanto $S$ tiver mais de um elemento retire mais um elemento e repita 2-4.

   Se ao final do processo ainda não atendermos a equação 5.16, podemos concluir que a borda se afasta do ponto $\boldsymbol{p}$ e por isso ele não vai mudar de meio (ar ou fluido) nesta iteração.

## 5.5 Ajuste da Função Distância em Pontos Vizinhos a Superfície

Depois da advecção pela velocidade, a função $\Phi$ usada para determinar as superfícies de nível, usualmente deixa de ser uma função distância à borda do fluido. É preciso que ela recupere essa propriedade, o que, entretanto, precisa ser feito sem modificar a topologia da borda do fluido e de forma que não seja computacionalmente pesada.

Para conservar a topologia, obriga-se que toda aresta da malha, caso $\Phi$ seja estimada nos vértices da malha, que seja cortada pela superfície do fluido antes da retificação, mantenha essa propriedade depois. O mesmo se aplica ao segmentos ligando centros de células contíguas cortados pela borda, quando $\Phi$ é estimada nos centros das células.

Se, no entanto, os valores de $\Phi$ nos extremos da aresta diferirem mais do que o comprimento dela, será necessário ajustá-los, caso contrário $\Phi$ não poderá ser uma distância.

Sejam $\boldsymbol{p_0}$ e $\boldsymbol{p_1}$ esses extremos. Uma forma usual de promover esse ajuste é dada por

$$\Phi(\boldsymbol{p_i}) = min(\Phi(\boldsymbol{p_i}), max(0, \Phi(\boldsymbol{p_{1-i}}) + h) \text{ se } \Phi(\boldsymbol{p_i}) > 0$$
$$\Phi(\boldsymbol{p_i}) = max(\Phi(\boldsymbol{p_i}), min(0, \Phi(\boldsymbol{p_{1-i}}) - h) \text{ se } \Phi(\boldsymbol{p_i}) \leq 0.$$

Essa formulação é viesada no sentido de que o último dos extremos tratado não muda de valor, toda a variação acontece no outro. Além disso, a comparação com 0, que teve de ser introduzida para evitar que $F$ mudasse de sinal num dos extremos, pode produzir alinhamentos artificiais entre a borda e componentes da estrutura da malha.

Ao invés de proceder dessa forma, preferimos preservar o ponto onde se anula a interpolação linear ao longo da aresta, dos valores da $\Phi$ computados em seus extremos. Depois substituímos o valor de $\Phi$ em cada extremo pela distância dele a esse ponto de anulação. Esse procedimento tem a propriedade desejável de não fazer $|\Phi(p_i)| < h$.

Observe que como tal procedimento não aumenta o $|\Phi(p_i)|$ não ha risco dessa propriedade ser perdida se o valor de $\Phi(p_i)$ for modificado quando se processar outra aresta adjacente a $p_i$ e a um vértice do outro lado da borda. O passo 2 da implementação do Fast Marching pode então ser implementado da seguinte maneira:

```
1 for cada vértice q vizinho de p e pertencente a massa fluida do
2 |   if (Φ_p − Φ_q) > h then
3 |   |   Φ_p = h/(Φ_p − Φ_q) Φ_p ;
4 |   |   Φ_q = h/(Φ_p − Φ_fq) Φ_fq ;
5 |   end
6 end
```

**Algoritmo 1**: Atualização da $\Phi$: Passo 1

onde $h$ é o espaçamento da malha.

A atualização da velocidade artificial $\boldsymbol{v}_a$ em um vértice $\boldsymbol{p}$ que pertence a porção ar é feita da forma indicada no algoritmo abaixo. Observe que quando fazemos essa atualização todos os vizinhos de $\boldsymbol{p}$ ou já foram computados ou estão no heap. Portanto $\Phi$ já foi definida para eles.

```
1  v_a(p) = (0, 0, 0);
2  for cada direção principal do
3  |  Sejam w e w^- os vizinhos de p segundo essa direção ;
4  |  soma_dos_pesos = 0.0;
5  |  if Φ(w) < Φ(w^-) then
6  |  |  w* = w ;
7  |  end
8  |  else
9  |  |  w* = w^- ;
10 |  end
11 |  if Φ(w*) < Φ(v) then
12 |  |  peso = Φ(p) − Φ(w*) ;
13 |  |  v_a(p) = v_a(p) + peso v_a(w*) ;
14 |  |  soma_dos_pesos = soma_dos_pesos + peso;
15 |  end
16 end
17 v_a(p) = v_a(p)/ soma_dos_pesos ;
```

**Algoritmo 2**: Computo da velocidade artificial

Observe que $\boldsymbol{p}$ tem pelo menos um vizinho tal que $\Phi(\boldsymbol{w}^*) < \Phi(\boldsymbol{v})$. Isso é decorrente da maneira que computamos $\Phi$. Portanto, *soma_dos_pesos* jamais será nulo ao final.

## 5.6 Marching Cubes

Marching Cubes é a denominação de um algoritmo criado por Lorensen and Cline, [53], para determinar uma malha poligonal representando uma isosuperfície de um campo escalar 3D amostrado nos vértices de uma grade regular. Sua versão 2D é conhecida por Marching Squares.

Para cada célula cúbica da grade o algoritmo considera o valor do campo em seus 8 vértices para montar uma representação da isosuperfície dentro da célula constituída por faces triangulares. As representações poligonais obtidas para cada

célula são, então, agregadas para formar a da isosuperfície.

Atribuindo 0 aos vértices cujo nível esta abaixo do da isosuperfície e 1 aos que estão num nível acima criamos uma representação binária com 256 possíveis valores para o conjunto de vértices de uma célula. Essa representação binária indexa uma lista onde cada elemento é por sua vez uma lista de triângulos só que a posição dos vértices desses triângulos não é estabelecida inteiramente. Apenas se indica a aresta da célula onde o vértice deve estar.

A localização precisa do vértice dentro da aresta é obtida por interpolação linear dos valores do campo nos nós adjacentes a ela. Normais às faces triangulares são estimadas tomando-se uma média das normais nos seus vértices. essas, por sua vez, são obtidas a partir dos gradientes do campo nos nós da aresta, tomando-se o vetor unitário de uma interpolação linear desses gradientes. Obter essas normais é necessário para que se possa aplicar um modelo de iluminação às faces da representação da isosuperfície.

A escolha das representações determinadas para duas células vizinhas devem ser coincidentes na face que elas tem em comum, o que pode não ser imediato quando a face contem 4 vértices da isosuperfície. Além disso configurações de valores do campo nos vértices da célula obtidas a partir de outras configurações por uma rotação devem dar origem a representações que também se relacionam por essa rotação. Duas configurações com valores de campo simétricas em cada nó devem gerar a mesma representação só sendo trocado o sinal do vetor normal. Considerando como equivalentes, configurações que se relacionam por uma rotação ou por uma operação de troca de sinal de todos os vértices, temos 15 casos.

Apesar da versão original representada na figura 5.7 abaixo, não cumprir inteiramente essa proposta versões mais recentes já corrigiram esse defeito (Ver [19]). Observamos que cada componente conexa da representação tem no máximo 6 vértices ou 4 triângulos e que quando há mais de uma componente cada uma delas é um triângulo ou um quadrilátero.

Neste trabalho vamos empregar a representação obtida pelo Marching Cubes dentro de cada célula de borda para computar os fluxos que passam de uma face a outra dentro da célula.

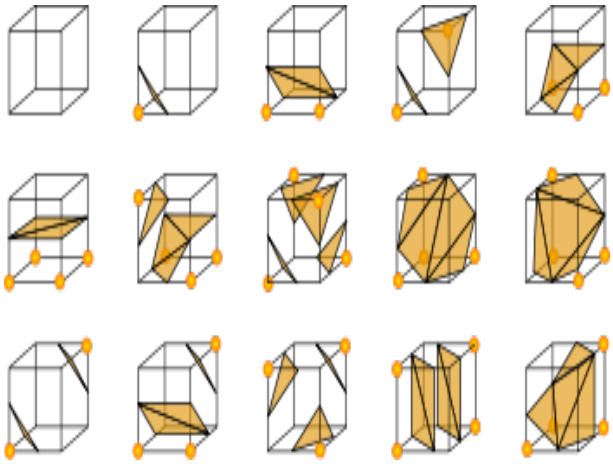


Figura 5.7: Marching Cubes: 15 casos da versão original.

# Capítulo 6

# Tratamento das Células de Borda

Como não existe uma metodologia para dar um tratamento diferenciado às células de borda que tenha se estabelecido, estamos propondo uma alternativa que é descrita nas seções seguintes.

## 6.1 Células Ar-Borda

Células de borda oferecem uma alternativa interessante para fazer com a que a simulação se atenha a realidade numa série de situações. Uma delas diz respeito ao espalhamento do fluido não viscoso pelo fundo do recipiente, o qual se dá ao longo de uma lâmina extremamente fina. A espessura dessa lâmina depende de especificidades do fluido e não da resolução da malha em que se realiza a simulação, como acontece num modelo Euleriano clássico. A tentativa de tornar atingível essa espessura, fazendo a resolução mais fina que ela, pode dependendo do modelo usado - por exemplo, assumindo que o fluido é invíscido e que os limites do container são SLIP - gerar velocidades ao longo do fundo maiores que as reais, determinando então, velocidades de subida nas paredes laterais também maiores do que o esperado. Em consequência o fluido sobe por elas mais do que deveria.

Conforme já comentado anteriormente, as células de borda oferecem uma alternativa interessante também em outras situações em que o volume fluido apresenta estruturas bastante finas, como no afinamento de um fluxo tubular que desce sob a ação da gravidade ou no encontro de frentes que se espalham por duas paredes na aresta de junção delas. No primeiro caso elas podem propiciar que a evolução

do fluido não seja detida com um investimento que consideramos menor que empregado por outras alternativas - usar partículas, por exemplo. No outro caso, filetes que eventualmente sobem pelas linhas de junção em alguns casos deixam de ser suprimidos.

No processo de construção do simulador nossa primeira motivação para dar um tratamento específico às células parcialmente cheias se deu no contexto, menos abordado na literatura, de se obter valores para as velocidades nas faces das células que foram recém introduzidas na massa fluida. Essa questão, definitivamente, não admite uma solução trivial que sirva para um conjunto abrangente de situações. Manter nulas as velocidades de faces que não sejam adjacentes a células fluido, não reproduz a situação real quando a célula se enche completamente e faz com que a resolução da equação de Poisson gere situações fisicamente inaceitáveis. Suponha que num problema 2D sem gravidade temos uma única célula fluida $C$ com velocidade $v > 0$ nas arestas verticais e 0 nas horizontais. Se ao se acrescentar a célula a direita dela ($C'$) à massa fluida, mantemos zeradas as velocidades nas arestas não comuns com $C$, a solução da equção de Poisson vai repartir o excesso de fluido que chega a $C'$, igualmente pelas três arestas não adjacentes a $C$ que assumirão, nesse caso, velocidades $v/3$. Assim o fluido se espalharia em todas as direções o que não é fisicamente correto nas condições do problema.

A alternativa de replicar valores de velocidades em faces de células vizinhas que já são fluidas também não é solução de uso geral. Há uma série de situações em que células vizinhas tem velocidades em faces paralelas que não são correlacionadas. Uma dessas situações ocorre quando o fluido chega ao fundo do container e o fluxo muda radicalmente de direção. Suponha, novamente num contexto 2D, que um fluxo tubular com diâmetro de uma célula chega ao fundo do container numa célula $C$ com velocidade vertical $V$. Assumindo-se que essa velocidade será repartida entre as arestas verticais de $C$, a cada uma delas será associada uma velocidade $v/2$. Se numa iteração próxima aquela em que o fundo é atingido, a célula $C'$ a direita de $C$, se tornar fluida, então se copiarmos para a sua aresta horizontal superior, o valor $v$ então teremos muito possivelmente uma situação irreal, dado que muito pouco fluxo deve estar atravessando essa aresta. Aplicando esse procedimento a todas as células tornadas fluidas enquanto o fluxo escorre pelo fundo pode fazer com que,

via Poisson, a velocidade de propagação assuma valores extremamente altos em completa dissintonia com a realidade física.

Esse último exemplo ressalta a necessidade de se ter um controle sobre a quantidade de fluido que atravessa uma célula parcialmente cheia, ou seja uma célula de borda, simplesmente para prover inicializações consistentes, às células que serão incorporadas a massa fluida. Essa é a finalidade da metodologia descrita neste capítulo que utiliza informações que podem vir da aproximação da borda na célula produzida pelo Marching-cubes ou por aproximações ainda mais simplistas do volume fluido dentro da célula cuja a inspiração são as empregadas pelos métodos da família Volum-of-Fluid descritos no capítulo 2. A diferença entre a metodologia introduzida neste capítulo e a mais abrangente, apresentada no próximo, é que aqui nos restringimos a utilizar informações relativas a borda para suprir valores iniciais adequados as células feitas fluido enquanto que no capítulo 7, essas informações vão influir no cômputo da advecção nas células de borda e na própria montagem do sistema de Poisson.

Deve-se esclarecer que o modelo empregado aqui visa especialmente a manutenção da incompressibilidade se tornando mais simples se assumirmos que no entorno do instante em que estamos fazendo a avaliação da velocidade numa célula de borda, o regime de fluxos que entra e sai dela se mantém pelo tempo em que ela é percorrida pelo fluido. Caso isso não aconteça e tenhamos que quando uma porção de fluido elementar que está saindo agora da célula, entrou nela, o regime era bastante diferente, esse modelo mais simples não se aplica. Assim ele é favorecido pelo aumento da resolução, mas o aplicamos com bons resultados em resolução baixa desde que não se dissocie o processo de definição de velocidades aplicado na borda do empregado para as células inteiramente "FLUIDO". O método em sua versão básica é sequencial. Um nível de paralelização, suficiente na grande maioria dos casos, não é difícil de se obter.

Nas seção seguinte a metodologia introduzida acima será apresentada em detalhes.

## 6.2 Tratamento das Células Ar-Borda

Considerando a figura 6.1, observamos que, por exemplo, a célula (2,6), é considerada uma célula do tipo Ar, pois seu centro está fora do interior do fluido, isto é, possui $Phi > 0$. No entanto, a mesma possui fluido em seu interior o qual contribui para velocidade do fluido que passa na célula (2,5).

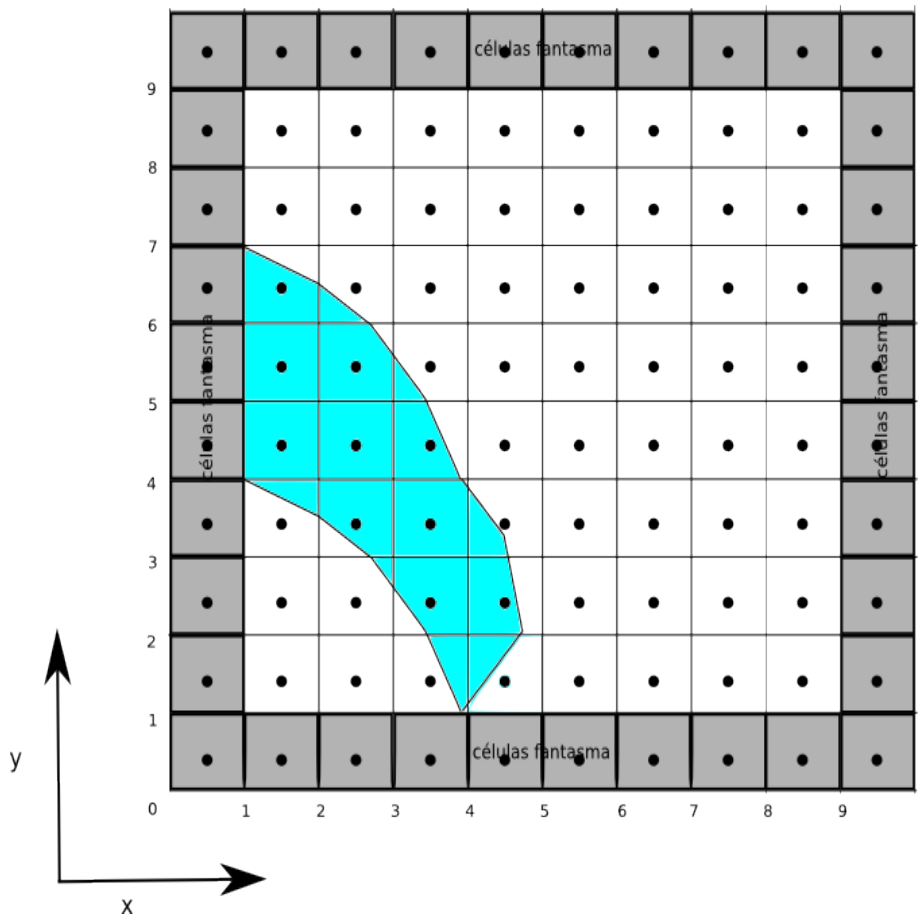



Figura 6.1: Células Ar-Borda

Assim, as células do tipo Ar que possuem alguma célula vizinha, por face, aresta ou vértice, do tipo Fluido, podem conter fluido e portanto influenciar no cômputo da velocidade em outras células. Essas células serão aqui denominadas células do tipo **Ar-Borda**. Observe que com essa definição excluímos estruturas finas onde a porção fluida corta sequências de células não adjacente a células fluidas.

### 6.2.1 Ordem de Precedência entre as Células Ar-Borda para Cálculo da Velocidade

Para garantirmos que a velocidade numa célula Ar-Borda, $A$, seja influenciada apenas por células vizinhas por face, cujo fluxo se dá de $V$ para $A$, definimos uma ordem de precedência, para a atualização da velocidade, entre as células Ar-borda:

1. Inserimos em uma lista específica, aqui denominada **ListBorderAir**, as células do tipo Ar-Borda.

2. Removemos da lista ListBorderAir as células onde estão definidas as condições de contorno da simulação, uma vez que essas células precisam de outro tipo de tratamento esepecial.

3. Para cada uma das células $A$ em ListBorderAir:

   (a) Verificamos para cada uma das vizinhas por face, se a mesma é uma célula do tipo Ar-Borda e se o sentido do fluxo é de $V$ para célula $A$. Isso define uma pré-ordem " $\prec$ " sobre o conjunto das células Ar-Borda. $V \prec A$, indica que $V$ é uma antecessora de $A$, ou seja, que a velocidade nessa célula vizinha, $V$, deve ser definida primeiro.

   (b) se não existirem antecessoras a célula $A$, então definimos a velocidade em suas faces, utilizando apenas as células que já possuem velocidades definidas (conforme subseção 6.2.2 ) e a retiramos da lista ListBorderAir.

4. Enquanto existir alguma célula na lista ListBorderAir:

   (a) percorremos a lista ListBorderAir verificando se as vizinhas Ar-Borda que precediam uma determinada célula foram processadas, e em caso afirmativo eliminamos as respectivas precedências.

   (b) Para cada uma das células $A$ em ListBorderAir: Repetimos o passo 3b.

   (c) Se nenhuma célula for processada em 4b, então processamos a que tiver o menor número de precedentes e a retiramos da lista ListBorderAir.

#### Considerações

Uma série de considerações precisam ser feitas em relação a esse processo:

1. Consideração relativa a pré-ordenação:

   Essa consideração diz respeito a como implementar a pré-ordenação de maneira que o tempo gasto nessa etapa varie de forma linear com o número de células Ar-Borda. Assim, buscando atualizar o campo velocidade nas células Ar-Borda em uma ordem que respeite $\prec$, e em tempo linear, o procedimento a seguir é sugerido:

   (a) Dadas duas células Ar-Borda - $A$ e $B$ - notamos que $A \prec B$ se existe um caminho de células Ar-Borda $\{A = C_0, C_1, \ldots, C_n = B\}$ tal que, na solução atual, há um fluxo de $C_i$ para $C_{i+1}$. Se a face $F_j$ separa essas duas células, isto significa que a velocidade nessa face, $\boldsymbol{v}_{F_j}$, aponta para $C_{i+1}$.

   (b) Particionamos o conjunto de células Ar-Borda em listas $L_i, i = 0, \ldots, 5$ de acordo com o número de antecessores, contíguos a esta, que uma célula tem.

   (c) Criamos cópias dessas listas ($L'_i$).

   (d) Enquanto $L'_0$ não estiver vazia, processamos seus elementos em qualquer ordem e para cada um deles, transferimos todos os seus sucessores da lista $L'_i$ para lista $L'_{i-1}$.

   (e) Removemos os elementos processados de $L'_0$ e se $L'_0$ tornar-se vazia e existirem ainda células não processadas, isso significa que algumas delas formam um ciclo, o que provavelmente somente acontece em um vórtice de borda. Nesse caso não há maneira de cumprir a precedência, mas para minimizar a violação escolhemos uma célula na primeira lista não vazia e a processamos como descrito acima.

   Além da inclusão/remoção determinada pela mudança de rótulo da célula, uma única operação é necessária para atualizar as listas $L_i$ para a próxima iteração: Se, para uma face $F$, adjacente a duas células de borda, $V_F$ muda de sinal quando $\left(\frac{\Delta t}{\rho}\right) \nabla P$ é adicionado à solução, então re-posicionamos cada célula adjacente em função desta inversão de precedência.

2. Consideração relativa a paralelização:

Em princípio, a ideia de se processar as células Ar-Borda segundo uma dada ordenação seria um entrave ao tratamento em paralelo dessas células. Isso no entanto pode ser minorado estabelecendo-se um raio máximo para o uso dos antecessores que influem no valor da velocidade em uma célula. Essa proposta só se torna factível porque temos as velocidades das faces das céluas Ar-Borda obtidas na iteração anterior.

Empregando essas velocidades e assumindo um modelo de paralelização em que há um processador para cada célula então, numa primeira iteração se considera apenas a relação de precedência entre células vizinhas. Na segunda iteração, já com os valores atualizados na primeira iteração extendemos a região de influência de uma célula de borda às vizinhas das vizinhas e assim sucessivamente. Fixado um raio de influência em $r$ células, $r$ iterações serão suficientes.

Esse é um expediente simples para atribuir um nível de paralelismo a resolução de sistemas lineares como o de Poisson.

3. Consideração relativa a advecção do campo de velocidades:

   O procedimento empregado para atualização da velocidade de uma célula Ar-Borda pode ser considerado como uma advecção em que a distância percorrida efetivamente pelo fluido de uma iteração a outra é igual ao tamanho de uma célula. Ao adotar essa hipótese procuramos nos livrar do cômputo de cortes no volume de massa fluida dentro da célula. Com ela é possível trabalhar apenas com os cortes efetuados nas faces, que são determinados pelo próprio algoritmo que faz o traçado da borda. A figura 6.2 ajuda a esclarecer essa questão.

   Quando não se usa o própio corte numa face $F$ de entrada na célula $C$ fornecido pelo Marching-Cubes, ele pode ser estimado levando-se em conta:

   (a) As arestas que a célula tem em contato com a massa fluida.

   (b) O sentido da velocidade na face já tratada $C^0$.

   (c) A área $A$ que esse corte deve ter e que também já foi obtida quando se processou $F$.

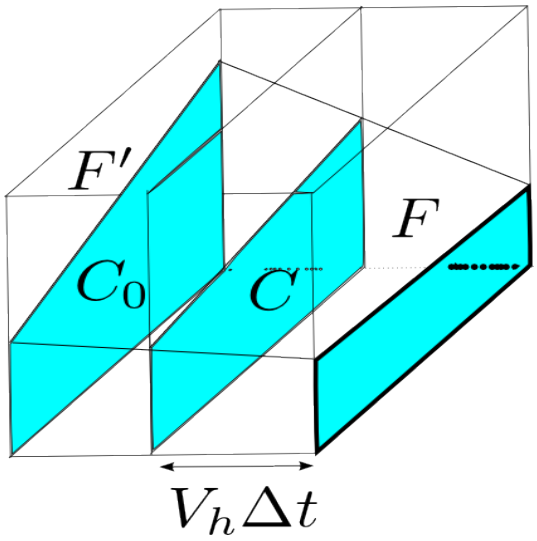



Figura 6.2: O objetivo é computar a influência do fluxo em F' na determinação da velocidade em F. Supondo $V\Delta t = h$ evitamos ter de computar o corte C determinado na massa fluida por um plano paralelo a F e distando dele $V_h\Delta t$. Ao invés de C empregamos $C_0$ que pode ser fornecido pelo Marching Cubes.

A espectativa é que empregando 3a e 3b se possa definir um vértice ou uma aresta que indicariam a situação de área zero.

Numa primeira estimativa as velocidades paralelas a $F$ em $C$ indicariam a direção do corte que seria feito normal a composição delas. O corte com a direção assim determinada e com área $A$ na porção fluida seria o adotado.

4. Consideração relativa a inicialização da velocidade em novas células fluido:

   Com este esquema que privilegia a precedência determinada pelo sentido do fluxo pretendemos obter melhores inicializações para as velocidades em células que são introduzidas na massa fluida, do que usando extrapolação ou critérios de similaridade.

### 6.2.2 Cálculo da Velocidade para uma Célula Ar-Borda

O cálculo da velocidade para uma célula Ar-Borda, $A$, é feito da segiunte forma:

Para cada uma das seis células $V \in \{C_{i-1,j,k}, C_{i+1,j,k}, C_{i,j-1,k}, C_{i,j+1,k}, C_{i,j,k-1}, C_{i,j,k+1}\}$, vizinhas por face a célula $A = C_{i,j,k}$, verificamos se a velocidade na face comum a célula $A$ aponta de $V$ para $A$, ou seja $V \prec A$ ver figura 6.3

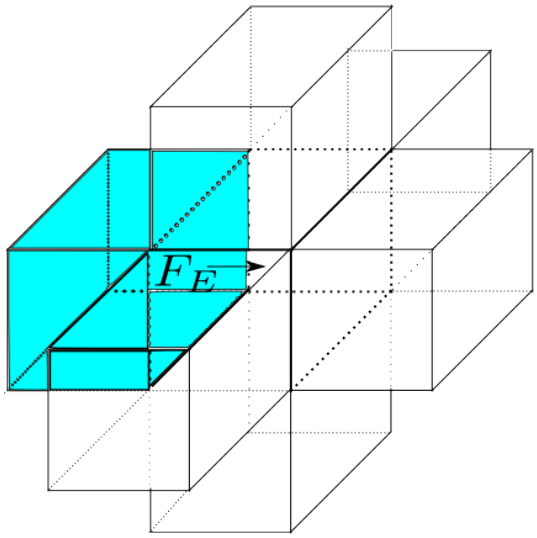



Figura 6.3: Célula $V$ (em azul) cuja velocidades na face comum a célula $A$ (célula central) aponta para o interior da célula $A$

Quando isso acontece calculamos como o fluxo na face comum a $A$ e a $V$, face $F_E$, com sentido $V \rightarrow A$, denominado fluxo de entrada, se distribui pelas demais faces de $A$, denominadas faces de saída. Isso é feito pensando que esse fluxo será mantido até que a face de saída seja atingida.

Seja $V = C_{i-1,j,k}$, suponha que a velocidade na face $F_E$, aponta no sentido de $V$, conforme figura 6.3.

Então:

1. Calculamos um vetor velocidade, temporário, $\boldsymbol{v}_{temp}$, para a face $F_E$, o qual será utilizado para determinar como o fluxo de entrada em $F_E$ se distribui entre as faces de saída de $A$. O cálculo do vetor $\boldsymbol{v}_{temp}$ é feito da seguinte forma:

   (a) Atribuímos a componente $u_{temp}$ do vetor $\boldsymbol{v}_{temp}$ o valor de $u_{i,j,k}$, a velocidade da face $F_E$.

   (b) Identificamos as faces de $V$ paralelas a direção de $u$ onde as velocidades apontam para o interior da célula $V$. Para cada par de faces com orientação $(D)$ temos três possíveis casos:

      i. Em nenhuma das duas faces com a orientação $D$ a velocidade aponta para dentro de $V$. Ver figura 6.4. Nesse caso, a componente de $\boldsymbol{v}_{temp}$ ortogonal a essas faces, $(\boldsymbol{z}_{temp})$, será nula.

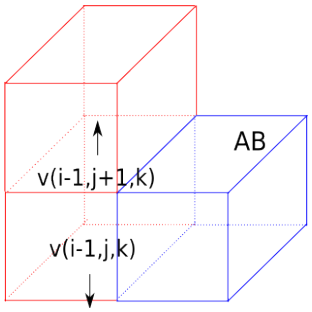



Figura 6.4: Caso 1(b)i: Em nenhuma face com a orientação $D$ a velocidade aponta para dentro de $V$.

ii. Em apenas uma das faces de $V$ com a orientação $D$ a velocidade aponta para dentro de $V$. Ver figura 6.5. Nesse caso $\boldsymbol{z}_{temp}$ assumirá o valor da velocidade dessa face.

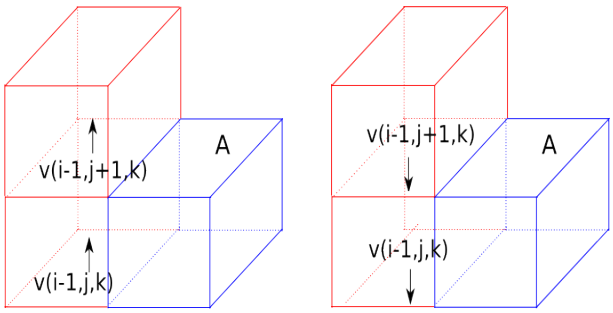



Figura 6.5: Caso 1(b)ii: Em apenas uma face de $V$ com a orientação $D$ a velocidade aponta para dentro de $V$.

iii. Em ambas as faces de $V$ com a orientação $D$, a velocidade aponta para $V$. Ver figura 6.6. Nesse caso $\boldsymbol{z}_{temp}$ será então a soma das velocidades dessas faces.

O tratamento dos três casos, 1(b)i, 1(b)ii e 1(b)iii, pode ser resumido da seguinte forma: $\boldsymbol{z}_{temp}$ será a soma das velocidades em faces ortogonais a face em questão e que apontam para dentro de $V$.

2. Calculado o vetor velocidade da face de entrada, $F_E$, o objetivo será podermos determinar como o fluxo de entrada em $F_E$ será repartido pelas faces de saída. Para isso, imaginemos inicialmente que $F_E$ está coberta de fluido.

   Dado um número real $k$ faça $k^- = max\{-k, 0\}$ e seja $sign(.)$ a função que atribui a $k$ o valor $-1, 0$ ou $1$ conforme $k$ seja negativo, nulo ou positivo.

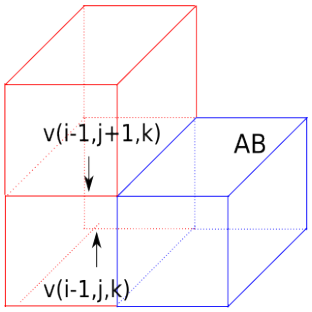



Figura 6.6: Caso 1(b)iii: Em ambas as faces com a orientação $D$ a velocidade aponta para dentro de $V$.

Lembrando que $A = c_{i,j,k}$ encontre o vértice $\boldsymbol{v}^-$ de coordenadas $\left(i + (sign\,(u_{temp}))^-\right), \left(j + (sign\,(v_{temp}))^-\right), \left(k + (sign\,(w_{temp}))^-\right)$. Esse vértice é definido pelas três faces que são de entrada quando o fluxo tem a direção de $v_{temp}$.

Considere agora um sistema de coordenadas $S_{\boldsymbol{v}^-}$ com origem em $\boldsymbol{v}^-$ e vetores unitários da forma $(\boldsymbol{v}' - \boldsymbol{v}^-)$ onde $\boldsymbol{v}'$ é um vértice adjacente a $\boldsymbol{v}^-$. Desta forma, $\boldsymbol{v}^-$ se projeta segundo a direção de $\boldsymbol{v}_{temp}$ numa única face de $A$. Seja $\boldsymbol{w}^+$ essa projeção.

No sistema $S_{\boldsymbol{v}^-}$ as coordenadas de $\boldsymbol{w}^+$ são dadas por

$$\frac{(|u_{temp}|, |v_{temp}|, |w_{temp}|)}{max\,\{|u_{temp}|, |v_{temp}|, |w_{temp}|\}}.$$

Assim $\boldsymbol{w}^+$ terá sempre uma coordenada unitária no sistema de coordenadas $S_{\boldsymbol{v}^-}$, aquela relativa a componente de $\boldsymbol{v}_{temp}$ de maior valor absoluto.

Há dois casos a considerar quanto a localização de $\boldsymbol{w}^+$:

(a) $\boldsymbol{w}^+$ pode está na face oposta a $F_E$ como mostrado na figura 6.7. Nesse caso, $\boldsymbol{w}^+$ pode ser dado, em coordenadas $S_{\boldsymbol{v}^-}$, por $\boldsymbol{w}^+ = (1, a_2, a_3)$. E as porções das faces de saída por onde o fluxo de entrada, em $F_E$, sairá, serão dadas por:

i. Na face oposta a $F_E$, um retângulo tendo como vértices opostos $\boldsymbol{w}^+$ e $\boldsymbol{v}^+$, o vérice de $A$ diametralmente oposto a $\boldsymbol{v}^-$. A área de retângulo considerando uma célula de tamanho normalizado é dada por $(1 - a_2)(1 - a_3)$. Ver figura 6.8.

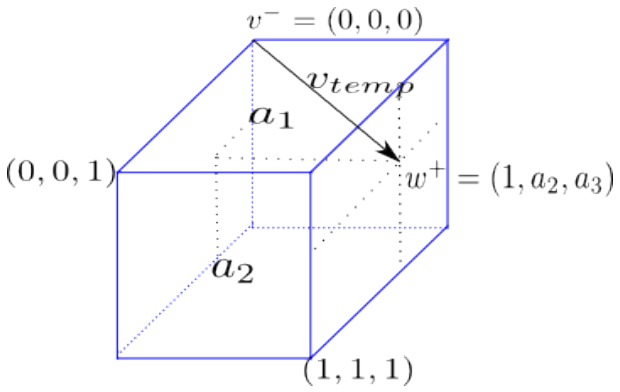



Figura 6.7: $w^+$ na face oposta

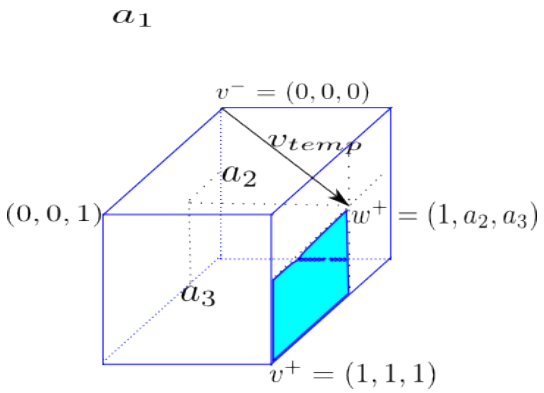



Figura 6.8: Projeção de $\boldsymbol{v}^-$ na face oposta

ii. Nas faces de saída adjacentes a $F_E$, trapézios formados pela aresta que a face de sáida tem em comum com $F_E$ e o segmento paralelo a ela que vai de $\boldsymbol{v}^+$ a projeção ortogonal de $\boldsymbol{w}^+$ sobre a face de saída. Em coordenadas $S_{\boldsymbol{v}^-}$ esses trapézio são definidos pelos vértices $\{(0,1,1),(0,0,1),(1,a_2,1),(1,1,1)\}$ e $\{(0,1,1),(0,1,0),(1,1,a_3),(1,1,1)\}$ A área de cada um deles é $(1+a_2)/2$ e $(1+a_3)/2$. Ver figuras 6.9 e 6.10.

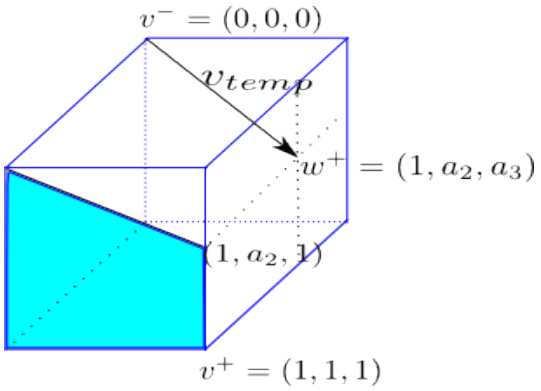



Figura 6.9: Área na face adjacente (frente)

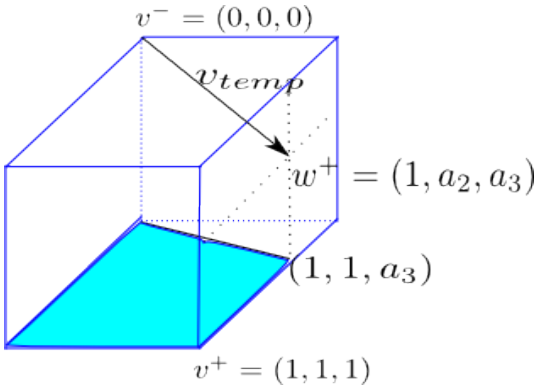



Figura 6.10: Área na face adjacente (base)

(b) $\boldsymbol{w}^+$ pode está em uma das faces adjacentes a $F_E$ como mostrado na figura 6.11. Neste caso, a face oposta a $F_E$ não será atingida pelo fluxo de entrada em $V$ por $F_E$. As porções das faces de saída adjacentes por onde o fluxo de entrada, em $F_E$, sairá, são dadas por:

i. Para a face contendo $\boldsymbol{w}^+$ temos um trapézio cuja área é dada por $a_1 * \left(1 - \frac{a_3}{2}\right)$. Ver figura 6.12.

