Simulação de Escoamentos com Superfícies Livres em um Ambiente de Memória Distribuida

Maurílio Boaventura

Prof. Dr. José Alberto Cuminato
Orientador

Prof[a]. Dr[a]. Maria Cristina Castro Cunha
Co-orientadora

Tese apresentada ao Instituto de Matemática, Estatística e Computação Científica, UNICAMP, como requisito parcial para a obtenção do Título de Doutor em Matemática Aplicada.

Simulação de Escoamentos com Superfícies Livres em um Ambiente de Memória Distribuida

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida por Maurílio Boaventura e aprovada pela comissão julgadora.

Campinas, 28 de julho de 1998.

Prof. Dr. José Alberto Cuminato
Orientador

Profª Drª Maria Cristina C. Cunha
Co-orientadora

Tese apresentada ao Instituto de Matemática, Estatística e Computação Científica, UNICAMP, como requisito parcial para obtenção do Título de DOUTOR em Matemática Aplicada.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Boaventura, Maurílio

B63s Simulação de escoamentos com superfícies livres em um ambiente de memória distribuída / Maurílio Boaventura -- Campinas, [S.P. :s.n.], 1998.

Orientadores : José Alberto Cuminato, Maria Cristina C. Cunha

Tese (doutorado) - Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Navier-Stokes, Equações de - Solução numérica. 2. Processamento paralelo. I. Cuminato, José Alberto. II. Cunha, Maria Cristina Castro. III. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. VI. Título.

Tese de Doutorado defendida e aprovada em 28 de julho de 1998

Pela Banca Examinadora composta pelos Profs. Drs.

Prof (a). Dr (a). JOSÉ ALBERTO CUMINATO

Prof (a). Dr (a). RUDNEI DIAS DA CUNHA

Prof (a). Dr (a). SAULO RABELLO MACIEL DE BARROS

Prof (a). Dr (a). JOSÉ VITÓRIO ZAGO

Prof (a). Dr (a). JOSÉ LUIZ BOLDRINI

Aos meus filhos Fábio e Marcelo e a minha esposa Inês.

Agradecimentos

Ao Prof. José Alberto Cuminato, pela orientação deste trabalho, pela amizade e pela confiança em mim depositada.

À Profª. Maria Cristina Castro Cunha, orientadora do programa de doutorado, pela confiança em mim depositada, pela amizade e ótimo convívio durante minha estada na UNICAMP.

Ao pessoal do CENAPAD/SP-UNICAMP, pela assessoria e por atenderem protamente aos meus pedidos.

Aos Professores Antonio Castelo Filho, do ICMSC-USP-SÃO CARLOS e Murilo Francisco Tomé, do Inst. Técnico de Lisboa, pela ajuda e constante estímulo durante a realização deste trabalho.

À minha família, por tanto carinho e confiança e também pela compreensão à minha ausência constante.

A três grandes amigas: Amélia, Eliana e Magda, que numca me deixaram desistir.

Aos colegas do DCCE/IBILCE/UNESP - São José do Rio Preto, especialmente aos da área de Análise Numérica, pelo estímulo e apoio, que foram decisivos à realização deste trabalho.

Aos amigos do curso de doutorado, pelos bons momentos que me proporcionaram.

Aos colegas do DMA e da Seção de Pós-Graduação do IMECC-UNICAMP, pela ajuda e apoio constante.

Aos professores do DMA-IMECC-UNICAMP, com os quais tive o prazer de conviver.

À CAPES/PICDT, pelo apoio financeiro.

A todos que direta ou indiretamente colaboraram para a realização deste trabalho.

Resumo

Apresentamos, neste trabalho, uma técnica paralela baseada em uma decomposição de domínio para resolver as equações de Navier-Stokes com superfícies livres em coordenadas cartesianas e cilíndricas em duas dimensões. Essa técnica é baseada no código apresentado por Tomé [1993] e Tomé e co-autores [1996], a qual por sua vez é baseada no método SMAC apresentado por Amsden e Harlow [1971], que resolve as equações de Navier-Stokes em três passos: a equação de momento, a equação de Poisson e o movimento das partículas. A primeira equação é discretizada por diferenças finitas explícitas. A paralelização é realizada dividindo-se o domínio original de cálculo em vários subdomínios verticais e atribuindo cada um deles a um processador. Todos os cálculos podem ser realizados usando comunicação somente com o processador vizinho mais próximo. No final, apresentamos testes comparando a performance do código paralelo com o sequencial e discutimos a questão do balanceamento de carga.

Abstract

A parallel technique based on domain decomposition for solving free surface Navier-Stokes equations in cartesian and cylindrical coordinates in two dimensions is described. It is based on the code by Tomé [1993] and Tomé et.al. [1996], which in turn is based on the SMAC method by Amsden & Harlow [1971], which solves the Navier-Stokes equations in three steps: the momentum equation and Poisson solvers and particle moviment. The first equation is discretized by explicit finite diferences. The parallelization is performed by splitting the computation domain into vertical strips and assigning each of these to a processor. All the computation can then be performed using nearest neighbour communication. We present run tests comparing the performance of the parallel with the serial code, and discuss the load balancing question.

Conteúdo

Capítulo 1

Introdução

A solução numérica de muitos problemas em dinâmica dos fluidos está se tornando cada vez mais viável graças aos recentes avanços da computação moderna, tanto no desenvolvimento de máquinas, favorecendo um aumento na velocidade de cálculos, quanto na área de programas, possibilitando técnicas computacionais muito mais eficientes do que há alguns anos atrás. Entre as classes mais importantes de problemas de dinâmica dos fluidos está a classe de problemas de escoamentos com superfícies livres, que são problemas presentes em nossa vida diária, tais como ondas quebrando na praia e água escorrendo de uma torneira. Poderíamos ficar horas citando exemplos desse tipo de problemas, porém destacamos os problemas industriais que, por razões econômicas, têm despertado grande interesse nos últimos anos. Entre os problemas industriais estão os das indústrias química, alimentícia e metalúrgica que necessitam preencher frascos ou moldes com determinado fluido.

A investigação desses problemas não é tarefa fácil devido à diversos fatores: a superfície livre pode estar continuamente mudando sua posição com o tempo e sua forma ser desconhecida. Surgem, também, complicações na aplicação das condições de contorno, que precisam ser satisfeitas sobre a superfície livre e, além do mais, muitos problemas de interesse prático envolvem a investigação de escoamentos em geometrias extremamente complexas.

Os problemas de escoamentos de fluidos viscosos incompressíveis bi-dimensionais são modelados pelas equações de Navier-Stokes, ou seja, as equações de conservação de massa:

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial (uv)}{\partial y} = -\frac{\partial p}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + g_x,$$

$$\frac{\partial v}{\partial t} + \frac{\partial (uv)}{\partial x} + \frac{\partial v^2}{\partial y} = -\frac{\partial p}{\partial y} + \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + g_y,$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0.$$

Em princípio existem várias técnicas de aproximação que podem ser utilizadas para resolver numericamente problemas de escoamentos com superfícies livres. Por exemplo, técnicas de elementos finitos em geometrias complexas foram propostas por Nickell, Tanner e Caswell [26], para resolver problemas estacionários e, também, por Graham, Burley e Carling [16], para problemas não estacionários. Técnicas Lagrangeanas de cálculos, propostas por Hirt, Cook e Butler [19], ou mistas, Lagrangeanas-Eulerianas, propostas por Chuan [7], foram empregadas para escoamentos em duas ou três dimensões. Contudo, a aproximação mais comum tem sido a aproximação Euleriana juntamente com técnicas de diferenças finitas.

O método MAC ("Marker-and-Cell") apresentado por Harlow e Welch [17], em 1965, foi um dos primeiros algoritmos bem sucedidos para resolver numericamente escoamentos com superfícies livres, e desde então esse método tem sido o protótipo para o desenvolvimento de técnicas similares.

1.1 O Método MAC e sua Evolução

O método MAC é uma técnica de diferenças finitas baseada em uma malha diferenciada para resolver numericamente as equações Eulerianas dependentes do tempo para escoamentos de fluidos incompressíveis, viscosos e com superfícies livres. Uma das principais características desse método é a utilização de partículas virtuais sem massa para representar o fluido. Estas partículas são conduzidas pelo campo de velocidade e são utilizadas para indicar a movimentação do fluido em qualquer instante de tempo, proporcionando, desta forma, uma visualização dos resultados. O propósito essencial da utilização dessas partículas é definir a posição da superfície livre, de modo que as condições de contorno possam ser impostas sobre ela. Contudo, em sua formulação original o método MAC se mostrou ineficiente no tratamento da equação de Poisson e na imposição das condições de contorno. Para superar essas dificuldades Amsden e Harlow [2] desenvolveram, em 1971, o método SMAC ("Simplified Marker-and-Cell"). Nesse método o ciclo de cálculo é dividido em duas partes: a primeira parte para calcular provisoriamente o campo de velocidade e a segunda para uma revisão na velocidade calculada anteriormente, empregando, para isso, uma função potencial de modo a satisfazer a equação de continuidade e assegurar a conservação de massa. Neste caso a equação de Poisson obtida é mais simples e requer somente condições de contorno homogêneas.

A metodologia SMAC, resolve as equações de Navier-Stokes em três passos: a equação de momento, a equação de Poisson e o movimento das partículas. As equações de momentos são discretizadas por fórmulas de diferenças finitas explícitas sobre uma malha diferenciada, ou seja, uma malha onde algumas variáveis são calculadas nos pontos de discretização e outras com pequeno deslocamento. A equação de Poisson é discretizada pela fórmula de 5 pontos aplicada no ponto central da malha.