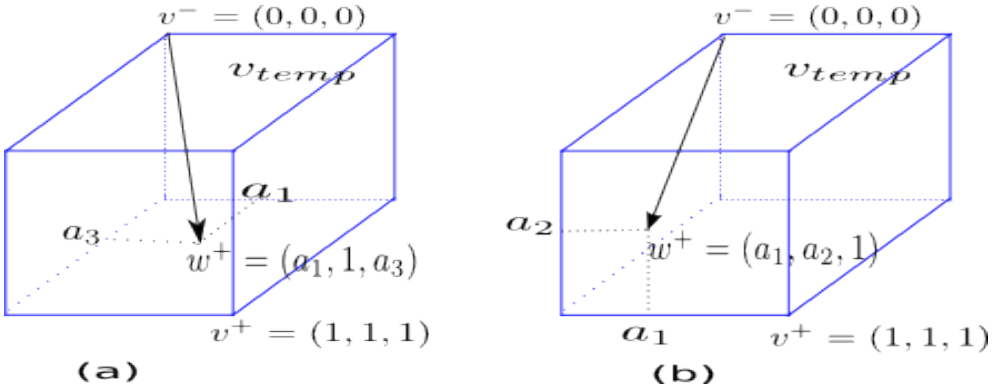



Figura 6.11: (a) $\boldsymbol{w}^+$ na face adjacente "base" (b) $\boldsymbol{w}^+$ na face adjacente "frente"

ii. Para a outra face de saída, definindo $v^* = (a_1, 1, 1)$, temos um triângulo cuja área é dada por $\frac{a_1}{2}$. Ver figura 6.12. Observe que $v^+$ não está mais na região de projeção da face $F_E$, $v^+$ não é atingido pelo fluxo, então $v^*$ é o ponto dentro da região de projeção, diametralmente opsto a $v^-$.

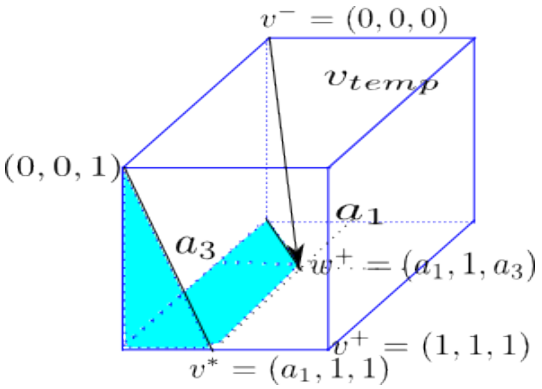



Figura 6.12: Área nas faces adjacentes

Assim, a partir da projeção de $\boldsymbol{v}^-$ segundo a direção de $\boldsymbol{v}_{temp}$, obtemos a área de cada face de saída $F_S$ que será atrevessada pelo fluxo de entrada por $F_E$. A área de saída numa face $F_S$ referente a face $F_E$ será referida a partir de agora por $A_{out}\,(F_E, F_S)$.

Para identificar as porções de $F_E$ pelas quais entra o fluxo que irá deixar $A$ por cada uma das faces de sáida $F_S$, que representaremos por $A_{in}\,(F_E, F_S)$, devemos fazer uma projeção em sentido contrário (contrário a $\boldsymbol{v}_{temp}$) de $\boldsymbol{v}^+$ ou $\boldsymbol{v}^*$ sobre a face de entrada.

Para considerar conjuntamente os dois casos indicados acima, lembrando que tanto $\boldsymbol{v}^+$ como $\boldsymbol{v}^*$ são os pontos atingidos pelo fluxo de entrada, em $F_E$, mais distantes de $\boldsymbol{v}^-$, vamos nos referir a ambos por $\boldsymbol{v}_{far}$.

Seja então $\boldsymbol{w}^-$ a projeção de $\boldsymbol{v}_{far}$ sobre $F$, ver figura 6.13, e $(0, a'_2, a'_3)$ as suas coordenadas $S_{\boldsymbol{v}^-}$. Nesse caso temos que:

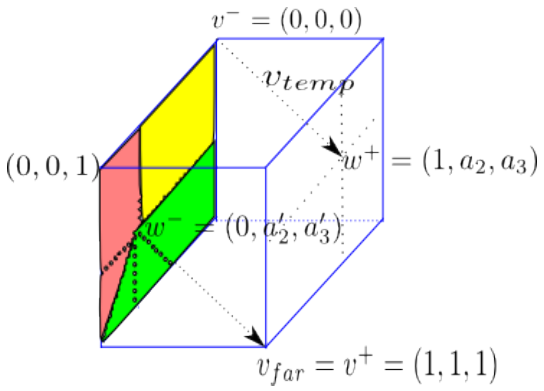



Figura 6.13: Porções de Entrada

1. A porção de $F_E$ que manda fluxo para face oposta é dada pelo retângulo com vértices opostos $\boldsymbol{v}^-$ e $\boldsymbol{w}^-$. No caso em que $\boldsymbol{v}_{far} = \boldsymbol{v}^*$ esse retângulo degenera num segmento. Sua área nula indica que o fluxo por $F_E$ não chegará a face oposta. Em qualquer caso a área do retângulo é dada por $a'_2.a'_3$.

2. A porção que manda fluxo para uma face adjacente $F_A$ é dada por um trapézio definido pela aresta $F_E \cap F_A$ e pelo segmento paralelo a ela que parte de $\boldsymbol{w}^-$ e fecha o trapézio. No caso em que $\boldsymbol{v}_{far} = \boldsymbol{v}^*$ o trapézio relativo a face de saída que não contém $\boldsymbol{w}^-$ degenera em um triângulo, pois um dos $a'_i$ é nulo. Em qualquer caso a área do trapézio terá a forma $(1 + a'_i)/2.(1 - a'_j)$ onde $i = 2$ e $j = 3$ ou $i = 3$ e $j = 2$.

Tendo obtido para uma dada face de saída $F_S$ as áreas $A_{in}(F_E, F_S)$ e $A_{out}(F_E, F_S)$ para toda face de entrada $F_E$ a velocidade em $F_S$ pode ser finalmente computada por

$$\boldsymbol{v}(F_S) = \frac{\sum_{F_E} \boldsymbol{v}(F_E) A_{in}(F_E, F_S)}{\sum_{F_E} \boldsymbol{A}_{out}(F_E, F_S)} \tag{6.1}$$

Se $\boldsymbol{v}_{temp}$ fosse o mesmo para todas as faces de entrada, teríamos $\sum_{F_E} \boldsymbol{A}_{out}(F_E, F_S) = 1$ e não haveria sobreposição das porções de $F_S$ atingidas por faces de entrada diferentes.

Pode acontecer do fluxo de entrada por uma face $F_{E_1}$ atingir outra face também de entrada $F_{E_2}$, esse fato não será levado em conta. Considera-se que essa situação configura o choque entre duas fontes de propagação que deve ser mediado pela ação das pressões quando se resolve o sistema de Poisson.

No texto acima assumimos que as faces de entrada estão completamente cheias. A introdução de limites para o fluido dentro da face acrescenta as dificuldades consideradas abaixo:

1. Como são estabelecidos esses limites? A idéia é aproveitar os próprios resultados obtidos pelo mecanismo de traçado de borda. Aqui assumiremos que esse mecanismo é constituido por um procedimento clássico utilizando Marching-Cubes no qual a intersecção da borda com uma face é constituida por normalmente um, ou no máximo dois segmentos de reta, dado que uma aresta só pode ser cortada uma única vez.

   Quando existem dois segmentos numa face de entrada $F_E$, temos duas situações. Na primeira a porção fluida de $F_E$, $F_E^{fluida}$ é desconexa. Nesse caso cada uma das duas componentes é tratada separadamente e a contribuição delas para o fluxo em cada face de saída é somada.

   Quando ao invés, a porção fluida é conexa, então, tomando-se o complementar se recai no caso anterior. Obtemos a contribuição de $F_E^{fluida}$ para uma face de sáida, subtraindo a contribuição desse complementar da que essa face receberia de $F_E$ se $F_E$ estivesse cheia. Assim, evitamos ter que dar tratamento específico ao caso em que a borda intercepta uma face em mais de um segmento, economizando memória se pretendemos tabular os valores dessas contribuições. Observamos que isso é possível dada a aditividade tanto do fluxo para uma

face de saída como da área dessa face atingida por esse fluxo quando se consideram conjuntos disjuntos de $F_E$. A figura 6.14 ilustra esse segundo caso.

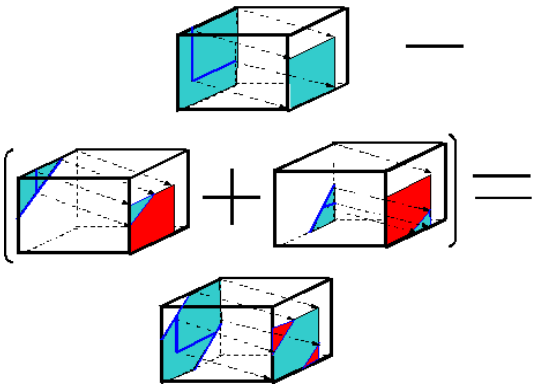


Figura 6.14: Área total menos as áreas do canto

Ocorre que ao utilizar dados produzidos pelo algoritmo de traçado de borda, que normalmente só existem durante a etapa de visualização, temos que trazê-los para a estrutura associada a uma face ou célula.

Na formulação clássica temos uma única componente de velocidade armazenada por face, um único valor de pressão e um label por célula. Assim, registrar em que ponto as arestas são cortadas, introduziria mais 4 dados para cada face, isso significaria acrescer a necessidade de memória para armazenar os dados da grade, um espaço maior do que eles dispendem no caso clássico.

Isso sugere que esses dados sejam armazenados a parte e acessados via um apontador que é setado apenas para as faces de células de borda.

Para independer de dados fornecidos pelo traçador, podemos adotar templates para ocupação da face pela massa fluida levando em conta a área ocupada por ela. Considera-se que essa área foi computada quando se tratou a célula da qual a face é de saída, na própria iteração corrente.

Além dessa área o template leva em conta o conjunto de vértices da face que pertencem a célula na massa fluida.

Esses vértices são indicados por pequenos círculos vermelhos na figura 6.15, em que são apresentados todos os casos possíveis levando-se conta não só esses conjuntos de vértices mas sua posição relativa a $v^-$ - no caso, se adjacentes

ou o vértice oposto. Na opção descrita na figura abaixo as bordas da massa fluida são mantidas paralelas às arestas da face $F_E$.

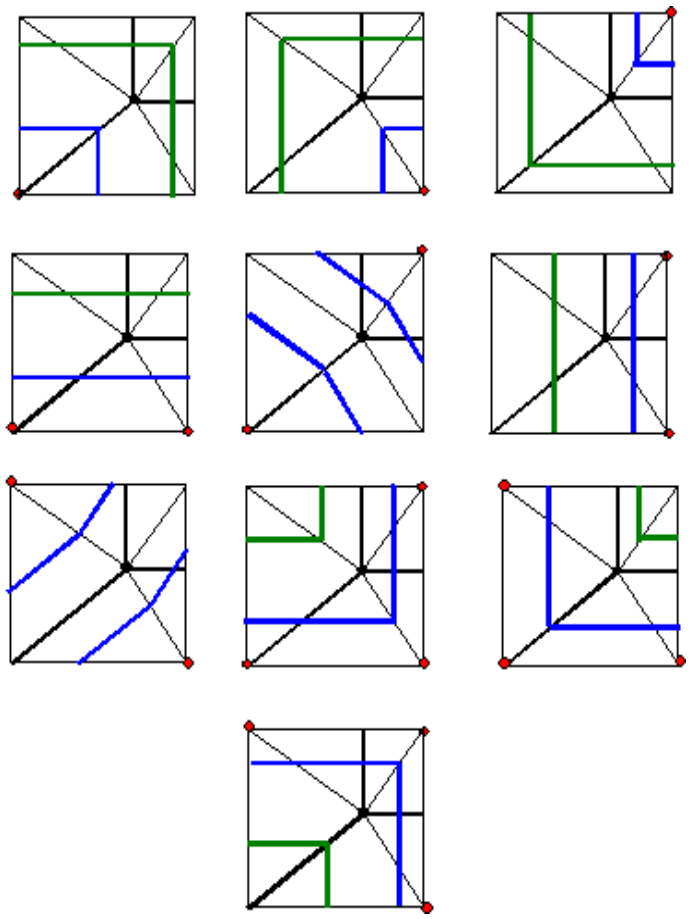


Figura 6.15: Todos os casos possíveis para a representação da massa fluida considerando-se uma configuração retângulos trapézios fixa. Os vértices vermelhos são aqueles que pertencem a massa fluida. O traçado em azul representa a borda quando o vértice comum ao retângulo e aos trapézios ($w^+$) está na porção ar. O traçado em verde representa a borda quando esse ponto está na porção fluido. Nos casos em que vértices opostos da face pertencem ao mesmo meio o traçado da borda tem duas componentes conexas.

Outra alternativa é fazer os limites da borda serem paralelos às raias que ligam os vértices da face de entrada à $w^+$, a retro-projeção de $v^+$ sobre essa face. Todas essas formulações levam a cômputos relativamente simples e as frações do fluxo de entrada na célula por $F_e$ que é direcionado para uma face de saida $F_s$. Além disso, as áreas das faces atingidas por esse fluxo podem ser tabeladas em função de três variáveis em $[0, 1]$, especificamente, as coordenadas de $w^+$,

num sistema de coordenadas local para a face e a área da porção fluida de $F_e$. Adotando uma precisão de 1/8 do lado da célula para as coordenadas de $w^+$ e em consonância de 1/64 para a área fluida em $F_e$ podemos listar

# Capítulo 7

# Tratamento das Células de Borda Utilizando os Resultados do Marching Cubes

Diferente da seção anterior em que o propósito de se usar células de borda era essencialmente o de introduzi-las gradualmente na massa fluida ao invés de fazer isso só quando elas estivessem completamente cheias, nesse capítulo vai-se apresentar uma metodologia que tem a pretensão de inter-relacionar procedimentos empregados na simulação, na evolução da borda e na visualização. Num esquema clássico há módulos para resolver a equação de Navier-Stokes, para evoluir a borda via Fast Marching/Level-Sets e para gerar a superfície do fluido por um mecanismo do tipo Marching-Cubes. Pode-se dizer que, essencialmente, o fluxo de informações de um módulo para o outro, segue a ordem dessa lista. O fluxo em sentido contrário tem relevância menor. Aqui se pretende integrar essas etapas nas células atravessadas pela borda, buscando na verdade se aproximar de uma série de idealizações que foram formuladas em função das dificuldades encontradas na aplicação dos modelos especificados nos dois capítulos anteriores. Esses objetivos, que listamos a seguir são todos atingidos em certa medida empregando-se a proposta descrita nas seções seguintes.

1. Aproveitar a informação relativa ao corte de cada face regular, produzido pela borda obtida através do Marching-Cubes, no cômputo da advecção da velocidade numa célula de borda e na montagem do Sistema de Poisson. Com

isso se pode evitar a estimação de velocidades em pontos que pertencem a porção Ar, que são requeridas para que se possa empregar um sistema semi-lagrangeano. O uso dessa informação serve para ajustar à forma recortada da célula, metodologias para o cômputo de derivadas convectivas. A introdução de células de borda no sistema de Poisson permite efetuar correções na velocidade de deslocamento da face externa usada para fazer a evolução da borda. Como atender a essa finalidade é o objeto das seções 7.1 e 7.4.

2. Utilizar formulações para o cômputo da área da face externa e de sua orientação que dependam apenas da dimensão da área fluida em cada face regular. A orientação deve ser tal que permita a separação dos vértices da célula pertencentes a massa fluida e a porção Ar. Isso pode tornar menos direto o cálculo da orientação em casos em que 5 arestas da célula são cortadas por uma mesma componente conexa da interseção borda/célula ou mesmo ser impossível de obter em situações em que 6 arestas são cortadas por uma componente. Nesse último caso se deve subdividir a célula. Esclarecemos que essas situações anômalas são raras e que de acordo com o modelo para o traçado da borda dentro de uma célula empregado pelo Marching-Cubes clássico, 6 arestas de uma célula é o máximo que uma componente da borda pode cruzar. A seção 7.3 enfoca essas questões.

3. Outro paradigma empregado para a escolha da orientação da face externa é o de que exista uma triangulação delimitada pelas arestas de borda nas faces regulares da célula, tal que uma reta com essa orientação corte cada triângulo não mais de uma vez. O que vai se mostrar na seção 7.3 é que se é possível separar os vértices da célula que estão em meios distintos, então se pode fazer isso atendendo também a esse outro paradigma.

4. Na medida do possível fazer com que todas as operações efetuadas num passo da simulação envolvendo uma célula de borda, incluindo as utilizadas para se obter a orientação e a área da face externa, possam ser feitas a partir de dados computados nas faces sem precisar explicitar o traçado da borda no interior da célula. Isso, entretanto, pode ser custoso computacionalmente, como pode ser o caso da advecção. Para simplificar podemos substituir o

corte da borda numa face $F$, pelo corte determinado em $F$ pela versão planar da face externa. Fazendo isso, passa-se a ter em $F$ um corte da borda para cada célula adjacente. A expectativa é que a descontinuidade produzida não seja relevante. A temática de independer do traçado da borda no interior da célula, nos restringindo a considerar apenas sua interseção com as faces da célula, está presente em todas as seções seguintes.

5. Computar novos pontos de interseção da borda com as arestas ao invés de obtê-los por interpolação a partir de estimativas da distância de seus vértices à borda. Mais especificamente, efetuar um procedimento de regularização - entendido como um processo que concilia informações produzidas em células adjacentes - ao nível das arestas e não para os vértices. Depois obter novos pontos de corte por interpolação. Essa alternativa deve ser preferida se/onde for detectada perda de volume. Uma proposta para ela é apresentada na seção 7.4.

6. Dispor de tabelamentos obtidos a priori para manter a complexidade das operações realizadas nas células de borda comparável a efetuada para células regulares. Essa possibilidade pode ser certamente vantajosa em implementações em CPU. Ajustá-las a um contexto de paralelização ainda é um desafio. Elas demandam ainda uma representação adequada para as variáveis envolvidas de forma que a própria indexação necessária para fazer consultas às tabelas não se torne onerosa a ponto de eliminar qualquer vantagem que o pré-processamento possa propiciar. Para reduzir o número de tabelas a serem armazenadas, a célula é rotacionada conforme sua classificação efetuada pelo Marching-Cubes ou o octante do vetor velocidade considerado. Exemplos de uso desses tabelamentos são apresentados nas seções 7.1 e 7.5.

O modelo proposto neste capítulo é uma tentativa de ir de encontro a todas as dificuldades que foram encontradas na aplicação das sucessivas versões empregando uma abordagem clássica. Células de borda permitem a representação de filetes mais finos que uma célula e que a dinâmica da simulação nas células adjacentes aos limites do container não seja dependente da resolução empregada, o que pode acarretar situações irrealistas. Fazendo a evolução da borda nos moldes descritos

na seção 7.4 se pode impedir a perda de volume. Esse modelo entretanto não se acha inteiramente implementado porque sua concepção é recente. Além disso, a ideia é fazê-lo coexistir com a proposta que já temos implementada sem aumentar consideravelmente as necessidades de memória. Observamos que essas necessidades já se tornam maiores quando fazemos uso das informações produzidas pelo Marching-Cubes antes restritas a sua aplicação - nesse caso os dados relativos a uma célula eram produzidos quando do tratamento dessa célula e depois descartados - acessíveis a todas as etapas de simulação. Por isso os resultados computacionais apresentados no capítulo seguinte não se referem ainda a aplicação do modelo descrito aqui embora parte dos mecanismos expostos neste capítulo já estejam implementados.

## 7.1 Representação Linear e Linear por Partes da Face Externa

Numa simulação clássica, montada numa grade regular, a borda não influi diretamente no cômputo das velocidades e é calculada a parte. Sua evolução é normalmente feita por um mecanismo que a desloca mantendo-a ajustada à grade. Ao considerarmos a influência da borda, teríamos um ganho computacional se adotássemos como representação da face externa, o próprio traçado determinado pelo mecanismo de responsável por essa etapa, por exemplo o Marching-Cubes. Só que isso pode ocasionar recortes mais complicados e não inteiramente definidos.

A complexidade vem por conta da interseção da borda com uma célula (cúbica) poder ter várias componentes. Nesse caso para cada componente conexa da interseção do volume fluido com a célula regular deve ser criada uma célula de borda. Quando, ao contrário, essa interseção é conexa, mas sua fronteira dentro da célula não é, temos o caso em que a face externa tem várias componentes cada uma delas com sua própria velocidade. Adiantamos que nesse último caso as velocidades dessas componentes serão obtidas a partir das definidas nas faces regulares, de forma a não complicar ainda mais o modelo. Ver figura 1.1.

A partir dessa representação pode-se obter outra mais simples e independente de opções, encontrando-se o plano que melhor se ajusta aos vértices dela, entre os que cortam as mesmas arestas da célula. A face seria então aproximada pela interseção

desse plano com a célula, constituindo o que vamos chamar de versão planificada da face externa. A partir de sua versão planificada se atribui uma normal a face externa e, é claro, que todas operações podem ser agilizadas se ela for empregada no lugar da representação inicial. Para algumas operações, entretanto, essa vantagem existe, mas poderia ser dispensada em benefício de maior precisão. Além disso, outras formas de representação podem ser usadas, mas na descrição dos procedimentos que tratam a célula de borda, que é feita a seguir, nos restringiremos a essas duas mais simples.

## 7.2 Advecção numa Célula de Borda

Em relação a advecção numa célula de borda seguimos em linhas gerais o procedimento descrito no capítulo anterior. Os princípios básicos adotados são os mesmos - variações na velocidade do meio fluido numa face de entrada $F_e$ são sentidas automaticamente em todas as faces de saída para onde se direciona o fluxo que entra por $F_e$ . Isso nos evita fazer cortes no volume fluido dentro de uma célula, ou interpolações empregando velocidades de faces ou vértices definidas por extrapolação no meio ar.

Tudo que se precisa calcular, então, é quanto da vazão que entra por uma dada face $F_e$ da célula de borda $C$ -incluindo possivelmente a externa - vai sair por uma face de saída $F_s$. As diferenças entre os fluxos de entrada e saída considerando-se apenas as faces regulares são atribuídas ao fluxo que atravessa a face externa $F_{ext}$. Para cada face de entrada se define uma velocidade de transporte dentro da célula do fluxo que entra por ela ($\boldsymbol{v}_e$). Essa velocidade pode ser definida tomando-se a velocidade da face $F_e$ e obtendo as outras componentes da célula de onde vem o fluxo que entra por $F_e$. Ver fig. 7.1.

Esse modelo se adequa às situações em que o número de faces de entrada não é menor que o de saída. No caso de termos o contrário deve-se empregar um procedimento dual definindo-se uma velocidade ($\boldsymbol{v}_s$) para cada face de saída $F_s$ e computando-se o fluxo em cada célula de entrada que movendo-se na direção dessa velocidade atingiria $F_s$.

Alternativamente pode se definir uma velocidade única $\boldsymbol{v}_C$ para todas as faces de entrada da célula. Se parte do fluxo que vem de uma face de entrada é direcionado

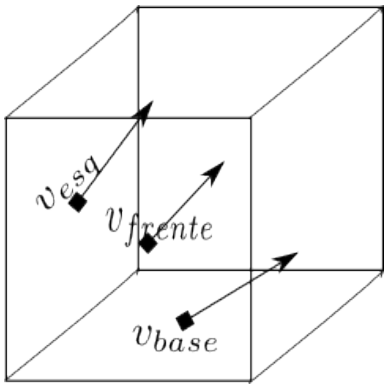



Figura 7.1: Velocidades 3D para cada face de entrada calculadas para se computar o fluxo que vai delas a cada face de saída.

para outra face, também de entrada, então esse fluxo não é descontado dessa última face, deixando-se para o sistema de Poisson resolver o acúmulo de fluido dentro da célula.

Assumimos que essa formulação é absolutamente geral sob condições de incompressibilidade. Com isso queremos dizer que para qualquer configuração de fluxos nas faces de uma célula, desde que eles somem zero, pode ser reproduzida definindo-se velocidades de faces que transportem o fluxo que chega/sai por elas suposto atingir a sua área de forma uniformemente distribuída. Esse resultado pode ser ainda mais forte: Se o número de faces de entrada é igual às de saída então existe de fato uma velocidade única para a célula tal que a repartição de fluxos determinada por essa velocidade reproduz os valores das velocidades de face.

Observamos que formas clássicas de cômputo da advecção da componente da velocidade definida em uma face empregam o valor dessa própria componente. Ocorre que para células de borda recém criadas esse valor pode ainda não existir e precisa, então, ser computado via extrapolação. A proposta apresentada aqui visa obter esses novos valores via advecção, que nesse caso é computada empregando-se valores obtidos para velocidades de outras faces da célula. Nada impede, entretanto, que o esquema proposto aqui seja adaptado de forma a empregar informação relativa à face para onde se está computando a adveção, desde que ela esteja disponível.

Em relação ao capítulo anterior, a diferença é que agora temos um traçado da fronteira ar-fluido em cada face regular fornecido pelo marching cubes. Isso vai nos permitir computar o fluxo que atravessa a face externa adotado-se, por exemplo, a

versão planar de $F_{ext}$.

Sem perda de generalidade podemos supor que a velocidade $\boldsymbol{v}_e$ tem todas as componentes positivas e que $F_e$ é a face definida por $w = 0$ sendo $u$ e $v$ as demais coordenadas de um sistema em que a célula $C$ é representada pelo o cubo unitário.

Se o vértice $v_{op} = (1, 1, 1)$, projetado segundo a direção de $\boldsymbol{v}_e$ sobre o plano $w = 0$, for um ponto $p = (0, a, b)$ de $F_e$ então o fluxo que entra por $F_e$ sairá pelas três faces adjacentes a $v_{op}$, incluida a contida em $w = 1$. Ver figura 7.2.

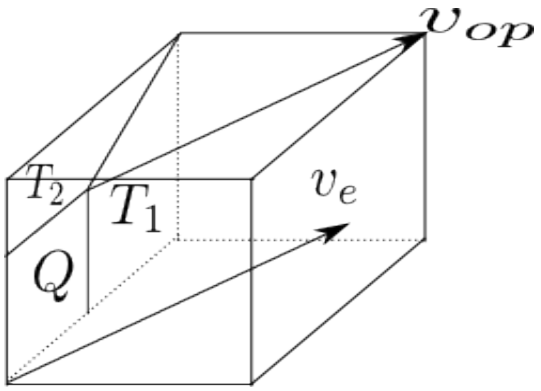



Figura 7.2: Frações de uma face de entrada por onde o fluxo passa para ir atingir 3 faces de saída, incluindo a oposta a ela.

Se $p \notin F_e$, seja $p_{\text{dist}}$ o ponto de $C$ mais distante da origem cuja projeção segundo uma direção paralela a $\boldsymbol{v}_e$ sobre $w = 0$ é um ponto $q$ de $F_e$. Ver figura 7.3.

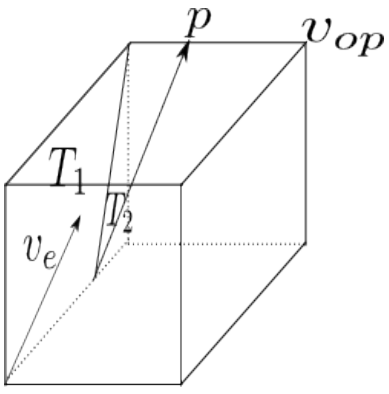



Figura 7.3: Frações de uma face de entrada por onde o fluxo passa para ir atingir 2 faces de saída.

Escolhendo adequadamente as coordenadas $u$ e $v$, $q$ pode ser escrito no sistema $(u, v, w)$ da forma $(0, 0, c)$. Fazendo, nesse segundo caso $a = 0$ e $b = c$, podemos dizer

que, em qualquer caso, o ponto $(0, a, b)$ define um retângulo, $Q = 0 \times [0, a] \times [0, b]$, e dois trapézios, ($T_1$ e $T_2$), separados pelo segmento que liga esse ponto a $(0, 1, 1)$, os quais particionam $F_e$. Suas áreas definem a porção de fluxo que será direcionada a cada face de saída. Isso assumindo que $F_e$ é inteiramente ocupada pela massa fluida.

Se $F_e$ não é inteiramente ocupada pela massa fluida, então sejam $\alpha$ e $\beta$ os parâmetros que definem os extremos do corte determinado em $F_e$ pela borda, inicialmente suposto ser constituído por um único segmento. A partir desses parâmetros e da classificação da célula pelo marching cubes se obtém a reta que contem esse corte determinando, por exemplo, seus coeficientes lineares $p$ e angular $m$, relativos ao sistema $(u, v)$. Se pretendemos tabelar as frações destinadas a cada face de saída em função de coeficientes que definem a borda, amostrados numa dada resolução, podemos utilizar os próprios parâmetros $\alpha$ e $\beta$ para manter o intervalo de amostragem em $[0, 1]$. Nesse caso precisamos ter apenas duas tabelas, uma para retas que cortam arestas opostas da face e outra para as que cortam arestas adjacentes. Todas as combinações relativas ao par de arestas cortadas e ao lado da borda onde está a massa fluida podem ser cobertas empregando-se complementaridade em relação a face cheia e trocando-se variáveis. A figura 7.4 indica como isso é feito. A tabela a ser utilizada bem como os valores efetivamente empregados na consulta ($\alpha, \beta$ e seus complementares) é determinada pela classificação do marching-cubes.

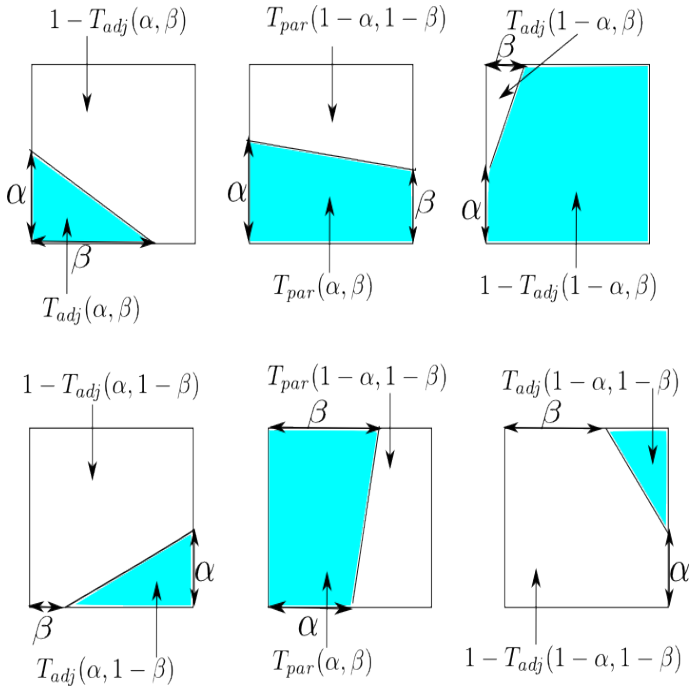



Figura 7.4: Todos os possíveis casos de interseção entre face e borda constituída por uma única aresta podem ser cobertas se tivermos duas tabelas $T_{adj}$ e $T_{par}$ que indicam o valor das áreas das porções delimitadas pela aresta de borda quando ela corta lados adjacentes e paralelos da face, respectivamente.

O caso em que o corte de $F_e$ pela borda é constituído por dois segmentos pode ser reduzido a se aplicar reiteradamente o tratamento especificado acima. Assim se a interseção da massa fluida com $F_e$ é desconexa obtemos a vazão que se destina a cada face de saída partindo de cada componente e somamos as duas. Se ela for conexa, a face terá duas componentes no meio ar. Então pensamos que o direcionamento do fluxo para as faces de saída é obtido aplicando-se o tratamento acima à face toda e aos dois triângulos que constituem a porção ar. Os resultados obtidos para esses triângulos são somados e descontados dos que se encontrou para a face toda. A figura 7.5 ilustra esses dois casos.

Encontrada a fração da entrada que se destina a cada face de saída pode-se passar a resolver quanto dessa vazão alcança a porção fluida da face de saída e quanto dela escapa pela face externa. Sejam, então: $e_e$ e $e_s$, respectivamente, as arestas de borda na face de entrada($F_e$) e saída ($F_s$).

Refira, ainda, por $P_{\boldsymbol{v}_e F_s}(e_e)$, a projeção, segundo a direção da velocidade $\boldsymbol{v}_e$ definida acima, de $e_e$ sobre $F_s$. Se empregarmos a aproximação planar da face

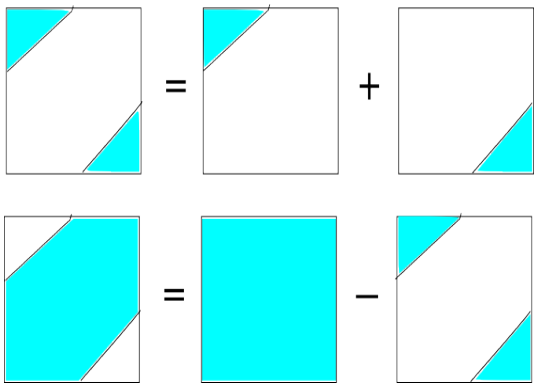


Figura 7.5: Em faces cuja intersecção com a massa fluida é desconexa, cada componente da borda é tratada separadamente. Quando é a porção Ar da face que é desconexa, dos resultados obtidos para a célula cheia são subtraidos os que seriam obtidos se a porção Ar fosse fluida.

externa, $P_{\boldsymbol{v}_e F_s}(e_e)$ não vai cortar $e_s$ em seu interior relativo. Assim temos apenas duas possibilidades a considerar:

1. $P_{\boldsymbol{v}_e F_s}(e_e)$ pertence a porção fluida de $F_s$. Nesse caso todo fluxo que vai da porção fluida de $F_e$ para $F_s$ é aproveitado. Nada sai pela face externa da célula. Para efeito da obtenção dessas vazões numa tabela pré-computada podemos desprezar $e_s$ e considerar apenas a restrição imposta por $e_e$, como já foi descrito acima.

2. $P_{\boldsymbol{v}_e F_s}(e_e)$ pertence a porção ar de $F_s$. Nesse caso todo o fluxo que chega a $F_s$ vindo de $F_e$ é aproveitado. O excedente que sai de $F_e$ vai atravessar a face externa. No contexto de se fazer esse cômputo usando dados pré-computados temos novamente 4 parâmetros (2 de $\boldsymbol{v}_e$ e 2 de $e_s$).

Caso não se queira computar uma velocidade com 3 componentes para a face externa e obter o fluxo que a atravessa como um ajuste, em função do que transita entre as faces regulares da célula podemos adotar o seguinte procedimento.

1. Ao computar a porção do fluxo que vai de $F_e$ para $F_s$ determina-se também a área de cada uma dessas faces atravessada pelo fluxo que transita de uma a outra. Considere $AE(e, s)$ a área contida na porção fluida de $F_e$ que se destina a face $F_s$, sem considerar a restrição imposta pela borda em $F_s$. Analogamente,

$AS(e,s)$ é a área da porção fluida de $F_s$ que recebe fluxo de $F_e$, sem considerar a restrição imposta pela borda nessa última face. No primeiro caso eliminamos a restrição de borda na saída e no segundo, na entrada. Além disso, considere $PrAE(e,s)$ a projeção de $AE(e,s)$ sobre $F_s$ segundo a direção de $\boldsymbol{v}_e$. Portanto, $AE(e,s)$ e $PrAE(e,s)$ se relacionam por:

$$PrAE(e,s) = AE(e,s), \text{ se } F_e \text{ e } F_s \text{ são paralelas}$$

e em caso contrário,

$$PrAE(e,s) = \frac{|v_{e,u}|}{|v_{e,v}|} AE(e,s),$$

onde $v_{e,u}$ e $v_{e,v}$ são as componentes de $\boldsymbol{v}_e$ nas direções paralelas a $F_e$ e $F_s$, respectivamente, e ortogonais a aresta comum entre essas faces.

Uma dessas áreas pode ser obtida de uma tabela e a outra via a relação acima, ou ambas podem ser tabeladas a priori, definida uma precisão. A figura 7.6 ajuda a entender essas relações.

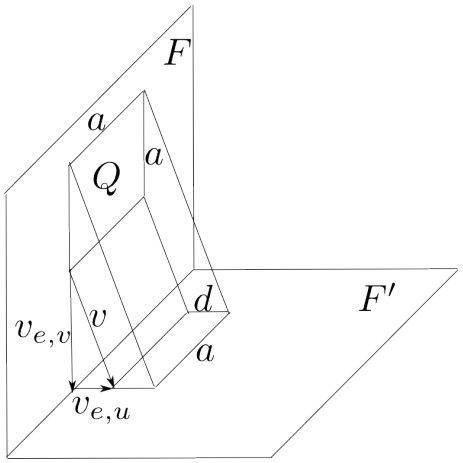



Figura 7.6: Relação entre a área de um quadrado $Q$ em uma face F com lados paralelos a interecção dessa face com uma adjacente $F'$ e a de sua projeção segundo a direção $\boldsymbol{v}_e$ sobre $F'$.

Seja $Q$ um quadrado elementar de lado $a$ situado numa face $F$ tendo um lado paralelo a interseção dessa face com outra $F'$ e outro perpendicular a essa interseção. Na projeção de $Q$ sobre $F'$ o lado paralelo a interseção mantém as dimensões. Por semelhança de triângulos a distância $d$ entre os lados paralelos

da projeção mantém a relação $d/|v_{e,v}| = a/|v_{e,u}|$. A relação entre a área da projeção(ad) e a do quadrado é dada, portanto por $\frac{|v_{e,u}|}{|v_{e,v}|}$. Se essa relação vale para quadrados com essa orientação vale para qualquer subconjunto de $F$, dado que ele pode ser expresso pela união de um conjunto desses quadrados, com interiores disjuntos dois a dois.

Do mesmo modo defina $PrAS(e,s)$ como sendo a projeção de $AS(e,s)$ sobre $F_e$ segundo a direção de $\boldsymbol{e}_e$. $PrAS(e,s)$ e $AS(e,s)$ guardam entre si as mesmas relações que $AE(e,s)$ e $PrAE(e,s)$. Seja ainda $\boldsymbol{n}_{ext}$ vetor normal a versão planar da face externa de $C$. A figura 7.7 ilustra a definição desses subconjuntos quando $F_e$ e $F_s$ são opostas e a figura 7.8 quando elas são adjacentes.

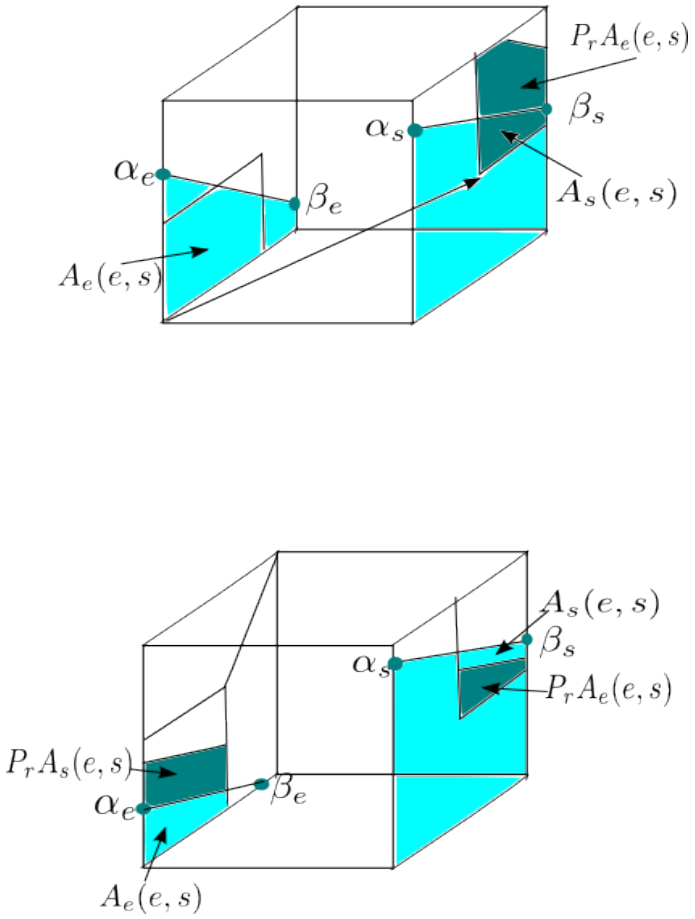



Figura 7.7: Subconjuntos de $F_e$ e $F_s$ relevantes para o cálculo da porção de fluxo transferida de uma a outra quando essas duas faces são opostas.

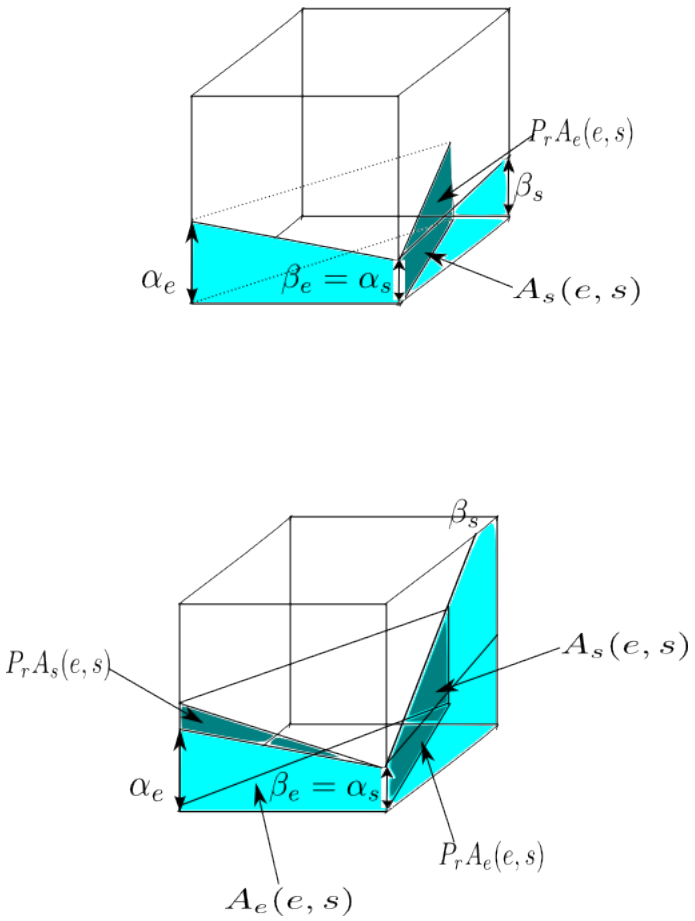



Figura 7.8: Subconjuntos de $F_e$ e $F_s$ relevantes para o cálculo da porção de fluxo transferida de uma a outra quando essas duas faces são adjacentes.

Os dois casos indicados nas figuras se referem às duas possibilidades consideradas no item 2 abaixo.

2. Se $\langle \boldsymbol{v}(e), N_{ext} \rangle \geq 0$ ou $F_e$ é uma face inteiramente no volume fluido execute:
   (a) $Fl_{delta} = \boldsymbol{v}(F_e) AE(e,s)$
   (b) $Fl(s) += Fl_{delta}$ (fluxo na face de entrada em questão que deveria acontecer na face de saída $s$ caso a mesma estivesse completamente coberta pelo fluido).
   (c) $Fl'(s) += Fl_{delta} \frac{AS(e,s)}{PrAE(e,s)}$ (fluxo que efetivamente acontece na face de saída $s$ e que tem origem na face de entrada em questão).
   (d) $AS(s) += AS(e,s)$
3. Em caso contrário faça:
   (a) $Fl_{delta} = \boldsymbol{v}(F_e) PrAS(e,s)$
   (b) $Fl'(s) += Fl_{delta}$ (fluxo na face de saída $s$ que deveria ter origem na respectiva face de entrada caso a mesma estivesse completamente coberta pelo fluido).
   (c) $Fl(s) += Fl_{delta} \frac{PrAE(e,s)}{AS(e,s)}$ (fluxo na face de saída $s$ que efetivamente tem origem na face de entrada em questão).
   (d) $AS(s) += AS(e,s)$
4. Ao final, quando todas as faces de entrada que enviam fluxo para $F_s$ tiverem sido consideradas, obtenha a velocidade em $F_s$ por: $\boldsymbol{v}(F_s) = \frac{Fl'(s)}{AS(s)}$. Além disso vá acumulando o fluxo que deve atravessar a face externa fazendo $Fl_{ext} += (Fl(s) - Fl'(s))$.
5. Ao final do processamento da célula o valor de $Fl_{ext}$ indicará a quantidade de fluxo que deve atravessar a face externa. Um valor de $Fl_{ext}$ positivo indicará que esse fluxo está saindo da porção fluida na célula, que assim deverá aumentar de volume. Ao contrário, um $Fl_{ext}$ negativo indicará um fluxo entrando na parte fluida que deverá ter seu volume reduzido.

O raciocínio que justifica a metodologia acima, fica mais fácil de assimilar observando o simples exemplo 2D representado na figura 7.9.

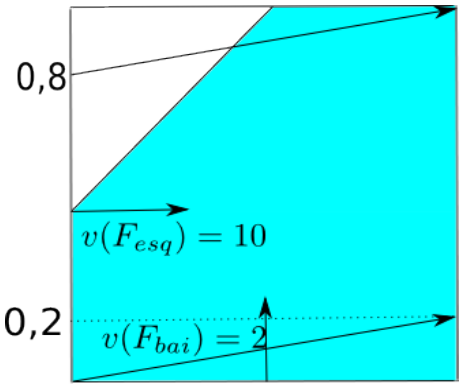



Figura 7.9: Fluxo através de uma célula 2D.

Nesse exemplo, $C$ é uma célula de lados unitários onde a velocidade da face da esquerda($F_{esq}$) é 10 e a da face de baixo ($F_{bai}$) é 2. As faces da direita ($F_{dir}$) e de cima ($F_{cim}$) são consideradas faces de saída a determinar. Usando uma extrapolação simples essas faces teriam velocidade 10 e 2, respectivamente. Para ambas as faces de entrada vamos associar uma velocidade para o trânsito do fluxo dentro da célula dada por(10,2). Nesse caso, o fluxo que chegaria a face da direita teria 0.8 de sua composição vindo da face da esquerda, que chegaria a ela a partir da altura 0.2 e 0.2 que sairia de $F_{bai}$ e chegaria a $F_{dir}$ até essa altura. A face de cima é inteiramente suprida pela porção de $F_{esq}$ a partir da altura 0.8.

Suponha agora que a borda corta os pontos médios de $F_{esq}$ e $F_{cim}$ estando a parte fluida abaixo dela. Nesse caso, teremos os seguintes valores:

1. $AE(esq, cim) = 0$, já que até a altura 0.5 $F_{esq}$ não manda fluxo para $F_{cim}$, só para a face da direita.

2. $AS(esq, cim) = 0.5$ já que toda porção fluida da face de cima vem da face esquerda.

3. $PrAS(esq, cim) = 0,1$ correspondendo ao trecho de $F_{esq}$ entre 0.8 e 0.9.

4. O produto escalar $\langle V(e), N_{ext}\rangle = \left\langle (10,2), (-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}) \right\rangle < 0$, de forma que devemos executar a segunda alternativa do procedimento acima, item 3 da enumeração anterior. Nesse caso teremos:

   (a) $PrAS(esq, cim) = 0.1$,

   (b) $Fl_{delta} = 10 * 0.1 = 1.0$,

(c) $Fl'(cim) = 1.0$ (fluxo na face de cima que deveria vir da face esquerda caso a mesma estivesse completamente coberta pelo fluido),

(d) $Fl(cim) = 0$ (fluxo na face de cima que efetivamente tem origem na face esquerda).

Resultando numa contribuição para o fluxo que atravessa a face externa de $Fl(cim) - Fl'(cim) = -1.0$. Dado que todo fluxo que chega a $F_{cim}$ vem de $F_{esq}$, resulta que $v_{Fcim} = \frac{Fl'(cim)}{AS(esq,cim)} = \frac{1}{0,5} = 2$.

5. $AE(esq, dir) = 0.5$ e $PrAS(esq, dir) = AS(esq, dir) = 0.8$ que é o comprimento dos segmentos $[(1, 0.2), (1, 1)]$ ou $[(0, 0), (0, 0.8)]$. Supondo que a face de entrada da esquerda é processada antes da de baixo temos, ao final desse primeiro processamento que $Fl'(dir) = 10 * 0.8 = 8$, $Fl(dir) = 10 * 0.5 = 5$ que implicam numa contribuição de $-3$ para o fluxo que atravessa a face externa e 0.8 para $AS(dir)$.

6. Considerando, em seguida o fluxo que vem da face de baixo temos $AE(bai, dir) = 1.0$ e $PrAE(e, s) = AS(bai, dir) = 0.2$. Como $F_{bai}$ está totalmente imersa na massa fluida devemos empregar a primeira alternativa acima obtendo $Fl_{delta} = 10 * 0.2 = 2$ e daí, $Fl(dir) = 5 + 2 = 7$, $Fl'(dir) = 8 + 2 = 10$ e $AS(dir) = 0.8 + 0.2 = 1.0$ de onde, finalmente, tiramos $V_{Fdir} = 10$. Somando as contribuições de cada caso, o fluxo que atravessa a face externa é -4. Temos então como resultado final que fluxos com valores 5, 2 e 4, entram, respectivamente, por $F_{esq}$, $F_{bai}$ e $F_{ext}$ e saem vazões de 10 e 1 pelas faces da direita e de cima.

Com respeito a idéia de se adaptar a metodologia descrita acima de forma a dar maior peso à informações obtidas no entorno da face para onde se está calculando a advecção começamos por reinterpretar os termos envolvidos nesse cálculo no caso de uma malha regular.

Na presença de incompressibilidade, a coordenada $i$ de $\nabla.(\boldsymbol{v}\boldsymbol{v}^T)$, $\left[\nabla.\left(\boldsymbol{v}\boldsymbol{v}^T\right)\right]_i$, pode ser expressa por :

$$\left[\nabla.\left(\boldsymbol{v}\boldsymbol{v}^T\right)\right]_i = v_i\frac{\partial v_i}{\partial x_i} + v_j\frac{\partial v_i}{\partial x_j} + v_k\frac{\partial v_i}{\partial x_k} \tag{7.1}$$

onde $x_i, x_j$ e $x_k$ são as coordenadas canônicas e $v_i, v_j$ e $v_k$ são as componentes de $\boldsymbol{v}$ relativas a elas. Há várias maneiras de discretizar a expressão acima. Preferimos aquelas que fazem com que, pelo menos para um $\Delta t$ suficientemente pequeno, o resultado da advecção de $v_i$ relativa a uma dada face seja uma combinação convexa dos valores de $v_i$ relativos a essa face e as faces vizinhas que tem a mesma orientação.

Se considerarmos que o valor de $v_i$ é estendido das faces para todo o volume fluido por uma interpolação contínua, a propriedade acima garante que o resultado da advecção de $v_i$ numa face no instante $t_k + \Delta t$ é o valor assumido por ela em algum ponto da massa fluida no instante $t_k$ como se espera de um processo de advecção.

Entre as formulações com essa propriedade, utilizada na equação $\frac{\partial v_i}{\partial t} = \left[\nabla . \left(\boldsymbol{v}\boldsymbol{v}^T\right)\right]_i$, a mais simples de todas, em que os coeficientes da combinação são obtidos a partir de dados de uma única célula, é explicitada a seguir para o caso em que $(i = 1, j = 2, k = 3)$ e também que:

1. A face onde se calcula a advecção é a face da direita $F_D$ de uma célula $C$ , indexada por $(m-1, n, p)$.

2. Todas as componentes de velocidade envolvidas são positivas.

3. A célula cúbica tem lado com comprimento igual a 1.

Nessas condições, a equação 7.1 discretizada toma a forma:

$$\begin{aligned} v_1^{t+\Delta t}(m,n,p) = \; & v_1(m,n,p) \\ & -\Delta t[(v_1(m,n,p) + v_1(m-1,n,p))/2][v_1(m,n,p) - v_1(m-1,n,p)] \\ & +\Delta t.v_2(m-1,n,p).v_1(m,n,p) \\ & -\Delta t.v_2(m-1,n+1,p).v_1(m,n+1,p) \\ & +\Delta t.v_3(m-1,n,p).v_1(m,n,p) \\ & -\Delta t.v_3(m-1,n,,p+1).v_1(m,n,p+1) \end{aligned} \tag{7.2}$$

Em termos do fluxo na célula $C$ no intervalo $\Delta t$, as linhas da expressão acima, 7.2, podem ser interpretadas da seguinte forma:

- A segunda linha expressa a quantidade de fluxo que no instante $t_k$ atravessa um corte $X$ paralelo a face $F_D$ feito na célula $C$, a uma distância de $F_D$,

dada por $\Delta t[(v_1(m,n,p) + v_1(m-1,n,p))/2]$. Isso assumindo que $v_1$ varia linearmente nessa célula e depende só de $x_1$.

- A terceira linha é uma aproximação da quantidade de fluxo que entrando em $C$ pela célula abaixo dela durante o intervalo $\Delta t$ , vai atingir $F_D$ em $t_k + \Delta t$.

- Analogamente a quarta linha estima a porção do fluxo que passa pelo corte paralelo a $F_D$, definido anteriormente, no instante $t_k$ e sai de $C$ pela célula de cima durante esse intervalo.

- A interpretação das linhas 5 e 6 é análoga às da terceira e da quarta apenas trocando as faces abaixo e acima de C pelas a frente e atrás respectivamente.

Expressamos assim, a advecção para $F_D$ em termos relativos a célula $C$ que são fáceis de adaptar para o caso em que a célula se torna uma célula de borda.

Podemos ajustar a descrição dos termos da equação acima a formulação que vimos empregando dizendo simplesmente:

1. À esquerda de $X$ empregamos velocidades associadas à célula $C$ no instante $t_k$ com o sentido de computar o fluxo que atravessa esse corte nesse momento. Para isso, podemos empregar a metodologia que vimos utilizando neste capítulo e no anterior para determinar uma velocidade $\boldsymbol{v}^{esq}$ que pode ser relativa à célula ou uma para cada face de entrada. Com essa velocidade determinamos o fluxo que atravessa $X$ em $t_k$.

2. À direita de $X$ empregamos velocidades mais próximas possíveis a face $F_D$ para expressar as trocas de fluxo nas faces dessa porção de $C$ durante o intervalo $\Delta t$. No contexto de nossa metodologia isso vai significar escolher uma velocidade $v^{dir}$ para essa região.

A localização de $X$ pode ser computada a partir de dados dos dois tipos. A idéia é aplicar conjuntamente $v^{esq}$ até atingir $X$ e depois $v^{dir}$. Assumindo que todas as componentes de ambos os lados tem o mesmo sinal - que assumiremos positivo- como deve acontecer na imensa maioria das células, então as clássicas figuras constituídas por um retângulo e dois trapézios tendo um vértice comum $P$ - o retângulo pode ser degenerado e nesse caso um dos trapézios vira um triângulo - de onde extraímos as

frações de fluxo se tornam mais complexas pela subdivisão do retângulo, que contem o fluxo que vem de $X$, também em um novo retângulo e mais dois trapézios, cada um deles associado a uma face de entrada a esquerda de $X$.

Essa configuração está representada na figura 7.10. Ela passa ter dois pontos de confluência de linhas separatrizes $P_1$ e $P_2$. Se $D$ é a distância entre $X$ e $F_D$, considerando-se os lados da célula unitários, então $P_1$ é a projeção sobre $F_D$ segundo a direção de $v^{dir}$ do ponto expresso em coordenadas locais de $C$ por $(1-D, 0, 0)$ enquanto $P2$ é o resultado da projeção da $(0, 0, 0)$ sobre $X$ segundo a direção de $v^{esq}$- seja $Q$ o resultado dessa projeção - e em sequência da projeção de $Q$ sobre $F_D$ segundo a direção de $v^{dir}$ (A figura 7.10 ilustra essa situação). Esses pontos tem as seguintes coordenadas:

$P_1 = (1, \frac{D.v_j^{dir}}{v_i^{dir}}, \frac{D.v_k^{dir}}{v_i^{dir}})$

$P_2 = (1, \frac{(1-D).v_j^{esq}}{v_i^{esq}} + \frac{D.v_j^{dir}}{v_i^{dir}}, \frac{(1-D).v_k^{esq}}{v_i^{esq}} + \frac{D.v_k^{dir}}{v_i^{dir}})$

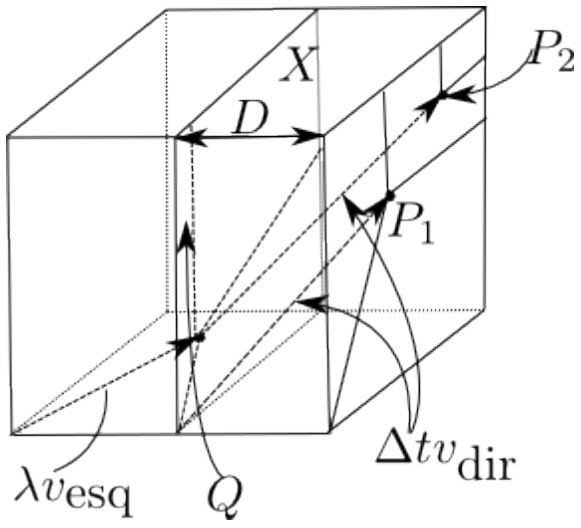



Figura 7.10: Projeção de direção $v^{esq}$ sobre X seguida de outra de direção $v^{dir}$ sobre a face da direita

As informações obtidas dessa figura mais complexa podem ser extraídas de duas partições de $F_D$ com o formato anterior mais simples. A primeira com ponto de confluência em $P_1$ e a segunda com ponto de confluência em $P_2 - P1$. Nesse último caso o corte da borda em $F_D$ deve ser transladado de - $P_1$.A figura 7.11 indica a repartição de $F_s$ considerando as duas projeções citadas acima.

Isso significa, pensando-se em usar dados pré-processados, duas consultas a

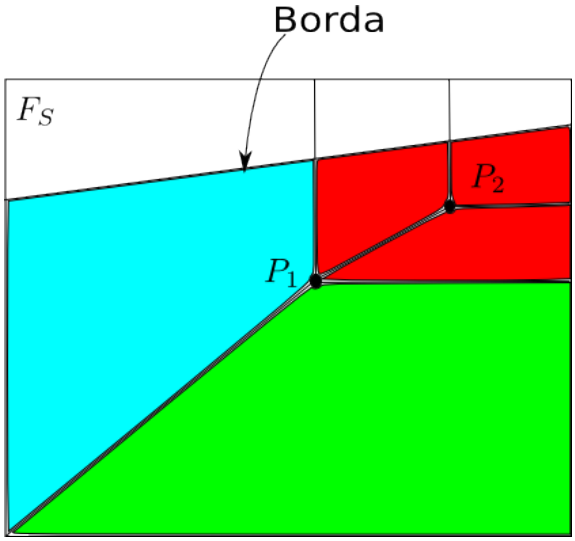



Figura 7.11: Duas subdivisões da face $F_s$ que precisam ser consideradas quando se empregam duas projeções em sequência.

mesma tabela ao invés de uma única.

Finalizando a seção algumas considerações a respeito de se empregar tabelas de resultados pré-computados para determinar áreas atingidas e frações de fluxo transferidas de uma face para outra. Todas as operações desse tipo descritas acima podem ser obtidas a partir do resultado de funções de até 6 variáveis reais restritas ao intervalo [0,1]- representando, por exemplo, cortes nas faces de entrada e saída e o unitário da velocidade. Se consideramos frações múltiplas de 1/32 para todas essas variáveis, tabelar o resultado de uma dessas funções demandaria 1 Gigabyte. Como precisamos de algumas dessas tabelas isso deixa de fazer sentido. É que nesse caso íamos precisar construir uma tabela para tratar cada célula individualmente que tem uma dimensão que queremos evitar para a malha toda, quando falamos em trabalhar em baixa resolução. Precisaríamos, então reduzir a precisão da tabela. Mas se ao invés de 6 tivéssemos apenas 4 variáveis, vamos precisar de apenas 1 Megabyte e nesse caso essa restrição não se aplicaria. Para isso, entretanto, algumas operações tem de ser feitas via mais de uma consulta a tabela, o que aumenta o tempo de resposta podendo torná-lo num caso extremo similar ao que seria necessário quando nenhum pre-cômputo é feito.

No sentido de reduzir o custo computacional da determinação dos índices a serem utilizados numa busca às tabelas, cortes nas arestas são expressos por números inteiros resultantes da divisão da distância a um vértice pela resolução empregada.

Velocidades tem representação dupla. Advecção é feita usando variáveis inteiras. Seu resultado, extraído a partir das tabelas, entretanto, é float ou double. O Sistema de Poisson é, então, resolvido com variáveis desse tipo, o que é importante por razões de convergência. Depois que o $\Delta t$ é computado ele multiplica as velocidades sendo o resultado de novo expresso como inteiros. Observe que num esquema clássico as velocidades são representadas por duas matrizes, se expressando o resultado da advecção calculado usando os dados de uma na outra. Aqui são estamos propondo que essas duas matrizes tenham tipos diferentes.

## 7.3 A área da face externa e sua orientação

Esta seção é dedicada a se obter uma aproximação da área da face externa, que seja de cômputo fácil, bem como a melhor orientação para sua representação planar.

Conforme já mencionado a geometria da face externa não é explicitada inteiramente, sabendo-se apenas que ela é delimitada pelas interseções da massa fluida com as faces regulares determinadas por um procedimento do tipo Marching Cubes.

Como se pretende computar apenas uma velocidade média para a face externa, para efeito do cálculo da vazão que sai por ela e também para o modelo que vamos utilizar para a evolução da superfície, o que importa é a área da projeção sobre um plano ortogonal a essa velocidade[1]. Daí a idéia de estimar apenas a valor da área dessa projeção. Na verdade vamos percorrer o sentido contrário, procurando determinar uma versão planar da borda e estimando a componente da velocidade ortogonal a ela, o que para nossos propósitos é equivalente. Num caso regular - que são a imensa maioria - a aproximação da área dessa versão planar é dada pela norma do gradiente de uma função, que discretizada nas faces da malha, toma em cada uma delas, o valor da área da face que é cortada pela massa fluida. Nos referiremos a essa aproximação por $\nabla A_F$.

Na realidade, nem sempre vamos empregar uma versão da face externa contida num único plano e nem sempre a versão simples, determinada por $\nabla A_F$, é apropriada. Isso porque, devemos levar em conta duas premissas básicas que precisam ser

[1]Ressaltamos que não podemos usar a própria área da triangulação $T$ obtida pelo Marching-Cubes porque a direção do fluxo que atravessa a face externa pode atravessar um triângulo de $T$ entrando na parte fluida e cortar um outro saindo dessa porção para o ar.

atendidas por uma versão planar da face externa:

- O plano $P_{ext}$ contendo essa versão deve separar os vértices da célula de borda contidos na massa fluida dos que estão na porção ar. Isso é o mínimo necessário para que a representação obtida para uma célula de borda se ajuste a versão corrente da função distância a borda empregada para fazê-la evoluir. Entretanto nem sempre é possível atender essa propriedade como no caso ilustrado na figura 7.16, em que uma mesma componente conexa da borda corta duas faces opostas segundo direções principais distintas.

  Essa impossibilidade, entretanto, só pode acontecer se o poligonal -$B'$ - que delimita a interseção da célula com a borda cortar todas as faces da célula, tendo, portanto 6 arestas. Nesse caso a alternativa é particionar a célula, o que pode ser feito introduzindo-se uma face diagonal definida por duas arestas paralelas opostas, de forma a separar em sub-células distintas as faces onde arestas paralelas são cruzadas pela borda.

  As condições para que um vetor $\boldsymbol{v}$ possua um plano ortogonal que separe os vértices da célula em meios distintos visam garantir que

$$\max_{\boldsymbol{v}_C \in V_{Fl}} \langle \boldsymbol{v}, \boldsymbol{v}_C \rangle \leq \min_{\boldsymbol{w}_C \in V_{Ar}} \langle \boldsymbol{v}, \boldsymbol{w}_C \rangle ,$$

  onde $V_{Fl}$ e $V_{Ar}$ são os conjuntos de vértices de $C$ no fluido e no ar, respectivamente. Essas condições são da forma $\langle \boldsymbol{v}, \Delta \rangle \geq 0$ onde $\Delta$ é um vetor diferença entre um vértice na porção ar e outro no fluido. Assim elas apenas especificam o sinal de coordenadas de $\boldsymbol{v}$ ou de somas ou diferenças dessas coordenadas.

- A projeção ortogonal sobre $P_{ext}$ de pelo menos uma triangulação restrita pela poligonal $B'$ e delimitada por ela deve se dar sem sobreposição - isto é, de forma que as projeções dos interiores de seus triângulos sejam disjuntas. Assim, um fluxo com direção ortogonal a $P_{ext}$ corta a triangulação uma vez só, sem sair, re-entrar e sair de novo, por exemplo.

  Para casos em que a poligonal tem 5 arestas pode acontecer que a aproximação dada por $\nabla A_F$ não satisfaça essa propriedade. No entanto, se a condição de separação dada no item anterior for satisfeita essa segunda condição também o será, conforme vai ser mostrado. Por isso vai se fazer uma "projeção"de $\nabla A_F$

sobre o cone de direções em que a condição de separação é satisfeita, para se obter a direção ortogonal a $P_{ext}$.

Para apresentarmos de forma mais específica e formal a abordagem delineada acima é preciso especificar uma série de elementos, o que passamos a fazer a partir de agora. Em primeiro lugar lembramos que um componente conexa da intersecção de uma célula com a superfície da massa fluida, determinada pelo Marching-Cubes é delimitada por um polígono de 3 a 6 arestas. Um ou sete vértices na massa fluida determinam uma poligonal $B'$ com um número de arestas - $n_{B'}$ - igual a três. Dois ou seis, 4 arestas. De uma forma geral K arestas significam K-2 ou 10-K vértices na massa fluida. A exceção é o caso em que os vértices num mesmo meio estão todos numa única célula em que $n_{B'} = 4$. Seja então $F_{i,j}$ a face regular de uma célula de borda $C$ descrita em coordenadas relativas a essa célula por $x_i = j$, $i = 1, 2, 3$ e $j = 0, 1$. Faça, então, $p_{i,j}$ ser o polígono que constitui a intersecção da massa fluida com a face $F_{i,j}$. $p_{i,j}$ pode ser vazio. Considere, então, uma triangulação T qualquer restrita pelas arestas de $B'$ e delimitada por ela. Quando $T$ estiver perfeitamente identificada, $t_k, k = 1, \ldots, K-2$ serão os triângulos de T e $n_k$ o vetor unitário normal ao plano de $t_k$ que aponta para fora do volume fluido. Se for necessário especificar T de forma explicita, vamos escrever $t_{T,k}$ e $n_{T,k}$, respectivamente. Seja, agora, $n_{\text{ext}}$ a solução do problema :

$$\min_{n \in S_2} \sum_{k \in K} \int_{t_k} ||n_k - n||^2 \tag{7.3}$$

$n_{\text{ext}}$ é o vetor unitário de $\sum_{k \in K} A_k n_k$. O plano $P_{\text{ext}}$, ortogonal a $n_{\text{ext}}$ passando pelo centro de massa de $B(T) = \bigcup_{k \in K} t_k$, é uma aproximação do plano que melhor se ajusta a $B(T)$, cuja expressão é mais direta. Determinar a melhor regressão linear de $B(T)$ demandaria encontrar o auto-vetor da matriz de covariância de $B(T)$ associado ao seu menor auto-valor. Essa é uma tarefa muito dispendiosa para ser realizada a cada célula de borda e a cada iteração. Deve-se observar que variando-se a triangulação $T$, o centro de massa muda. Isso, entretanto, será irrelevante para nossos propósitos já que vamos precisar apenas da orientação do plano que representa a borda em $C$ e não de sua localização precisa. Ao contrário, a definição de $n_{\text{ext}}$ é independente da triangulação adotada já que ele pode ser expresso por

$$e_1 \times e_2 + (e_1 + e_2) \times e_3 + \ldots + (e_1 + \ldots + eK - 2) \times e_K = \sum_{i=1,K-2} \sum_{j=i+1,K-1} (e_i \times ej) \tag{7.4}$$

onde $e_i, i = 1, \ldots, K$ são vetores representando as arestas de $B'$ percorridas no sentido positivo. Considere ainda, os operadores $A(.)$,$Pr_{\text{ext}}(.)$ e $Pr_i(.)$ aplicáveis a subconjuntos de $C$ os quais se referem, respectivamente, à área, à projeção ortogonal a $P_{\text{ext}}$ e à projeção ortogonal sobre os planos das faces $F_{i,.}$. Definidos todos esses elementos podemos, finalmente, enunciar e demonstrar um primeiro resultado que é essencial para dar agilidade a um modelo inteiramente baseado nas faces da borda.

**Lema 1** *O vetor* $N = (A(p_{1,0}) - A(p_{1,1}), A(p_{2,0}) - A(p_{2,1}), A(p_{3,0}) - A(p_{3,1}))$ *é ortogonal a* $P_{ext}$ *definido pelo problema de otimização acima. Se o número de arestas de* $B'$ *for* $\leq 4$ *a norma desse vetor é a área -* $A_{Pr_{ext}B(T*)}$ *- da projeção ortogonal de qualquer triangulação* $T^*$ *de B' sobre* $P_{ext}$*. O mesmo ocorre se* $B'$ *for um hexágono e a configuração dos vértices contidos no fluido ou no ar admitir um plano separador. Em ambos os casos existe um plano paralelo a* $P_{ext}$ *que separa os vértices de* $C$ *no fluido e no ar.*

**Demonstração:**

Seja $P_{\text{ort}}$ a projeção ortogonal sobre um plano $P$ e $V$ um volume limitado cuja superfície é diferenciável em quase todo ponto. Nesse caso, a seguinte identidade é verdadeira:

$$\int_{\partial V} P_{\text{ort}}(x) sign(< n_x, n_P >) dx = \int_{\partial V*} < n_x, n_P > dx = 0, \tag{7.5}$$

onde $\partial V$ é a superfície de $V$ $\partial V*$ a porção diferenciável dessa superfície, $n$ a normal a $\partial V*$ que aponta para fora(ou para dentro) de $V$ e $n_P$ um vetor unitário normal a $P$.
Agora se $V$ for o volume fluido em $C$ e $P$ o plano de uma das faces $F_{i,j}$ então essa identidade toma a forma:

$$A(p_{i,1}) - A(p_{i,0}) + \sum_{k \in K} A(Pr_{\text{i}}(t_k)).sign(< n_k, e_i >) = 0 \Longrightarrow$$
$$A(p_{i,0}) - A(p_{i,1}) = \sum_{k \in K} A(Pr_{\text{i}}(t_k)).sign(< n_k, e_i >) =$$
$$\sum_{k \in K} A(t_k).(< n_k, e_i >) =< \sum_{k \in K} A(t_k).n_k, e_i >)$$

Considerando o resultado acima para i=1,2,3 obtemos: $N = \sum_{k \in K} A(t_k).n_k$. Portanto $N$ é um múltiplo $n_{\text{ext}}$ e o quadrado de sua norma $\|N\|^2$ é dado por:

$$< N, N >=< \sum_{k \in K} A(t_k).n_k, \|N\| n_{\text{ext}} >= \|N\| \sum_{k \in K} A(t_k) < n_k, n_{\text{ext}} >=$$
$$\|N\| \sum_{k \in K} A(Pr_{\text{ext}}(t_k)) sign(< n_k, n_{\text{ext}} >)$$

o que implica,

$$\|N\| = \sum_{k \in K} A(Pr_{\text{ext}}(t_k)) sign(< n_k, n_{\text{ext}} >)$$

Para provar, portanto, que existe uma triangulação $T^*$ tal que $\|N\| = A(Pr_{\text{ext}}(B(T^*)))$ basta mostrar que $\text{sign}(< n_{T*,k}, n_{\text{ext}} >) = 1$ ou, equivalentemente que $< n_{T*,k}, N >$ é positivo. Observe que ao fazer isso mostramos que $Pr_{\text{ext}}(t_{T*,k})$ e $Pr_{\text{ext}}(t_{T*,j}), j \neq k$ tem interiores disjuntos.

O caso em que $n_{B'} = 3$ isso é verdade já que $T^*$ consiste de um único triângulo. O plano desse triângulo é obviamente um separador do único vértice que está num dado meio dos demais vértices.

Os dois casos em que $n_{B'} = 4$ são representados nas figuras 7.12 e 7.13 respectivamente.

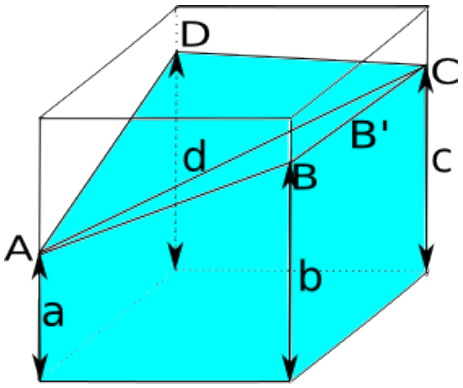



Figura 7.12: Primeira das duas possibilidades para 8 casos em que $n_{B'} = 4$. A linha tracejada no interior da face externa representa a aresta interior da triangulação estrita por $B'$

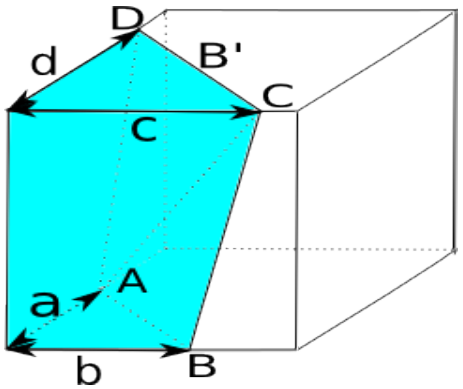



Figura 7.13: Segunda das duas possibilidades para 8 casos em que $n_{B'} = 4$.

Nessas figuras se supõe um sistema de referência com origem no vértice inferior esquerdo à frente e com eixos coordenados definidos pelos vetores que vão desse vértice aos vizinhos na ordem horizontal($e_x$), frontal($e_y$) e vertical($e_z$). Desse modo os eixos formam um triedo tri-ortogonal positivo. x,y,z vão ser as coordenadas relativas a esses eixos na ordem em foram apresentados. Nessas figuras representaremos por uma letra maiúscula o ponto de corte de uma aresta pela borda e com a mesma letra, só que minúscula, para indicar a coordenada não necessariamente inteira desse ponto.

Assim na figura 7.12 os pontos de corte serão A,B, C e D que tem por coordenadas, no sistema adotado $(0, a, 0), (1, b, 0), (1, c, 1)$ e $(0, d, 1)$, respectivamente. Nesse caso, os triângulos que contem a diagonal A-C tem (2 x área) e orientação representados pelos vetores $(B-A)\times(C-A)$ e $(C-A)\times(D-A)$ que são $n_{t1} = (a-b, b-c, 1)$ e $N_{t2} = (d-c, a-d, 1)$. O vetor $N$, que é a soma desses dois será, portanto, $(a+d-(b+c), a+b-(d+c), 2)$. Assim, o produto escalar $< n_{t1}, N >$ vale $(a-b)^2+(b-c)^2+2+\{(a-b).(d-c)+(b-c).(d-a)\}$. As duas primeiras parcelas são não negativas e portanto a soma das três primeiras vale pelo menos 2 e para que ele seja exatamente 2, $a = b = c$. Cada uma das duas parcelas restantes é o produto de dois números em $[-1, 1]$ e portanto cada uma delas é maior que -1. Daí segue que sua soma é $\geq -2$ e para que elas somem exatamente -2, $\|a - b\|$. Assim a soma das três primeiras não pode ser 2 e simultaneamente a das duas últimas -2, resultando que $\langle n_{t1}, N\rangle > 0$. Raciocínio idêntico mostra que $\langle n_{t2}, N\rangle$ também é positivo. Além disso da expressão de $N = (a + d - (b + c), a + b - (d + c), 2)$, verificamos que $|N_x| + |N_y| \leq N_z$ que a condição para que os 4 vértices da face de cima que estão no ar, sejam separados dos da de baixo, contidas no fluido, por um plano ortogonal a $N$.