Amsden e Harlow escreveram um programa específico, chamado ZUNI, que emprega toda a metodologia SMAC. Esse código foi desenvolvido para simular escoamentos de fluidos bidimensionais em coordenadas retangulares e cilíndricas, sujeitos a contornos de entrada e de saida e condições de contorno de escorregamento livre ou de não escorregamento impostas sobre as paredes do contorno rígido. Além do mais, um obstáculo retangular pode ser incorporado à região de solução. No entanto, o domínio de solução é suposto retangular e é restrito a duas dimensões.

Nos anos que sucederam o desenvolvimento dos métodos MAC e SMAC, vários outros métodos empregando metodologias similares foram apresentados. Alguns introduzindo modificações para melhorar a performance, outros para solucionar problemas específicos e outros, ainda, para ampliar a aplicabilidade desses métodos, como por exemplo, a extensão do domínio de solução para domínios mais gerais e consequentemente mais complexos.

O quebrar das ondas, por exemplo, tem várias aplicações em problemas de engenharia. Com este propósito, Chan e Street [7], apresentaram, em 1971, o método SUMMAC ("Stanford University Modified Marker and Cell"), para simular ondas solitárias na água. Subsequentemente, nos anos de 1985 e 1986, Miyata e Sishimura [23] e [24], desenvolveram o método TUMMAC ("Tokyo University Modified Marker and Cell"), para simular ondas geradas por navios de configurações arbitrárias e para ondas quebrando sobre corpos circulares ou elípticos.

Para aumentar a performance do código SMAC, que no caso de problemas com número de Reynolds pequeno impõe uma séria limitação no tamanho do passo, Deville [13], Pradht [29] e Golafshani [15] introduziram esquemas do tipo implícito. Em particular, Pracht [30] desenvolveu um tratamento implícito para a velocidade, usando malha computacional Lagrangeana-Euleriana arbitrária, permitindo, com isso, o cálculo de escoamentos envolvendo contornos irregulares e/ou em movimento. Mais recentemente, McQueen e Rutter [21] e Markham e Proctor [20] desenvolveram modificações para o código ZUNI com a finalidade de aumentar a performance. Basicamente eles empregaram uma rotina automática para calcular o passo no tempo e o método dos gradientes conjugados com pré-condicionamento para o tratamento da equação de Poisson.

Uma solução baseada na filosofia original MAC, para condições de contorno de escorregamento livre impostas em um domínio arbitrário foi apresentada por Viecelli [37] e [38].

Este método é aplicável mesmo nos casos em que todos os contornos têm forma arbitrária e os pontos da malha não coincidem com os pontos do contorno. Os cálculos relatados até o momento são para duas dimensões, mas em princípio a técnica pode ser estendida para três. O método, chamado ABMAC ("Arbitrary Marker and Cell") por Viecelli, foi aplicado a vários problemas específicos e bons resultados foram relatados. Contudo sua extensão para a filosofia SMAC não está clara, principalmente quando são impostas condições de não escorregamento sobre contornos arbitrários.

Uma das grandes dificuldades no desenvolvimento de métodos do tipo MAC é a aplicação das condições de tensão sobre a superfície livre, uma vez que a superfície livre não é conhecida com exatidão e, além do mais, está continuamente se modificando. Originalmente as condições de tensão no método MAC foram incorporadas impondo pressão igual a zero em todas as células da superfície. Para melhorar essas condições, Hirt e Sharnnon [18] desenvolveram aproximações com expressões de diferenças finitas bastante simples. Supondo pequena a curvatura local da superfície livre, as equações de tensão normal e tangencial se tornam bastante simplificadas, pois a superfície livre é considerada localmente horizontal ou vertical. Expressões para uma curva na qual a direção normal à superfície livre apresenta uma inclinação de 45^0 também foram desenvolvidas. Contudo, essas condições foram aplicadas nos centros das células da superfície. Subsequentemente Nichols e Hirt [25] apresentaram um tratamento melhor, impondo essas condições sobre a superfície livre ao invés de nos centros das células.

Mais recentemente, em 1994, incorporando a metodologia SMAC e influenciados por muitos dos trabalhos mencionados anteriormente, principalmente pelos trabalhos de McQueen e Rutter [21], Markham e Proctor [20] e Viecelli [37] e [38], Tomé e McKee [32] desenvolveram o método GENSMAC para resolver numericamente escoamentos com superfícies livres em domínios gerais. O método é uma extensão do método SMAC. As principais características do método GENSMAC são: o tratamento de contornos irregulares, principalmente quando condições de não escorregamento são impostas sobre eles e até então não consideradas em outros métodos; o método possui uma rotina para calcular automaticamente o espaçamento no tempo; utiliza o método dos gradientes conjugados para a equação de Poisson; contém aproximações bastante precisas para as condições de tensão sobre a superfície livre; pode simular escoamentos envolvendo vários contornos de entrada e de saida e várias superfícies livres, mais detalhes deste trabalho podem ser encontrados em Tomé [33]. A idéia empregada por esses autores para tratar o contorno irregular, foi aproximá-lo por um outro coincidindo com as linhas da malha (contorno virtual) e, então, resolver a equação de Poisson no domínio definido pelo contorno virtual. As condições de contorno para a velocidade são impostas sobre o contorno irregular original.

Motivados pela potencial aplicabilidade do método SMAC para simular a injeção de mate-

rial em moldes de configurações arbitrárias e complexas, Tomé e McKee, desenvolveram uma ferramenta bastante poderosa para tratar problemas de escoamentos de fluidos viscosos incompressíveis em domínios gerais. Inicialmente a metodologia GENSMAC foi implementada em coordenadas cartesianas em um domínio geral bi-dimensional. No final de 1996, Tomé e co-autores [34], apresentaram uma nova versão do código, na qual foi incluida a solução em coordenadas cilíndricas para solucionar problemas tri-dimensionais eixo-simétricos.

Uma das grandes restrições na implementação do método SMAC e, consequentemente, GENSMAC, é o armazenamento das partículas virtuais, uma vez que são alocadas posições de memória para cada uma delas e isso traz um inconveniente muito grande para malhas muito finas ou problemas grandes que envolvam o armazenamento de muitas partículas. Além do mais, a movimentação dessas partículas no ciclo de cálculo, dependendo da quantidade, pode consumir mais da metade do tempo de processamento total. Para contornar essas restrições, Tomé e co-autores [35], apresentaram uma outra versão da implementação do método GENSMAC, na qual são armazenadas e movimentadas somente as partículas que representam a superfície livre. Com isso, houve uma diminuição substancial no tempo de processamento e, também, no armazenamento das partículas, cujas posições de memória são agora alocadas dinamicamente.

1.2 Contribuição Original deste Trabalho

Dentre os problemas em mecânica dos fluidos a classe de problemas de escoamento de fluidos incompressíveis tem recebido especial destaque nas últimas décadas. Como os equacionamentos matemáticos que modelam esses problemas, na maioria das vezes não apresenta solução exata conhecida, se faz necessário a busca de técnicas matemáticas de aproximações. Neste contexto as filosofias MAC, SMAC e suas variantes, são os protótipos para esses modelos. Dentro dessa filosofia a metodologia GENSMAC se mostrou uma ferramenta poderosa e é um dos algoritmos que vem comprovando ser eficaz e robusto na solução de tais problemas.

Apesar de todos os avanços ocorridos na metodologia GENSMAC, no sentido de diminuir o tempo de processamento numérico, o tempo gasto na execução do código para problemas grandes ou com malhas muito finas, ainda é uma séria restrição, pois, com a discretização do problema, o espaçamento no tempo pode ser muito pequeno e a execução do algoritmo se tornar bastante lenta, demorando demais para atingir o tempo de escoamento desejado. Com isso surge, ainda, a necessidade de se testar técnicas computacionais eficientes que possam realmente contribuir nesse sentido.

O foco das atenções de muitos pesquisadores em todo o mundo nos últimos anos, tem sido as técnicas de paralelização de algoritmos computacionais, pois esta é uma maneira de se atingir um alto desempenho computacional. Neste contexto, nossa contribuição é propor uma técnica de decomposição de domínio, onde o domínio original de solução é decomposto em vários subdomínios, de tal modo que o mesmo algoritmo possa ser executado paralelamente em cada um deles, buscando, desta forma, atingir um melhor desempenho computacional quando comparado com o código sequencial.

Acreditamos que a técnica de decomposição de domínios que estamos propondo possa ser estendida para outras metodologias de processamento similares à metodologia apresentado por Tomé e co-autores. No entanto, até o momento, realizamos sua implementação apenas para a metodologia GENSMAC, pois, em nossa opinião, é a metodologia que vem apresentando o melhor desempenho.

A técnica de decomposição que estamos propondo consiste em decompor o domínio de solução em vários subdomínios de mesmo tamanho, ou o mais próximo possível desta situação, dividindo o domínio original com cortes verticais e aplicando o mesmo algoritmo sobre cada um deles, ou seja, o mesmo algoritmo deve operar sobre diferentes conjuntos de dados.

As figuras 1.1 e 1.2 apresentadas a seguir retratam essa situação para um caso particular.

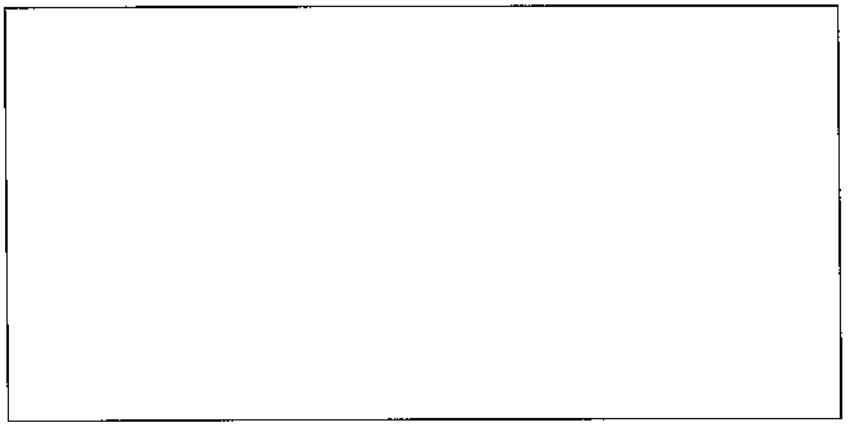


Figura 1.1: Domínio original de solução

Considerando um domínio de solução retangular como o mostrado na figura 1.1, a divisão em subdomínios toma a forma apresentada na figura 1.2, para o caso particular de três subdomínios.