No caso da figura 7.13, adotando-se o mesmo sistema de referência os pontos em que as arestas cortam a borda são dados por $A = (0, 0, a)$, $B = (b, 0, 0)$,$C = (c, 1, 0)$ e $D = (0, 1, d)$ e os vetores que representam área e orientação dos triângulos $ABC(t_1)$ e $ACD(t_2)$ são $(a, a(b - c), b)$ e $(d, c(a - d), c)$. $N$ será então $(a + d, a(b - c) + c(a - d) = ab - cd, b + c)$.

O produto $< n_{t1}, N >$ será então dado pela soma de parcelas positivas exceto $ac(b - c)(a - d) = a^2(c(b - c) - ad(c(b - c)$ cujo sinal é indeterminado. Como $\|c(b - c)\| \leq 1$ o módulo dessa última parcela é $\leq a^2 + ad = n_{t1,x}N_x$. Assim mesmo que ela seja negativa ela será compensada pelo produto $n_{t1,x}N_x$. Portanto $< n_{t1}, N >$ será $\geq 0$. Como na expressão de $< n_{t1}, N >$ ainda restam elementos $\geq 0$ que só podem se anular se $b = c = 0$, o que faria a face externa de $C$ ser igual a uma regular ou mesmo uma aresta, esse produto escalar resulta positivo. Da expressão $N = (a + d, ab - cd, b + c)$ verificamos que $N_x \geq 0$, $N_z \geq 0$ e $|N_y| \leq \min N_x, N_z$ que são as condições para que $P_{ext}$ separe os dois vértices da

aresta $(0,.,0)$ que estão na porção fluido dos demais que estão no ar.

Um caso menos óbvio diz respeito ao caso em que $n_B = 6$ e a configuração dos vértices no fluido e no ar admite um separador planar. Nesse caso dos 4 vértices em cada meio, um é adjacente aos outros três. Esse caso via uma série de rotações da célula pode assumir a configuração representada na figura 7.14 onde os vértices de $B'$ são $A = (0, a, 0)$, $B = (b, 0, 0)$, $C = (1, 0, c)$, $D = (1, d, 1)$, $E = (e, 1, 1)$ e $f = (0, 1, f)$.

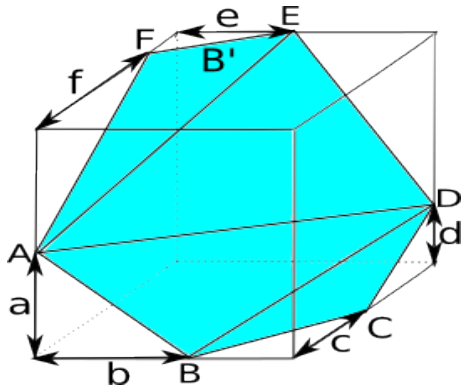



Figura 7.14: Corte quando $n_{B'} = 6$ e existe um separador planar dos vértices na massa fluida e no ar.

Os vetores representando o dobro da área e a orientação dos triângulos ABC,ACD,ADE e AEF tem as seguintes coordenadas, respectivamente:

$$(-ac, -bc, a(1-b))$$
$$(-a(1-c) - cd, c-1, d)$$
$$(d-1, e-1, 1-a-e(d-a) \geq 1-a-(d-a) = 1-d)$$
$$((a-1)(1-f), -ef, e(1-a))$$

Como todas as variáveis acima são números em [0,1] resulta que todas as coordenadas x e y desses 4 vetores são negativas e as 4 coordenadas z são positivas. O mesmo se verifica portanto, para a soma deles - que é $N$. Observe então que $N_x, N_y \leq 0$ e $N_z \geq 0$ atendendo-se portanto, as condições de separabilidade do conjunto formado por (0,0,1) e seus vértices vizinhos, supostos estarem no ar, em relação aos demais vértices de $C$. O produto escalar de $N$ com qualquer desses vetores vai ser, então, uma soma de produtos de dois números com o mesmo sinal. Por conseguinte o produto escalar é não negativo. Ele é de fato positivo porque como três vértices de $B'$ não podem ser colineares, a área

de qualquer triângulo com vértices nos de $B'$ é não nula. Portanto $n_{ti}$ terá pelo menos uma coordenada não nula. Como essa coordenada em $N$ terá o mesmo sinal e módulo maior, o produto das duas será positivo o mesmo se podendo dizer de $< N, n_{ti} >$. Assim em todos casos mostramos que existe uma triangulação T* tal que todos os triângulos se projetam sem sobreposição sobre $P_{ext}$. Como: $(i)$ A definição desse plano não depende da triangulação, $(ii)$ Qualquer triângulo definido pelos vértices de $B'$ pertence a uma triangulação que pode pode ser feita coincidir com T* por rotação e translação de vértices na mesma aresta da célula. $(iii)$ As variáveis a-f são quaisquer em [0,1]. Então o resultado vale para qualquer triangulação de $B'$ encerrando a prova.

◇

Para efeito de se atender às duas premissas relativas a orientação da face externa, o caso em que $n_{B'} = 5$ é aquele em que o processo de determinação de $P_{ext}$ depende da posição dos pontos de corte das arestas pela borda. Tal como se pode ver na figura 7.15, esse caso se caracteriza por $B'$ ter duas arestas ($E_1$ e $E_2$ (na figura elas são AE e AB) adjacentes num vértice $v$ (E na figura) as quais ligam cada uma, pontos em duas arestas paralelas da célula $C$.

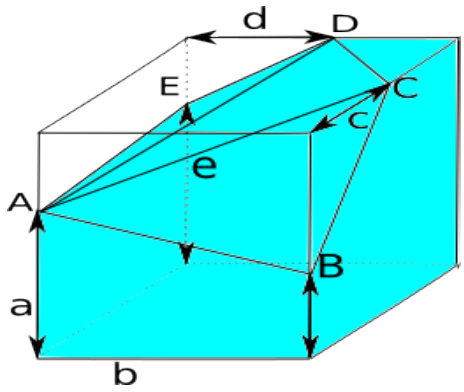



Figura 7.15: Corte quando $n_{B'} = 5$.

Considere então as faces de $C$ ortogonais à aresta -$e_v$- de $C$ contendo $v$. Uma delas - $F_0$ - não será cortada pela borda, tendo, portanto, todos os vértices no mesmo meio. A outra - $F_1$ - terá apenas um vértice(v*-(1,1,1)na figura)nesse meio e os outros três - $v_0$ e $v_2$, adjacentes a v*, e $v_1$ - no outro meio. $v* \in e*$, a aresta oposta a $e_v$. Seja ainda w* - que está em - $F_0$ - o vértice oposto a v* e considere o sistema de referência local com origem em w* e eixos coordenados definidos pelos

unitários de $v*-v_0$,$v_1-w*$ e $v*-v_2$. Na figura esse sistema é o próprio sistema canônico.

Se $F_0$ está no ar e o plano ortogonal a $N'$ separa os vértices de $C$ nos dois meios,então temos:

1. $\langle N', v*\rangle > \langle N', v_i\rangle \rightarrow N'_i > 0, i = 0, 2.$

2. $\langle N', v_1\rangle < \langle N', w*\rangle \rightarrow N'_1 < 0$. Em função de 1 verificamos que:

   (a) O vértice de $F_1$ que está no fluido, onde $\langle N', .\rangle$ é maior é ou $v_0$ ou $v_2$.

   (b) O vértice de $F_0$ onde $\langle N', .\rangle$ é menor é $w*$.

   Isso significa que para garantir a condição de separabilidade desejada precisamos ainda impor que:

3. $\langle N', v_i\rangle < \langle N', w*\rangle \rightarrow (N'_i + N'_1) < 0, i = 0, 2.$

Definidas essas condições podemos enunciar o resultado abaixo relativo a obtenção de um plano onde uma triangulação T* restrita por $B'$ se projete sem sobreposições. Qualquer que seja T* esse plano existe - por exemplo, os planos paralelos a face $F_0$. A questão é escolhê-lo de forma a aproximar $B(T*)$ e satisfazer ao mesmo tempo a condição de separabilidade.

**Lema 2** *Se $n_{B'}$ for 5 e se $N$ pertencer ao octante $Oc_0$ definido pelas condições (1) e (2) acima, então existe uma triangulação $T^*$ de $B'$ tal que $T^*$ se projeta ortogonalmente sobre $P_{ext}$ sem sobreposição. Essa propriedade se mantém se $N$ é substituído por sua projeção sobre $Oc_0$.*

**Demonstração:** Na configuração representada na figura 7.15 - que pode ser sempre obtida via rotações de $C$ - os vértices de $B'$ são $A = (0, a, 0)$, $B = (1, b, 0)$, $C = (1, 1, c)$, $D = (d, 1, 1)$ e $E = (0, e, 1)$. Os vetores apontando para o meio ar, que representam a orientação e 2x área dos triângulos adjacentes a $(A = v*)$ - São eles $t_1 = ABC, t_2 = ACD$ e $t_3 = ADE$ - tem as seguintes coordenadas:

$$n_0 = (c(b-a), -c, 1-b)$$
$$n_1 = ((1-a)(1-c), -1+cd, (1-a)(1-d))$$
$$n_2 = ((1-e), -d, d(e-a))$$

N, portanto, vale $(2-(a+c+e)+bc, -1-(c+d)+dc, 2-(a+d+b)+de)$. A segunda coordenada é sempre negativa indicando que a condição 2 acima é sempre satisfeita. Para que a condição 1 também seja aceita devemos ter, então, que

$$2+bc > (a+c+e) \text{ e} \tag{7.6}$$
$$2+de > (a+d+b).$$

Para verificar que atendidas essas condições $< N, n_0 > \geq 0$ observe que levando em conta as desigualdades dadas em 7.6, a única possibilidade de coordenadas idênticas desses dois vetores terem sinais diferentes, são as primeiras se $b < a$ e as terceiras se $e < a$. Se isso não ocorre, $< N, n_0 >$ será a soma de três valores não-negativos. Para que o valor desse produto interno seja nulo é preciso que simultâneamente $b = 1$ e $c = 0$ o que indica coincidência de vértices de B', que não teria, então, 5 vértices. Em caso contrário considere a expressão completa de $< N, n_0 >$:

$$\{(2-(a+c+e)+bc)c(b-a)\}+\{c.(1+c+d-dc)\}+\{2-(a+d+b)+de\}(1-b). \tag{7.7}$$

Nessa expressão os termos em d e e são lineares e portanto assumirão valores mínimos quando essas variáveis valerem 0 ou 1 conforme o sinal do coeficiente que as multiplica. Assim se $-c(b-a)+d(1-b) > 0$, $e = 0$ o que sempre acontece pois $b < a$.

Fazendo $e = 0$, a condição em $d$ se torna $c(1-c)-(1-b) > 0 \rightarrow d = 0$, caso contrário $d = 1$. Se $d = 0$ e $e = 0$ a expressão 7.7 se torna:

$$\{(2-(a+c)+bc)c(b-a)\}+\{c.(1+c)\}+\{2-(a+b)\}(1-b). \tag{7.8}$$

Como $b < a$ temos que $(2-(a+c)+bc) < 2-(a+c)+ac = 1+(1-a)(1-c)$ e o último termo é maior que $(2-2a)(1-b)$.

Juntando tudo, 7.8 é maior que {1+(1-a)(1-c)}c(b-a)+{c.(1+c)}+2(1-a)(1-b).

Essa expressão è linear em $b$ e portanto tem valores extremos para $b = 0$ ou $b = 1$. Esse último caso pode ser descartado pois contradiz $b < a$. Em relação ao primeiro temos:

$$-\{1+(1-a)(1-c)\}ca+\{c.(1+c)\}+2(1-a) \geq$$
$$-(1-a)(1-c)ca+c^2+2(1-a) = (1-a)(2-(1-c)ca)+c^2 \geq 0.$$

Se $d = 1$ então a segunda parcela de 7.7 se torna $2c$. Como $(a+c+e) > bc$, a primeira parcela é maior que $2(b-a)c \geq -2c$ e assim o termo negativo é compensado pela segunda parcela. Como algumas desigualdades empregadas no desenvolvimento acima são estritas, completa-se desse modo a prova de que se a condição 1 for satisfeita $< N, n_0 >$ ¿ 0. De forma análoga mostramos que $< N, n_2 >< 0$.

Se $e \geq a$ esse produto escalar é a soma de três valores não negativos que não podem ser todas nulas.

Se, ao contrário, $e < a$, observando que a expressão completa $< N, n_2 >$ pode ser obtida a partir da de $< N, n_0 >$ trocando $b$ por $e$ e $c$ por $d$, podemos afirmar que fazendo essas trocas no raciocínio acima, obtemos para $n_2$ o resultado já obtido para $n_0$.

Finalmente devemos observar que se as condições expressas em 7.6 forem satisfeitas então as coordenadas de $N$ e $n_1$ tem todas o mesmo sinal e como as segundas coordenadas desses vetores são não nulas seu produto escalar será estritamente positivo.

Observamos agora que se $N_0$ for negativo $N_2$ será obrigatoriamente positivo. Caso contrário, como $a, c$ e $d \leq 1$ teríamos simultaneamente:

$$2 + bc < (a + c + e) \Rightarrow 1 + bc < (c + e) \Rightarrow (1 - e) < c(1 - b) \leq 1 - b \text{ e}$$
$$2 + de \leq (a + d + b) \Rightarrow 1 + de \leq (d + b) \Rightarrow (1 - b) \leq d(1 - e) \leq 1 - e$$

gerando-se uma contradição.

Da mesma forma $N_2$ negativo acarreta $N_0$ positivo. Das inequações de 7.9 se extrai também que $N_i = 0 \Rightarrow N_{2-i} \geq 0, i = 0, 2$. Entretanto, observe que a anulação das duas coordenadas ao mesmo tempo implica em: $i$Caso $c = 0$ ou $d = 0$: teremos, conforme o caso, simultaneamente, $c = 0$ e $b = 1$ ou $c = 0$ e $e = 1$ ambos configurando coincidência de vértices de B, situação que podemos desconsiderar. $ii$Caso $c > 0$ e $d > 0$: o produto escalar$< N, n_i >, i = 0, 2$ será dado pelo produto das segundas coordenadas que serão ambas positivas. Portanto, esta situação não constitui problema e podemos admitir no desenvolvimento que será feito a seguir que apenas uma das coordenadas $N_2$ ou $N_0$ pode se anular.

Suponha, agora que para satisfazer a condição 1, se $N_i, i = 0, 2$ for $\leq 0$, seu valor seja substituido por um positivo arbitràriamente pequeno($\epsilon$). Nesse caso $< N, n_i >$ será positivo.Para ver isso no caso em que $i = 0$ observe que se $N_0 \leq 0$ então $N_2 > 0$ e $b < 1$ pois $b = 1 \Rightarrow N_1 = (2 - (a + e)) \geq 0$, sendo estritamente positivo, exceto se $a = e = 1$, situação que dado que também $b = 1$ acaba acarretando coincidência de vértices de $B'$.

Assim, a última parcela de 7.7, $N_2(1 - b) > 0$. Como a segunda é sempre não negativa e o módulo da primeira foi reduzido a $\epsilon$, $< N, n_0 >$ é positivo. Menos imediato é mostrar que $< N, n_{2-i} >$ também será positivo.

Suponha que $N_2 \leq 0$ e que substituimos esse valor por $\epsilon$. Então, como nesse caso $N_0$ será positivo, $< N, n_0 >$ só poderia ser não positivo se $b \leq a$ devendo se ter ainda que:

1. Caso $c > 0$: $(2 - (a + c + e) + bc)(a - b) \geq .(1 + c + d - dc)$.Entretanto,o lado

esquerdo dessa desigualdade ou é nulo ou é $< (2-a)(a) \leq 1$ enquanto o lado direito é claramente $\geq 1$ Portanto ela é sempre falsa.

2. Caso $c = 0$ Nesse caso $< N, n_0 >= \epsilon$ se $b < 1$. $c = 0 e b = 1$ significa que $B'$.

Desse modo, o fato de eliminar coordenadas negativas de N para satisfazer as propriedades de separação dos vértices em meios distintos não elimina o fato dos triângulos de T* se projetarem sem sobreposição.

Resta ainda analisar, o que ocorre quando para atender a condição 3, reduzimos o valor de $N_0$ ou $N_2$ de modo que ele fique menor em módulo que $N_1$. A idéia é nesse caso fazê-los iguais a $|N_1| - \epsilon$. Se fazemos isso com $N_0$ então $< N, n_0 >> (c(b-a)+c)(1+c+d-dc) - c(b-a).\epsilon) + \max \epsilon, N_2(1-b) > 0$ se $c > 0$. Se $c = 0$ a primeira parcela se anula, mas nesse caso $(N_0 > |N_1|) \Rightarrow (1 > a + e + d)$, o que faz com que $N_2$ seja positivo. Então como $b < 1$, senão teremos coincidência de vértices, a segunda parcela será $> 0$. Se ao invés de reduzir $N_0$, fazemos isso com $N_2$, então:

$$< N, n_0 >= \max\{\epsilon, N_0\}c(b-a) + ((1-b)+c)(1+c+d-dc) - \epsilon(1-b). \tag{7.9}$$

Se $b \geq a$ então a primeira parcela desse produto escalar é não negativa e como a segunda é estritamente positiva se não tivermos simultaneamente $c = 0$ e $b = 1$, caso que podemos descartar,o resultado será positivo. Assuma agora que $b < a$. Se a primeira parcela é em módulo $\leq \epsilon$, a positividade da segunda vai prevalecer. Se mantivemos o valor de $N_0$ é por que ele era menor que $|N_1|$ e nesse caso temos $N_0.c(b-a) > (1+c+d-dc).c(b-a) > (1+c+d-dc).\{c(b-1)-c\}$ que é o simétrico do segundo termo de 7.9. Desse modo $< N, n_0 >$ continuará positivo em qualquer dos casos.

Se tivermos que reduzir ao mesmo tempo $N_0$ e $N_2$ então $< N, n_0 >$ se tornará $(1+c+d-dc)(c(1-(b-a))+(1-b)) > 0$ a menos que $c = 0$ e $b = 1$, que mais uma vez descartamos. Resultados obtidos para $< N, n_0 >$ podem de forma análoga serem obtidos $< N, n_2 >$. Ressaltamos que todas essas transformações não afetam a positividade de $< N, n_1 >$ dado que ele continua sendo a soma de produtos de coordenadas que tem o mesmo sinal ou produtos que valem $\epsilon$. Portanto, a projeção sobre o cone de direções definido pelas condições 1,2 e 3, não afeta a separabilidade da projeção dos triângulos de T* sobre $P_{ext}$.

Fica faltando apenas considerar o caso em que $N_{B'} = 6$ e não é possível encontrar um plano separador para os conjuntos de vértices da cela em meios distintos.Por meio de rotações essa situação pode sempre ser representada pela configuração apresentada na

figura 7.16 onde $A = (0, a, 0), B = (b, 0, 0), C = (1, 0, c), D = (1, 1, d), E = (e, 1, 1) eF = (0, f, 1)$.

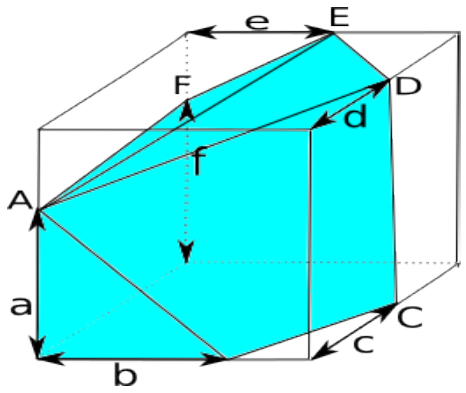



Figura 7.16: Caso em que não é possível separar por um plano os vértices da célula que estão na massa fluida daqueles que estão no ar.

Nesse caso é preciso particionar a célula $C$ em duas semi-células obtidas pelo corte de $C$ pelo plano definido por duas arestas paralelas opostas. Esse corte deve ser definido de forma que as arestas $AF$ e $CD$- aquelas que ligam pontos em arestas paralelas de uma face de $C$- fiquem em sub-células distintas. Por exemplo, o corte definido pelo conjunto de vértices de $C$, $\{(1, 0, 0), (1, 0, 1), (0, 1, 1) e (0, 1, 0)\}$ satisfaz essa propriedade.Esse corte introduz um novo vértice $G \in EF$ na subcélula contendo a origem teremos então os vértices $A, B, C, GeF$ A separação dos vértices dessa semi-célula se tornou factível, senão vejamos. Se $(0, 0, 0)$ está na porção fluida, $(0, 0, 1)$ e $(1, 0, 1)$ também estarão. Os vértices da sub-célula na porção ar são $(1, 0, 0), (0, 1, 0)$ e $(0, 1, 1)$. Esses dois conjuntos são separáveis por um plano ortogonal a um vetor $N$ com as seguintes propriedades$N_0 > 0$,$N_2 < 0$ e $N_1 > \max N_0, -N_2$. Resultado análogo pode ser obtido para os vértices da outra semi-célula.Procedimento similar ao desenvolvido para o caso de $n_{B'} = 5$ garante a não sobre-posição dos triângulos que representam a borda em cada sub-célula quando projetados sobre o plano $P_{\text{ext}}$ obtido para ela.

Então o resultado vale para qualquer triangulação de $B'$ encerrando a prova. $\diamond$

Em função do verificado acima se pode pensar em aproximar $A(P_{\text{ext}} \bigcap C$ por $\|N\|$. Essa aproximação se torna exata se $B'$ for planar. De toda forma qualquer que seja a triangulação, não exatamente T*, $\|N\|$ representa a área que se deve multiplicar uma velocidade com a direção $n_{\text{ext}}$ para computar a vazão que deixa massa fluida em $C$ por toda superfície de $T*$.

## 7.4 A evolução da borda

Dispondo da componente da velocidade na face externa que é normal a ela podemos pensar num sistema alternativo para a evolução da borda que seja mais resistente às situações de retenção/perda de volume determinadas por um sistema clássico usando abordagens do tipo level sets/fast-marching. Num sistema desse tipo é necessário definir na região da borda velocidades nos vértices (3 componentes) para fazer a advecção da função distância a borda. Essas 3 componentes serão substituídas por um valor por aresta que indica o ponto onde ela deve ser cortada. Como o número de arestas é menor que 3 vezes o de vértices não há acréscimo na memória necessária para implementar essa nova alternativa. Como o dado relativo a uma aresta é computado a cada iteração sem que seja preciso guardar o valor anterior para fazê-lo, tal como acontece com as velocidades de vértices no esquema clássico, e uma vez que eles tem praticamente a mesma disponibilidade de memória pode-se pensar em passar de um a outro esquema quando for necessário e reverter ao sistema original num momento posterior.

Parte do procedimento empregado na evolução também pode ser feito mais simples: A advecção da função distância é realizada via a interpolação - usualmente trilinear - dos valores que ela assume nos 8 vértices de uma célula. Isso também determina que se tenha que estimar velocidades nos vértices das células de borda mesmo que as arestas adjacentes a ele não sejam cortadas pela borda. Os dados relativos a essa função - valor estimado ou simplesmente o sinal - computados num vértice pelo esquema que será proposto aqui são obtidos comparando os valores relativos às até 6 arestas adjacentes ao vértice que são cortadas pela borda. Em vértices não adjacentes a arestas cortadas pela borda não é necessário avaliar a função de distância. Na verdade, se precisarmos apenas da informação de que um vértice está dentro ou fora da massa fluida, então essa informação pode ser obtida diretamente dos dados das arestas e nem precisamos de uma estrutura para descrever a função distância computada nos vértices.

Em outros pontos há uma completa equivalência computacional. Por exemplo, procedimentos para o cômputo de velocidades, num vértice de uma célula de borda, empregam para cada componente, dados das até 4 faces ortogonais a essa componente que são adjacentes a ele e interceptam a massa fluida. O ponto de corte de

uma aresta é determinado considerando-se dados que vem das também 4 células adjacentes a ela.

Em linhas gerais pode-se dizer comparando-se as complexidades computacionais dos dois métodos que promover a evolução da superfície ao longo das arestas de uma célula de borda ou das adjacentes a ela, na nova metodologia, é mais complexo que qualquer dos procedimentos empregados pela abordagem padrão. Entretanto, esse procedimento substitui vários que provêm a evolução e o próprio traçado da borda na abordagem clássica. Em vista disso, essa maior complexidade pode ser compensada.

Entretanto, em relação à robustez da proposta que será descrita aqui devemos fazer a seguinte consideração. O fato de tanto a velocidade na face externa como também, conforme veremos, a coordenada -$X(e)$- do ponto de corte da borda em uma aresta $e$, serem obtidos por quocientes não favorece a estabilidade numérica. Não tanto no que se refere a $X(e)$, em cujo cômputo, se inclui um processo de minimização que elimina valores altos. Mas fluxos e áreas da face externa muito pequenos, entretanto, aumentam a possibilidade de erro no cômputo da velocidade normal a essa face encontrada ao final da advecção.

Para que o esquema proposto possa funcionar bem é necessário estabelecer normas que o caracterizam e diferenciam de uma abordagem comum via level-sets.

Em primeiro lugar vai se mudar o cômputo da distância à massa fluida empregada. Em lugar da distância euclidiana vai se considerar uma distância L1 ou Manhattan que é dada pelo comprimento do menor caminho entre dois pontos constituído de arestas na grade.

Assumida essa métrica, observamos ainda que vamos estar interessados apenas em mensurar a distância entre um vértice($v$) da malha e um conjunto($X$) determinado pela interseção de um plano $P$ -por exemplo, paralelo ao da representação planar de uma face externa - com um octante do qual $v$ pertence.

Assumindo ainda, que $P$ é cortado por uma aresta adjacente a $v$ então a distância L1 entre $v$ e $X$ é dada pelo comprimento do menor segmento que vai do ponto ao conjunto e está contido numa aresta. Em termos da metodologia que vamos introduzir aqui, isso significa que para esses vértices - que são adjacentes a arestas da malha intersectando a borda deslocada - se considerarmos distâncias apenas ao

longo de arestas, ainda assim vamos estar computando uma distância ponto conjunto determinada por uma métrica.

Sejam (conforme ilustrados nas figuras 7.17 e 7.18):

1. $A_{\text{ext}}$, a área da face externa $F_{\text{ext}}$ de uma célula de borda $C$.
2. $n_{\text{ext}} = (n_x, n_y, n_z)$, o vetor unitário da componente normal a $F_{\text{ext}}$ da velocidade computada nessa face - $v_{\text{ext}}$.
3. $P_{\text{ext}} = \{V | \phi_{\text{ext}}(V) \triangleq (\langle n_{\text{ext}}, V \rangle - d_{\text{ext}}) = 0\}$, o plano da versão planar de $F_{\text{ext}}$.
4. $D_{\text{ext}}$ o deslocamento dessa face determinado por $v_{\text{ext}}\Delta t$.
5. $P_\Delta = \{V | \phi_\Delta(V) \triangleq < n_{\text{ext}}, V > -d_\Delta = 0\}$, o plano que contem a face externa deslocada por $D_{\text{ext}}$.

A busca dos novos pontos de corte nas arestas de $C$ ou adjacentes a ela, será orientada, em primeiro lugar pela identificação do octante $Oc(C)$ determinado por:

1. O vértice $V_o(C)$ de $C$ mais longe de $F_{\text{ext}}$ entre os que estão do lado contrário ao indicado por $n_{\text{ext}}$.
2. Os semi-eixos coordenados cujo sentido é determinado pela componente respectiva de $n_{\text{ext}}$.

Considere agora, o sistema de coordenadas com origem em $V_o(C)$ onde o octante $Oc(C)$ passa a ser o octante positivo e os vetores da base tem o tamanho de uma aresta. Se $C$ é a célula $(i, j, k)$ então as coordenadas de $V_o(C)$ são dadas por:

$$(i + \frac{1}{2}(1 - sign(n_x)), j + \frac{1}{2}(1 - sign(n_y)), k + \frac{1}{2}(1 - sign(n_z))) \tag{7.10}$$

Nesse novo sistema o vetor que representa $n_{\text{ext}}$, que vamos grafar $|n|_{\text{ext}}$, tem todas as coordenadas não negativas e $C$ se torna o cubo unitário.

Definido esse sistema, as arestas da grade contidas em $Oc(C)$ que são adjacentes a $C$, embora não pertençam a ela, serão analisadas em três grupos, cada um deles definido pela soma das coordenadas dos vértices de $C$ de onde saem. Conforme essa soma seja 1,2 ou 3 faremos uma análise diferenciada. Da origem - soma 0 - não saem

arestas que se enquadrem nas condições acima. As arestas de $C$ compõem um outro grupo.

No texto a seguir $w$ vai representar uma das coordenadas $x, y$ ou $z$. Suponha ainda, inicialmente, que o deslocamento da borda por $D_{\text{ext}}$ leva a interseções apenas com arestas vizinhas às que são cortadas presentemente. Essa condição é verdadeira se $\|D_{\text{ext}}\| < \frac{\sqrt{2}}{2}h$, sendo $h$, o lado da célula daqui por diante assumido igual a 1, considerando a normalização feita pelo sistema de referência que estamos empregando. Observe que essa limitação é só uma pouco mais restrita que a condição CFL que estamos empregando, que pode ser expressa por $\|D_{\text{ext}}\| < 1$. Além disso se a condição acima não for verdadeira o processo pode ser sempre repartido em estágios nas quais ela se verifica.

A versão planar da face externa de $C$ e sua versão deslocada por $D_{\text{ext}}$ definem um prisma - $PRI_{\Delta}$ - cujas bases são ortogonais às outras faces. Dessas outras faces vamos considerar aquelas definidas por $D_{\text{ext}}$ e pela aresta - $a_w$ - da versão planar de $F_{\text{ext}}$ que está contida no plano $w = 1$. Se essa aresta existir faça $F_w$, ser a face que ela define e $P_w$, o plano que contem essa face. Todos esses elementos são representados na figura 7.17 abaixo.

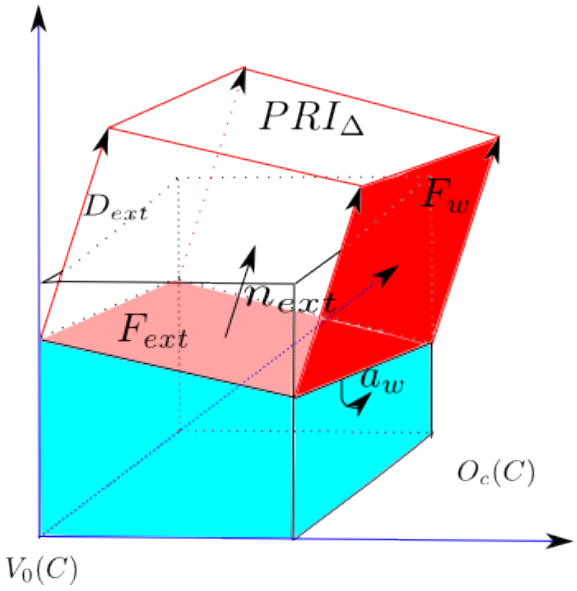



Figura 7.17: Prisma $PRI_{\Delta}$.

Além de restringi-la ao octante $Oc(C)$ vamos manter a evolução da borda dentro de um politopo $(Q)$ contendo a origem e determinado por um conjunto de planos. Os planos que delimitam esse politopo dependem do grupo a que a aresta, sendo

cortada pela face externa, em questão pertença. Eles serão escolhidos da seguinte forma:

1. Se a aresta pertence a $C$ então Q é o semi-espaço-$SE_\Delta$- contendo a origem determinado por $P_\Delta$.

2. Se a aresta tem coordenada livre $w$ e as outras ($u$ e $v$) iguais a zero - isto é pertence ao grupo 1 - então além de $P_\Delta$, Q é restrito por um plano $w = W$, sendo $W$ computado da seguinte maneira:

   (a) Se a componente da velocidade de direção $w$ está definida na face de $C$ contida em $w = 1$, ($F_{w=1}$), então:

   $$W = \max\left\{\left(\Delta t - \frac{\Phi_{ext}(V_e)}{v_{ext}}\right) v_w(F_{w=1})\right\} = \max\left\{-\phi_\Delta(V_e)\frac{v_w(F_{w=1})}{v_{ext}}\right\}. \tag{7.11}$$

   Observamos que nesse caso $\phi_\Delta(V_e) < 0$. O operador max trata do caso em que $v_w(F_{w=1}) < 0$.

   (b) Em caso contrário, $P_{ext}$ corta $e'$, a continuação de $e$ em $C$. Nesse caso pelo menos $v(F_{w=0})$ estará definida e podemos adotá-la para o deslocamento na direção de $w$:

   $$\max\{0, -\phi_\Delta(V_e)\} \max\left\{0, \frac{v(F_{w=0})}{v_{\text{ext}}}\right\}. \tag{7.12}$$

   Nesse caso temos a possibilidade de tanto $-\phi_\Delta(V_e)$ como $v(F_{w=0})$ serem negativos e $W$ não pode ser negativo. Alternativamente pode-se tomar como velocidade de propagação o próprio $v_{ext}$ obtendo-se então:

   $$\max\{0, -\phi_\Delta(v_e)\}. \tag{7.13}$$

   Os dois casos são apresentados, respectivamente, nas figuras 7.18 e 7.19.

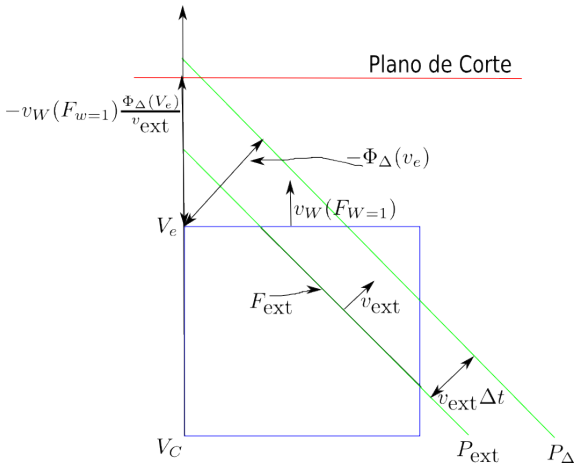



Figura 7.18: Componente da velocidade de direção $w$ está definida na face de $C$ contida em $w = 1$, $(F_{w=1})$.

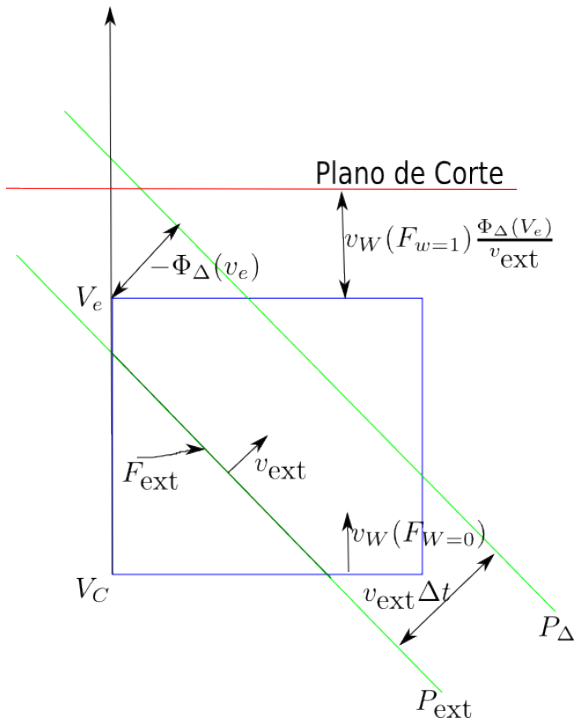



Figura 7.19: $P_{ext}$ corta $e'$, a continuação de $e$ em $C$.

3. Se apenas uma das outras coordenadas(u) for 1 e a outra(v) zero então além de $P_\Delta$, o politopo sera restrito pelos seguintes planos.

   (a) Pelo valor obtido por 7.12 ou 7.13, se o plano $w = 1$ não é cortado.

   (b) Pelo valor obtido por 7.11, se a aresta w=u=1 é cortada.

   (c) Por $P_w$, se a aresta dada por $v = 0$ e $w = 1$ for cortada.

Devido a orientação de $n_{\text{ext}}$, uma das duas arestas adjacentes a $e$ contidas em $w = 1$ é necessariamente cortada mas apenas uma delas. Os três casos indicados podem ser vistos, respectivamente, nas figuras 7.20, 7.21 e 7.22 dadas a seguir.

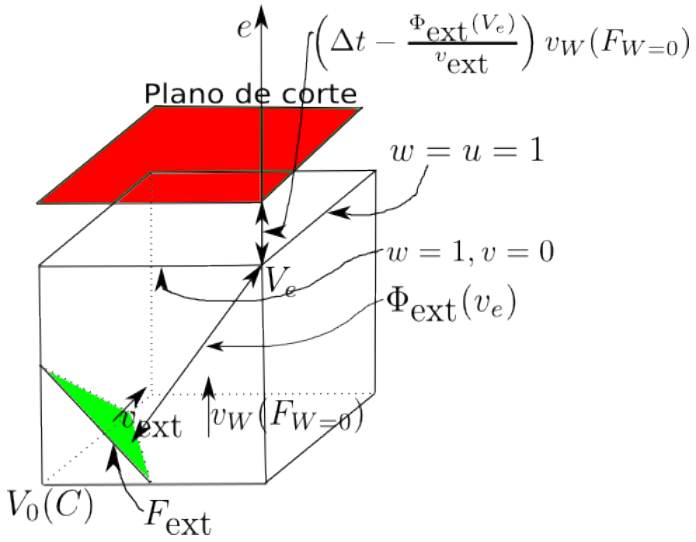



Figura 7.20: Politopo se o plano $w = 1$ não é cortado.