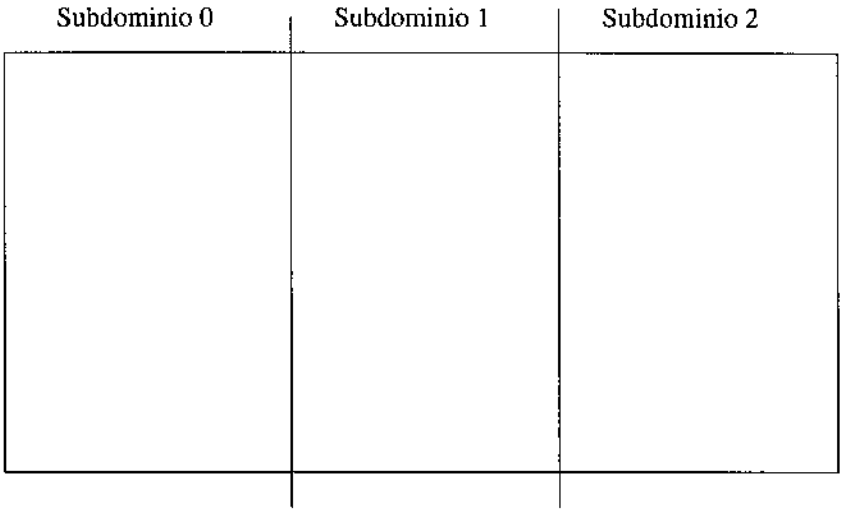


Figura 1.2: Divisão do domínio de solução

O mesmo algoritmo deve ser aplicado sobre cada um desses subdomínios. Os dados referentes ao subdomínio 0 devem ser processados pelo processador 0, os dados referentes ao subdomínio 1 devem ser processados pelo processador 1 e, assim por diante. No final da aplicação, a junção dos resultados obtidos em cada subdomínio fornecerá o resultado global procurado.

Como já mencionamos anteriormente, existem duas versões de códigos para a metodologia GENSMAC. Na primeira versão são utilizadas partículas virtuais para se representar toda a massa de fluido, enquanto que na segunda, são utilizadas partículas virtuais somente para a representação da superfície livre. Embora a segunda versão da implementação sequencial da metodologia GENSMAC já tenha representado uma diminuição significativa no tempo total de processamento computacional quando comparado com a execução da primeira, o tempo de processamento ainda não foi reduzido a limites razoáveis para problemas de grande porte que exijam muito tempo de processamento. Com a finalidade de reverter ou pelo menos amenizar essa situação apresentamos, neste trabalho, implementações paralelas baseadas na decomposição de domínio citada anteriormente para ambas as versões do código GENSMAC.

Apesar da técnica de decomposição de domínio que estamos propondo ser bastante simples, sua implementação em arquiteturas paralelas é bastante complexa, pois além das dificuldades normais que surgem da arte de desenvolver algoritmos paralelos, tivemos ainda que enfrentar as dificuldades advindas da geometria associada ao problema, principalmente quando surgem situações complexas em regiões próximas às vizinhanças de subdomínios e envolvendo mais de um subdomínio. Em um processo de escoamento de fluido, a superfície livre pode tomar formas arbitrárias, muitas vezes pode formar um "zigue-zagues" que, por sua vez, pode estar dividido entre dois subdomínios, conduzindo a um alto grau de dificuldade na implementação computacional. É muito difícil descrever e, em um simples relato

sem descrever minúcias, dar a idéia de toda a complexidade envolvida na problemática da implementação paralela.

A implementação paralela é descrita principalmente nos capítulos 4 e 6 deste trabalho. Gostaríamos, porém, de antecipar neste momento dois pontos de alta complexidade. O primeiro deles é a implementação paralela das condições de tensão sobre a superfície livre e, o segundo, é a questão da movimentação do fluido, representado por partículas virtuais, principalmente na versão do código na qual são utilizadas tais partículas somente para a representação da superfície livre.

A dificuldade da implementação paralela das condições de tensão sobre a superfície livre está diretamente relacionada a sua forma, que pode estar constantemente se modificando, conduzindo a situações bastante complexas, situações estas que podem localizar-se parte em um subdomínio parte em outro, exigindo uma implementação bastante cuidadosa, principalmente no sentido de abranger todas as situações que podem ocorrer.

Na primeira versão a implementação paralela, no tocante à movimentação do fluido na qual são armazenadas todas as partículas virtuais, requer estruturas de dados mais simples e, consequentemente, exige menor esforço no desenvolvimento do código paralelo. Nessa primeira versão, após serem atualizadas todas as posições das partículas, é realizado um teste para se verificar se elas devem permanecer nos subdomínios em que estão ou se devem ser transferidas para outros subdomínios. Esse processo de transferência ou de eliminação de partículas em cada subdomínio é realizado sem maiores problemas, pois essas partículas são armazenadas sem uma ordem pré-estabelecida na forma de pares ordenados. Já para a implementação paralela da segunda versão do código GENSMAC, na qual são armazenadas somente as partículas que representam a superfície livre houve a necessidade de técnicas de programação e estruturas de dados bem mais elaboradas, aumentando em muito o grau de dificuldade da implementação computacional.

Na implementação sequencial da movimentação do fluido dessa segunda versão do código GENSMAC, é utilizada uma lista orientada do tipo fechada, para se representar cada corpo de fluido. A orientação da lista é essencial para se determinar a localização do fluido. A implementação paralela dessa estrutura não é viável, pois um mesmo corpo de fluido pode estar distribuido por mais de um subdomínio, dentre os subdomínios considerados na decomposição. Desta forma, tivemos que alterar a estrutura de dados utilizada na implementação sequencial, pois a representação em forma de lista do tipo fechada não é mais admissível nesta situação. A estrutura adotada então, foi representar cada corpo de fluido por uma lista orientada do tipo aberta e dividida em componentes, sendo que cada subdomínio pode conter uma, mais de uma ou mesmo nenhuma componente desse corpo de fluido. A movimentação das partículas, com essa estrutura, tornou-se bem mais complexa, exigindo muito cuidado no sentido de cobrir todas as situações que podem ocorrer. Uma descrição mais

completa dessa implementação é apresentada no capítulo 6, dedicado inteiramente a esta técnica.

No final deste trabalho, apresentamos testes computacionais que foram realizados comparando-se o desempenho do código paralelo com o sequencial, o que comprova a eficiência da implementação e mostra que a decomposição de domínio realizada é adequada. Cabe-nos ressaltar ainda que o tipo de problema que estamos lidando, nem sempre permite uma distribuição equilibrada de trabalho entre os processadores que irão colaborar na solução do problema em questão. Para alguns problemas, uma certa quantidade de processadores pode causar um desbalanceamento na carga de trabalho que cada processador deve realizar, alguns processadores podem receber mais trabalho que outros, conduzindo a resultados de performance não muito satisfatórios. Para outros problemas, a mesma quantidade de processadores pode ser a ideal, proporcionando uma distribuição uniforme da carga de trabalho entre os processadores envolvidos. Desta forma, para se atingir o máximo em performance deve-se combinar o problema a ser resolvido com a quantidade de processadores a ser utilizada. Muitas vezes essa combinação é perfeitamente exeqüível.

Tomé e co-autores [31] apresentaram, no início de 1997, uma nova versão da metodologia e da implementação do código GENSMAC para a solução de problemas de escoamentos em domínios gerais tri-dimensionais, mas a implementação em paralelo dessa recente versão é, ainda, assunto para um futuro trabalho.

1.3 Considerações Finais

Nos capítulos seguintes, abordamos tópicos referentes ao modelamento matemático de problemas de escoamento de fluidos viscosos incompressíveis, as técnicas computacionais MAC, SMAC e GENSMAC para se solucionar numericamente esses problemas, a técnica de decomposição de domínio que estamos propondo, bem como aspectos das implementações dos algoritmos sequencial e paralelo e, no final, relatamos alguns resultados de testes de eficiência que realizamos.

Este trabalho está organizado da forma que descrevemos a seguir.

No capítulo 2, é apresentado o modelamento matemático para solucionar escoamentos de fluidos com superfícies livres e, também, o tratamento numérico utilizado no método MAC.

No capítulo 3, descrevemos a metodologia GENSMAC, fazendo comparações, quando necessário, com a metologia SMAC.

A técnica de decomposição de domínio que estamos propondo, bem como aspectos das

implementações sequencial e paralela são apresentados no capítulo 4.

Apresentamos a versão do método GENSMAC para solucionar problemas eixo-simétricos e, também, os aspectos computacionais tanto para o caso sequencial, como para o caso paralelo, no capítulo 5.

No capítulo 6, descrevemos a técnica utilizada para se representar apenas as partículas que constituem a superfície livre, bem como as dificuldades de implementação dessa técnica para o caso paralelo.

Alguns problemas que foram utilizados para testar o código paralelo durante o seu desenvolvimento, são apresentados no capítulo 7.

No capítulo 8, são apresentados resultados de performance obtidos comparando-se execuções dos códigos sequencial e paralelo para os mesmos problemas e, também, apresentamos uma análise desses resultados.

No capítulo 9, apresentamos as conclusões gerais e propostas de trabalhos futuros e, ao final, existe um apêndice com a estrutura dos códigos sequencial e paralelo.

Capítulo 2

Solução Numérica de Escoamento de Fluidos na Presença de Superfícies Livres

Neste capítulo, vamos descrever rapidamente o método MAC desenvolvido para investigar numericamente a dinâmica de um fluido viscoso incompressível. Primeiramente apresentamos o modelo matemático que descreve o movimento de um fluido, em seguida apresentamos as condições de contorno e as condições de tensão sobre a superfície livre e, finalmente, descrevemos o método MAC, incluindo as equações de diferenças finitas.

2.1 O Equacionamento Matemático

O movimento de um fluido pode ser descrito seguindo-se a trajetória de qualquer partícula do corpo desse fluido ou, fornecendo-se a velocidade de cada partícula do fluido em cada instante de tempo, que são, respectivamente, as formulações Lagrangeana e Euleriana. Em um escoamento contínuo onde as partículas do fluido podem entrar e sair do sistema, a descrição Euleriana é mais apropriada, pois permite maior flexibilidade em se determinar o escoamento em todos os pontos do espaço em um determinado tempo.

Durante o movimento do fluido a massa de qualquer elemento desse fluido deve ser conservada. Denotando por ρ a densidade, isto é, a quantidade de massa por unidade de volume, e por u a velocidade de uma partícula, obtemos a primeira equação básica, que expressa a conservação de massa:

$$\frac{D\rho}{Dt} + \nabla \cdot \mathrm{u} = 0, \tag{2.1}$$

onde,

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathrm{u}.\nabla$$

é a derivada temporal seguindo o movimento do fluido, também chamada derivada material. Para líquidos, é necessária uma alteração bastante substancial na pressão para que se altere significativamente a densidade e desta forma a suposição de incompressibilidade é uma aproximação geralmente adotada. Supondo-se ρ constante em (2.1), a equação de conservação de massa se reduz a

$$\nabla \cdot \mathrm{u} = 0. \tag{2.2}$$

Neste trabalho, o fluido será considerado Newtoniano, ou seja, a tensão de cisalhamento será suposta proporcional ao gradiente de velocidade, com μ, a constante de proporcionalidade, denotando a viscosidade dinâmica. Além disso, vamos supor o fluido essencialmente isotérmico, ou seja, todas as quantidades e, em particular, a viscosidade será considerada constante com relação à temperatura. Assim, utilizando (2.2), as equações que representam o movimento do fluido podem ser escritas como (Batchelor [3]):

$$\frac{\partial \mathrm{u}}{\partial t} + (\mathrm{u} \cdot \nabla)\mathrm{u} = -\nabla p + \nu \nabla^2 \mathrm{u} + \mathrm{g}, \tag{2.3}$$

onde p é a pressão e ν é a viscosidade cinemática. Estas equações descrevem a produção local de momento em termos das seguintes fontes: $(\mathrm{u} \cdot \nabla \mathrm{u})$ descreve a condução de momento pelo movimento do fluido, ∇p é a alteração do momento que surge das forças da pressão normal, $\nu \nabla^2 \mathrm{u}$ representa a difusão do momento por processos viscosos e g descreve a produção de momento pelas forças da gravidade. A equação (2.3) nada mais é do que a equação de momento de Navier-Stokes para um fluido Newtoniano. Esta equação pode ser escrita de duas formas modificadas, que são frequentemente usadas na construção de muitos métodos numéricos:

$$\frac{\partial \mathrm{u}}{\partial t} + \nabla \cdot (\mathrm{uu}) = -\nabla p + \nu \nabla^2 \mathrm{u} + \mathrm{g} \tag{2.4}$$

e

$$\frac{\partial \mathrm{u}}{\partial t} + \nabla \cdot (\mathrm{uu}) = -\nabla p + \nu \nabla \times \nabla \times \mathrm{u} + \mathrm{g}. \tag{2.5}$$

Ambas são obtida com o auxílio de (2.2). Considerando as variáveis adimensionais,

$$\mathrm{u} = U\bar{\mathrm{u}}, \quad x = L\bar{x}, \quad t = \frac{L}{U}\bar{t} \text{ e } p = \rho U^2 \bar{p},$$

e introduzindo-as em (2.2), (2.4) e (2.5), obtemos as seguintes equações adimensionais:

$$\nabla \cdot \mathrm{u} = 0, \tag{2.6}$$

$$\frac{\partial \mathrm{u}}{\partial t} + \nabla \cdot (\mathrm{uu}) = -\nabla p + \left(\frac{1}{Re}\right) \nabla^2 \mathrm{u} + \left(\frac{1}{Fr^2}\right) \mathrm{g} \tag{2.7}$$

e

$$\frac{\partial \mathrm{u}}{\partial t} + \nabla \cdot (\mathrm{uu}) = -\nabla p - \left(\frac{1}{Re}\right) \nabla \times \nabla \times \mathrm{u} + \left(\frac{1}{Fr^2}\right) \mathrm{g}, \tag{2.8}$$

onde,

$Re = UL/\nu$ é o número de Reynolds;
$Fr = U/\sqrt{Lg}$ é o número de Froude;
U é uma velocidade típica;
L é uma escala típica;
g é a constante gravitacional;
$\mathrm{g} = (g_x, g_y, g_z)^T$ é o vetor gravitacional unitário;
$\mathrm{u} = (u, v, w)^T$ são as componentes adimensionais da velocidade; e
p é a pressão adimensional.

As barras foram abandonadas para simplificar a notação.

As equações (2.6)-(2.7) ou (2.6)-(2.8), com adequadas condições iniciais e de contorno, constituem um sistema de equações diferenciais parciais para as incógnitas u e p.

2.2 Condições de Contorno

Para qualquer problema específico é necessário fornecer um conjunto apropriado de condições de contorno, que podem ser de diversos tipos:

1. Contorno rígido de não escorregamento: um contorno rígido de não escorregamento representa uma parede viscosa que oferece uma certa resistência, impedindo o deslizamento

natural na movimentação do fluido. Isto é modelado matematicamente impondo-se ambas as velocidades normal e tangencial iguais a zero no contorno, ou seja,

$$u_n = 0 \quad \text{e} \quad u_m = 0.$$

Os subscritos n e m serão usados para denotar, respectivamente, as componentes normal e tangencial.

2. Contorno rígido de escorregamento livre: pode representar uma linha de simetria ou uma superfície que não oferece resistência para o movimento do fluido. Isto é obtido exigindo-se que a velocidade normal e a derivada normal da velocidade tangencial se anulem no contorno, ou seja,

$$u_n = 0 \quad \text{e} \quad \frac{\partial u_m}{\partial n} = 0.$$

3. Contorno de entrada pré-estabelecido: representa um segmento de contorno onde o campo de velocidade é pré-estabelecido. Tipicamente a velocidade normal é não nula, enquanto que a velocidade tangencial é nula, ou seja,

$$u_n = U_i \quad \text{e} \quad u_m = 0.$$

4. Contorno de saida: representa um segmento de contorno onde o campo de velocidade é pré-estabelecido ou onde o gradiente espacial não admite alterações, chamado, neste caso, de contorno de saida contínuo. No primeiro caso tipicamente a velocidade normal é não nula, enquanto que no segundo, a velocidade tangencial se anula na direção normal. Estes casos são modelados por:

i) Contorno de saida pré-estabelecido:

$$u_n = U_0 \quad \text{e} \quad u_m = 0;$$

ii) Contorno de saida contínuo:

$$\frac{\partial u_n}{\partial n} = 0 \quad \text{e} \quad \frac{\partial u_m}{\partial n} = 0,$$

onde,

u_n é a componente de velocidade normal;
u_m é a componente de velocidade tangencial;
n é a direção normal externa ao contorno.

2.3 Condições de Tensão sobre a Superfície Livre

Condições de contorno apropriadas para uma superfície livre, requerem que ambas os tensões, normal e tangencial, se anulem, as quais, na ausência de tensões superficiais, são dadas por (Batchelor [3]):

$$n.\sigma.n = 0, \tag{2.9}$$

$$m.\sigma.n = 0, \tag{2.10}$$

onde n e m são, respectivamente, os vetores unitários normal e tangencial à superficie livre e σ é o tensor de tensão.

2.4 O Método MAC

O método MAC é uma técnica de solução de diferenças finitas para simular numericamente escoamentos viscosos incompressíveis transientes com superfícies livres. Para se obter a localização da superfície livre como uma função do tempo, são introduzidas partículas virtuais sem massa que representam o fluido durante o escoamento. Essas partículas são conduzidas pelo campo de velocidade, mas não têm nenhuma importância na determinação das velocidades ou do campo de pressão.

As equações básicas são constituidas pela forma cartesiana bi-dimensional de (2.4) e pela equação de conservação de massa (2.2), isto é,

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial (uv)}{\partial y} = -\frac{\partial p}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + g_x, \tag{2.11}$$

$$\frac{\partial v}{\partial t} + \frac{\partial (uv)}{\partial x} + \frac{\partial v^2}{\partial y} = -\frac{\partial p}{\partial y} + \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + g_y, \tag{2.12}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0. \tag{2.13}$$

No método MAC e técnicas similares, o domínio de escoamento é representado por uma malha de células retangulares fixas, como mostra a figura 2.1. Cada retângulo, que representa uma célula na malha, é delimitado por linhas paralelas aos eixos x e y do sistema cartesiano.