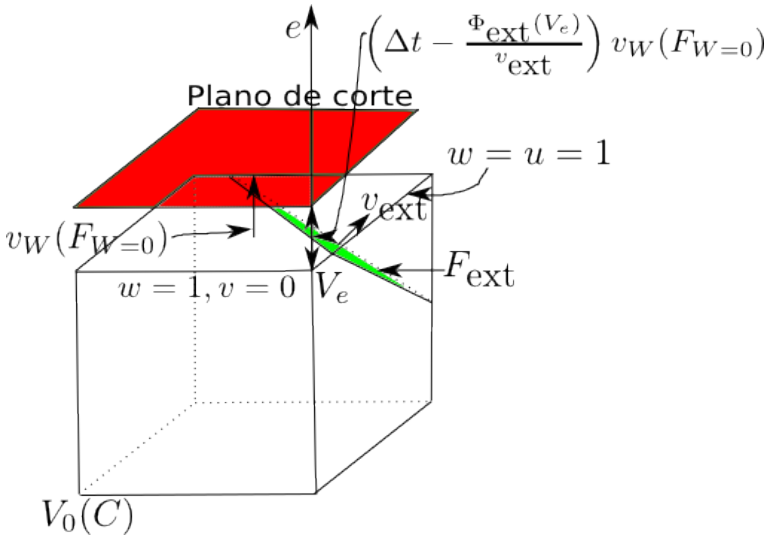



Figura 7.21: Politopo se a aresta w=u=1 é cortada.

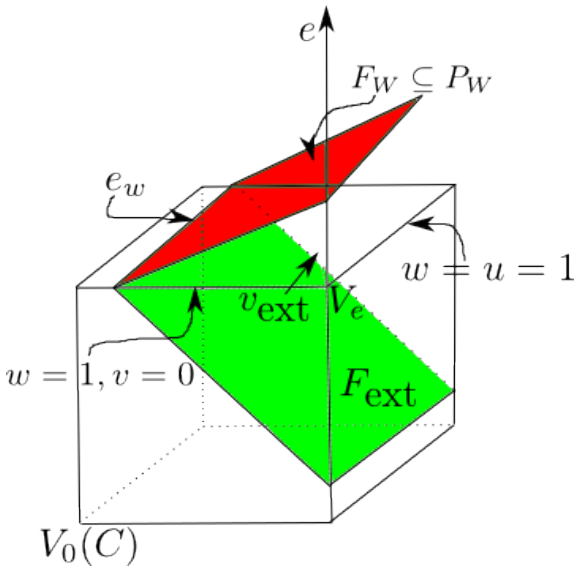



Figura 7.22: Politopo se a aresta dada por $v = 0$ e $w = 1$ for cortada.

4. Se a aresta tem coordenada livre $w$ e as outras iguais a 1 - grupo 3-, o politopo é definido pelos planos $P_\Delta$ e $P_w$. Se $P_w$ não estiver definido então a aresta em questão não será atingida pela borda deslocada dado que a cada iteração o deslocamento de $F_{\text{ext}}$ é menor que o tamanho de uma aresta. A figura 7.23 representa esse caso.

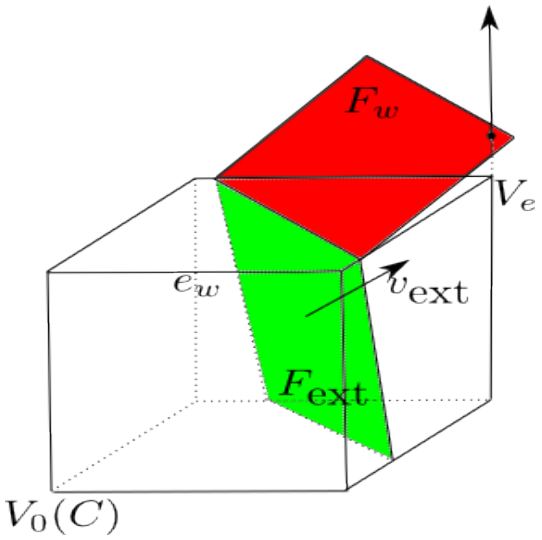



Figura 7.23: Aresta não atingida pela borda deslocada.

Observamos que para a metodologia funcionar adequadamente as arestas que já são cortadas pela borda atual devem ser processadas primeiro que aquelas que passam a ser cortadas após o deslocamento da borda. Entre essas novas arestas

só precisam ser consideradas as adjacentes a uma que era cortada mas se tornou inteiramente fluido após o deslocamento da borda. Note que pode acontecer que o deslocamento da borda computado para uma dada célula adjacente a ela, faça uma aresta $e$ se tornar toda fluido. Nesse caso se deveria procurar um novo ponto de corte em cada aresta $e' \neq e$ contendo o último vértice de $e$ a ser atingido. Mas, pode ocorrer que considerando depois, o deslocamento computado para uma outra célula adjacente a ela, $e$ continue sendo cortada pela borda. Como a alternativa mais conservadora deve prevalecer, o ponto de corte eventualmente encontrado em $e'$ não deve ser considerado senão seria gerada uma configuração inconsistente.

Voltando às regras especificadas acima, a série de exemplos 2D representados nas figuras 7.24 - 7.26 dadas a seguir ajudam a entender a opção por algumas delas.

Dentro da célula $C$ não consideramos as restrições dadas pelo prisma $PRI_{\Delta}$ por que elas dariam origem a falsos pontos da interface caso a borda nas células vizinhas evolua de forma contínua. Assim no exemplo da figura 7.24 o ponto $q$ seria determinado inutilmente porque, de fato, não pertence a borda.

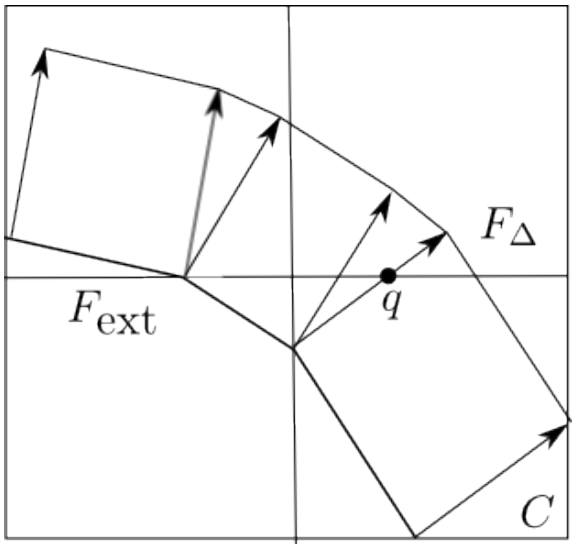



Figura 7.24: O ponto $p$ que pertence a uma face lateral do prisma definido por $F_{\text{ext}}$ e $v_{\text{ext}}$ não é um bom candidato a interseção do novo traçado da superfície do fluido com a aresta.

Se a aresta pertence ao grupo 1 então se fossem consideradas as restrições dadas pelo prisma nenhum de seus pontos poderia pertencer a Q. Agora imagine que essa aresta($e$) é do grupo 1 não só para $C$ mas para qualquer célula adjacente a ela cuja borda deslocada venha a intersectá-la. Pode acontecer que o ângulo formado por $e$

e as versões planares das faces externas dessas células seja muito pequeno tal como acontece na figura 7.25.

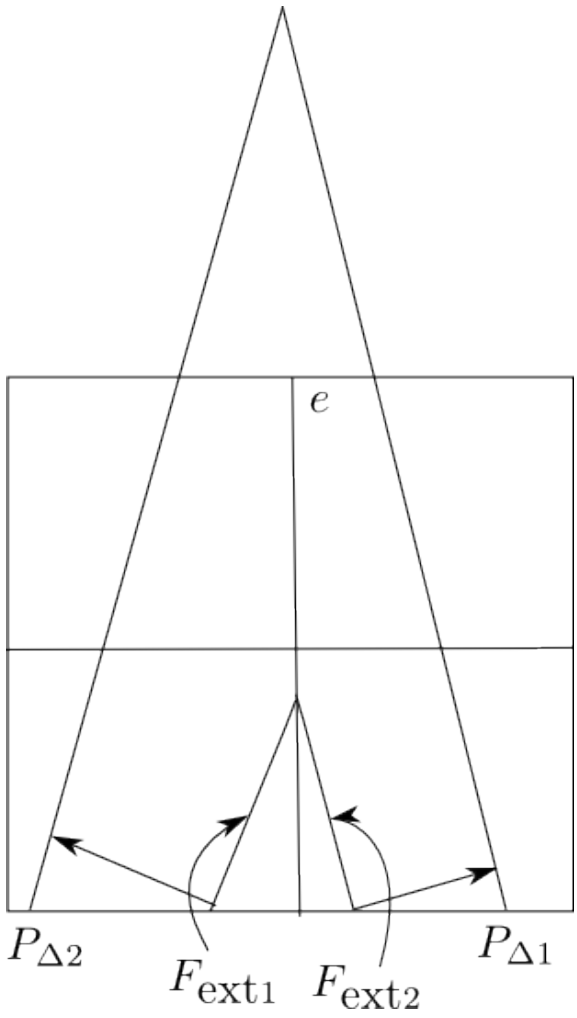



Figura 7.25: Aresta que é do grupo 1 para todas as células adjacentes contendo fluido.

Nesse caso, um deslocamento de tamanho módico dessas faces externas vai determinar que a aresta fique inteiramente em $SE_\Delta$. Assim essa aresta inteira mudaria de meio numa única iteração, o que não é plausível, em particular porque estamos limitando o deslocamento de fluido numa iteração por $h$. Para limitá-la de forma mais condizente, suponha, primeiro que o vértice de $e$ em $C(v_e)$ é atingido na transição entre $P_{\text{ext}}$ e $P_\Delta$. Quando $V_e$ é atingido já se passou parte da transição mas ainda falta a ser percorrido um percurso de tamanho $-|\Phi_\delta| = d_\Delta - |n|_w$ na direção de $n_{\text{ext}}$.

Esse percurso seria percorrido num tempo

$$d_t = \frac{-\Phi_\delta}{V_{\text{ext}}} \tag{7.14}$$

Imagine, então que nesse tempo que falta, a borda se desloque na própria direção de $e$, que é a que mais a faria avançar ao longo dessa aresta. Isso equivale a assumir que a partir do momento que passa por $v_e$ a borda se torna ortogonal a aresta $e$. Essa orientação, por outro lado, é a que minimiza a distância a $V_e$ do ponto em que um plano distando desse vértice um dado valor pré-fixado $R$, corta a aresta. A figura 7.26 ilustra essa condição de max-min.

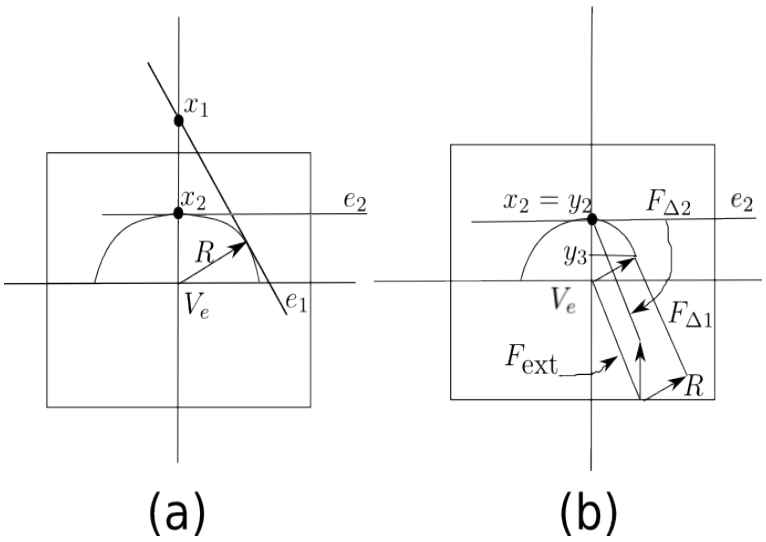



Figura 7.26: (a) Entre todas as retas que distam $R$ de $V_e$, aquela em que a interseção com $e$ é mais próxima de $V_e$ é exatamente a ortogonal a $e$ (Característica Min). (b) Entre todos os deslocamentos da face externa aquela que mais avança na direção de $e$ é logicamente o efetuado em sua direção (Característica Max).

Se a borda vai se deslocar na direção de $e$ a partir do momento que passa por $V_e$, então é preciso estimar com que velocidade($V_w$) se faz esse deslocamento. Ocorre que nessa circunstância $v(F_{w=1})$ pode não estar definido. Nesse caso $v(F_{w=0})$ ou o próprio $v_{\text{ext}}$ é empregado. Se $V(e)$ não muda de meio na transição entre $P_{\text{ext}}$ e $P_\Delta$, então se pode fazer a borda atual retroagir até passar por $V(e)$ e empregar o raciocínio acima chegando-se a uma restrição idêntica com respeito a $w$. Só que agora podemos assegurar que $v_w(F_{w=1})$ está definido e pode ser usado como a velocidade de propagação ao longo de $e$.

Se a aresta pertence aos grupos 2 e 3, então temos limitantes superiores melhor informados para restringir a posição do ponto em que $e$ corta a borda deslocada que o empregado no caso do grupo 1. Esses limitantes são estabelecidos pelos planos contendo as faces paralelas a $D_{\text{ext}}$ do prisma $PRI_{\Delta}$. Entre esses planos o mais restritivo conforme vamos mostrar a seguir é exatamente $P_w$. A figura 7.27 vai ajudar a entender a demonstração desse fato.

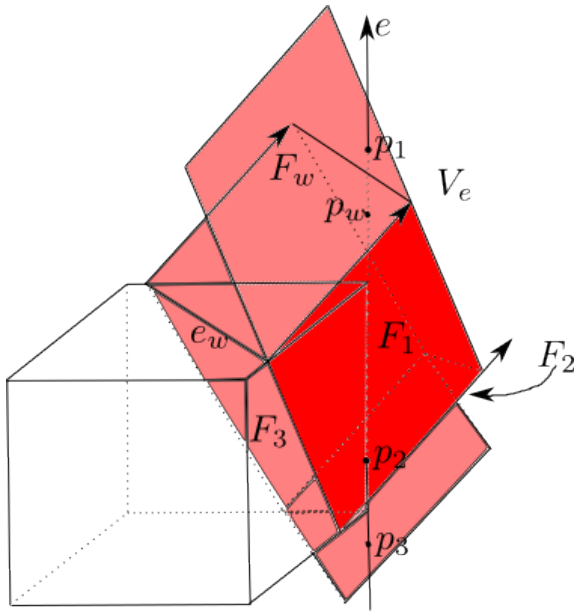



Figura 7.27: Entre as faces paralelas a $n$ do prisma $PRI\Delta$, $F_1, F_2, F_3$ e $F_w$, aquela cujo plano oferece a restrição mais estrita dentro da aresta $e$ é $F_w$, a face que contém a aresta contida em $w = 1$. Pode-se ver que o ponto de interseção do plano dessa face com $e$, $p_w$, é mais baixo que $p_1$ (determinado pelo plano de $F_1$) enquanto os planos de $F_2$ e $F_3$ não cortam $e$. Suas interseções com a reta suporte $e$ se dão nos pontos $p_2$ e $p_3$ que estão abaixo de $V_e$.

Começamos refletindo que se $P_w$ não estiver definida - o que significa dizer que a versão planar não corta o plano $w = 1$ - então, como estamos supondo que a borda atual corta pelo menos uma aresta de $C$ adjacente a $e$, essa aresta tem que ser a que é colinear com $e$ - $e'$. Os planos das faces de $PRI_{\Delta}$ definidos pelas arestas da versão planar que se encontram num ponto de $e'$, já cortam $e'$ e portanto não vão cortar $e$ e assim não fornecem restrições. Considere agora as retas suporte das outras arestas da versão planar. Essas retas ou estão contidas em planos paralelos a $e$ e que distam dela exatamente 1 ou estão no plano $w = 0$ que também dista 1 de $e$. Assim, como

$\|D_{\text{ext}}\| < 1$, não tem como os planos das faces definidas por essas arestas encontrarem $e$ dentro do semi espaço viável definido por $P_\Delta$. Ou seja para esses planos a restrição dada por $P_\Delta$ é sempre mais forte. Se o plano w=1 é cortado, as restrições com que a imposta por $P_w$ precisa competir são aquelas determinadas pelas arestas da versão planar com um vértice em que $w = 1$. As demais arestas tem retas suporte distando mais de 1 de $e$ e portanto conforme vimos as restrições estabelecidas por elas serão dominadas pela imposta por $P_{\text{ext}}$. As arestas adjacentes a $w = 1$ que precisam ser consideradas tem de ser co-planares com $e$ - senão novamente estarão em planos que distam 1 de $e$ - não podendo também cortar $e'$. Seja $a*$ uma dessas arestas.

Se $e$ é uma aresta do grupo 3 então o plano P* formado por $e$ e $a*$ pode ser escrito da forma $u = 1$ para alguma coordenada u. Se v é a coordenada restante então a reta suporte dessa aresta pode ser expresa dentro do plano $u = 1$ da forma:

$$\rho(w, v) \triangleq |n_w|w + |n_v|v = d_{\text{ext}} - |n_u|. \tag{7.15}$$

Isso significa que a intersecção dessa reta com a reta suporte de $e$, que se nos restringimos ao plano em questão se expressa por $v = 1$ é dada por

$$(w* = \frac{d_{\text{ext}} - |n_u| - |n_v|}{|n_w|}, v* = 1). \tag{7.16}$$

Como $v_e$ - que no contexto se escreve (1,1) - está do outro lado da origem em relação a $P_{\text{ext}}$ então:

$$|n_w| + |n_v| > d_{\text{ext}} - |n_u|. \tag{7.17}$$

Isso e o fato de $|n_w|$ ser positivo determinam que $w* < 1$. Desse modo um plano qualquer que contenha $a*$ vai cortar a reta suporte de $e$ fora dela. Portanto não interceptará $e$ não se estabelecendo assim restrição à evolução da borda ao longo dessa aresta.

Se, ao contrário, $e$ está no grupo 2, então, temos que considerar as duas possibilidades para P*. Se ele pode ser escrito da forma $u = 0$ ou $u = 1$. No primeiro caso $v_e$ mais uma vez se escreve(1,1) e podemos repetir o raciocínio acima para mostrar mais uma vez que a intersecção de P* com a reta suporte de $e$ se dá fora dela. O segundo caso - em que $v_e$ se escreve $(1, 0)$ - a aresta da face $w = 1$ que é cortada é $u = w = 1$, caso em que não se aplicam restrições determinadas por planos contendo as faces de $PRI_\Delta$ paralelas a $D_{ext}$.

Para mostrar que o custo computacional da proposta apresentada acima é plausível, é preciso, agora, mostrar que o cômputo das intersecções de $P_\Delta$ e P* com aresta podem ser obtidos com simplicidade. No caso de $P_\Delta$ temos duas situações. Na primeira $e \in C$ e já é cortada pela borda na versão atual no ponto especificado por $w = w_0$. Novo ponto de corte será dado por $w_D = w_0 + \frac{\|D_{\text{ext}}\|}{n_w}$. Se $w_D$ indicar um ponto fora de $C$, então, vai-se pesquisar intersecções em arestas adjacentes a que é atualmente cortada. Se $e$ é uma dessas arestas, seja $v(e)$ o vértice que ela tem em comum com essa outra. Nesse caso

$$w_D = \frac{d_\Delta - < n, v(e) >}{n_w}. \tag{7.18}$$

No caso de P*, vamos por simplicidade supor que $w = z$. A intersecção da versão planar com o plano $z = 1$ tem a direção de

$$n \otimes (0, 0, 1) = (n_y, -n_x, 0). \tag{7.19}$$

O vetor ortogonal ao plano P* tem então a direção de

$$n \otimes (n_y, -n_x, 0) = (n_x.n_z, n_y.n_z, n_z^2 - 1) = n_z.n - (0, 0, 1). \tag{7.20}$$

Seja $p_0$ um ponto qualquer de a*. Nesse caso o ponto de corte de P* em $e$ será o determinado por $w_D$ obtido por

$$< n_z.n - (0, 0, 1), v_e - p_0 + w_D.(0, 0, 1) >= 0. \tag{7.21}$$

Como $(v_e - p_0)$ é ortogonal a $(0, 0, 1)$ tiramos:

$$w_D = \frac{n_z}{1 - n_z^2}(< n, v_e > -d_{\text{ext}}) \tag{7.22}$$

Para definir qual dos dois - entre $P_\Delta$ e P* - fornece a restrição mais estrita, podemos escrever:

$$\text{Se } \frac{(< n, v_e > -d_{\text{ext}})}{d_\Delta - < n, v(e) >} + 1 > \frac{1}{n_w^2}, P_\Delta, \text{ senão } P^*. \tag{7.23}$$

Observamos que substituindo as faces do prisma $PRI_\Delta$ pelos planos que as contem forçamos a obtenção de intersecções com as arestas da grade que se nos restringíssemos ao prisma não existiriam. Teríamos então que efetuar algum processo de preenchimento de buracos para recuperar a topologia da borda deslocada.

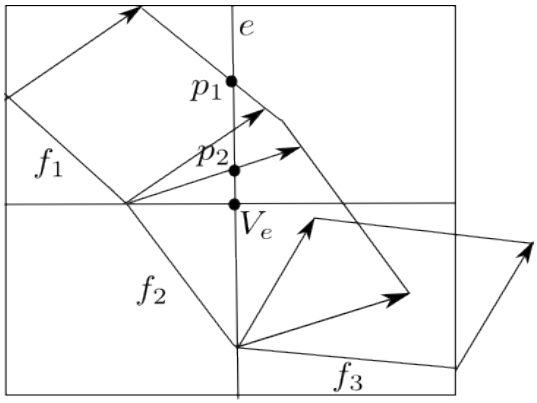
No texto acima especificamos como pode ser obtida a estimativa para o ponto de corte da borda deslocada numa aresta $e$ da grade, considerando a informação

fornecida por uma dada célula de borda atual. Como pode haver até 12 delas adjacentes a $e$ é preciso conciliar de alguma maneira a informação proveniente de cada uma delas. Seja uma vez mais $w$ a coordenada não fixa na aresta. Se simultâneamente células onde $w$ assume valores acima e abaixo dos de $e$, produzirem candidatos a ponto de corte da borda deslocada em $e$ então essa aresta será, numa primeira abordagem, considerada como inteiramente contida num mesmo meio. Numa abordagem mais elaborada essa condição pode disparar um mecanismo de sub-divisão das células adjacentes a $e$ ou submetê-las a um modelo de interpolação não linear. Pensando na abordagem mais simples a determinação do ponto de corte em $e$ envolve dados que provêm de até 8 células. Seja agora, Cel($e$), o conjunto de células contendo $e$ e Viz($e$), o daquelas que contêm pelo menos um vértice de $e$. Para toda célula $C'$ tal que $e \in Oc(C')$ seja $w_e(C')$ = a distância entre o ponto de corte de $e$ determinado considerando as restrições relativas ao deslocamento da face externa de $C'$ e o vértice de $e$ que já era fluido ou que se tornou fluido com esse deslocamento. Assumindo que esse vértice é único a distância a ele do ponto de corte de $e$ escolhido se considerando todas as células que estabelecem restrições sobre essa aresta é dado por:

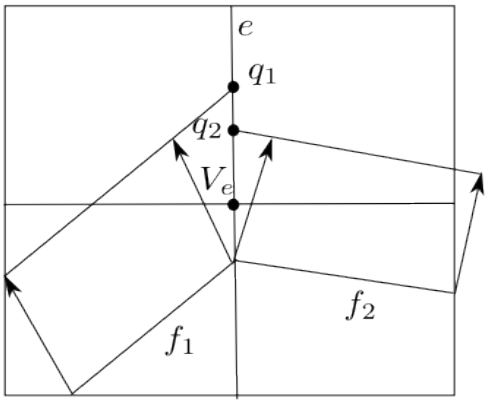
$$W(e) = \min_{C \in \text{Cel(e)}} \max_{C' | Oc(C') \supseteq C} |w_e(C')| \tag{7.24}$$

Isso significa eleger como ponto de corte, o ponto em $e$ mais distante de $v_e$ que pertence ao interior de uma representação local do volume fluido no próximo estágio construída da seguinte forma: Em cada quadrante determinado por planos das faces contendo $e$, qualquer ponto satisfazendo às restrições relativas a uma célula $C'$ tal que $Oc(C')$ intercepte o quadrante, será considerado fluido.Assim dentro de cada quadrante se considera a restrição mais relaxada entre as impostas pelas células. A distancia $W(e)$ é a maior que permite permanecer no fluido considerando os 4 quadrantes. Então como as restrições em cada quadrante são lineares isso significa que a uma distância de $v_e$ arbitrariamente próxima de $W(e)$ é possível centrar uma pequena esfera inteiramente dentro dessa representação local da parte fluida. A figura 7.28 ilustra o problema max-min acima.

O procedimento descrito acima substitui as rotinas de fast-marching e level sets




(a)




(b)

Figura 7.28: Na figura (a) as restrições determinadas pelo deslocamento de $f_1$ e $f_2$ que estão no mesmo lado em relação a $e$ - em $R^3$ seria preciso considerar o mesmo quadrante determinado pela reta suporte de $e$ - deve-se tomar a menos restritiva, isto é, a relativa a um ponto de interseção mais longe de $V_e$ - no exemplo da figura, $p_1$. Na figura (b), entre as restrições que estão em lados distintos em relação a reta suporte de $e$ deve-se tomar a mais restritiva, ou seja, a que determina num ponto de interseção mais próximo de $V_e$ - no caso $q_2$ . Daí a utilização de uma função min-max. Mínimo para lados (quadrantes em $R^3$) diferentes e máximo no mesmo lado ou quadrante.

dentro de uma abordagem clássica tornando certamente o código mais compacto. Ele também propicia minorar situações típicas de amortecimento da dinâmica da

simulação com a consequente perda de volume. Para mostrar como isso se dá, considere uma longa sequência($C_1, \ldots C_n$) de células 2D empilhadas na direção vertical, sendo $C_1$ a mais baixa e $C_0$ a que vem imediatamente abaixo dela, tal como é mostrado na figura 7.29. Todas as células da sequência estão contidas na massa fluida e cada vértice dessas células dista da borda um valor $\epsilon$ muito pequeno. Dentro da célula $C_0$ a borda também passa a uma distância $\epsilon$ de sua aresta($[V_0, V_1]$) em comum com $C_1$. Se $f$ a distância com sinal à borda, temos nesse caso $f(V_0) = f(V_1) = -\epsilon$ e para os demais vértices de $C_0$-[$X_0$ e $X_1$]-$f(X_0) = f(X_1) = 1 - \epsilon$. Assuma que a velocidade computada em todas as células empilhadas tem direção vertical apontando para baixo e vale $v = \frac{v_{max}}{2}$, sendo $v_{max}$ a componente da velocidade de valor máximo considerando-se todas as células da grade. Nesse caso devido a condição CFL que empregamos temos $v = \frac{1}{2.\Delta t}$. Não há razão para que essa velocidade não seja transferida aos vértices das células da pilha e assim a advecção de $f$ em qualquer vértice $V$ da pilha será efetuada por

$$f(V) = f(V + v\Delta t) = f(V + \frac{\Delta t.e_y}{2.\Delta t}) = f(V + 0.5e_y) = 0.5(f(V) + f(V + e_y)). \quad (7.25)$$

Assim,ao longo da pilha teremos $F(V) = 0.5(-\epsilon + -\epsilon) = -\epsilon$ e para os vértices $X_i, i = 1, 2$,$f(X_i) = 0.5(1 - \epsilon - \epsilon) = 0.5 - \epsilon$.$C_0$ não será portanto incorporada a massa fluida e supondo que ela se mantenha sem alterações na próxima iteração teremos que $F(V)$ continuará valendo $-\epsilon$ para todos os vértices de células da pilha e $f(X_i) = 0.5(0.5 - \epsilon - \epsilon) = 0.25 - \epsilon; i = 1, 2$. Se as condições se mantiverem após k estágios teremos $f(X_i) = 2^{-k} - \epsilon$ e $C_0$ ainda não terá sido incorporada a massa fluida. Lógicamente, se a entrada de fluido se manteve durante esses estágios houve considerável perda do volume fluido. A metodologia corrente ao invés, na primeira transição de estágio levaria o corte da borda em $[V_i, X_i]$ a um ponto distando $0.5 + \epsilon$ de $V_i$ e na tieração seguinte a celula $C_0$ seria inteiramente fluida.

Finalmente, um comentário relativo a paralelização do procedimento acima. Na descrição que fizemos nos fixamos em uma célula de borda e procuramos novos pontos de corte nas arestas adjacentes a ela determinados em função do deslocamento de sua face externa. Para efeito de paralelizar o método se deve fazer o contrário, focar numa aresta e analisar as restrições advindas de células da borda adjacentes

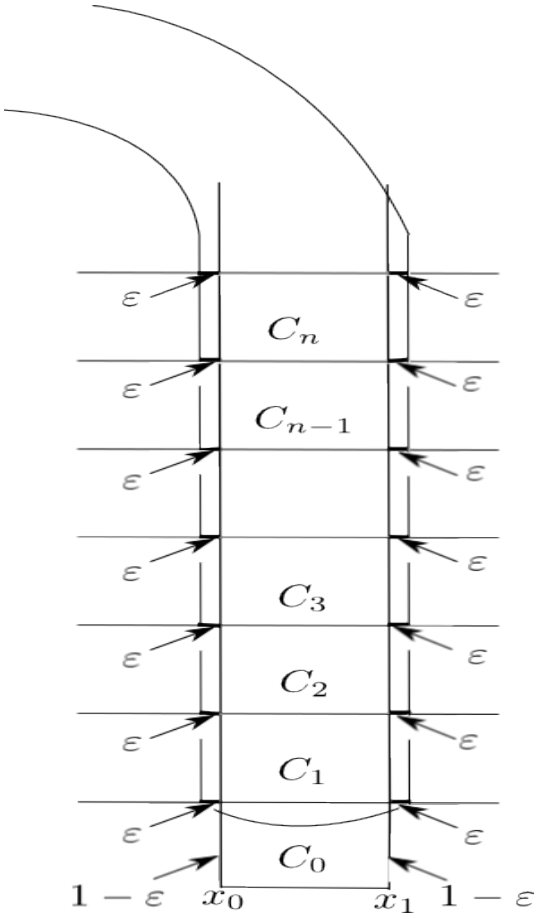



Figura 7.29: Exemplo onde há perda de volume se empregando a estratégia padrão usando level-sets

determinam sobre onde ela será cortada no próximo estágio. Nesse caso devemos lembrar que arestas já cortadas pela borda precisam ser tratadas antes das demais.

## 7.5 Montagem do Sistema de Poisson

A montagem do sistema de Poisson considerando células irregulares, ainda mais com uma face para a qual temos uma velocidade ortogonal com as três componentes - e não apenas uma projeção sobre a normal da face- não pode ser feita dos moldes em que esse sistema é construido no caso de uma malha regular. Um dos pontos que se tem de observar é que não se pode assumir que a velocidade da face de borda de uma célula seja uma constante em toda a face. Ao contrário ela deve ser pensada como um valor médio que para efeito de montagem do sistema de Poisson consideraremos aplicado no centro geométrico da face já planificada. Caso contrário,

teríamos descontinuidades no valor das velocidades ao longo da intersecção entre a face de borda da célula e as regulares.

O modelo mais simples que podemos estabelecer preservando a continuidade nessas junções e considerando o fluxo que atravessa a face de borda pode ser formulado assim:

A componente $V_i(p)$ da velocidade em em um ponto $p$, qualquer, de $C - (p = (p_i; i = 1, 2, 3))$ já em coordenadas nomalizadas- depende apenas da coordenada $p_i$ sendo obtido pelo valor em $p_i$ dado por:

1. Uma parábola $a_i p_i^2 + b_i p_i + c_i$ cujos coeficientes devem ser tais que ela atenda a :

   (a) $\int_{p \in F_B} (a_i p_i 2 + b_i p_i + c_i) = V_{FB,i} Area(F_B)$,

   (b) $V_i(0) = V_{F0,i}$,

   (c) $V_i(1) = V_{F1,i}$, quando as duas faces regulares de $C$ ortogonais a direção $i$ forem cortadas pela massa fluida. Nelas estão definidas as componentes $V_{F0,i}$ e $V_{F1,i}$.

2. Um segmento $b_i p_i + c_i$, determinado como no caso anterior só que fazendo $a_i = 0$ e empregando apenas uma das restrições impostas pelas faces regulares, quando só uma delas interceptar a massa fluida.

Fazendo isso o valor médio do divergente em C pode ser calculado por:

$$\frac{\sum_i \int_{p \in MFC} (a_i p_i + b_i) dp}{Vol(MFC)} = \frac{\sum_i a_i \int_{p \in MFC} p_i dp}{Vol(MFC)} + \sum_i b_i, \tag{7.26}$$

onde Vol(MFC) representa o volume da massa fluida em C(MFC).

Agora observe que tanto o sistema empregado para computar os coeficientes $a_i$ e $b_i$ a partir de $(V_{FB}, i, V_{F0,i}, V_{F1,i})$, bem como a integral que aparece na expressão acima só dependem da geometria da massa fluida em $C$ o que nos permite re-escrever 7.26 da forma:

$$\sum_i \langle T_i(MFC), (V_{FB,i}, V_{F0,i}, V_{F1,i}) \rangle \tag{7.27}$$

onde o vetor $T_i(MFC)$ pode ser tabelado a priori na precisão estabelecida. A entrada dessa tabela tem dimensão 3 dado que depende apenas do plano que define $MFC$.

Podemos computar assim o lado direito do sistema de Poisson, já para computar os coeficientes que expressam o Laplaciano da pressão, outra estratégia terá que ser empregada. O Laplaciano de uma função amostrada no centro da célula, como é o caso da pressão, é obtido, emulando-se o procedimento clássico empregado no caso de uma malha regular, computando-se o divergente do gradiente da função , cujas componentes são computadas nas faces da célula. Essa última operação, que só envolve dados relativos a célula em questão, pode ser efetuada como foi explicitado acima para o caso da velocidade.

Computar o gradiente da pressão numa face regular, entretanto, requer dados das duas células adjacentes a face, estendendo o número de variáveis relativas a geometria da massa fluida a serem consideradas e inviabilizando o tabelamento a priori nos moldes a que nos propomos. Por isso ao invés de um valor médio para a célula vamos computar um valor pontual estimado no centro da face.

Um problema nesse caso é que, diferentemente do caso regular, os centros da $MF$ de duas células vizinhas não está mais alinhado com o centro da $MF$ da célula que as separa. Uma forma simples de lidar com esse desalinhamento consiste em fazer o seguinte:

1. Sejam $q_1$ e $q_2$ os centros de massa da $MF$ de duas células de borda vizinhas $(C1, C2)$ e $q_{12}$ o centro da face que as separa $(F)$. Faça a projeção ortogonal de $q_{12}$ nos planos paralelos a $F$ que passam por $q_1$ e $q_2$, obtendo, respectivamente os pontos $r_1$ e $r_2$.

2. Se tivermos uma estimativa da pressão em $r_1$ e $r_2$ podemos aproximar a componente $i$ do gradiente de pressão, que é ortogonal a $F$, no ponto $q_{12}$ fazendo;

$$\frac{\partial P}{\partial p_i} q_{12} \approx \frac{P(r2) - P(r1)}{(r_{2i} - r_{1i}} \tag{7.28}$$

3. Para estimar $P(r_i)$ empregando apenas dados da célula onde $r_i$ empregamos o seguinte raciocínio: Ao longo da borda a pressão é a assumida para a porção Ar do container$(P_A r)$. Portanto ela é uma superfície de nível da pressão e assim $\nabla P$ é perpendicular a borda em qualquer de seus pontos. Como a face de borda $(F_{Bi})$ da célula $C_i$ aproxima a borda em $C_i$, então, em toda $F_{Bi}$, $\nabla P$ tem a direção da normal a $F_{Bi}$, cuja melhor estimativa é dada pela normal

de sua versão planificada. Assumindo que esse gradiente se mantém em toda célula $C_i$ então sua norma e sentido sairão de :

$$\frac{P(q_i) - P_{Ar}}{\langle n_{FBi}, q_i - p_{FBi} \rangle} \tag{7.29}$$

onde $n_{FBi}$ e $q_{FBi}$, são respectivamente a normal a $F_{Bi}$ e um ponto qualquer de sua versão planar.

4. Finalmente, tendo uma estimativa de $\nabla P$ em $C_i$, aproximamos $P(r_i)$ por:

$$P(q_i) + \langle \nabla P, r_i - p_i \rangle \tag{7.30}$$

Juntando as expressões 7.26, 7.29 e 7.30 obtemos:

$$\frac{\partial P}{\partial p_i}(q_{12}) \approx \frac{\alpha_1 P(r_1) + \alpha_2 P(r_2) + \alpha_3 P_{Ar}}{\alpha_4} \tag{7.31}$$

onde

$$\alpha_i = \langle n_{FBi}, r_i - p_{FBi} \rangle, i = 1, 2,$$

$$\alpha_3 = \sum_i \langle n_{FBi}, (qi - ri) \rangle$$

e

$$\alpha_4 = \langle n_{FBi}, (q_i - p_{FBi} \rangle .$$

Os coeficientes $\alpha_i$ dependem apenas da geometria das células $C_i$ e podem ser computados a priori gerando uma tabela com 3 entradas, os coeficientes que definem a versão planar de $FB$. Observamos que todos os $\alpha_i$ tem módulo $\leq 1$.

Para finalizar esta seção alguma s observações referentes a aplicação de forças externas.

Para forças externas independentes do ponto e do tempo, como a de gravidade, desde que cada componente da velocidade advectada no estágio seguinte $(t + \Delta t)$ seja uma combinação convexa de componentes de velocidade obtidas em $t$, não há diferença em fazer isso antes ou depois da advecção. Entretanto, no que diz respeito ao fato de novas células serem atingidas na advecção isso certamente importa porque mudando a velocidade externa de uma célula, uma outra pode passar a ser ou deixar de ser atingida. Adotamos a ordem preconizada em [73] e adicionamos as forças externas antes da advecção. Observamos, entretanto, que empregando pré-processamento é possível até considerar a ação das forças externas durante a

advecção sem um maior ônus computacional - isso não será considerado-. Isso se justifica não apenas pela complexidade introduzida para efetuar uma correção de ordem menor que $O((\Delta t)^2)$, mas pelo fato de que esse pré-cômputo ter de ser realizado a cada inicialização, que é quando se conhecem as forças externas que serão empregadas, ao invés de gerar uma tabela independente dos parâmetros da aplicação corrente, meramente lido de um arquivo quando se abre a sessão.