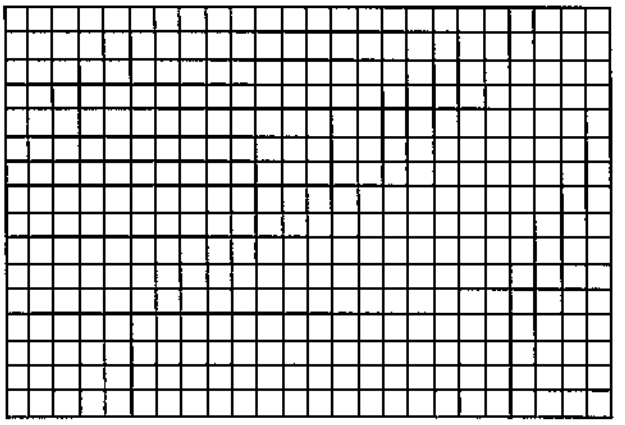


Figura 2.1: Domínio de solução

Como podemos observar na ilustração apresentada na figura 2.1, existem 24 células na direção x contra 16 células na direção y dos eixos coordenados, perfazendo um total de 384 células em todo o domínio.

As células dentro do domínio de solução podem ser marcadas com S, B, F, E, I ou O, onde:

1) S - representam as células que constituem a superfície livre (células "SURFACE");

2) B - representam as células que compõem o contorno rígido (células "BOUNDARY");

3) E - representam as células vazias, ou seja, as células que não fazem parte do contorno e nem contém fluido no instante de tempo considerado (células "EMPTY");

4) F - representam as células que indicam a presença de fluido naquela região e não têm células E adjacentes a nenhuma de suas faces (células "FULL");

5) I - representam as células do contorno de entrada (células "INFLOW");

6) O - representam as células do contorno de saida (células "OUTFLOW").

A marcação inicial das células dentro da malha, se dá da seguinte forma: a partir da definição do contorno rígido associado ao problema, são identificadas todas as células que estão sobre esse contorno e, também, as que são exteriores a ele. Essas células são, então, marcadas com B. Da mesma forma, são marcadas com I as células que estão sobre o contorno de entrada e com O as células que estão sobre o contorno de saida. As demais células, que são interiores ao contorno rígido são marcadas com E. Com o desenvolver do processo de escoamento, as células que foram marcadas com E podem vir a se tornar células S, F ou

mesmo voltar a ser células E. Essas células são dinâmicas por natureza. As figuras 2.2 e 2.3 a seguir ilustram o exemplo de uma malha de células com suas respectivas marcações para um problema típico.

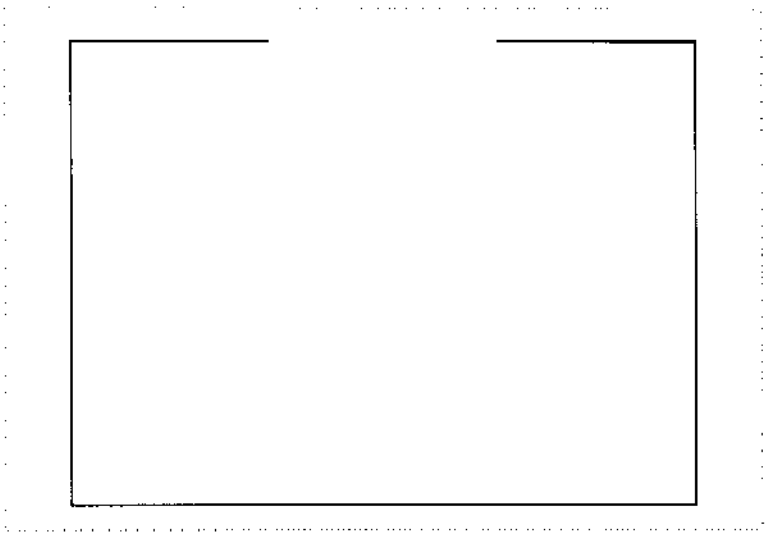


Figura 2.2: Molde do recipiente

A figura 2.2 representa o molde do recipiente para um caso típico. A figura 2.3 representa o domínio de solução dividido em células com suas respectivas marcações no instante inicial de tempo e, na sequência, a figura 2.4 representa a atualização dessas células após percorrido um determinado período de tempo no processamento do escoamento.

B	B	B	B	B	B	B	B	I	I	I	I	I	I	I	B	B	B	B	B	B	B	B
B	B																				B	B
B	B																				B	B
B	B																				B	B
B	B																				B	B
B	B																				B	B
B	B																				B	B
B	B																				B	B
B	B																				B	B
B	B																				B	B
B	B																				B	B
B	B																				B	B
B	B																				B	B
B	B																				B	B
B	B																				B	B
B	B																				B	B
B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B

Figura 2.3: Domínio Ω e a marcação de células

As células que aparecem em branco na figura 2.3 e, também, na figura 2.4 são células

"EMPTY", ou seja, são células marcada com E e são mostradas desta forma apenas para efeito de visualização.

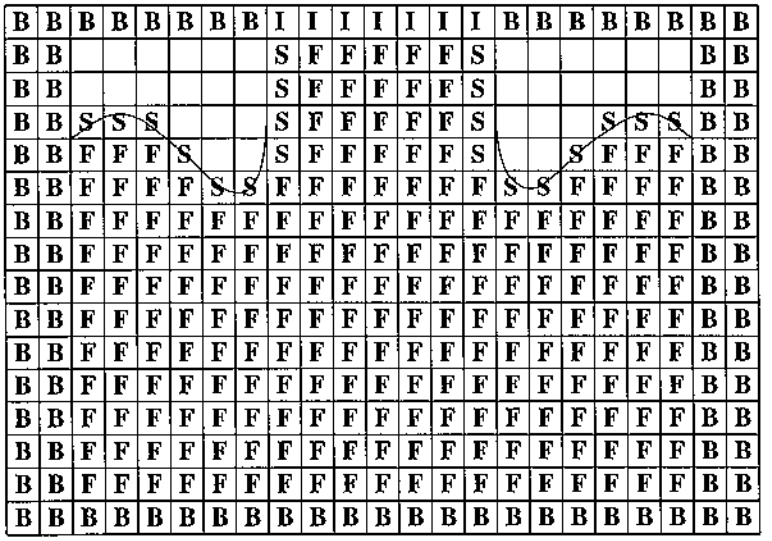


Figura 2.4: Domínio Ω e tipos de células

As figuras 2.3 e 2.4 retratam um recipiente que está sendo preenchido com um fluido. Esse fluido está sendo injetado pela parte superior do recipiente, através de um contorno de entrada. Neste exemplo não está previsto nenhum contorno de saida. Como podemos observar, o domínio de solução é constituido por células quadradas de mesmo comprimento, que estão marcadas com S, B, F, I ou E.

Após realizada essa divisão do domínio em células retangulares fixas, para determinar a solução do problema de escoamento em função do tempo, as equações (2.11), (2.12) e (2.13) são discretizadas por equações de diferenças finitas. É, então, empregada uma malha diferenciada com índices i e j denotando a localização dos centros das células, com i contando as colunas na direção x e j contando as linhas na direção y. A localização das variáveis na malha é mostrada na figura 2.5:

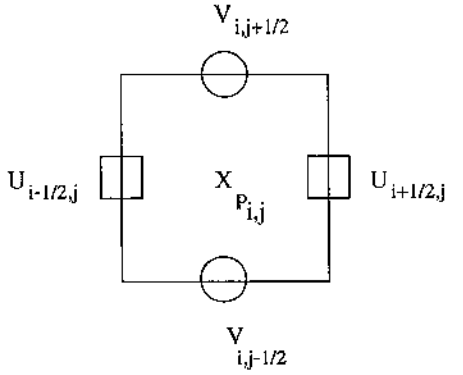


Figura 2.5: Localização das variáveis em uma malha diferenciada

As equações (2.11), (2.12) e (2.13) são aproximadas por equações de diferenças finitas, respectivamente, em u, v e p. São empregadas, para isso, as seguintes aproximações:

$$\left[\frac{\partial u}{\partial t}\right]_{i+\frac{1}{2},j} = \frac{\left(u^{n+1}_{i+\frac{1}{2},j} - u^{n}_{i+\frac{1}{2},j}\right)}{\delta t} + O(\delta t),$$

$$\left[\frac{\partial u^2}{\partial x}\right]_{i+\frac{1}{2},j} = \frac{\left(u^2_{i+1,j} - u^2_{i,j}\right)}{\delta x} + O(\delta x^2),$$

$$\left[\frac{\partial (uv)}{\partial y}\right]_{i+\frac{1}{2},j} = \frac{\left((uv)_{i+\frac{1}{2},j+\frac{1}{2}} - (uv)_{i+\frac{1}{2},j-\frac{1}{2}}\right)}{\delta y} + O(\delta y^2),$$

$$\left[\frac{\partial^2 u}{\partial x^2}\right]_{i+\frac{1}{2},j} = \frac{\left(u_{i+\frac{3}{2},j} - 2u_{i+\frac{1}{2},j} + u_{i-\frac{1}{2},j}\right)}{(\delta x)^2} + O(\delta x^2),$$

$$\left[\frac{\partial^2 u}{\partial y^2}\right]_{i+\frac{1}{2},j} = \frac{\left(u_{i+\frac{1}{2},j-1} - 2u_{i+\frac{1}{2},j} + u_{i+\frac{1}{2},j+1}\right)}{(\delta y)^2} + O(\delta y^2),$$

$$\left[\frac{\partial p}{\partial x}\right]_{i+\frac{1}{2},j} = \frac{(p_{i+1,j} - p_{i,j})}{\delta x} + O(\delta x^2),$$

onde os índices superiores denotam os níveis de tempo e os índices inferiores representam as localizações no espaço. δt é o espaçamento no tempo, δx e δy os espaçamentos, respectivamente, nas direções x e y.