# Capítulo 8

# Interface Gráfica e Resultados Computacionais

Considerando que um dos objetivos deste trabalho é criar um arcabouço para o desenvolvimento de sistemas para simulação computacional de fluidos, apresentamos, inicialmente, neste capítulo a estrutura do sistema com essa finalidade que foi construido para testar a metodologia proposta na tese. Apresentamos também a interface gráfica empregada para a visualização prévia do comportamento da simulação. Em seguida apresentamos alguns resultados computacionais, sejam relativos a:

1. Problemas encontrados em nossos experimentos como a não rara perda de volume.

2. A performance de alternativas distintas que foram tentadas para implementar fases do processo, como a resolução do sistema de Poisson. Nesse caso, especificamente, se vai comparar a aplicação de um método de gradiente conjugado simples sem um pré-condicionamento adequado com opções mais elaboradas que em contrapartida requerem bem mais memória e são mais custosas por iteração, embora possam convergir em menos iterações.

3. Mostrar que o ônus computacional determinado por propostas introduzidas aqui, como o tratamento que é aplicado às células de borda, é reduzido.

Exemplos utilizados na obtenção desses resultados, bem como o vídeo resultante da renderização de suas saídas, estão disponíveis em `http://www.lcg.ufrj.br/`

Members/ceduardo. As tabelas geradas foram obtidas empregando-se um computador com processador Core i7 e com 4 GB de memória.

## 8.1 Principais Módulos do Sistema

A estrutura do sistema que foi desenvolvido é constituida dos seguintes módulos:

**Inicialização:** Neste módulo podemos ler num arquivo de configuração, contendo texto semelhante ao apresentado abaixo onde são especificados os parâmetros da simulação que incluem as dimensões do problema, constantes físicas empregadas, a indicação de opções metodológicas, limites para o número de operações de procedimentos específicos e condições de fronteira empregadas. Também nesse módulo se faz a alocação de memória para as estruturas de dados empregadas pelo sistema em consonância com os dados estipulados no texto.

No exemplo do texto de entrada que se segue, em cada linha se faz um comentário - introduzido por // - esclarecendo o significado do parâmetro nela especificado ou indicando as demais opções para ele.

```
// Exemplo: Enchimento de um recipiente (tanque) cúbico

dimension {
    length     <1.0, 1.0, 1.0>
    resolution < 32, 32, 32>
}

parameter {
   prstep   5   // intervalo entre as pré-visulaizações
   itermax   5000  // número máximo de iterações para se
                   // resolver a equação de Poisson
   eps      1e-9   // resíduo máximo para a solução da equação
                  // de Poisson
   deltmax   1.00  // passo máximo no tempo
```

```
    tfconv    1.0   // parâmetro para condição de CFL
    reynolds  1000   // número de Reynolds a partir do qual se
                // obtém o coeficiente de viscosidade
    gx       0.0    // componente x da aceleração constante
    gy      -0.3   // componente y da aceleração constante
    gz       0.0    // componente z da aceleração constante
    alpha  1.0 // Parâmetro para a aproximação dos termos convectivos;
              // 0= diferença central, 1=upwind
    ConvectiveTerms   SMART // DC, QUICK, HLPA, SMART, VONOS
}

box {  // conjunto de células com específica condição de fronteira
       coords <14,1,14>,<18,32,18>    //
       slip       //
}
```

**Atualização:** Nesse módulo se executa um "loop" constituido pelos seguintes sub-módulos:

1. Atualização do campo velocidades e do campo de pressão nas células da massa fluida, incluindo as células de borda.
2. Reinicialização da função distância através do método Fast-Marching;
3. Advecção da superfície do fluido através do método Level-Set que pode ser substituido por um procedimento do tipo forward baseado na representação da superfície dada pelo Marching-Cubes como introduzido no capítulo 7;
4. Pré-visualização do campo de velocidades e da superfície do fluido através do método Marching-Cubes;

**Renderização do fluido:** A renderização é feita a parte exportando-se os dados para um software específico. A produzida nos exemplos que serão apresentados nas seções seguintes foram produzidas empregando-se o pacote PovRay;

Os detalhes sobre a implementação de cada módulo podem ser encontrados no apêndice B.

## 8.2 Interface Gráfica do Sistema

A interface gráfica para acompanhamento da simulação durante seu processamento é constituida de 4 telas. Duas representando vistas do tanque e duas representando a projeção sobre um plano de corte com uma das orientações principais, do campo de velocidades computado nos pontos desse plano. Essas velocidades podem ser não só as definidas na staggered grid mas também as computadas em vértices ou faces empregadas para fazer a advecção da função distância a borda. As velocidades são representadas de forma proporcional a sua magnitude o que pode provocar sobreposição de várias delas. Representar apenas seu vetor direção reduziria o problema mas impediria que detectássemos que a velocidade explodiu em alguma dada célula, o que foi útil, em particular durante a construção do sistema. A representação é restrita a um conjunto de iterações indicado nos dados de entrada e atualizada com uma periodicidade também estabelecida pelo usuário. Em cada vista do tanque a massa fluida é representada empregando-se uma metodologia de shading padrão para a superfície do fluido produzida pelo Marching-Cubes.

A figura abaixo apresenta uma saida do pré-visualizador descrito acima. No alto a esquerda estão representadas as velocidades das faces cortadas pelo plano $w = D/2$, onde D é o tamanho do lado do tanque. A sua direita se representa uma vista do tanque a partir de um ponto pertencente ao eixo $x = y = D/2$ Na figura a esquerda em baixo, a vista se refere a um ponto do eixo $y = w = D/2$. Finalmente, à direita dessa figura estão representadas as velocidades de célula usadas na evolução da borda.

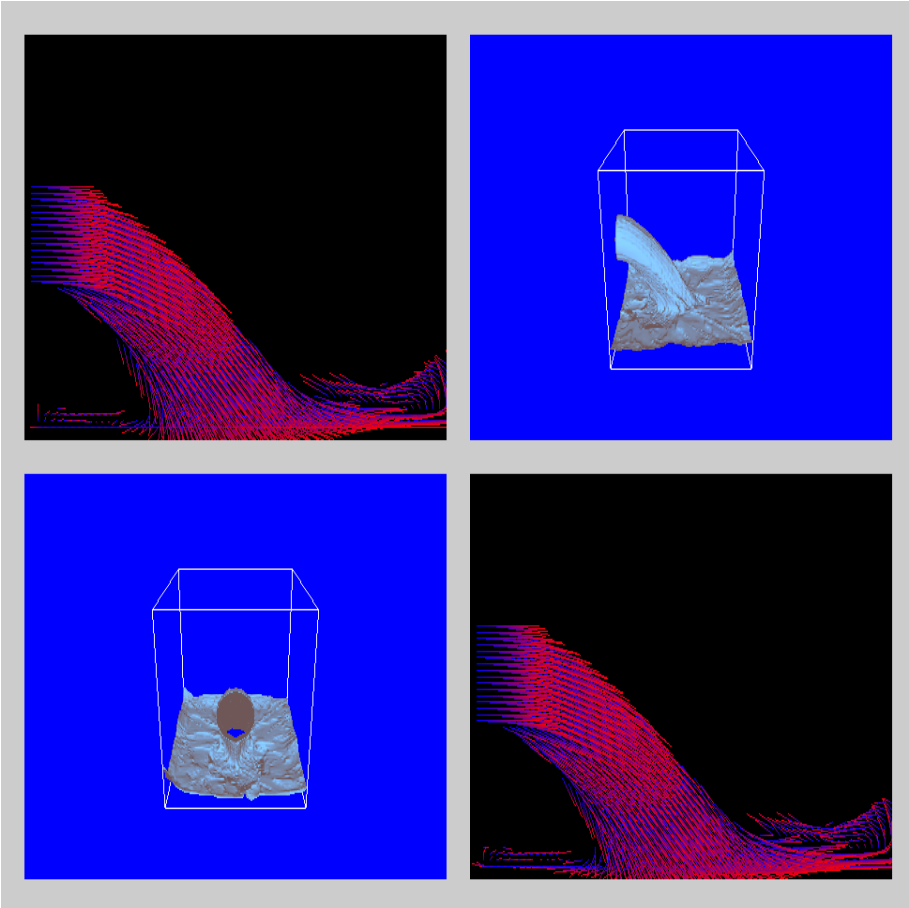


Figura 8.1: Exemplo de saída do Pré-Visualizador

A figura a seguir apresenta o frame produzido pelo Visualizador(Pov-Ray) correspondente a situação representada na figura 8.1.

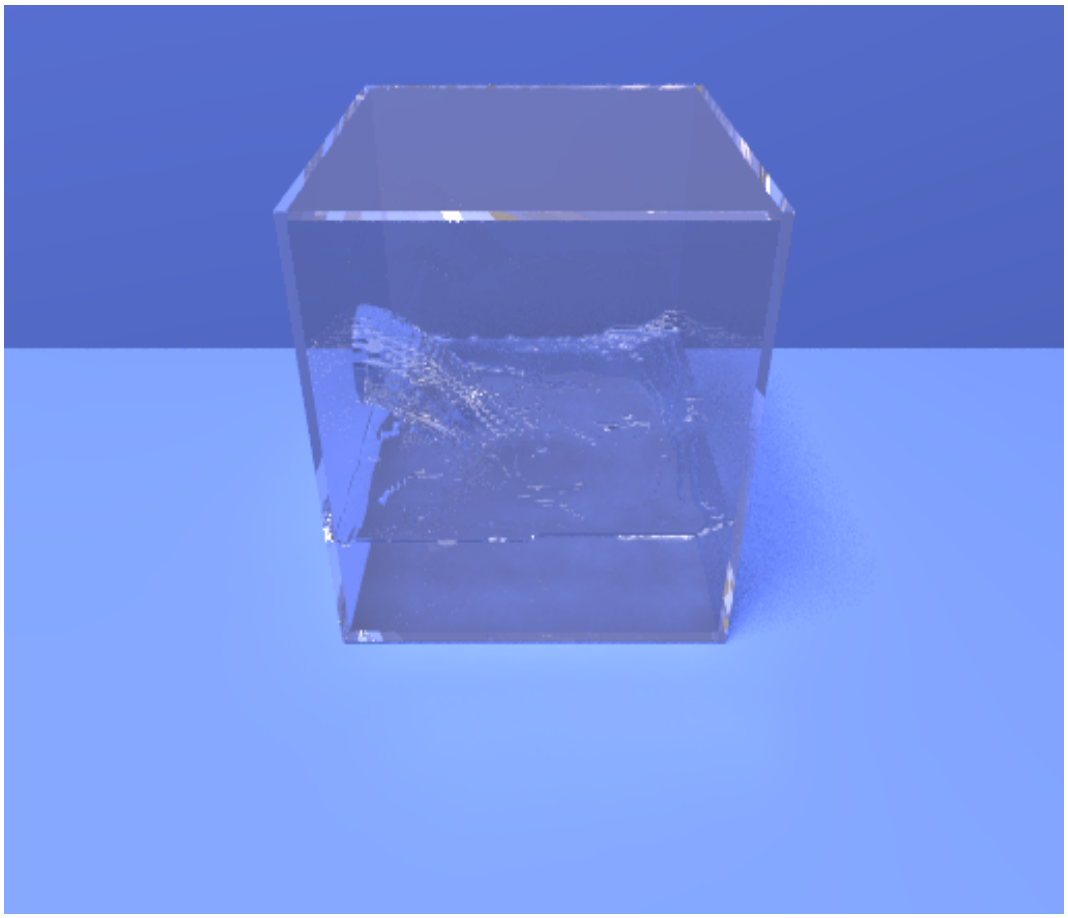


Figura 8.2: Renderização obtida pelo Pov-Ray, correspondente a situação pré-visualizada na figura anterior.

O exemplo reportado em todos os casos é o de enchimento de um tanque cúbico determinado pela entrada de fluido por um orifício circular cujo centro coincide com o centro de uma parede lateral, o tanque é representado por uma grade 32x32x32. A dinâmica desse exemplo contem algumas situações que podem causar dificuldades a simulação: Choque com o fundo e o deslocamento laminar ao longo dele com direções diferentes. Encontro nos limites do fluxo tubular e também na junção das paredes do tanque.

## 8.3 Tempo Gasto no Tratamento das Células de Borda

A tabela 8.1 apresenta o tempo total da iteração, e uma comparação entre o tempo gasto no tratamento das células de borda o tempo despendido para resolver o sistema de Poisson. Através dela podemos observar que o tempo gasto no tratamento das células de borda torna-se rapidamente insignificante em relação ao tempo total na medida que a massa fluida aumenta.

Pode-se observar que o número de células de borda aumenta até que o fundo esteja coberto e depois passa a decair porque o "tubo"por onde o fluido cai começa a ser coberto.

Lâminas por onde o fluido ascende junto as paredes laterais fazem com que o número de células de borda continue crescendo um pouco além da situação de cobertura do fundo. Próximo desse momento - entre as iterações 300 e 400 como mostra a tabela - o número de células interiores a massa fluida passa a ser maior que o de células de borda e a partir daí o custo para processar essas últimas vai se tornando comparativamente menor até se tornar menor que 1/34 do custo de se resolver o Poisson na iteração 3600. Mesmo no início do processo, quando elas são mais numerosas que as outras, o tratamento específico aplicado a elas já é menos dispendioso do que as demais operações do processo.

A presença de loops (curvas limites de uma estrutura de vórtice) na superfície do fluido pode tornar o tratamento de borda mais oneroso, como acontece entre as iterações 1500 e 2000, apesar da diminuição do número de células de borda. A presença desses loops ao redor da área em que o fluxo tubular que cai encontra a

superfície é de ser esperar. O maior processamento determinado por eles decorre do tamanho das cadeias ordenadas pela relação " $\leq$ " - ver capítulo 6 - que podem, então, ser formadas. Lembramos que estamos nos referindo a uma implementação inteiramente seqüencial.

| Iteração - Tempo Total da Iteração | Número de células fluido | Número de células de borda | Tempo por iteração | |
|---|---|---|---|---|
| | | | Poisson Solver de Melhor Performace | Processamento da Borda |
| 100 - 0.01s | 787 | 920 | 0.01s | <0.01s |
| 200 - 0.03s | 1532 | 1758 | 0.02s | <0.01s |
| 300 - 0.04s | 2313 | 2466 | 0.03s | <0.01s |
| 400 - 0.06s | 3802 | 2515 | 0.05s | <0.01s |
| 500 - 0.10s | 5153 | 2280 | 0.07s | <0.01s |
| 1000 - 0.15s | 10530 | 1901 | 0.12s | <0.01s |
| 1500 - 0.23s | 14944 | 1701 | 0.18s | 0.01s |
| 2000 - 0.74s | 18704 | 1640 | 0.68s | 0.01s |
| 3000 - 0.37s | 25229 | 1551 | 0.31s | <0.01s |
| 3600 - 0.41s | 29811 | 1422 | 0.34s | <0.01s |

Tabela 8.1: Relação entre o tempo de tratamento da células de borda e o tempo despendido na resolução da equação de Poisson.

## 8.4 Perda de Volume

O método em que se evolui a borda por meio da advecção semi-lagrangeana de uma função de level-set é conhecido por acarretar perda de volume devido a causas como a difusão numérica e o uso de modelos de interpolação (por exemplo trilineares) em situações em que eles não correspondem a realidade. Exemplos podem ser produzidos em que a simulação simplesmente estanca - caso do fluxo tubular que vai afinando até que para no ar - que na literatura se propõe resolver com empregos de partículas (Métodos da família dos Particle Level Sets).

Aqui, nos preocupamos em identificar de quanto pode ser essa perda quando se usa um esquema lagrangeano típico num caso padrão como o tratado aqui. No caso

a que se refere a tabela abaixo, a entrada foi dimensionada de forma a produzir um filete descendente com largura suficiente para alcançar o fundo sem ser detido. Mas o orifício de entrada foi dimensionado de forma que o tanque encha mais lentamente e o intervalo de tempo em cada iteração escolhido de forma a permitir no máximo o avanço de apenas uma fração ($\leq \frac{1}{2}$) do comprimento do lado da célula. Enfim, foram estabelecidas condições que favorecem a perda de volume se a borda evoluir pela advecção de um level set.

A tabela 8.2 mostra o quão considerável pode ser a perda nessas condições quando se usa um procedimento padrão sem nenhuma salva-guarda para evita-la. Essa tabela enfoca a fase inicial da simulação, incluindo a parte anterior ao momento em que o fluxo atinge o fundo do reservatório e a parte do espalhamento sobre o fundo. Observamos que a perda continua mesmo depois do fluxo atingir o fundo do tanque, isso se dá porque a perda está vinculada à diferença entre a direção da velocidade (horizontal) e a do projetor que leva o vértice no ponto da borda mais próximo (vertical).

| Iteração | Volume dado por area de inflow x velocidade de inflow x tempo ($m^3$) | Volume dado por Números de células fluidas ou de borda x $h^3$ ($m^3$) | Percentual de perda de volume |
|---|---|---|---|
| 20 | 42.7765 | 43.3438 | -1.3261% |
| 40 | 65.1008 | 67.9688 | -4.40537% |
| 60 | 88.0016 | 91.6562 | -4.15295% |
| 80 | 111.512 | 110.094 | 1.27162% |
| 100 | 148.067 | 142.594 | 3.69628% |
| 120 | 178.159 | 159.938 | 10.2275% |
| 140 | 207.303 | 178.312 | 13.9847% |
| 160 | 228.665 | 194.219 | 15.0642% |
| 180 | 259.373 | 211.5 | 18.4571% |
| 200 | 302.54 | 234.75 | 22.4069% |

Tabela 8.2: Percentual da perda de volume utilizando-se o método semi-lagragenao padrão para advectar a função level set.

Considere agora uma metodologia em que cada vértice $V$ é submetido a um "backtraking"como no esquema semi-lagrageano, só que a distância a borda da posição $V'$ obtida dessa forma é computada tomando-se como base a aproximação da superfície fluida dentro da célula $V'$ dada pelo Marching Cubes como foi descrito na seção 7.2. A tabela 8.3 indica que empregando-se essa nova metodologia a perda de volume é praticamente evitada, pois o volume das células fluidas e de borda se mantém muito próximo do fluxo acumulado até uma dada iteração.

O volume das células cortadas pelo fluido superestima o da massa fluida e a relação entre os dois está sujeita a oscilações. A coluna de 8.3 relativa ao percentual de perda de volume pode ser interpretada da seguinte maneira: Na fase inicial da simulação (número de interações $< 80$) o percentual de células de borda, cujo volume é superestimado, é relativamente grande, o que justifica um pequeno ganho de volume, numa segunda fase, ainda antes do fluido alcançar o fundo do tanque (número de iterações entre 80 e 120) o percentual de células de borda diminui e há uma pequena perda de volume. A partir do momento que o fundo do tanque é atingido pelo fluido o número de células de borda aumenta consideravelmente o que justifica a volta dos valores negativos na parte baixa da tabela.

Esclarecemos que não testamos ainda uma versão em que a borda evolua do modo "forward"descrito na seção 7.3. A idéia é que essa alternativa seja empregada apenas quando e onde for necessária para evitar a perda de volume. Aplica-lá a situações em que a advecção de level-sets funciona bem não ocasionando perda de volume pode, por outro lado, sacrificar a suavidade da solução obtida. Ainda se busca a formulação ideal para identificar quando passar de um método para o outro.

| Iteração | Volume dado por area de inflow x velocidade de inflow x tempo | Volume dado por Números de células fluidas ou de borda x $h^3$ | Percentual de perda de volume |
|---|---|---|---|
| 20 | 42.481 | 42.9688 | -1.1482% |
| 40 | 62.1868 | 64.2812 | -3.36797% |
| 60 | 81.6057 | 84 | -2.93402% |
| 80 | 102.902 | 101.844 | 1.0284% |
| 100 | 134.285 | 131.125 | 2.3529% |
| 120 | 170.113 | 170.5 | 0.22748% |
| 140 | 199.307 | 199.781 | -0.238024% |
| 160 | 221.699 | 227.75 | -2.72935% |
| 180 | 237.298 | 244.375 | -2.98235% |
| 200 | 258.172 | 267.344 | -3.55261% |

Tabela 8.3: Percentual da perda de volume empregando-se a função distância a borda obtida a partir da aproximação produzida pelo Marching-Cubes.

## 8.5 Comparação entre o Método Gradiente e o Método Gradiente Bi-Conjugado

O uso de gradientes conjugados para a resolução do sistema de Poisson é indicação prevalente na literatura em especial associada ao emprego de pré-condicionadores conforme descrito no capítulo 4. Algumas observações de crater genérico devem ser feitas em relação a implementação de gradientes conjugados que se deve adotar.

1. Sistemas de Poisson monofásicos (um único fluido) são simétricos, mas quando existe uma superfície livre e se quer impor que a pressão zero na borda seja obtida pela interpolação da pressão em células fluidas e no ar, essa propriedade pode ser perdida. Expressões desse processo de interpolação substituem ou são embutidas nas equações que exprimem o Laplaciano em células junto a borda destruindo-se assim a simetria do sistema. Para abranger também essa formulação não simétrica o esquema usando gradientes conjugados clássico é subtituido por um empregando gradientes bi-conjugados, também descrito no capítulo 4.

2. O método bi-conjugado, entretanto, tem que manter variáveis relativas a matriz transposta e portanto ter requesitos de memória maiores. Na implementação clássica da versão dita estabilizada, que adotamos nesse trabalho, oito matrizes adicionais de dimensões iguais a da grade das velocidades são empregadas, o que considerando restrições de memória razoáveis ($\leq 8Gb$) limita consideravelmente o tamanho dessa grade. Isso a menos que se recorra a mecanismos de swap, o que se contrapõe ao fato de que estamos empregando essas 8 estruturas, exatamente, para agilizar o processo.

3. Tendo em vista, reduzir os requisitos de memória, fomos a outro extremo, empregando um modelo em que se efetua um pré-processamento básico, dispensando essas matrizes adicionais. Com isso ampliam-se consideravelmente as dimensões dos problemas que podem ser tratados, mas ficou a questão de quanto se perde em performance por usar um sistema simplista assim. Para avaliar esse aspecto fizemos um confronto entre essa implementação e a do método bi-conjugado para nosso exemplo padrão que está definido numa ma-

lha (32x32x32). O resultado está expresso na tabela abaixo. A precisão exigida como condição de parada foi a mesma nos dois casos.

| Iteração | Gradiente Conjugado | | Gradiente Biconjugado | |
|---|---|---|---|---|
| | Número de células fluido | Tempo por iteração | Número de células fluido | Tempo por iteração |
| 100 | 804 | 0.01s | 787 | 0.01s |
| 200 | 1536 | 0.01s | 1532 | 0.02s |
| 300 | 2320 | 0.02s | 2313 | 0.03s |
| 400 | 3752 | 0.06s | 3802 | 0.05s |
| 500 | 5101 | 0.12s | 5152 | 0.07s |
| 1000 | 10200 | 0.36s | 10530 | 0.12s |
| 1500 | 14995 | 0.60s | 14944 | 0.20s |
| 2000 | 18434 | 0.84s | 18704 | 0.68s |
| 3000 | 26218 | 1.33s | 25229 | 0.31s |
| 3600 | 31634 | 1.78s | 29811 | 0.34s |

Tabela 8.4: Comparação entre os métodos Gradiente Conjugado e Gradiente Biconjugado para solução da equação de Poisson.

Na tabela acima pode-se observar que a medida que aumenta a massa fluida a proposta mais elaborada vai se tornando notoriamente mais eficiente apesar de uma iteração dela ser consideravelmente mais longa. O número de iterações até a convergência, é, entretanto bem menor. Os resultados dessa tabela indicaram a seguinte formulação para resolver o trade-off entre performance e dispêndio de memória:

1. Dimensões reduzidas - Gradiente Bi-Conjugado Estabilizado. O dispêndio de memória entre as diferentes versões do método Bi-Conjugado é basicamente

2. Dimensões maiores até um determinado limite - Gradiente Conjugado com pré-condicionamento simples.

3. Além desse limite - Empregar iterações de Jacobi como é feito usualmente em implementações em GPU.

**Observação:** Nos restringimos a reportar esse único caso, devido ao fato dos resultados em relação ao tempo relativo de tratamento da borda e o percentual de perda de volume ficarem bastante próximos em várias situações, onde utilizamos combinações das seguintes variações:

- Resolução: Grades 32x32x32 ou 64x64x64;
- Posições de fluxo de entrada: Orifícios de entrada de fluido definidos em diferentes paredes do cubo;
- Diâmetro dos orifícios de entrada: Raios 4, 8 e 16;

# Capítulo 9

# Conclusões e Trabalhos Futuros

## 9.1 Conclusões

Grande parte do trabalho realizado na construção desta tese foi despendida na construção de um sistema para simulação de fluidos e interface tracking segundo uma abordagem clássica via Fast Marching/Level Sets. Pelo uso intensivo desse sistema e observando as circunstâncias em que ele não respondia de forma inteiramente adequada se pode apresentar uma série de propostas que enfocam especificamente esses casos.

Elementos de inovação - e aí reside a contribuição científica do trabalho - existem tanto na forma de efetuar a advecção em uma célula de borda, quanto na obtenção das aproximações planares de sua face externa e na própria forma de evoluir a borda ou ainda de modelar o sistema de Poisson, como especificado nas seções 7.2, 7.3, 7.4 e 7.5. A abordagem não inteiramente física especificada no capitulo 6, tem objetivos mais modestos mas permite introduzir uma nova célula a massa fluida de forma gradativa.

Limitações para as abordagens introduzidas aqui são diversas. Entre elas o fato do nível de detalhamento estar vinculado ao das aproximações poligonais obtidas pelo Marching-Cubes, a complexidade de ter que lidar com retalhos de células e a utilização de esquemas cuja paralelização não pode se dar de forma completa.

Por outro lado uma metodologia em que os pontos de corte da borda nas arestas são computados diretamente num esquema "forward" pode preservar estruturas mais finas muito embora também possa gerar soluções mais rugosas em situações em que

o esquema clássico funciona bem. É em função dessas propriedades complementares que surgiu a idéia de se aplicar uma ou outra metodologia conforme as circunstâncias, o que também tem um sentido inovador.

## 9.2 Trabalhos Futuros

Há toda uma sequência de trabalhos ainda por ser desenvolvidos para se ter uma completa implementação da proposta de se fazer uma evolução da interface segundo metodologias intercambiáveis. Isso, aproveitando todas as possibilidades do conceito de célula de borda e integrada ainda à classificação empregada pelo Marching-Cubes para a determinação do traçado da borda dentro de uma célula. Antes de tudo se tem que completar a implementação da proposta apresentada no capitulo 7, colocando-a dentro desse contexto de atuação localizada seja espacialmente ou no tempo de simulação. A evolução do trabalho deve se realizar enfocando os temas especificados abaixo:

1. Alguns dos tópicos a serem tratados, se referem ao aprimoramento das metodologias apresentadas neste trabalho. Entre as questões a serem abordadas temos: Será que é suficiente conhecer apenas a componente normal à borda da velocidade com que ela se desloca? Se $\Delta t$ é pequeno, isso certamente é verdade. Mas a perspectiva, aqui é trabalhar com deslocamentos que podem ter o tamanho de uma aresta num contexto de baixa resolução. Se considerarmos mais adequado conhecer todas as componentes da velocidade de borda há algumas formas de estimá-las a serem avaliadas. O mais importante, entretanto, é como adaptar o procedimento empregado para fazer a evolução - descrito na seção 7.4 - de forma a se utilizar proveitosamente essas coordenadas adicionais da velocidade.

   Além disso há a questão de se usar aproximações planares da borda dentro de cada célula para fazer a atualização da velocidade. Ter uma aproximação planar por célula significa admitir descontinuidades da borda nas faces, que conforme já foi dito, esperamos não ser relevantes. Uma formulação que efetue essa atualização empregando apenas as interseções da borda com as faces, sem recorrer a aproximações planares é mais custosa e as vantagens de sua

implementação precisam ser avaliadas.

2. O conceito de célula de borda possibilita tratar de forma prática o escoamento junto ao piso de um tanque quando um fluido invíscido escorre formando uma lâmina fina. Basta estabelecer uma altura para a lâmina de água nas células vizinhas àquelas em que o jorro atinge o piso. A grosso modo essa altura se manterá ao redor do ponto de impacto se expressando numa vizinhança dos limites da evolução do escoamento. Essa situação pode ser simulada com o uso das células de borda até sem refletir inteiramente o modelo físico que a determina.

   A idéia é modelar o escoamento junto ao piso tendo como paradigma, vídeos que já dispomos, os quais retratam situações reais simples. Isto é pretendemos nos calcar num padrão visual esperado e tentamos ajustar os mecanismos da simulação de forma a reproduzi-lo.

3. Com respeito a detecção de condições que determinem a passagem de um esquema de evolução da borda para outro há que distinguir as que são globais - perda de volume global - e as que se referem a contextos localizados - imobilidade junto ao encontro de paredes do recipiente, por exemplo, onde a simulação costuma se deter por não ser capaz de produzir um filete fino subindo pela junção das paredes. Como essas condições locais podem ser mais complexas de detectar, se pretende num primeiro momento restringi-las ás que podem ser identificadas apenas pelas coordenadas da célula de borda - exemplos: fundo do recipiente ou junção de paredes. Condições que dependam da configuração da massa fluida serão tratadas posteriormente. Exemplo, o caso geral de células onde a velocidade apresente descontinuidade.

   Assim numa vizinhança da junção de duas paredes se passa a empregar uma metodologia que propicie a geração de filetes sem mais considerações.

   Quando a mudança de metodologia for localizada espacialmente o problema crucial é delimitar o range em que ela dever ser efetuada e evitar que artefatos possam, eventualmente, ser produzidos na zona de transição.

4. Nas implementações em CPU o uso de dados tabelados a priori, definida uma

dada resolução, pode manter para as células de borda, o custo computacional de operações equivalentes as efetuadas para células regulares.

5. A modelagem para o traçado da borda dentro de uma célula efetuada pelo algoritmo de Marching-Cubes não possibilita a interseção das faces sem que as arestas sejam cruzadas, e nesse caso em apenas um ponto cada.

6. Recentemente Heo [43] apresentou trabalho em que a função distância à borda dentro de uma célula é modelada por uma representação polinomial obtida a partir de uma distribuição de pontos, incluídos entre os chamados pontos de quadratura - Ver [43].

7. O conceito de célula de borda é propício para o tratamento de situações em que o contorno do recipiente é irregular simulando a presença de recifes ou outros obstáculos. Essa irregularidade pode ser descrita definido-se uma orientação e um offset para cada célula cortada pelo contorno. Assim uma célula de borda onde a fração passível de se tornar fluida é inteiramente fluida se encaixa na definição de célula de borda dada acima. Entretanto quando isso não acontece passamos a ter duas orientações para os limites do fluido dentro da célula - uma da borda e outra da superfície do container. (Tratamento de limites da massa fluida com orientação qualquer.)

8. Nas renderizações produzidas a partir do sistema que já temos implementado, apenas rudimentarmente se efetua a produção de bolhas obtidas como um by-product de um procedimento de Particle Level Sets. Uma representação de tamanho fixo, era gerada em situações como quando uma partícula se afastava mais que um limite da massa fluida. Agora o que se pretende focar é a formação de espuma a partir das células de borda onde há um "choque de fluxos com diferentes direções"como aquelas em que o jorro vindo da torneira encontra a massa fluida ou quando o escoamento junto ao piso deixa de ser uma lâmina fina. Além disso se pretende aplicar um modelo para a dinâmica das bolhas de ar que as permita variar de tamanho e eventualmente se agregarem ou estourarem em situações de contato. Tudo isso a partir de parâmetros como o número de bolhas geradas por célula, dimensões iniciais e tempo de vida que devem ser semi-aleatórios. (Introdução de bolhas e espumas.)

9. Neste trabalho também não se abordou questões relativas á entrada e saída de fluido do sistema. A manutenção de um fluxo de entrada constante numa área de input sobre uma parede do container, mesmo quando essa área se torna coberta pelo fluido, pode levar a resultados irrealistas. Outra questão diz respeito ao fechamento de uma fonte de entrada em que um conjunto de células de borda pode ter de ser gerado instantaneamente.

10. No que diz respeito a implementação em GPU, a idéia mais bem definida é a de se partir para uma implementação em CUDA onde os threads sejam alocados dinamicamente apenas às células cortadas pela massa fluida. Em grande parte das simulações não se alcança a situação de ter o container cheio, sendo muitas vezes o volume ocupado pelo fluido muito menor. Restringir o processamento, portanto, às células fluidas pode significar poder trabalhar com uma resolução maior. Afora isso, o investimento em paralelização deve enfocar temas específicos como a resolução do sistema de Poisson, para a qual há propostas de implementação em GPU bastante simplórias.

# Apêndice A

# Método do Gradiente, Gradientes Conjugados e Gradientes Biconjugados

Apresentamos a seguir um conjunto de métodos que resolve um sistema linear da forma $Ax = b$ onde $A$ é inicialmente simétrica e definida positiva. Essa condição pode ser desconsiderada em relação ao método dos gradientes biconjugados apresentados na última seção desse apêndice.

Todos os métodos descritos aqui, permitem preservar a esparsividade da matriz $A$ por não recorrerem a pivotemamento.

## A.1 Método do Gradiente

Apesar do método do gradiente não se mostrar eficaz na solução dos sistemas gerados em nossas simulações, apresentamo-os aqui como um pré-requisito didático para os métodos mais eficazes.

Seja A uma matriz $m \times m$, simétrica, $a_{i,j} = a_{j,i} \quad ; \quad 1 \leq i, j \leq m$ e definida positiva, isto é tal que $p^T Ap \geq 0 \quad ; \quad \forall p$. Consideremos uma função quadrática auxiliar:

$$f(x) = (1/2)\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x} + c, \tag{A.1}$$

que transforma vetores em números reais. Como a função é quadrática, há apenas

um vetor que minimiza $f$ e é exatamente o ponto crítico de $f$, ou seja, a solução de

$$\nabla f(\boldsymbol{x}) = 0.$$

Neste caso, temos

$$\nabla f(\boldsymbol{x}) = (1/2)(A^T + A)\boldsymbol{x} - \boldsymbol{b} = A\boldsymbol{x} - \boldsymbol{b} = 0.$$

Assim, se encontrarmos o ponto crítico (ponto de mínimo) de $f$, ele será solução do sistema linear $A\boldsymbol{x} = \boldsymbol{b}$.

O método do gradiente é um método iterativo de otimização onde a cada iteração tomamos

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + a_n \boldsymbol{r}_n,$$

de forma que tenhamos, para $f$ definida como em A.1, $f(\boldsymbol{x}_{n+1}) < f(\boldsymbol{x}_n)$.

A direcção de busca $\boldsymbol{r}_n$ é dada por

$$\boldsymbol{r}_n = -\nabla f(\boldsymbol{x}_n) = b - A\boldsymbol{x}_n \tag{A.2}$$

a qual é a direção de máximo decrescimento de $f$ em $x_n$.

Para encontrarmos o valor $a_n$ que minimiza $f$, igualamos a zero a derivada de $f$ em relação a $a_n$,

$$\begin{aligned} \tfrac{d}{da_n} f(\boldsymbol{x}_n + a_n \boldsymbol{r}_n) &= \nabla f(\boldsymbol{x}_n + a_n \boldsymbol{r}_n).\boldsymbol{r}_n &&= \\ &= (b - A(\boldsymbol{x}_n + a_n \boldsymbol{r}_n)).\boldsymbol{r}_n &&= 0. \end{aligned} \tag{A.3}$$

Substituindo A.2 em A.3 obtemos

$$\boldsymbol{r}_n.\boldsymbol{r}_n - a_n A\boldsymbol{r}_n.\boldsymbol{r}_n = 0,$$

ou seja, o valor mínimo de $f$ será obtido quando

$$a_n = \frac{\boldsymbol{r}_n.\boldsymbol{r}_n}{A\boldsymbol{r}_n.\boldsymbol{r}_n}.$$

Assim, podemos representar o método através do algoritmo 3

```
1 r = b − Ax  //gradiente de f no ponto  x ;
2 residuo = r.r ;
3 while (residuo > ε) and (n_iteracoes < maximo_iteracoes) do
4 |   a = r.r / Ar.r ;
5 |   x = x + ar ;
6 |   r = b − Ax  //gradiente de f no ponto  x ;
7 |   residuo = r.r ;
8 |   n_iteracoes = n_iteracoes + 1 ;
9 end
```

**Algoritmo 3**: Método do Gradiente

## A.2 Método dos Gradientes Conjugados

O método do gradiente conjugado pode ser derivado a partir dos métodos de projeção, onde a idéia básica é extrair uma uma solução aproximada para o sistema, dentro de um subespaço de $R^n$.

Seja $K$ o subespaço das soluções aproximadas, denominado subespaço de pequisa, se $K$ possui dimensão $m$, então, em geral, $m$ restrições, denominadas condições de ortogonalidade, devem ser impostas para definir a solução restrita a $K$, a saber, o vetor $b - Ax$, para essa solução, deve ser ortogonal a $m$ vetores linearmente independentes. Tais vetores definem um outro subespaço $L$ de dimensão $m$, denominado subespaço de restrições.

Os métodos de projeção podem ser classificados de acordo com os subespaços $K$ e $L$, se $K$ e $L$ são iguais o método é denominado método de projeção ortogonal, caso contrário, o método é denominado método de projeção oblíquo.