Dentre as expressões anteriores existem termos que não foram definidos dentro da malha de células. Para avaliar estes termos, são utilizadas médias simples como, por exemplo,

$$u_{i,j} = \frac{1}{2}(u_{i+\frac{1}{2},j} + u_{i-\frac{1}{2},j}),$$

$$u_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{1}{2}(u_{i+\frac{1}{2},j} + u_{i+\frac{1}{2},j+1}),$$

$$v_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{1}{2}(v_{i,j+\frac{1}{2}} + v_{i+1,j+\frac{1}{2}}).$$

Com as discretizações realizadas anteriormente, a forma discretizada de (2.11) pode ser escrita da seguinte maneira:

$$u_{i+\frac{1}{2},j}^{n+1} = F_{i+\frac{1}{2},j}^{n} - \frac{\delta t}{\delta x}\left(p_{i+1,j}^{n+1} - p_{i,j}^{n+1}\right), \tag{2.14}$$

onde,

$$\begin{aligned} F_{i+\frac{1}{2},j}^{n} &= u_{i+\frac{1}{2},j}^{n} + \delta t \left[-\frac{u_{i+1,j}^{2} - u_{i,j}^{2}}{\delta x} - \frac{(uv)_{i+\frac{1}{2},j+\frac{1}{2}} - (uv)_{i+\frac{1}{2},j-\frac{1}{2}}}{\delta y} \right. \\ &+ g_x + \nu \left[\frac{u_{i+\frac{3}{2},j} - 2u_{i+\frac{1}{2},j} + u_{i-\frac{1}{2},j}}{\delta x^2} \right. \\ &\left. \left. + \frac{u_{i+\frac{1}{2},j-1} - 2u_{i+\frac{1}{2},j} + u_{i+\frac{1}{2},j+1}}{\delta y^2} \right] \right]. \end{aligned} \tag{2.15}$$

Analogamente, a forma discretizada de (2.12) é dada por:

$$v_{i,j+\frac{1}{2}}^{n+1} = G_{i,j+\frac{1}{2}}^{n} - \frac{\delta t}{\delta y}\left(p_{i,.j+1}^{n+1} - p_{i,j}^{n+1}\right), \tag{2.16}$$

onde,

$$\begin{aligned} G_{i,j+\frac{1}{2}}^{n} &= v_{i,j+\frac{1}{2}}^{n} + \delta t \left[-\frac{v_{i,j+1}^{2} - v_{i,j}^{2}}{\delta y} - \frac{(uv)_{i+\frac{1}{2},j+\frac{1}{2}} - (uv)_{i-\frac{1}{2},j+\frac{1}{2}}}{\delta x} \right. \\ &+ g_y + \nu \left[\frac{v_{i+1,j+\frac{1}{2}} - 2u_{i,j+\frac{1}{2}} + u_{i-1,j+\frac{1}{2}}}{\delta x^2} \right. \\ &\left. \left. + \frac{u_{i,j+\frac{3}{2}} - 2u_{i,j+\frac{1}{2}} + u_{i,j-\frac{1}{2}}}{\delta y^2} \right] \right]. \end{aligned} \tag{2.17}$$

A equação de continuidade é discretizada nos centros das células usando-se as velocidades nos níveis avançados no tempo, ou seja,

$$D_{i,j}^{n+1} = \frac{(u_{i+\frac{1}{2},j}^{n+1} - u_{i-\frac{1}{2},j}^{n+1})}{\delta x} + \frac{(v_{i,j+\frac{1}{2}}^{n+1} - v_{i,j-\frac{1}{2}}^{n+1})}{\delta y}, \tag{2.18}$$

onde $D_{i,j}$ é a dilatação para a célula (i,j).

A substituição de (2.14) e (2.16) em (2.18) produz a seguinte equação discreta de Poisson para a pressão:

$$\left[\frac{(p_{i-1,j} - 2p_{i,j} + p_{i+1,j})}{\delta x^2} + \frac{(p_{i,j-1} - 2p_{i,j} + p_{i,j+1})}{\delta y^2} \right]^{n+1} =$$

$$= \frac{1}{\delta t}\left[\frac{(F_{i+\frac{1}{2},j} - F_{i-\frac{1}{2},j})}{\delta x} + \frac{(G_{i,j+\frac{1}{2}} - G_{i,j-\frac{1}{2}})}{\delta y}\right]. \tag{2.19}$$

Substituindo-se os diversos termos de (2.15) e (2.17) no lado direito de (2.19), obtemos:

$$\left[\frac{(p_{i-1,j} - 2p_{i,j} + p_{i+1,j})}{\delta x^2} + \frac{(p_{i,j-1} - 2p_{i,j} + p_{i,j+1})}{\delta y^2}\right]^{n+1} =$$

$$= \frac{D^n_{i,j}}{\delta t} - \left[\frac{(u^2_{i-1,j} - 2u^2_{i,j} + u^2_{i+1,j})}{\delta x^2} + \frac{(v^2_{i,j-1} - 2v^2_{i,j} + v^2_{i,j+1})}{\delta y}\right.$$

$$+2\left(\frac{(uv)_{i+\frac{1}{2},j+\frac{1}{2}} - (uv)_{i+\frac{1}{2},j-\frac{1}{2}} - (uv)_{i+\frac{1}{2},j-\frac{1}{2}} + (uv)_{i-\frac{1}{2},j-\frac{1}{2}}}{\delta x \delta y}\right)$$

$$\left.-\nu\,\frac{(D_{i-1,j} - 2D_{i,j} + D_{i+1,j})}{\delta x^2} + \frac{(D_{i,j-1} - 2D_{i,j} + D_{i,j+1})}{\delta y^2}\right]^n. \tag{2.20}$$

Esta última equação representa a equação de Poisson, para o método MAC, que deve ser resolvida a cada passo de tempo. Uma vez resolvida essa expressão, as velocidades seguem de (2.14) e (2.16).

Como pode ser observado na equação (2.20), a avaliação dessa expressão nas células contíguas ao contorno requer que termos como $u^2_{i-1,j}$ e $D_{i-1,j}$ sejam avaliados fora do domínio de escoamento. Para se ter uma idéia de como estas condições são aplicadas, considere a parede vertical com condição de contorno de não escorregamento. Com relação à figura 2.6, os índices i, j denotam uma célula dentro do domínio do escoamento e os índices $i-1, j$ são relativos à célula que está do lado de fora desse domínio. Então, se a equação (2.20) é aplicada no nó i, j, os valores externos ao domínio são obtidos da seguinte forma (Welch e co-autores [39]):

1) $v_{i-1,j+\frac{1}{2}} = -v_{i,j+\frac{1}{2}}$ e $v_{i-1,j-\frac{1}{2}} = -v_{i,j-\frac{1}{2}}$,

2) $u_{i-\frac{1}{2},j} = 0$,

3) $u^2_{i-1,j} = u^2_{i,j}$,

4) $(uv)_{i-\frac{1}{2},j+\frac{1}{2}} = 0$ e $(uv)_{i-\frac{1}{2},j-\frac{1}{2}} = 0$,

5) $p_{i-1,j} = p_{i,j} - g_x\delta_x - \frac{2\nu}{\delta x}(u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j})$,

6) $D_{i-1,j} = D_{i,j}$.

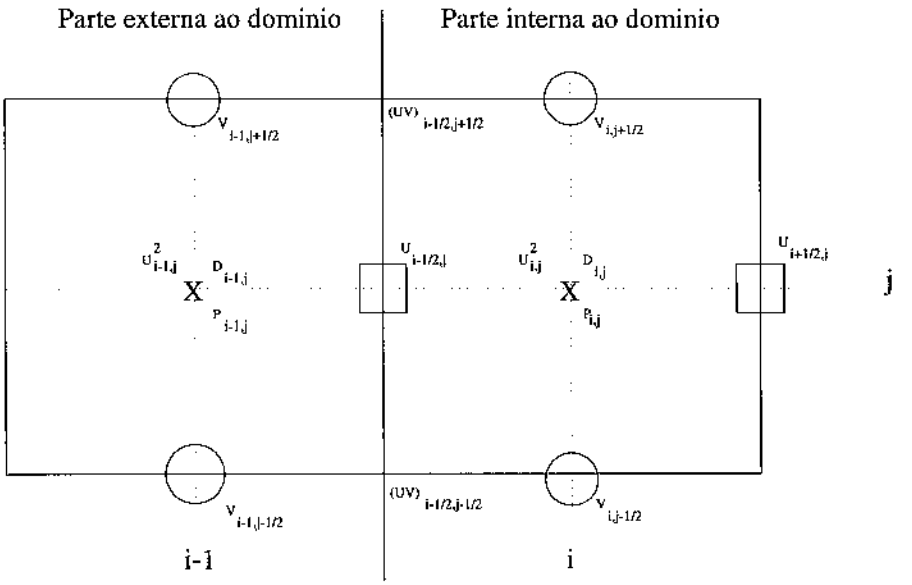


Figura 2.6: Posições das variáveis para uma parede esquerda

Como pode ser observado das expressões de 1) a 6) anteriores, a avaliação do lado direito de (2.20) introduz algumas variáveis fictícias. Os outros tipos de condições de contorno sobre as paredes rígidas são tratados de maneira similar.

Para implementar esse tratamento das condições de contorno foi necessário definir vários tipos de células dentro da malha. No total foram especificados onze tipos diferentes, o que causou um substancial aumento na lógica de programação.

A superfície livre é representada por um conjunto de células da superfície, as chamadas células "SURFACE", as quais permitem a imposição das condições de contorno. Supôs-se, também, no método MAC, a inexistência de forças viscosas e, neste caso, as condições de tensão na superfície livre (equações (2.9) e (2.10)) foram substituidas pela condição de viscosidade nula. Além do mais, foi suposta, para essas células, a condição de incompressibilidade. Desta forma as exigências básicas sobre a superfície livre são:

$$p_{i,j} = 0 \quad \text{e} \quad D_{i,j} = 0.$$

Diante das discretizações anteriores e dessas suposições, o procedimento para obter os campos de velocidade e de pressão em cada instante de tempo, pode ser resumido da seguinte forma:

1) Resolver uma equação de Poisson discreta para a pressão. Esta equação é obtida impondo-se que as equações de momento discretas gerem um novo campo de velocidade satisfazendo

as condições de incompressibilidade.