Assim, dado o sitema $A\boldsymbol{x} = \boldsymbol{b}$, e os dois subespaços de $R^n$, $K$ e $L$, o problema de aproximação através de um método de projeção no subespaço ortogonal a $L$, pode ser definido como:

$$\text{Encontrar } \tilde{\boldsymbol{x}} \in K \quad \text{tal que} \quad \boldsymbol{b} - A\tilde{\boldsymbol{x}} \perp L,$$

onde $\boldsymbol{b} - A\tilde{\boldsymbol{x}} \perp L$, significa $\langle \boldsymbol{b} - A\tilde{\boldsymbol{x}}, \boldsymbol{w} \rangle = 0, \quad \forall \boldsymbol{w} \in L.$

Se desejarmos explorar uma dada aproximação inicial $\boldsymbol{x}_0$, denominada solução inicial, então a nova aproximação $\tilde{\boldsymbol{x}}$ deve ser pesquisada no subespaço afim $\boldsymbol{x}_0 + K = \{\boldsymbol{v} \in R^n / \boldsymbol{v} = \boldsymbol{x}_0 + \boldsymbol{\delta}, \text{ onde } \boldsymbol{\delta} \in K\}$, e o problema de aproximação passa a ser redefinido como:

$$\text{Encontrar } \tilde{\boldsymbol{x}} \in \boldsymbol{x}_0 + K \quad \text{tal que} \quad \boldsymbol{b} - A\tilde{\boldsymbol{x}} \perp L. \tag{A.4}$$

Tomando

$$\tilde{\boldsymbol{x}} = \boldsymbol{x}_0 + \boldsymbol{\delta}, \text{ onde } \boldsymbol{\delta} \in K,$$

e $\boldsymbol{r}_0$, vetor residual como,

$$\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0, \tag{A.5}$$

temos que

$$\boldsymbol{b} - A\tilde{\boldsymbol{x}} = \boldsymbol{b} - A\left(\boldsymbol{x}_0 + \boldsymbol{\delta}\right) = \boldsymbol{b} - A\boldsymbol{x}_0 - A\boldsymbol{\delta} = \boldsymbol{r}_0 - A\boldsymbol{\delta} \tag{A.6}$$

e o problema A.4 passa ser equivalente a:

$$\text{Encontrar } \boldsymbol{\delta} \in K \quad \text{tal que} \quad \langle \boldsymbol{r}_0 - A\boldsymbol{\delta}, \boldsymbol{w} \rangle = 0, \quad \forall \boldsymbol{w} \in L. \tag{A.7}$$

Os métodos de projeção em geral, fazem uso de sucessivos passos de projeção, dados por A.7, onde novos pares de subespaços $K$ e $L$ são determinados a cada iteração, e o resultado da aproximação anterior é utilizado como solução inicial da iteração atual.

Seja $V = [\boldsymbol{v}_1, ..., \boldsymbol{v}_m]$ uma matriz $n$x$m$ cujos vetores-coluna formam uma base de $K$ e, similarmente, $W = [\boldsymbol{w}_1, ..., \boldsymbol{w}_m]$ uma matriz $n$x$m$ cujos vetores-coluna formam uma base de $L$. Se considerarmos a solução aproximada como

$$\tilde{\boldsymbol{x}} = \boldsymbol{x}_0 + V\boldsymbol{y}, \tag{A.8}$$

então multiplicando por $(-A)$ e somando $b$ em ambos o lados, temos

$$\boldsymbol{b} - A\tilde{\boldsymbol{x}} = \boldsymbol{b} - A\boldsymbol{x}_0 - AV\boldsymbol{y} \tag{A.9}$$

utilizando A.6 para substituir o lado esquerdo de A.9 temos,

$$\boldsymbol{r}_0 - A\boldsymbol{\delta} = \boldsymbol{b} - A\boldsymbol{x}_0 - AV\boldsymbol{y}, \tag{A.10}$$

multiplicando A.10 por $W^T$ temos,

$$W^T\left(\boldsymbol{r}_0 - A\boldsymbol{\delta}\right) = W^T(\boldsymbol{b} - A\boldsymbol{x}_0) - W^T AV\boldsymbol{y}$$

pela condição de ortogonalidade, dada por A.7,temos que $W^T(\boldsymbol{r}_0 - A\boldsymbol{\delta}) = 0$, logo

$$W^T(\boldsymbol{b} - A\boldsymbol{x}_0) = W^T AV\boldsymbol{y}$$

ou seja,

$$W^T\boldsymbol{r}_0 = W^T AV\boldsymbol{y} \tag{A.11}$$

ou ainda,

$$\boldsymbol{y} = \left(W^T AV\right)^{-1} W^T\boldsymbol{r}_0.$$

Agora substituindo $y$ em A.8, temos

$$\tilde{x} = \boldsymbol{x}_0 + V\left(W^T AV\right)^{-1} W^T\boldsymbol{r}_0. \tag{A.12}$$

Um protótipo de um método de projeção é representado pelo algoritmo 4.

```
1 repeat
2 |  Selecione um par de suespaços K e L;
3 |  Escolha as bases V = [v_1, ..., v_m] e W = [w_1, ..., w_m] ;
4 |  r = b − Ax;
5 |  y = (W^T AV)^{-1} W^T r ;
6 |  x = x + Vy ;
7 until até convergência ;
```

**Algoritmo 4**: Projeção

Observa-se a restrição de $W^T AV$ ser não-singular para o funcionamento do algoritmo 4. O teorema A.2.1 garante o cumprimento de tal restrição para alguns casos.

**Teorema A.2.1** *Se $A, L$ e $K$ satisfazem uma das duas condições:*
*i. $A$ é postiva definida e $L = K$ ou*
*ii. $A$ é não singular e $L = AK$*
*Então a Matriz $W^T AV$ é não-singular para quaisquer bases $V$ e $W$ de $K$ e $L$ respectivamente.*

A demonstração deste teorema, pode ser encontrada em [64].

## A.3 Método de Ortogonalização Total

Dado o sitema $A\boldsymbol{x} = \boldsymbol{b}$, o subespaço de Krylov relacionado a tal sistema e a um dado vetor $v$, $\text{Krylov}_m(A, v)$, é o subespaço gerado por $\{\boldsymbol{v}, A\boldsymbol{v}, A^2\boldsymbol{v}, ..., A^{m-1}\boldsymbol{v}\}$. Diversas propriedades sobre subespaços de Krylov e alguns métodos de projeção baseados nesses subespaços, são apresentados em [64]. Dentre tais métodos está o método de ortogonalização total, o qual servirá de base para descrição do método do gradiente conjugado.

No método de ortogonalização total (método OT) utiliza-se o algoritmo de Arnoldi, algoritmo 5, para construir uma base ortogonal para o subespaço de pesquisa $K$, aqui representado por $K_m$ por ser um subespaço de Krylov.

```
1  Escolha um vetor v_1 de norma 1;
2  for j = 1 to m do
3  |   for i = 1 to j do
4  |   |   h_{i,j} = ⟨A v_j, v_i⟩ ;
5  |   end
6  |   w_j = A v_j − Σ_{i=1}^{j} h_{i,j} v_i ;
7  |   h_{j+1,j} = ‖w_j‖_2 ;
8  |   if h_{j+1,j} == 0 then
9  |   |   Stop
10 |   end
11 |   v_{j+1} = w_j / h_{j+1,j}
12 end
```

**Algoritmo 5**: Arnoldi

A cada passo o algoritmo multiplica o vetor prévio de Arnoldi, $v_j$, por $A$ e então ortogonaliza o vetor resultante em relação a todos os prévios vetores $v_i$ por um procedimento padrão de Gram-Schmidt. Normaliza então o vetor resultante dessa ortogonalização e então atribui o resultado a $v_{j+1}$. O algoritmo irá parar se o resultado da ortogonalização, $w_j$, calculado na linha 6 se anular (este caso será examinado mais adiante). Abaixo, citamos alguns teoremas referentes a esse algoritmo, que são demonstrados em [64].

**Teorema A.3.1** *Assuma que o algoritmo 5 não para antes da m-ésima iteração.*

*Então os vetore resultantes $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_m$ formam uma base ortonormal do subespaço de Krylov $K_m$ gerado por $\{\boldsymbol{v}, A\boldsymbol{v}, A^2\boldsymbol{v}, ..., A^{m-1}\boldsymbol{v}\}$.*

**Teorema A.3.2** *Denote por $V_m$ a matriz nxm com vetores coluna $\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_m$, por $\bar{H}_m$ a matriz de Hessenberg $(m+1)$x$m$ cujas entradas $h_{i,j}$ diferentes de zero, são dadas pelo algoritmo 5, e por $H_m$ a matriz obtida deletando a última linha da matriz $\bar{H}_m$. Então as seguintes relações são obtidas*

$$AV_m = V_m H_m + \boldsymbol{w}_m \boldsymbol{e}_m^T = V_{m+1}\bar{H}_m \tag{A.13}$$

$$V_m^T A V_m = H_m. \tag{A.14}$$

*onde $\boldsymbol{e}_j$ é a j-ésima coluna da matriz identidade m*x*m.*

De posse do algoritmo de Arnoldi e de tais propriedades, podemos passar para descrição do método OT:

Dada a solução inicial $\boldsymbol{x}_0$ para o sistema $A\boldsymbol{x} = \boldsymbol{b}$, toma-se $L = K = k_m(A, \boldsymbol{r}_0)$, onde $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$ e $K_m(A, \boldsymbol{r}_0)$, é o subespaço gerado por $\{\boldsymbol{r}_0, A\boldsymbol{r}_0, A^2\boldsymbol{r}_0, ..., A^{m-1}\boldsymbol{r}_0\}$. E então busca-se uma solução aproximada $\boldsymbol{x}_m$ a partir do subespaço afim $\boldsymbol{x}_0 + K_m$ de dimensão $m$, impondo a condição $\boldsymbol{b} - A\boldsymbol{x}_m \perp K_m$, também conhecida como condição de Galerkin.

Tomando a solução aproximada $\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m\boldsymbol{y}_m$, como em A.8, e considerando $V = W = V_m$, uma vez que $K = L$, por A.11 temos que

$$\begin{aligned} V_m^T\boldsymbol{r}_0 = W^T\boldsymbol{r}_0 &= W^T A V\boldsymbol{y}_m \\ &= V_m^T A V_m \boldsymbol{y}_m \end{aligned} \tag{A.15}$$

Além disso, se tomarmos $\boldsymbol{v}_1 = \frac{\boldsymbol{r}_0}{\|\boldsymbol{r}_0\|_2}$ no algoritmo de Arnoldi, e $\beta = \|\boldsymbol{r}_0\|_2$, considerando que o algoritmo de Arnoldi não para antes da $m$-ésima iteração, teremos

$$V_m^T\boldsymbol{r}_0 = V_m^T(\beta\boldsymbol{v}_1) = \beta\boldsymbol{e}_1. \tag{A.16}$$

uma vez que pelo teorema A.3.1, $V_m$ forma uma base ortonormal.

Então, por A.14 e A.15 temos

$$V^T\boldsymbol{r}_0 = H_m\boldsymbol{y}_m, \tag{A.17}$$

e uma vez que $W = V_m$, por A.16 e A.17 temos

$$y_m = H_m^{-1}(\beta\boldsymbol{e}_1). \tag{A.18}$$

Dessa forma, o algoritmo do método OT é dado pelo algoritmo 6

1 $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$;
2 $\beta = \|\boldsymbol{r}_0\|$;
3 $\boldsymbol{v}_1 = \frac{\boldsymbol{r}_0}{\beta}$ ;
4 **for** $j = 1$ **to** $m$ **do**
5 $\quad \boldsymbol{w}_j = A\boldsymbol{v}_j$ ;
6 $\quad$ **for** $i = 1$ **to** $j$ **do**
7 $\quad\quad h_{i,j} = \langle \boldsymbol{w}_j, \boldsymbol{v}_i \rangle$ ;
8 $\quad\quad \boldsymbol{w}_j = \boldsymbol{w}_j - h_{i,j}\boldsymbol{v}_i$ ;
9 $\quad$ **end**
10 $\quad h_{j+1,j} = \|\boldsymbol{w}_j\|_2$ ;
11 $\quad$ **if** $h_{j+1,j} == 0$ **then**
12 $\quad\quad$ m = j ;
13 $\quad\quad$ goto 17 ;
14 $\quad$ **end**
15 $\quad \boldsymbol{v}_{j+1} = \boldsymbol{w}_j / h_{j+1,j}$ ;
16 **end**
17 $\boldsymbol{y}_m = H_m^{-1}(\beta \boldsymbol{e}_1)$ ;
18 $\boldsymbol{x}_m = x_0 + V_m \boldsymbol{y}_m$ ;

**Algoritmo 6**: Ortogonalização Total

O algoritmo do método OT, algoritmo 6, depende de um parâmetro $m$, o qual é a dimensão do subespaço de Krylov. Na prática esse parâmetro é definido de uma maneira dinâmica descrita em [64].

**Teorema A.3.3** *Assuma que o algoritmo de Arnoldi é aplicado a uma matriz simétrica. Então os coeficientes $h_{i,j}$ gerados pelo algoritmo são tais que*

$$\begin{aligned} h_{i,j} &= 0 \quad \text{para} \quad i < j-1 \quad \text{ou} \quad i > j+1 \\ h_{j,j+1} &= h_{j+1,j} \quad ; \quad j = 1, 2, ..., m \end{aligned} \tag{A.19}$$

O teorema A.3.3, demonstrado em [64], garante que: Quando a matriz $A$ é

simétrica, a matriz $H_m$ vem a ser uma matriz tridiagonal e simétrica:

$$H_m = \begin{bmatrix} h_{1,1} & h_{1,2} & & & \\ h_{1,2} & h_{2,2} & h_{2,3} & & \\ & & \ldots & & \\ & & h_{m-2,m-1} & h_{m-1,m-1} & h_{m-1,m} \\ & & & h_{m-1,m} & h_{m,m} \end{bmatrix}, \tag{A.20}$$

ou ainda, tomando-se $\alpha_j = h_{j,j}$ e $\beta_j = h_{j-1,j}$, temos

$$H_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & & \ldots & & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{bmatrix}. \tag{A.21}$$

Assim, podemos simplificar o algoritmo de ortogonalização total, algoritmo 6, de forma a obter um novo algoritmo de ortogonalização, denominado algoritmo de Lanczos, algoritmo 7 dado a seguir. Nesse algoritmo, simplesmente se computam os valores de $\alpha_j$ e $\beta_j$ suficientes para obter a matriz $H_m$ no caso de $A$ ser simétrica.

```
1  r_0 = b − Ax_0;
2  β = ‖r_0‖;
3  v_1 = r_0/β ;
4  for j = 1 to m do
5  |  w_j = Av_j − β_j v_{j−1} (se j = 1 faça β_1 = 0) ;
6  |  α_j = ⟨w_j, v_j⟩ ;
7  |  w_j = w_j − α_j v_j ;
8  |  β_{j+1} = ‖w_j‖_2 ;
9  |  if β_{j+1} == 0 then
10 |  |  m = j ;
11 |  |  goto 15 ;
12 |  end
13 |  v_{j+1} = w_j/β_{j+1} ;
14 end
15 y_m = H_m^{−1}(βe_1) ;
16 x_m = x_0 + V_m y_m ;
```

**Algoritmo 7**: Lanczos

Observe, que ainda precisamos calcular $H_m^{-1}$ para obtermos a solução final. Aplicando a fatoração $LU$ a matriz $H_m$ obtemos $H_m = L_m U_m$,

$$L_m = \begin{bmatrix} 1 & & & & \\ \lambda_2 & 1 & & & \\ & \ddots & & & \\ & \lambda_{m-1} & 1 & & \\ & & \lambda_m & 1 \end{bmatrix} \text{ e } U_m = \begin{bmatrix} \eta_1 & \beta_2 & & & \\ & \eta_2 & \beta_3 & & \\ & & \ddots & & \\ & & & \eta_{m-1} & \beta_m \\ & & & & \eta_m \end{bmatrix} \tag{A.22}$$

onde $L_m$ é uma matriz bidiagonal inferior, $U_m$ é uma matriz bidiagonal superior, os coeficientes $\beta_j$ são dados pelo algoritmo de Lanczos, algoritmo 7, os coeficientes $\lambda_j$ por

$$\begin{aligned} \lambda_1 &= 0 \\ \lambda_j &= \tfrac{\beta_j}{\eta_{j-1}} \text{ se } j > 1, \end{aligned} \tag{A.23}$$

e os coeficientes $\eta_j$ por

$$\eta_j = \alpha_j - \lambda_j \beta_j, \tag{A.24}$$

onde $\alpha_j$ também é dado pelo algoritmo de Lanczos, algoritmo 7.

Após a fatoração $LU$ da matriz $H_m$, temos que a solução aproximada $\boldsymbol{x}_m$ do algoritmo de Lanczos, pode ser dada por

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + V_m U_m^{-1} L_m^{-1}(\beta \boldsymbol{e}_1). \tag{A.25}$$

Definindo

$$P_m = V_m U_m^{-1} \tag{A.26}$$

e

$$\boldsymbol{z}_m = L_m^{-1}(\beta \boldsymbol{e}_1) \tag{A.27}$$

temos

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + P_m \boldsymbol{z}_m. \tag{A.28}$$

Por A.26 temos que

$$P_m U_m = V_m \tag{A.29}$$

equacionando a última coluna da relação matricial A.29 temos que

$$\sum_{i=1}^{m} u_{i,m} \boldsymbol{p}_i = \boldsymbol{v}_m \tag{A.30}$$

logo o vetor $p_m$ pode ser obtido a partir dos vetores $p_i$ de índice menor e do vetor $v_m$:

$$\boldsymbol{p}_m = \frac{1}{u_{m,m}} \left( \boldsymbol{v}_m - \sum_{i=1}^{m-1} u_{i,m} \boldsymbol{p}_i \right) \tag{A.31}$$

Temos então que:

$$P_m \boldsymbol{z}_m = P_{m-1} \boldsymbol{z}_{m-1} + \zeta_m \boldsymbol{p}_m \tag{A.32}$$

onde

$$\begin{aligned} &\zeta_1 = \beta = \|r_0\|, \\ &\zeta_m = -\lambda_m \zeta_{m-1} \quad ; \quad m > 1. \end{aligned} \tag{A.33}$$

Como resultado, podemos atualizar $\boldsymbol{x}_m$ a cada passo como $\boldsymbol{x}_m = \boldsymbol{x}_{m-1} + \zeta_m \boldsymbol{p}_m$, o que induz ao algoritmo 8, conhecido como a versão direta do algoritmo de Lanczos. Dado o fato de $U$ ser bidiagonal temos: $\boldsymbol{p}_m = \frac{1}{\eta_m} \left( \boldsymbol{v}_m - \beta_m \boldsymbol{p}_{m-1} \right)$.

$\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$;
$\zeta_1 = \beta = \|\boldsymbol{r}_0\|$;
$\lambda_1 = \beta_1 = 0$ ;
$\boldsymbol{p}_0 = \boldsymbol{0}$;
$\boldsymbol{v}_1 = \frac{1}{\beta}\boldsymbol{r}_0$ ;
**for** $m = 1$ **to** *convrgência* **do**
  $\boldsymbol{w} = A\boldsymbol{v}_m - \beta_m \boldsymbol{v}_{m-1}$ ;
  $\alpha_m = \langle \boldsymbol{w}_m, \boldsymbol{v}_m \rangle$ ;
  **if** $m > 1$ **then**
    $\lambda_m = \frac{\beta_m}{\eta_{m-1}}$ ;
    $\zeta_m = -\lambda_m \zeta_{m-1}$ ;
  **end**
  $\eta_m = \alpha_m - \lambda_m \beta_m$;
  $\boldsymbol{p}_m = \frac{1}{\eta_n}(\boldsymbol{v}_m - \beta_m \boldsymbol{p}_{m-1})$ ;
  $\boldsymbol{x}_m = \boldsymbol{x}_{m-1} + \zeta_m \boldsymbol{p}_m$ ;
  **if** $\boldsymbol{x}_m$ *convergiu* **then**
    Pare
  **end**
  $\boldsymbol{w} = \boldsymbol{w} - \alpha_m \boldsymbol{v}_m$ ;
  $\beta_{m+1} = \|\boldsymbol{w}\|_2$ ;
  $\boldsymbol{v}_{m+1} = \frac{1}{\beta_{m+1}}\boldsymbol{w}$ ;
**end**

**Algoritmo 8**: Versão Direta de Lanczos

## A.4 Método dos Gradientes Conjugados

O método dos gradientes conjugados (método GC), se dá através dos resultados estabelecidos pelo seguinte teorema A.4.1, :

**Teorema A.4.1** *Sejam* $\boldsymbol{r}_m = \boldsymbol{b} - A\boldsymbol{x}_m, m = 0, 1, ...$ *os vetores residuais produzidos pelos algoritmos de Lanczos e de Lanczos direto, (7 e 8) e* $\boldsymbol{p}_m, m = 0, 1, ...$ *os vetores auxiliares produzidos pelo algoritmo 8. Então,*

1. *Cada vetor residual* $\boldsymbol{r}_m$ *é tal que* $\boldsymbol{r}_m = \sigma_m \boldsymbol{v}_{m+1}$ *onde* $\sigma_m$ *é um escalar. Como resultado, os vetores residuais são ortogonais uns aos outros.*

2. *Os vetores auxiliares* $\boldsymbol{p}_m$ *formam um conjunto A-ortogonal, ou A-conjugado, isto é,* $\langle A\boldsymbol{p}_i, \boldsymbol{p}_j \rangle = 0$ *para* $i \neq j$.

Uma consequência do teorema A.4.1, demonstrado em [64], é que uma versão do algoritmo 8 pode ser derivada impondo as condições de ortogonalidade e conjugação. Isto gera o método GC que passamos a descrever agora.

O vetor $\boldsymbol{x}_{j+1}$ pode ser expresso como

$$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j \boldsymbol{p}_j. \tag{A.34}$$

Então o vetor residual deve satisfazer a recorrência

$$\boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j. \tag{A.35}$$

Se os vetores $\boldsymbol{r}_j$ são ortogonais então é necessário que $\langle \boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j, \boldsymbol{r}_j \rangle = 0$ e como resultado

$$\alpha_j = \frac{\langle \boldsymbol{r}_j, \boldsymbol{r}_j \rangle}{\langle A\boldsymbol{p}_j, \boldsymbol{r}_j \rangle}. \tag{A.36}$$

Temos também, que a próxima direção de pesquisa $\boldsymbol{p}_{j+1}$ é uma combinação linear de $\boldsymbol{r}_{j+1}$ e $\boldsymbol{p}_j$. Para um dos vetores com essa direção o coeficiente de $r_{j+1}$ será 1. Escolhendo esse vetor como $p_{j+1}$ podemos escrever

$$\boldsymbol{p}_{j+1} = \boldsymbol{r}_{j+1} + \beta_j \boldsymbol{p}_j. \tag{A.37}$$

Escrevendo $\boldsymbol{p}_{j+1}$ com em A.37 e considerando que $\langle A\boldsymbol{p}_j, \boldsymbol{p}_{j+1} \rangle = 0$ temos que

$$\beta_j = -\frac{\langle A\boldsymbol{p}_j, \boldsymbol{r}_{j+1} \rangle}{\langle A\boldsymbol{p}_j, \boldsymbol{p}_j \rangle}. \tag{A.38}$$

Note que a partir de A.35

$$A\boldsymbol{p}_j = -\frac{1}{\alpha_j}(\boldsymbol{r}_{j+1} - \boldsymbol{r}_j) \tag{A.39}$$

logo, utilizando também A.36, temos que

$$\beta_j = \frac{1}{\alpha_j} \frac{\langle \boldsymbol{r}_{j+1} - \boldsymbol{r}_j, \boldsymbol{r}_{j+1} \rangle}{\langle A\boldsymbol{p}_j, \boldsymbol{p}_j \rangle} = \frac{\langle \boldsymbol{r}_{j+1}, \boldsymbol{r}_{j+1} \rangle}{\langle \boldsymbol{r}_j, \boldsymbol{r}_j \rangle}. \tag{A.40}$$

Juntando essas relações obtemos o seguinte algoritmo

$\boldsymbol{r}_0 = b - A\boldsymbol{x}_0$;
$\boldsymbol{p}_0 = \boldsymbol{r}_0$;
**for** $j = 0$ **to** *convrgência* **do**
  $\alpha_j = \frac{\langle \boldsymbol{r}_j, \boldsymbol{r}_j \rangle}{\langle A\boldsymbol{p}_j, \boldsymbol{p}_j \rangle}$ ;
  $\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j \boldsymbol{p}_j$ ;
  $\boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j$ ;
  $\beta = \frac{\langle \boldsymbol{r}_{j+1}, \boldsymbol{r}_{j+1} \rangle}{\langle \boldsymbol{r}_j, \boldsymbol{r}_j \rangle}$ ;
  $\boldsymbol{p}_{j+1} = \boldsymbol{r}_{j+1} + \beta \boldsymbol{p}_j$ ;
**end**

**Algoritmo 9**: Gradientes Conjugados

## A.5 Pré-Condicionamento

Toda matriz simétrica $A$ admite uma decomposição da forma $A = LL^T$ onde $L$ é uma matriz triangular inferior. Isso propicia repartir a solução do sistema $A\boldsymbol{x} = \boldsymbol{b}$ em $L\boldsymbol{y} = \boldsymbol{b}$ e $L^T\boldsymbol{x} = \boldsymbol{y}$, o primeiro é resolvido de cima para baixo e o outro de baixo para cima, ambos em tempo linear com o número de coeficientes não nulos de $L$. Isso constitui resolver o sistema pelo chamado Método de Cholesky.

Ocorre que ao se passar de $A$ para $L$ pode ser perder o grau de esparsidade que a $A$ possui, tornado a estratégia contraproducente. É o que acontece no caso em que $A$ é a matriz do sistema de Poisson. Pode-se entretanto substituir $A$ por uma aproximação $\tilde{A}$ também simétrica, que pode ser obtida como o produto de uma matriz triangular inferior $\tilde{L}$, quase tão esparsa quanto $A$, por sua transposta. $\tilde{L}$ pode ser obtida diretamente a partir de $A$ pela expressão

$$\tilde{L} = A_{low} D^{-1} + D$$

once $A_{low}$ é a porção abaixo da diagonal principal de $A$ e $D$ é uma matriz diagonal. O termo relativo a célula $(i, j, k)$ dessa matriz, $D_{i,j,k}$, pode ser obtido a partir dos valores lexicograficamente menores que $(i, j, k)$ . Conforme a variante do método de Cholesky Incompleto empregado, variam a expressão de $D_{i,j,k}$ e a similaridade entre $A$ e $\tilde{A}$.

Quando a expressão de $D_{i,j,k}$ é dada por

$$D_{i,j,k} = \sqrt{\begin{array}{l} A_{(i,j,k),(i,j,k)} - \left(\frac{A_{(i-1,j,k),(i,j,k)}}{D_{(i-1,j,k)}}\right)^2 - \left(\frac{A_{(i,j-1,k),(i,j,k)}}{D_{(i,j-1,k)}}\right)^2 \\ - \left(\frac{A_{(i,j,k-1),(i,j,k)}}{D_{(i,j,k-1)}}\right)^2 \end{array}} \tag{A.41}$$

conseguimos que $A_{low}$ e $\tilde{L}$ tenham zeros nas mesmas posições e fazemos $\tilde{A} = A$ onde $A$ for não nulo.

Quando $D_{i,j,k}$ é obtida por

$$D_{i,j,k} = \sqrt{\begin{array}{l} A_{(i,j,k),(i,j,k)} - \left(\frac{A_{(i-1,j,k),(i,j,k)}}{D_{(i-1,j,k)}}\right)^2 - \left(\frac{A_{(i,j-1,k),(i,j,k)}}{D_{(i,j-1,k)}}\right)^2 \\ - \left(\frac{A_{(i,j,k-1),(i,j,k)}}{D_{(i,j,k-1)}}\right)^2 \\ -A_{(i-1,j,k),(i,j,k)} \left(\frac{A_{(i-1,j,k),(i-1,j+1,k)} + A_{(i-1,j,k),(i-1,j,k+1)}}{D^2_{(i-1,j,k)}}\right) \\ -A_{(i,j-1,k),(i,j,k)} \left(\frac{A_{(i,j-1,k),(i+1,j-1,k)} + A_{(i,j-1,k),(i,j-1,k+1)}}{D_{(i,j-1,k)}}\right)^2 \\ -A_{(i,j,k-1),(i,j,k)} \left(\frac{A_{(i,j,k-1),(i+1,j,k-1)} + A_{(i,j,k-1),(i,j+1,k-1)}}{D_{(i-1,j,k)}}\right)^2 \end{array}} \tag{A.42}$$

essas propriedades se mantém com a diferença de que a igualdade entre os coeficientes de $\tilde{A}$ e $A$ não nulos não inclui mais os elementos da diagonal principal que passam a ser escolhidos de forma que a soma dos elementos em cada linha de $\tilde{A}$ e $A$ seja a mesma.

A possibilidade de obter de obter $\tilde{L}$ e de inverter $\tilde{A}$ em tempo linear propicia a criação de uma variante do Método de Gradientes Conjugados que é equivalente a aplicar a forma clássica ao sistema

$$B\boldsymbol{x}' = \tilde{L}^{-1}b \tag{A.43}$$

onde

$$B = L^{-1}A\left(L^{-1}\right)^T$$

e fazer

$$\boldsymbol{x} = \left(L^{-1}\right)^T \boldsymbol{x}'.$$

A vantagem é que o número de iterações do Gradiente Conjugado para resolver um sistema linear não depende só das dimensões da matriz mas também de seu condicionamento avaliado pela relação entre o maior e o menor autovalor. Como $B$ é bem melhor condicionado que $A$ a convergência para solução do sistema A.43 se dá de forma mais rápida do que no caso do sistema original. Na verdade, já se

provou que se $\tilde{L}$ for gerada empregando-se a matriz diagonal $D$ obtida por **??** então o número de iterações necessárias para resolver A.43 é $O\left(n^{\frac{1}{2}}\right)$.

Além disso, não é preciso obter a matriz $B$ nem computar explicitamente $\boldsymbol{x} = \left(L^{-1}\right)^T \boldsymbol{x}'$. Pode-se adaptar o algoritmo de Gradientes Conjugados clássico aplicado a $A\boldsymbol{x} = \boldsymbol{b}$ simplesmente multiplicando alguns elementos por $\tilde{A}^{-1} - D$ que é linear. O algoritmo adaptado, conhecido como Gradiente Conjugado Pré-Condicionado, toma então a forma

1 $\boldsymbol{r}_0 = b - A\boldsymbol{x}_0$;
2 $\boldsymbol{p}_0 = \tilde{A}^{-1}\boldsymbol{r}_0$;
3 **for** $j = 0$ **to** *convrgência* **do**
4 $\quad \alpha_j = \frac{\left\langle \tilde{A}^{-1}\boldsymbol{r}_j, \boldsymbol{r}_j \right\rangle}{\left\langle A\boldsymbol{p}_j, \boldsymbol{p}_j \right\rangle}$ ;
5 $\quad \boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j \boldsymbol{p}_j$ ;
6 $\quad \boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j$ ;
7 $\quad \beta = \frac{\left\langle \tilde{A}^{-1}\boldsymbol{r}_{j+1}, \boldsymbol{r}_{j+1} \right\rangle}{\langle \boldsymbol{r}_j, \boldsymbol{r}_j \rangle}$ ;
8 $\quad \boldsymbol{p}_{j+1} = \tilde{A}^{-1}\boldsymbol{r}_{j+1} + \beta \boldsymbol{p}_j$ ;
9 **end**

**Algoritmo 10**: Gradientes Conjugados

## A.6 Método dos Gradientes Biconjugados

O método do gradiente biconjugado é um processo de projeção oblíqua sobre

$$K_m(A, v_1) \text{ gerado por } \{v_1, Av_1, ..., A^{m-1}v_1\}$$

ortogonalmente a

$$K_m(A^T, w_1) \text{ gerado por } \{w_1, A^T w_1, ..., A^{T^{m-1}} w_1\}.$$

Usualmente se faz $v_1 = r_0/\|r_0\|_2$. O vetor $w_1$ é arbitrário, desde que $\langle v_1, w_1 \rangle \neq 0$, usualmente $w_1 = v_1$. Procedendo da mesma maneira para derivação do método do gradiente conjugado a partir do método de Lanczos simétrico, nós escrevemos a decomposição $LDU$ de $T_m$ como

$$T_m = L_m U_m$$

e definimos

$$P_m = V_m U_m^{-1}.$$

Então a solução pode ser expressa como

$$\begin{aligned} x_m &= x_0 + V_m T_m^{-1}(\beta e_1) \\ &= x_0 + V_m U_m^{-1} L_m^{-1}(\beta e_1) \\ &= x_0 + P_m L_m^{-1}(\beta e_1). \end{aligned}$$

Note que a solução $x_m$ é atualizada a apartir de $x_{m-1}$ numa maneira similar ao método do gradiente conjugado. Assim como no método do gradiente conjugado os vetores $r_j$ e $r_j^*$ estão na mesma direção de $v_{j+1}$ e $w_{j+1}$ respectivamente. Definindo a matriz

$$P_m^* = W_m L_m^{-T}$$

temos que

$$(P_m^*)^T A P_m = L_m^{-1} W_m^T A V_m U_m^{-1} = L_m^{-1} T_m U_m^{-1} = I,$$

ou seja, os vetores $p_i^*$ de $P^*$ e os vetores $p_i$ de $P$ são $A$-Conjugados. Utilizando esta informação, um algoritmo semelhante a algoritmo do gradiente conjugado, algoritmo do gradiente biconjugado, pode ser derivado a partir do algoritmo de Lanczos.

1 $r_0 = b - Ax_0$;
2 Escolha $r_0^*$ tal que $\langle r_0, r_0^* \rangle \neq 0$ ;
3 $p_0 = r_0$;
4 $p_0^* = r_0^*$;
5 **for** $j = 1$ **to** *convrgência* **do**
6 $\quad \alpha_j = \langle r_j, r_j^* \rangle / \langle Ap_j, p_j^* \rangle$ ;
7 $\quad x_{j+1} = x_j + \alpha_j p_j$ ;
8 $\quad r_{j+1} = r_j - \alpha_j A p_j$ ;
9 $\quad r_{j+1}^* = r_j^* - \alpha_j A^T p_j^*$ ;
10 $\quad \beta = \langle r_{j+1}, r_{j+1}^* \rangle / \langle r_j, r_j^* \rangle$ ;
11 $\quad p_{j+1} = r_{j+1} + \beta_j p_j$ ;
12 $\quad p_{j+1}^* = r_{j+1}^* + \beta_j p_j^*$ ;
13 **end**

**Algoritmo 11**: Gradientes Biconjugados

# Apêndice B

# Descrição das Principais Rotinas Utilizadas no Sistema

Tendo em vista que em termos de re-usabilidade de software a documentação se mostra de vital importância, e que um dos objetivos deste trabalho é criar uma plataforma inicial para futuros projetos na área de simulção de fluidos, apresentamos neste apêndice, a descrição das principais rotinas utilizadas no sistema, a fim de, juntamente com o manual de referência do sistema, disponibilizado em minha página no laboratório de computação gráfica, `http://www.lcg.ufrj.br/Members/ceduardo`, facilitar a compreensão do código fonte do sistema, possibilitando sua re-implementação de forma mais eficiente e/ou ajustada a outro propósito específico.

Para ilustrar algumas dessas rotinas descritas aqui, utilizamos como exemplo, a simulação $fillContainerSimulation$, cujo vídeo também está disponibilizado em minha página.

O sistema está dividido basicamente em duas etapas, inicialização e atualização, descritas a seguir.

## B.1 Etapa de Inicialização

A etapa de inicialização é representa em nosso sistema pela rotina **void Simulation::InitializeApplication()**, a mesma é executada uma única vez a cada simulação, em síntese, tal rotina aloca espaço de memória para algumas das principais estruturas utilizadas no sistema, instância as principais classes do sitema, e define o

volume e o campo de velocidade iniciais.

### B.1.1 Rotinas que compõem a Etapa de Inicialização

- Scene::Scene() 1
- void Simulatoin::AlocateMemory() 2
- void Simulation::SurfaceInitial() 3
- Navier::Navier() 4
- void Navier::NavierSolverInitialization() 4b
- void Navier::InitInflowAndLabelSlipOutflow() 4(b)iv
- void Navier::InitializeBorderFacesLists(int index) 4(b)ivC
- void Navier::LabelSlipOutflowCells(int index) 4(b)ivD
- LevelSet::LevelSet(...) 5
- FastMarch::FastMarch(...) 6

### B.1.2 Descrição da Rotina void Simulation::InitializeApplication()

1. Instancia a classe *Scene* na qual estão definidos alguns parâmetros computacionais da simulação, como por exemplo, o tamanho da grade ($nx$ x $ny$ x $nz$), o espaçamento da grade ($h$) e o número máximo de iterações para o método iterativo utilizado na solução da equação de poisson.

2. Aloca espaço de memória para as estruturas de dados que irão armazenar:

   (a) As componentes de velocidade $(u, v, w)$:

      i. $gridCellUVW[3][(nx + 4)$ x $(ny + 4)$ x $(nz + 4)]$, armazena as componentes de velocidade definidas no centro de cada célula. Observe que aqui como em diversas outras estruturas utilizadas em nossa implementação, são alocados 4 espaços de memória a mais em cada

direção (dois em cada sentido), formando quatro paredes de células extras em cada direção (duas em cada sentido), esses espaços são denominados células fantasmas pois não fazem parte da malha real, e são utilizados apenas para facilitar o tratamento das condições de fronteira;

ii. $gridUVW[3][2*(nx+4)$ x $(ny+4)$ x $(nz+4)]$, armazena as componentes de velocidade definidas nas faces de cada célula, conforme figura 4.3, tal grade tem tamanho dobrado em relação as demais, pois armazena a versão atual e a nova versão das componentes de velocidade, respectivamente em sua primeira e segunda metade, comutandoas a cada iteração.

(b) A função distância (Level-Set), $gridPhi[(nx+4)$ x $(ny+4)$ x $(nz+4)]$;

(c) Os rótulos das céluas, $flag[(nx+4)$ x $(ny+4)$ x $(nz+4)]$, onde cada elemento de $flag$ pode assumir um dos seguintes valores:

i. `AIR = 0x2000`, células não preenchidas pelo fluido;

ii. `BORDER_AIR = AIR + 256`, células AIR que são vizinhas por face a uma célula preenchida pelo fluido (células armazenadas na lista $ListBorderAir$, descrita mais adiante);

iii. `PROCESSED_BORDER_AIR = BORDER_AIR + 1024`, células AIR que já foram processadas, (células armazenadas na lista $Navier :: ListAirToFluid$, descrita mais adiante);

iv. `SLIP_AIR = 0x3000`, células AIR que possuem condição de contorno do tipo slip;

v. `OUTFLOW_AIR = 0x4000`, células ainda não preencchida pelo fluido que caracterizam uma saída de fluido;

vi. `SLIP_FLUID = 0xD000`, células preenchidas pelo fluido que possuem condição de contorno do tipo slip;

vii. `FLUID = 0xE000`, células preenchidas pelo fluido;

viii. `INFLOW = 0xF000`, células que caracterizam uma entrada de fluido;

(d) Listas que irão armazenar a posição das células que receberão tratamento especial:

i. $ListInflow$, lista de células (índices) do tipo `INFLOW`;

ii. $ListSlipOutflow$, lista de células do tipo `OUTFLOW_AIR` ou `SLIP_AIR`;

iii. $ListBorderFaceAir$, lista de células `AR` que possuem uma célula vizinha `FLUID`, nessa lista pode existir células repetidas;

iv. $ListBorderFaceFluid$, lista de células `FLUID` que possuem uma célula vizinha `AR`, nessa lista pode existir células repetidas;

v. $ListBorderAir$; lista de células `AIR` que possuem uma célula vizinha `FLUID`, nessa lista não existe célula repetida;

vi. $ListBorderFluid$; lista de células `FLUID` que possuem uma célula vizinha `AR`, nessa lista não existe célula repetida;

vii. $ListFluid$, lista de células `FLUID`;

viii. $ListFluidEnd$;

ix. $BorderAirNon_DiagonalSize$;

x. $BorderFluidNonDiagonalSize$;

xi. $FM2LS - FluidList$;

xii. $FM2LS - AirList$;

3. Define as células que compõe o volume inicial do fluido, ou seja, define a função distância inicial, $gridPhi$ inicial. Na simulação $fillContainerSimulation$, utilizamos a função inicial dada por B.1, a figura B.1 apresenta o volume inicial do fluido, localizado na parede lateral esquerda do recipiente.

$$\Phi_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} = \begin{cases} h * max\left(-\frac{1}{2}, \sqrt{(j+\frac{1}{2}-cj)^2+(k+\frac{1}{2}-ck)^2}-r\right); & i = 0\, ou\, 1 \\ h * \sqrt{\left(max\left(0, \sqrt{(j+\frac{1}{2}-cj)^2+(k+\frac{1}{2}-ck)^2}-r\right)\right)^2+(i+\frac{1}{2}-2.0)^2}; & i > 1 \end{cases} \tag{B.1}$$

4. Instancia a classe $Navier$, a qual é responsável pela solução das equações de Navier-Stokes, cuja solução a cada iteração, determina campo velocidade do escoamento. E através do construtor **Navier::Navier()**:

   (a) Define uma tabela de contatos 64x6, binária, onde cada entrada indica se o vizinho j (por face) é fluido (1) ou Ar (0), e cada linha representa uma

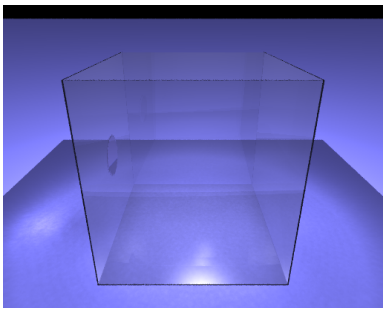


Figura B.1: Volume Inical

das 64 possibilidades, tal tabela será utilizada futuramente no cômputo da velocidade nas células com condição de contorno.

(b) Chama a função **Navier::NavierSolverInitialization()**, que :

i. Inicia o valor do passo no tempo, $dt$;

ii. Aloca espaço de memória para as estruturas de dados (vetores de double) que serão utilizadas na resolução da equação de Poisson.

iii. Inicia o rótulo de todas as células como `AIR`, com exceção das células que compõe os dois planos mais externos em cada um dos dois sentidos de cada uma das três direções do grid, ou seja, com exceção das células fantasmas. Iremos nomear os planos mais externos da grade, assim como as faces de uma dada célula, conforme figura B.2.

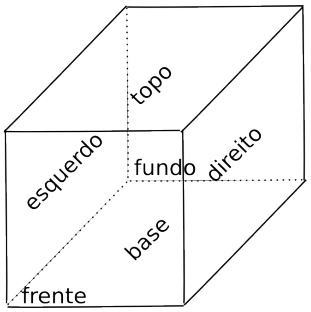



Figura B.2: Nome dos planos externos

Os planos externos são rotuladas da seguinte maneira:

A. Plano esquerdo: rótulo = 1;

B. Plano direito: rótulo = 2;

C. Plano base: rótulo = 3

D. Plano topo: rótulo = 6

E. Plano fundo: rótulo = 9

F. Plano frente: rótulo = 18

iv. Chama a função **void Navier::InitInflowAndLabelSlipOutflow()** que:

A. Define como `INFLOW` o rótulo das células que compõe o fluido inicial (observe que o fluido inicial foi definido no passo 3), atribui a velocidade do fluxo de entrada (valores $inflow\_value$ definidos na classe $Scene$) a tais células (tanto para as componentes de velocidade definidas nas faces da célula, como para as componentes definidas no centro da célula), e as insere na lista $ListInflow$.

B. Rotula as células que estão em posições pré-estabelecidas como `OUTFLOW_AIR` ou `SLIP_AIR`, e as insere em suas respectivas listas, $ListSlipOutflow$ ou $ListSlipOutflow$. A figura B.3 ilustra a configuração inicial do exemplo $fillContainerSimulation$:

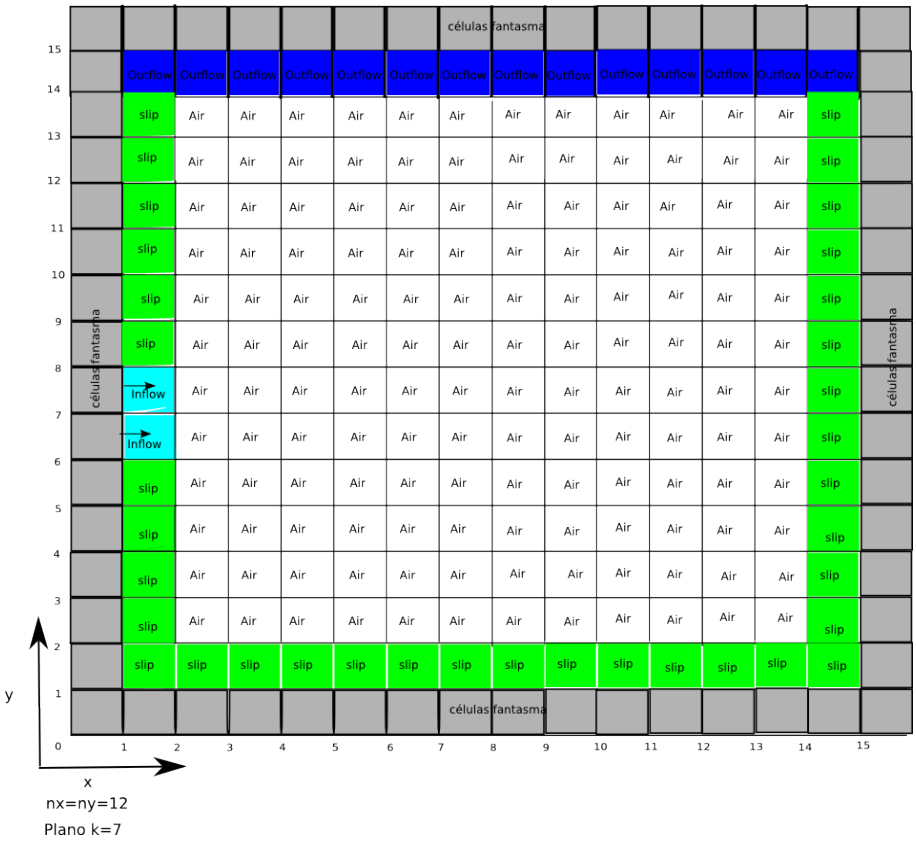



Figura B.3: Configuração Inicial

C. Para cada uma das células inseridas na lista $ListInflow$, chama a função void **Navier::InitializeBorderFacesLists(int index)**, a qual:

- Completa a definição da velocidade nas células do tipo `INFLOW` que possuem uma célula vizinha posterior (vizinha a face direita, a face topo ou a face frente) do tipo `AIR`, ou seja, atribui os respectivos valores $inflow\_value$ as respectivas faces posteriores dessas células, conforme figura B.4.

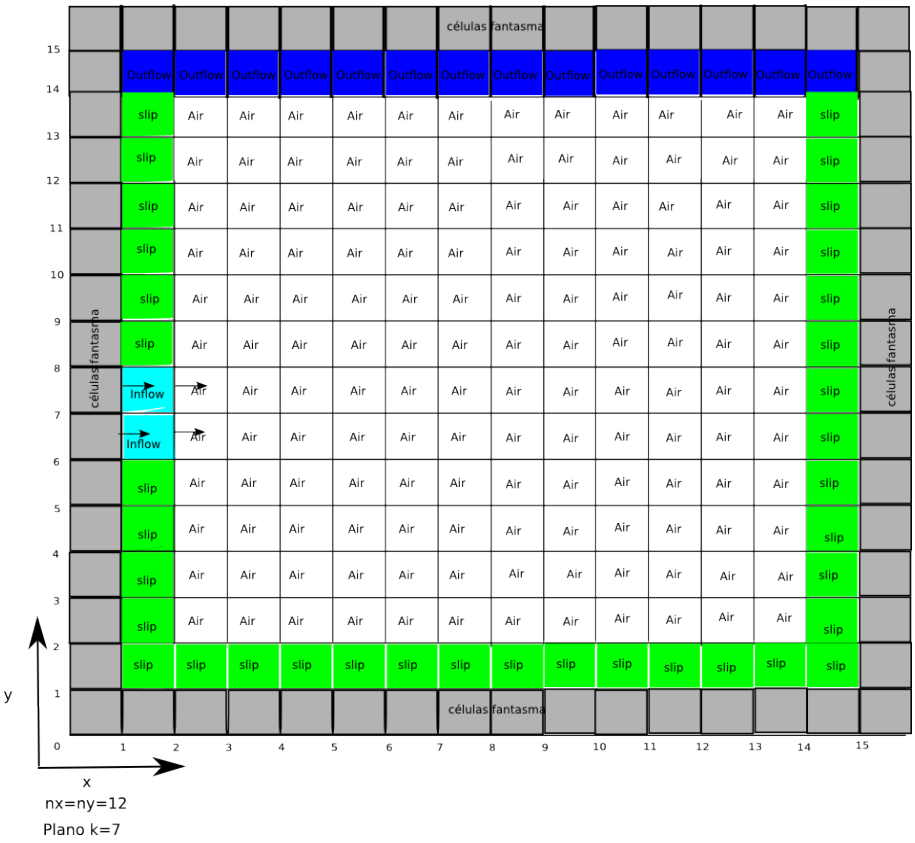



Figura B.4: Células Inflow com o Campo Velocidade Completado

- Insere na lista $ListBorderFaceFluid$ as células do tipo `INFLOW` que possuem uma célula vizinha por face de algum tipo diferente de `INFLOW` (nesse caso células do tipo `AIR` ou `SLIP_AIR`).
- Insere na lista $ListBorderFaceAir$ as células do tipo `AIR` ou `SLIP_AIR` que possuem uma célula vizinha por face do tipo `INFLOW`.

  Observe que ambas as listas $ListBorderFaceAir$ e $ListBorderFaceFluid$, podem ter uma mesma célula inserida mais de uma vez. Tais listas, em nosso exemplo, são apresentadas na figura B.5

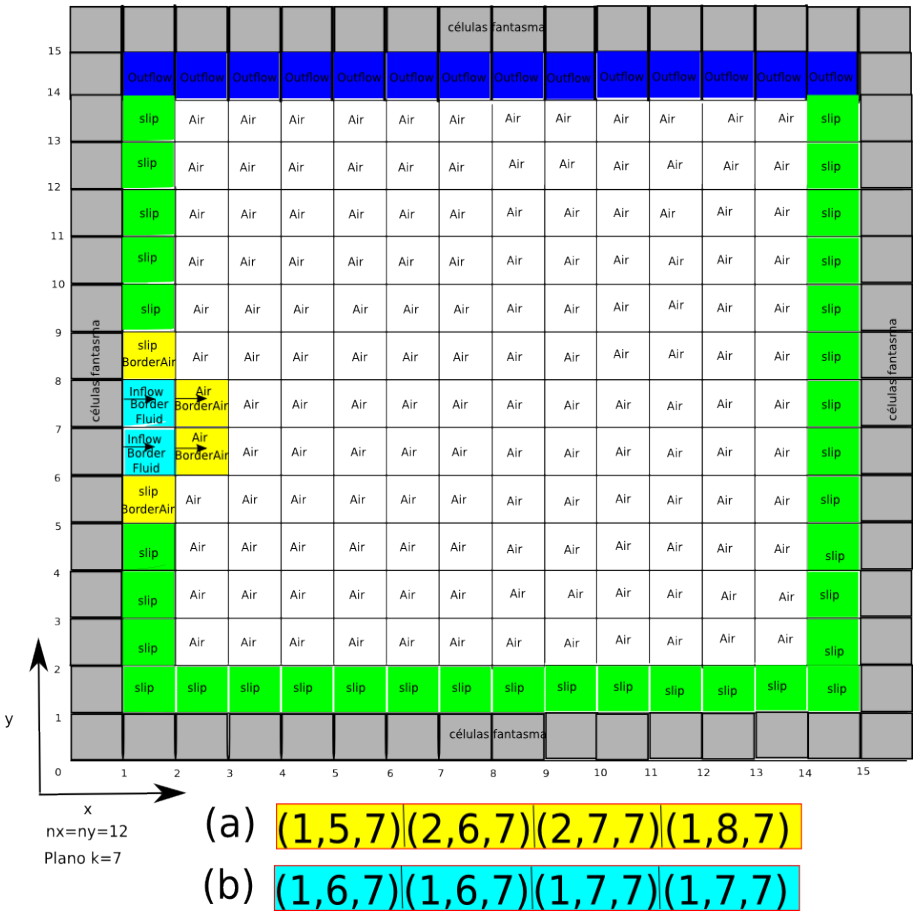



Figura B.5: (a)Lista BordeFaceAir (b)Lista BorderFaceFluid

D. Para cada uma das células inseridas na lista *ListSlipOutflow* chama a função void **Navier::LabelSlipOutflowCells(int index)**, a qual verifica o grau de liberdade de uma célula `SLIP_AIR`, ou seja, o número de vizinhos por face do tipo `AIR` dessa célula. De acordo com tal número de vizinhos, procede da seguinte forma:

- 1 vizinho do tipo `AIR`: Então acende no rótulo da respectiva célula o bit 64 e o bit referente ao vizinho em questão , a saber:
  - vizinho (i-1,j,k) acende o bit 1,
  - vizinho (i+1,j,k) acende o bit 2,
  - vizinho (i,j-1,k) acende o bit 4,
  - vizinho (i,j+1,k) acende o bit 8,
  - vizinho (i,j,k-1) acende o bit 16
  - vizinho (i,j,k+1) acende o bit 32.

- Mais de 1 vizinho do tipo `AIR`: Acende apenas os bits referentes aos respectivos vizinhos.

E. Faz $dt = min\left(dt, \frac{h}{\|inflow\_value\|+1E-15}\right)$;

5. Instancia a classe *LevelSet* , a qual representa a interface Fluido/Ar, ou seja, representa superfície livre do fluido.

6. Instancia a classe *FastMarch* , utilizada para ajustar os valores da função level set, armazenada em *gridPhi*, de maneira que a mesma defina uma função distância a superfície livre do fluido. E através do construtor **FastMarch::FastMarch(...)**:

   (a) Aloca espaço de memória para as estruturas de dados:

   - $FMHeap[(nx + 4)$ x $(ny + 4)$ x $(nz + 4)]$: (std::vector int) que irá armazenar o índice de cada elemento inserido na pilha.
   - $HeapPosition[(nx+4)$ x $(ny+4)$ x $(nz+4)]$: (int[]) que irá armazenar a posição na pilha de cada elemento elemento inserido na mesma.

   (b) Atribui $-\infty$ a $HeapPosition[(nx+4)$ x $(ny+4)$ x $(nz+4)]$ nas posições correspondentes aos planos fantasmas mais externos.

.

## B.2 Etapa de Atualização

A etapa de atualização é representa em nosso sistema pela rotina **void Simulation::UpdateConfiguration()**, a mesma é executada em loop até atingir um estado estacionário ou ser encerrada pelo usuário, e em síntese, a cada iteração, obtém a solução das equações de Navier-Stokes e evolui, de acordo com o campo de velocidade obtido através dessa solução, a superfície do fluido, representada implicitamente e manipulada através do método level set em conjunto com o método fast marching.

### B.2.1 Rotinas que Compõem a Etapa de Atualização

- void Navier::DoIt(...) 1

- void Navier::DiagonalBorderCells() 1f
- void Navier::ProcessingBorderAirList() 1h
- void Navier::OrderedNewFluidCellInitializer(...) 1(h)iii

### B.2.2 Descrição da Rotina void Simulation::UpdateConfiguration()

1. Chama a função **void Navier::DoIt()**, a qual representa um solver para solução das equações de Navier-Stokes, e funciona da seguinte maneira:

   (a) Para cada elemento na lista $ListBorderFaceAir$, verifica se o mesmo mudou de `AIR` para `FLUID` e caso tenha mudado retira-o da lista $ListBorderFaceAir$ e o insere na lista $ListAirToFluid$, simultaneamente verifica se um elemento da lista $ListBorderFluid$ mudou de `FLUID` para `AIR` e caso tenha mudado retira-o da lista $ListBorderFaceFluid$ e o insere na lista $ListFluidToAir$. Caso ocorra uma troca simultânea (mesmo índice nas duas listas) ao invés de eliminar os elementos, troca os de lista ($ListBorderFaceAir <-> ListBorderFaceFluid$).

   (b) Para cada uma das células na lista $Navier :: ListFluidToAir$ verifica se a mesma é uma vizinha por face de uma célula `FLUID`, e em caso afirmativo insere-a na lista $ListBorderFaceAir$ e também insere a célula vizinha em questão na lista $ListBorderFaceFluid$.

   (c) Para cada uma das células na lista $Navier :: ListAirToFluid$ verifica se a mesma é uma vizinha por face de uma célula `AIR`, e em caso afirmativo insere-a na lista $ListBorderFaceFluid$ e também insere a célula vizinha em questão na lista $ListBorderFaceAir$.

   (d) Cria uma nova lista denominada ListBorderAir a partir dos elementos da lista ListBorderFaceAir, onde as diferenças entre essas duas listas são:

      - Na ListBorderAir não existem elementos repetidos;
      - O flag de uma célula `AIR` inserida na ListBorderAir passa a ter o bit 256 acesso, caracterizando que a célula já foi inserida em tal lista;

- Existe uma lista associada a ListBorderAir, ListCostAir, que informa a direção entre a célula `FLUID` vizinha que deu origem a inserção desta célula `AIR` e a própria célula `AIR` que está sendo inserida (1 = direção do eixo x, 2 = direção do eixo y, e 4 = direção do eixo z), o registro de tal direção tem por finalidade facilitar a inserção futura de células `AIR` vizinhas por aresta ou vizinhas por vértice a célula `FLUID` em questão.

(e) Cria uma nova lista denominada ListBorderFluid a partir dos elementos da lista ListBorderFaceFluid, onde as diferenças entre essas duas listas são:

- Na ListBorderFluid não existem elementos repetidos;
- O flag de uma célula `FLUID` inserida na ListBorderFluid passa a ter o bit 256 acesso, caracterizando que a célula já foi inserida em tal lista;
- Existe uma lista associada a ListBorderFluid, ListCostFluid, que informa a direção entre a célula `AIR` vizinha que deu origem a inserção desta célula `FLUID` e a própria célula `FLUID` que está sendo inserida, o registro de tal direção tem por finalidade facilitar a inserção futura de células `FLUID` vizinhas por aresta ou vizinhas por vértice a célula `AIR` em questão.

(f) Chama a função **void Navier::DiagonalBorderCells(...)** para a lista $ListBorderAir$, a qual:

i. Insere em $ListBorderAir$ as células de "algum tipo `AIR`" (`AIR`, `BORDER_AIR`, `PROCESSED_BORDER_AIR`, `SLIP_AIR` ou `OUTFLOW_AIR`) vizinhas por aresta ou por vértice a uma célula `FLUID` (sem repetição).

ii. Verifica se uma das células inseridas em $ListBorderAir$ passou a ter o valor de $gridPhi < 0$, e em caso afirmativo, ajusta seu rótulo para rótulo = rotulo + 1024, e a insere na lista $Navier :: ListAirToFluid$.

Observe que uma dada célula deve ser inserida em $ListBorderAir$ mesmo que tenha sido inserida em $AirToFluid$ uma vez que a mesma precisa ter sua velocidade iniciada.

(g) Chama a função **void Navier::DiagonalBorderCells(...)** para a lista $ListBorderFluid$, e atua de maneira semelhante ao item anterior;

Observe que a lista $ListBorderFluid$ será utilizada apenas na varredura em que se verifica quais células do tipo `FLUID` passaram a ser do tipo `AIR`, enquanto a lista $ListBorderAir$ é utilizada na inicialização da velocidade nas novas células do tipo `FLUID`.

(h) Chama a função **void Navier::ProcessingBorderAirList**, a qual:

i. Retira as células do tipo SlipAir ou OutflowAir da lista $ListBorderAir$, visto que o valor da velocidade em suas faces não vai ser atualizado aqui mas em rotina específica para isso. Essas células retiradas são passadas para uma lista específica, $Navier :: BorderSlipOutflowAir$.

ii. Para cada uma das vizinhas por face de uma célula da lista $ListBorderAir$, verificamos se o seu label também é `BORDER_AIR`. Se for este o caso seja $D$ a direção ortogonal à face comum com a vizinha. Verificamos se $gridCellUVW(D, celulaVizinha) + gridCellUVW(D, celulaCorrente)$ tem o sentido da célua vizinha para a célula corrente. Se for esse o caso se "acende o bit" relativo a esse vizinho na posição corrente em um vetor de inteiros, $antecessorBorder[currPosition]$, que indicará então que esta célula vizinha do tipo `BORDER_AIR` deve ser inicializada antes da célula corrente. Os vizinhos (faces) de uma célula são rotuladas da seguinte maneira, ver figura B.6:

A. vizinho a direita = 1;

B. vizinho a esquerda = 2;

C. vizinho ao topo = 4;

D. vizinho a base = 8;

E. vizinho ao topo = 16;

F. vizinho a fundo = 32

iii. Se $antecessorBorder[currPosition] = 0$, ou seja, nenhum bit acesso, então a célula corrente não tem precedentes na lista $listBorderAir$

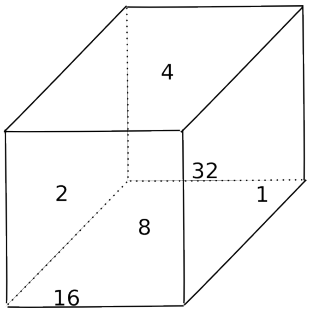



Figura B.6: Rótulo em decimal das faces de uma célula

ainda não processados e portanto pode ser inicializada e retirada da lista. A inicialização é feita pela rotina **OrderedNewFluidCellInitializer(...)** , que tem como parametro a célula corrente

iv.

# Apêndice C

# Visualização

Além da validação numérica dos modelos aplicados, uma visualização realística do fluxo é de grande importância pois possibilita a validação intuitiva. Logo, apresentamos aqui os principais passos no processo de visualização do fluxo.

A principal ferramente utilizada no processo de visualização é o pacote de rotinas POV-Ray, o qual é um pacote open source utilizado para criar imagens tridimensionais, foto-realistas utilizando a técnica ray-tracing [referência]. Em síntese POV-Ray é utilizado para ler um arquivo texto que descreve os objetos e a iluminação em uma cena, e gerar uma imagem dessa cena do ponto de vista de uma câmera também descrita no arquivo texto. O ray-tracing não é um processo rápido, mas produz imagens de alta qualidade com reflexões, sombreado, perspectiva e outros efeitos realistas.

# Referências Bibliográficas


[1] Adalsteinsson, D., and Sethian, J. A. The fast construction of extension velocities in level set methods. *Journal of Computational Physics 148* (1997), 2–22.

[2] Ajol, J. F., and Aubert, G. Signed distance functions ann viscosity solutions of discontinuous hamilton jacobi equations. Tech. rep., July 2002. INRIA.

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

[4] Amsden, A. A., and Harlow, F. H. A simplified mac technique for incompressible fluid flow calculations. *Journal of Computational Physics 6*, 2 (1970), 322–325.

[5] Atencio, Y. P., Alzamora, G. S., and Esperança, C. Enhanced physically-based animation of deformable bodies using shape-matching. *Comput. Entertain. 7*, 4 (2009), 1–19.

[6] Aulisa, E., Manservisi, S., and Scardovelli, R. A surface marker algorithm coupled to an area-preserving marker redistribution method for three-dimensional interface tracking. *Journal of Computational Physics 197*, 2 (2004), 555–584.

[7] Bargteil, A. W., Goktekin, T. G., O'brien, J. F., and Strain, J. A. A semi-lagrangian contouring method for fluid simulation. *ACM Trans. Graph. 25* (January 2006), 19–38.

[8] Blazek, J. *Computational Fluid Dynamics: Principles and Applications.* ELSEVIER, Kidlington, Oxford OX5 IGB, UK, 2001.

[9] Brackbill, J. U. Introduction to harlow's scientific memoir. *J. Comput. Phys. 195* (April 2004), 413–413.

[10] Bridson, R. *Fluid Simulation for Computer Graphics.* A K Peters, Ltd, Wellesley, Massachusetts, 2008.

[11] Brochu, T., and Bridson, R. Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing 31*, 4 (2009), 2472–2493.

[12] Bronstein, A. M., and Bronstein, M. M. *Numerical Geometry of Non-Rigid Shapes.* Springer, New York, NY 10013, USA, 2008.

[13] Bruce R. Munson, Donald F. Young, T. H. O., and Huebsch, W. W. *Fundamentals of Fluid Mechanics.* John Wiley and Sons Inc., Ames, Iowa, USA, 2006.

[14] Carlson, M. *RIGID, MELTING, AND FLOWING FLUID.* PhD thesis, College of Computing Georgia Institute of Technology, 2004.

[15] Carlson, M., Mucha, P. J., and Turk, G. Rigid fluid: animating the interplay between rigid bodies and fluid. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 377–384.

[16] Cecil, T. C., Osher, S. J., and Qian, J. Essentially non-oscillatory adaptive tree methods. *J. Sci. Comput. 35*, 1 (Apr. 2008), 25–41.

[17] Chen, F., and Hagen, H. A Survey of Interface Tracking Methods in Multiphase Fluid Visualization. In *Visualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling and Engineering (IRTG 1131 Workshop)* (Dagstuhl, Germany, 2011), A. Middel, I. Scheler, and H. Hagen, Eds., vol. 19 of *OpenAccess Series in Informatics (OASIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 11–19.

[18] Chentanez, N., Feldman, B. E., Labelle, F., O'Brien, J. F., and Shewchuk, J. R. Liquid simulation on lattice-based tetrahedral meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, pp. 219–228.

[19] Chernyaev, E. V. Marching cubes 33: Construction of topologically correct isosurfaces. Tech. rep., Institute for High Energy Physics, 1995.

[20] Chorin, A. J. A numerical method for solving incompressible viscous flow problem. *JOURNAL OF COMPUTATIONAL PHYSICS 135* (1997), 118–125.

[21] CRANE, L., and TARIQ. *Real Time Simulation and Rendering of 3D Fluids In GPU Gems 3 ed.* Addison Wesley Edited H. Nguyen, Boston, MA, 2008.

[22] Desjardins, O., and Pitsch, H. A spectrally refined interface approach for simulating multiphase flows. *J. Comput. Phys. 228*, 5 (Mar. 2009), 1658–1677.

[23] Dijkstra, E. W. *A Discipline of Programming.* Prentice-Hall, New York, NY 10013, USA, 1976.

[24] Enright, D., Fedkiw, R., Ferziger, J., and Mitchell, I. A hybrid particle level set method for improved interface capturing. *J. Comput. Phys 183* (2002), 83–116.

[25] Enright, D., Marschner, S., and Fedkiw, R. Animation and rendering of complex water surfaces. *ACM Trans. Graph. 21* (July 2002), 736–744.

[26] Enright, D., Nguyen, D., Gibou, F., and Fedkiw, R. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In *In Proc. 4th ASME-JSME Joint Fluids Eng. Conf., number FEDSM2003-45144. ASME* (2003), pp. 2003–45144.

[27] Evans, M., Harlow, F., and Bromberg, E. The particle-in-cell method for hydrodynamic calculations. *J. ACM 4* (April 1957), 137–142.

[28] Fedkiw, R., Stam, J., and Jensen, H. W. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 15–22.

[29] Ferreira, V. G. Modelagem metematica e simulacao numérica em dinamica dos fluidos. *Notas em Matemática Aplicada-BMAC 15* (2005), 68 p.

[30] Ferreira, V. G., Kurokawa, F. A., Queiroz, R. A. B., Kaibara, M. K., Oishi, C. M., Cuminato, J. A., Castelo, A., Tomé, M. F., and McKee, S. Assessment of a high-order finite difference upwind scheme for the simulation of convection–diffusion problems. *International Journal for Numerical Methods in Fluids 60*, 1 (2009), 1–26.

[31] Foster, N., and Fedkiw, R. Practical animation of liquids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 23–30.

[32] G, M., and V, P. M. Modifications to the two-dimensional incompressible fluid flow code zuni to provide enhanced performance. Tech. rep., 1983. C.E.G.B. Report TPRD /L/0063/M82.

[33] Gaskell, P. H., and Lau, A. K. C. Curvature-compensated convective transport: Smart, a new boundedness - preserving transport algorithm. *International Journal for Numerical Methods in Fluids 8 Issue 6* (2005), 617 – 641.

[34] Geiger, W., Leo, M., Rasmussen, N., Losasso, F., and Fedkiw, R. So real it'll make you wet. In *ACM SIGGRAPH 2006 Sketches* (New York, NY, USA, 2006), SIGGRAPH '06, ACM.

[35] Glimm, J., Grove, J. W., Li, X. L., and Zhao, N. Simple front tracking, 1999.

[36] Guendelman, E., Selle, A., Losasso, F., and Fedkiw, R. Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph. 24* (July 2005), 973–981.

[37] GUEYFFIER, LI, NADIM, SCARDOVELLI, AND ZALESKI. Volume-of-fluid interface tracking with smoothed surface stress methods for three-dimensional flows. *J.Comput. Phys 152*, 2 (1999), 423–456.

[38] H., F., AND HARLOW. Fluid dynamics in group t-3 los alamos national laboratory: (la-ur-03-3852). *Journal of Computational Physics 195*, 2 (2004), 414–433.

[39] HARLOW, F. H. Hydrodynamic problems involving large fluid distortions. *J. ACM 4* (April 1957), 137–142.

[40] HARLOW, F. H. Pic and its progeny. *Computer Physics Communications 48* (Jan. 1988), 1–10.

[41] HARLOW, F. H., AND AMSDEN, A. A. A numerical fluid dynamics calculation method for all flow speeds. *Journal of Computational Physics 8*, 2 (1971), 197–213.

[42] HARLOW, F. H., AND WELCH, J. E. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids 8*, 12 (1965), 2182–2189.

[43] HEO, N., AND KO, H. Detail-preserving fully-eulerian interface tracking framework. *ACM Transactions on Graphics 29*, 6 (2010), 176.

[44] HIRT, C., AND NICHOLS, B. Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of Computational Physics 39*, 1 (1981), 201–225.

[45] HOUSTON, B., NIELSEN, M. B., BATTY, C., NILSSON, O., AND MUSETH, K. Hierarchical rle level set: A compact and versatile deformable surface representation. *ACM Trans. Graph. 25*, 1 (Jan. 2006), 151–175.

[46] JF, M., AND P, R. Outline description of a recently implemented fluid flow code zuni. Tech. rep., 1981. C.E.G.B. Report LM /PHYS/0063/258.

[47] KASS, M., AND MILLER, G. Rapid, stable fluid dynamics for computer graphics. *SIGGRAPH Comput. Graph. 24* (September 1990), 49–57.

[48] Kim, D., Song, O.-y., and Ko, H.-S. A Semi-Lagrangian CIP Fluid Solver without Dimensional Splitting. *Computer Graphics Forum 27*, 2 (Apr. 2008), 467–475.

[49] Kim, D., Song, O.-y., and Ko, H.-S. Stretching and wiggling liquids. In *ACM SIGGRAPH Asia 2009 papers* (New York, NY, USA, 2009), SIGGRAPH Asia '09, ACM, pp. 120:1–120:7.

[50] Kohno, H., and Tanahashi, T. Numerical analysis of moving interfaces using a level set method coupled with adaptive mesh refinement. *International Journal for Numerical Methods in Fluids 45*, 9 (2004), 921–944.

[51] LeVeque, R. J. *Finite Volume Methods for Hyperbolic Problems*. CAMBRIDGE UNIVERSITY PRESS, New York, NY 10011-4211, USA, 1985.

[52] Lopes, F. Fenômenos de transporte i. UNIVERSIDADE FEDERAL DA BAHIA, ESCOLA POLITÉCNICA, DEPARTAMENTO DE ENGENHARIA QUÍMICA, 2009.

[53] Lorensen, W. E., and Cline, H. E. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21* (August 1987), 163–169.

[54] Losasso, F., Gibou, F., and Fedkiw, R. Simulating water and smoke with an octree data structure. *ACM Trans. Graph. 23* (August 2004), 457–462.

[55] Losasso, F., Talton, J., Kwatra, N., and Ronald, F. Two-way coupled sph and particle level set fluid simulation. *IEEE Transactions on Visualization and Computer Graphics 14*, 4 (2008), 797–804.

[56] M., J., and Hyman. Numerical methods for tracking interfaces. *Physica D: Nonlinear Phenomena 12*, 1-3 (1984), 396–407.

[57] M. Griebel, T. D., and Neunhoeffer, T. *Numerical Simulation in Fluid Dynamics, a Practical Introduction*. SIAM, Philadelphia, PA, 1998.

[58] M. Kass, A. W., and Terzopoulos, D. Snakes: Active contour models. *International Journal of Computer Vision l(4)* (1987), 321–331.

[59] McDonough, J. M. Lectures in computational fluid dynamics of incompressible flow: Mathematics, algorithms and implementations. Departments of Mechanical Engineering and Mathematics University of Kentucky, 2007.

[60] McKee, S., Tomé, M., Ferreira, V., Cuminato, J., Castelo, A., Sousa, F., and Mangiavacchi, N. The mac method. *Computers Fluids 37*, 8 (2008), 907–930.

[61] Miyata, H. Finite-difference simulation of breaking waves. *J. Comput. Phys. 65* (July 1986), 179–214.

[62] Miyata, H., Nishimura, S., and Masuko, A. Finite difference simulation of nonlinear waves generated by ships of arbitrary three-dimensional configuration. *Journal of Computational Physics 60*, 3 (1985), 391–436.

[63] Osher, S., and Fedkiw, R. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, New York, NY, USA, 2003.

[64] Saad, Y. *Iterative methods for sparse linear systems; 2nd ed.* SIAM, Philadelphia, PA, 2003.

[65] Sethian, J. *Level Set Method: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, New York, NY, 1996.

[66] Sethian, J., and Osher, S. Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations. *J. Computer Phys.*, 79 (1988).

[67] Shu, C.-W., and wang Shu, C. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. Springer, pp. 325–432.

[68] Sicilian, J., and Hirt, C. An efficient computation scheme for tracking contaminant concentrations in fluid flows. *Journal of Computational Physics 56*, 3 (1984), 428–447.

[69] Sod, G. A. *Numeric methods in fluid dynamics.* CAMBRIDGE UNIVERSITY PRESS, New York, NY 10022, USA, 2002.

[70] Song, B., Liu, G. R., Lam, K. Y., and Amano, R. S. On a higher-order bounded discretization scheme. *International Journal for Numerical Methods in Fluids 32*, 7 (2000), 881–897.

[71] Soo Kim, M., and Lee, W. I. A new vof-based numerical scheme for the simulation of fluid flow with free surface. part i: New free surface-tracking algorithm and its verification. *International Journal for Numerical Methods in Fluids 42*, 7 (2003), 765–790.

[72] Soo Kim, M., Sun Park, J., and Lee, W. I. A new vof-based numerical scheme for the simulation of fluid flow with free surface. part ii: application to the cavity filling and sloshing problems. *International Journal for Numerical Methods in Fluids 42*, 7 (2003), 791–812.

[73] Stam, J. Stable fluids. *In SIGGRAPH 99 Conference Proceedings, Annual Conference Series 15* (1999), 121–128.

[74] Sweby, P. K. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM Journal on Numerical Analysis 21* (1984), 995–1011.

[75] Tome, M. F., and McKee, S. Gensmac: A computational marker and cell method for free surface flows in general domains. *Journal of Computational Physics 110* (Jan. 1994), 171–186.

[76] Tomé, M. F., Filho, A. C., Cuminato, J. A., Mangiavacchi, N., and McKee, S. Gensmac 3d: A numerical method for solving unsteady three-dimensional free surface flows. *International Journal for Numerical Methods in Fluids 37*, 7 (2001), 747–796.

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

[78] Wojtan, C., Thurey, N., Gross, M., and Turk, G. Deforming meshes that split and merge. *ACM Trans. Graph. 28* (July 2009), 76:1–76:10.

[79] Wojtan, C., Thurey, N., Gross, M., and Turk, G. Physics-inspired topology changes for thin fluid features. *ACM Trans. Graph. 29* (July 2010), 50:1–50:8.