2) Calcular as novas velocidades a partir das equações de momento de diferenças finitas explícitas.

3) Atualizar as posições das partículas virtuais utilizando, para isso, as velocidades calculadas mais recentemente. As velocidades das partículas são, então, determinadas por interpolação linear utilizando-se as velocidades das células mais próximas.

4) Atualizar as células: as células dentro da malha são dinâmicas por natureza. Por exemplo, uma célula da superfície pode se tornar uma célula cheia ou vazia. Partículas também podem ser geradas ou desprezadas.

Os cálculos envolvendo os passos de 1) a 4), descritos anteriormente e o que chamaremos daqui para a frente de ciclo de cálculo, são executados em cada instante de tempo até que seja atingido um determinado tempo ou, então, até que seja atingido um estado constante.

2.5 Considerações Finais

O método MAC foi uma das primeiras técnicas desenvolvidas para solucionar problemas de escoamentos modelados pelas equações Eulerianas dependentes no tempo para fluidos viscosos incompressíveis com superfícies livres. A principal característica do método é o uso de partículas virtuais sem massa para a representação do fluido. Essas partículas são movimentadas em função das velocidades calculadas mais recentemente e, desta forma, é possível determinar, em cada instante de tempo, a posição de cada partícula e, com isso, obter a visualização dos resultados. O próposito essencial dessas partículas é definir a posição da superfície livre de tal modo que as condições de contorno possam ser impostas. Contudo, em sua formulação original o método se mostrou ineficiente no tratamento da equação de Poisson e pela exigência da imposição das condições de contorno. Para superar essas dificuldades Amsden e Harlow [9] desenvolveram o método SMAC ("Simplified Marker-and-Cell"), no qual o ciclo de cálculo foi dividido em duas partes: primeiro é realizado o cálculo provisório da velocidade e depois uma revisão da mesma, empregando para isso uma função potencial de modo a satisfazer a equação de continuidade. A equação de Poisson resultante neste método é mais simples e requer aplicação somente de condições de contorno homogêneas.

No próximo capítulo vamos discutir o método GENSMAC, que é a base de todo o nosso trabalho, comentando suas semelhanças e diferenças com o método SMAC, quando entendermos que esses comentários se fazem necessários.

Capítulo 3

A metodologia GENSMAC

Como já mencionamos anteriormente, Harlow e Welch desenvolveram, em 1965, uma técnica de diferenças finitas baseada em uma malha diferenciada para investigar a dinâmica de um fluido viscoso incompressível. Essa técnica, chamada MAC, emprega as variáveis primitivas de pressão e velocidade. Subsequentemente, em 1970, Amsden e Harlow, desenvolveram um método MAC simplificado, chamado de SMAC, no qual o ciclo de cálculo foi dividido em duas partes: uma para calcular a velocidade e outra para calcular a pressão, uma vez que não existe um procedimento de interação envolvendo velocidade e pressão.

Recentemente, em 1994, Tomé e McKee empregaram a mesma idéia apresentada por Amsden e Harlow de dividir o ciclo de cálculo em duas partes, no desenvolvimento de um outro método do tipo MAC, chamado por esses autores de GENSMAC.

Neste capítulo, é apresentada a metodologia empregada por Tomé e McKee no desenvovimento do método GENSMAC e nesse contexto são discutidos tópicos tais como: o equacionamento matemático, as equações de diferenças finitas, condições de contorno, condições de tensão sobre a superfície livre, o processo de atualização das células, a movimentação das partículas, o tratamento do contorno irregular e as condições de contorno sobre ele, o procedimento para calcular o espaçamento no tempo, o método para calcular a equação de Poisson e, no final, expomos uma técnica proposta por esses mesmos autores para determinar uma curva que aproxima a superfície livre dentre o conjunto de pontos que representam as partículas virtuais. Embora esse último tópico fuja aos objetivos desse trabalho, decidimos por incluí-lo apenas a título de curiosidade, para mostrar as dificuldades que advinham da necessidade de se ajustar uma curva para representar a superfície livre. Além do mais, segundo Tomé [33], essa técnica é mais simples que outras apresentadas por diversos autores.

A implementação do método GENSMAC tem as seguintes características:

i) o código foi escrito em FORTRAN estruturado;

ii) foi projetado para resolver escoamentos de fluidos com superfícies livres em domínios arbitrários;

iii) apresenta uma rotina para calcular automaticamente o espaçamento no tempo;

iv) utiliza o método dos gradientes conjugados para resolver o sistema linear resultante da equação de Poisson;

v) contém uma aproximação bastante precisa para as condições de tensão sobre a superfície livre;

vi) pode simular escoamentos envolvendo vários contornos de entrada e de saída.

3.1 A Metodologia GENSMAC

As equações básicas para escoamentos viscosos incompressíveis bi-dimensionais dependentes do tempo, são as equações de Navier-Stokes (2.8) juntamente com a equação de incompressibilidade (2.6) que, na forma adimensional, são dadas por:

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right) + (1/Fr^2)g_x, \tag{3.1}$$

$$\frac{\partial v}{\partial t} + \frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y} = -\frac{\partial p}{\partial y} - \frac{1}{Re}\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right) + (1/Fr^2)g_y, \tag{3.2}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0. \tag{3.3}$$

A idéia básica do método SMAC e, também, GENSMAC é resolver estas equações usando uma malha diferenciada e mover as partículas virtuais resolvendo as EDO: $\frac{d\mathrm{X}}{dt}$ =u (x=$(x, y)^T$ e u=$(u, v)^T$) em função das velocidades calculadas mais recentemente.

Mais precisamente, este procedimento pode ser descrito como segue. Suponha que em um dado tempo t_0, a velocidade u(x,t_0) é conhecida e as condições de contorno para a velocidade e pressão são dadas. A atualização da velocidade u(x,t), onde $t = t_0 + \delta t$, é, então, calculada da maneira descrita a seguir.

i) Seja $\tilde{p}(\mathrm{x}, t_0)$ uma pressão arbitrária satisfazendo exatamente a condição de pressão sobre a superfície livre.

ii) Calcula-se a velocidade intermediaria $\tilde{u}(x,t)$, a partir de:

$$\frac{\partial \tilde{u}}{\partial t} = \left[-\frac{\partial u^2}{\partial x} - \frac{\partial uv}{\partial y} - \frac{\partial \tilde{p}}{\partial x} + \frac{1}{Re}\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right) + (1/Fr^2)g_x \right], \tag{3.4}$$

$$\frac{\partial \tilde{v}}{\partial t} = \left[-\frac{\partial uv}{\partial x} - \frac{\partial v^2}{\partial y} - \frac{\partial \tilde{p}}{\partial y} - \frac{1}{Re}\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right) + (1/Fr^2)g_y \right], \tag{3.5}$$

com $\tilde{u}(x,t_0) = u(x,t_0)$, usando-se as condições de contorno corretas para $u(x,t_0)$.

As equações (3.4) e (3.5) são resolvidas por um esquema explícito de diferenças finitas.

Supondo que $(u(x,t), p(x,t))$ seja solução de (3.4) e (3.5), então, tem-se:

$$\frac{\partial u}{\partial t} = \left[-\frac{\partial u^2}{\partial x} - \frac{\partial uv}{\partial y} - \frac{\partial p}{\partial x} + \frac{1}{Re}\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right) + (1/Fr^2)g_x \right], \tag{3.6}$$

$$\frac{\partial v}{\partial t} = \left[-\frac{\partial uv}{\partial x} - \frac{\partial v^2}{\partial y} - \frac{\partial p}{\partial y} - \frac{1}{Re}\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right) + (1/Fr^2)g_y \right]. \tag{3.7}$$

Subtraindo (3.4) de (3.6) e (3.5) de (3.7), obtemos:

$$\frac{\partial(u-\tilde{u})}{\partial t} = -\frac{\partial(p-\tilde{p})}{\partial x},$$

$$\frac{\partial(v-\tilde{v})}{\partial t} = -\frac{\partial(p-\tilde{p})}{\partial y}. \tag{3.8}$$

Estas equações na forma vetorial se tornam:

$$\frac{\partial}{\partial t}(u-\tilde{u}) = -\nabla(p-\tilde{p}). \tag{3.9}$$

Tomando o rotacional sobre ambos os lados de (3.9), obtemos:

$$\nabla \times \left[\frac{\partial}{\partial t}(u-\tilde{u})\right] = 0. \tag{3.10}$$

Invertendo-se a ordem dos operadores em (3.10), obtém-se:

$$\frac{\partial}{\partial t}\left[\nabla \times (u-\tilde{u})\right] = 0 \implies \frac{\partial}{\partial t}(w-\tilde{w}) = 0 \Longrightarrow w-\tilde{w} = f(x)$$

mas,

$$u = \tilde{u} \text{ em } t = t_0 \Longrightarrow w = \tilde{w} \quad \text{em} \quad t = t_0 \Longrightarrow f(x) = 0.$$

Assim,

$$w = \tilde{w}, \qquad \forall x.$$

Isso mostra que $\tilde{\mathrm{u}}(\mathrm{x}, t)$ possui a vorticidade correta no tempo t.

Definindo a seguinte velocidade:

$$\bar{\mathrm{u}}(\mathrm{x}, t) \;=\; \tilde{\mathrm{u}}(\mathrm{x}, t) - \nabla\psi(\mathrm{x}, t),$$

com

$$\nabla^2\psi \;=\; \nabla \cdot \tilde{\mathrm{u}}(\mathrm{x}, t),$$

temos que $\bar{\mathrm{u}}(\mathrm{x}, t)$ satisfaz:

$$\nabla \cdot \bar{\mathrm{u}}(\mathrm{x}, t) \;=\; 0,$$

e a vorticidade permanece inalterada.

Desta forma $\bar{\mathrm{u}}(\mathrm{x}, t)$ pode ser identificada como sendo a velocidade atualizada u(x,t).

O raciocínio acima sugere o seguinte procedimento para atualização da velocidade u e pressão p:

iii) Resolve-se a equação de Poisson

$$\nabla^2\psi = \nabla \cdot \tilde{\mathrm{u}}(\mathrm{x}, t). \tag{3.11}$$

iv) Atualiza-se a velocidade:

$$\mathrm{u}(\mathrm{x}, t) = \tilde{\mathrm{u}}(\mathrm{x}, t) - \nabla\psi(\mathrm{x}, t) \tag{3.12}$$

v) Calcula-se a pressão: uma vez que temos calculada a velocidade $\mathrm{u}(\mathrm{x}, t)$, a pressão segue de (3.9), isto é,

$$-\frac{\nabla_d\psi}{\delta t} \;=\; -\nabla_d(p - \tilde{p}),$$

onde ∇_d denota o operador gradiente discreto.

Assim,

$$0 = \nabla_d(p - \tilde{p} - \frac{\psi}{\delta t})$$

e, portanto,

$$p = \tilde{p} + \frac{\psi}{\delta t}. \tag{3.13}$$

Desta forma, o método GENSMAC resolve as equações de momento explicitamente e um sistema esparso simétrico (a equaçao de Poisson discreta) para a velocidade potencial ψ.

O próximo passo é a movimentação das partículas.

vi) O movimento das partículas: as partículas são criadas nos contornos de entrada e são injetadas no domínio de solução para representar o fluido. Estas são as chamadas partículas virtuais, cujas coordenadas são armazenadas em cada passo de tempo e, então, atualizadas resolvendo-se as EDO abaixo pelo método de Euler,

$$\frac{dx}{dt} = u \quad \text{e} \quad \frac{dy}{dt} = v. \tag{3.14}$$

Este procedimento determina o movimento de uma partícula, mostrando se ela está se movendo de uma célula para outra ou não, ou mesmo se está saindo do domínio de solução.

3.1.1 As Equações de Diferenças Finitas Básicas

De maneira análoga ao método MAC, nas metodologias SMAC e, também, GENSMAC as equações básicas são aproximadas sobre uma malha diferenciada, ou seja, as variáveis de pressão $\tilde{p}_{i,j}$, a velocidade potencial $\psi_{i,j}$ e o divergente $D_{i,j}$ são posicionados no centro da célula, enquanto que as velocidades $u_{i,j}$ e $v_{i,j}$ são deslocadas por uma translação, respectivamente, de $\frac{\delta x}{2}$ e de $\frac{\delta y}{2}$.

As equações de momento (3.4) e (3.5) são discretizadas e aplicadas, respectivamente, nos nós u e v. As equações discretizadas por diferenças finitas para estas equações são (Amsden e Harlow [2]):

$$\frac{\tilde{u}^{n+1}_{i+\frac{1}{2},j} - u^n_{i+\frac{1}{2},j}}{\delta t} = \frac{\tilde{p}_{i,j} - \tilde{p}_{i+1,j}}{\delta x} + \frac{u^n_{i+\frac{1}{2},j-\frac{1}{2}} v^n_{i+\frac{1}{2},j-\frac{1}{2}} - u^n_{i+\frac{1}{2},j+\frac{1}{2}} v^n_{i+\frac{1}{2},j+\frac{1}{2}}}{\delta y}$$

$$+ \frac{u^n_{i+\frac{1}{2},j} u^n_{i-\frac{1}{2},j} - u^n_{i+\frac{3}{2},j} u^n_{i-\frac{1}{2},j}}{\delta x}$$

$$+\frac{1}{Re}\left[\frac{u^n_{i+\frac{1}{2},j+1}+u^n_{i+\frac{1}{2},j-1}-2u^n_{i+\frac{1}{2},j}}{\delta y^2}\right.$$

$$\left.-\frac{v^n_{i+1,j+\frac{1}{2}}-v^n_{i+1,j-\frac{1}{2}}-v^n_{i,j+\frac{1}{2}}+v^n_{i,j-\frac{1}{2}}}{\delta x\delta y}\right]+\frac{1}{Fr^2}g_x, \qquad (3.15)$$

$$\frac{\tilde{v}^{n+1}_{i,j+\frac{1}{2}}-v^n_{i,j+\frac{1}{2}}}{\delta t} = \frac{\tilde{p}_{i,j}-\tilde{p}_{i,j+1}}{\delta y}+\frac{u^n_{i-\frac{1}{2},j+\frac{1}{2}}v^n_{i-\frac{1}{2},j+\frac{1}{2}}-u^n_{i+\frac{1}{2},j+\frac{1}{2}}v^n_{i+\frac{1}{2},j+\frac{1}{2}}}{\delta x}$$

$$+\frac{v^n_{i,j+\frac{1}{2}}v^n_{i,j-\frac{1}{2}}-v^n_{i,j+\frac{3}{2}}v^n_{i,j+\frac{1}{2}}}{\delta y}$$

$$-\frac{1}{Re}\left[\frac{2v^n_{i,j+\frac{1}{2}}+v^n_{i+1,j+\frac{1}{2}}-v^n_{i-1,j+\frac{1}{2}}}{\delta x^2}\right.$$

$$\left.+\frac{u^n_{i+\frac{1}{2},j+1}-u^n_{i+\frac{1}{2},j}-u^n_{i-\frac{1}{2},j+1}+u^n_{i-\frac{1}{2},j}}{\delta x\delta y}\right]+\frac{1}{Fr^2}g_y. \qquad (3.16)$$

A equação de Poisson (3.11) é discretizada nos centros das células usando os cinco pontos Laplacianos e pode ser reescrita como:

$$4\psi_{i,j}-\psi_{i+1,j}-\psi_{i-1,j}-\psi_{i,j+1}-\psi_{i,j-1}=-h^2\tilde{D}_{i,j} \qquad (3.17)$$

onde,

$$\tilde{D}_{i,j} = \frac{\tilde{u}_{i+\frac{1}{2},j}-\tilde{u}_{i-\frac{1}{2},j}}{\delta x}+\frac{\tilde{v}_{i,j+\frac{1}{2}}-\tilde{v}_{i,j-\frac{1}{2}}}{\delta y}$$

e h é o tamanho do espaçamento na malha (supondo $\delta x = \delta y$).

3.1.2 Atualização das Células

As células dentro do domínio de solução têm uma representação dinâmica e é empregado um esquema para monitorar suas definições. Isto é obtido através de uma sessão de atualização chamada por "reflagging", que é executada em cada iteração do ciclo de cálculo. Um ciclo de cálculo, na metodologia GENSMAC, é constituido pelos passos de i) a vi), descritos na seção 3.1.

Esse procedimento de atualização ou remarcação das células é realizado no ínicio de cada ciclo. Como o fluido não é estacionário, as células dentro do domínio podem mudar

constantemente. Desta forma é necessário verificar, em cada ciclo de cálculo, se as células precisam ser atualizadas ou não. Isto é realizado da seguinte maneira:

i) faz-se uma varredura em todas as partículas para verificar quais células contém partículas e quais não as contém;

ii) se uma célula E contém partículas, então ela se torna uma célula S;

iii) todas as células S que não contém mais partículas se tornam células E e as velocidades que estão sobre as faces dessas células, contíguas a células E, tornam-se zero;

iv) se uma célula F tem como vizinho uma célula E, então ela torna-se uma célula S;

v) finalmente, todas as células S que não contém células E em sua vizinhança tornam-se células F.

3.1.3 Condições de Tensão sobre a Superfície Livre

As aproximações utilizadas no cálculo das condições de contorno sobre a superfície livre, são exatamente as mesmas em ambas as metodologias, SMAC e GENSMAC. Tais aproximações são descritas a seguir.

Como já dissemos no capítulo 2, as condições de contorno apropriadas para a superfície livre são as componentes da tensão normal e tangencial se anulando, dadas pelas equações (2.9) e (2.10). Estas equações podem ser reescritas como (Hirt e Shannon [18]):

para a tensão normal,

$$p - (2/Re)\left[n_x n_x \frac{\partial u}{\partial x} + n_x n_y \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) + n_y n_y \frac{\partial v}{\partial y}\right] = 0, \tag{3.18}$$

e, para a tensão tangencial,

$$\left[2n_x m_x \frac{\partial u}{\partial x} + (n_x m_y + n_y m_x)\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) + 2n_y m_y \frac{\partial v}{\partial y}\right] = 0, \tag{3.19}$$

onde n= (n_x, n_y) é o vetor normal unitário externo à superfície livre e m= (m_x, m_y) é o vetor tangente.

Supondo-se que a malha seja suficientemente fina de modo que a superfície livre intercepte uma célula em somente dois lados, (3.18) e (3.19) podem ser aproximadas localmente por diferenças finitas de acordo com um dos três casos descritos a seguir.

i) Células S com somente um lado contíguo a uma célula E: para estas células vamos supor que a superfície livre corte dois lados opostos da célula, de modo que n_x ou n_y seja pequeno. Então, (3.18) e (3.19) podem ser aproximadas por:

$$p - \left(\frac{2}{Re}\right)\left(\frac{\partial u_n}{\partial n}\right) = 0 \tag{3.20}$$

e

$$\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = 0, \tag{3.21}$$

onde n é a direção x ou a direção y.

Para efeito de ilustração considere o caso mostrado na figura 3.1:

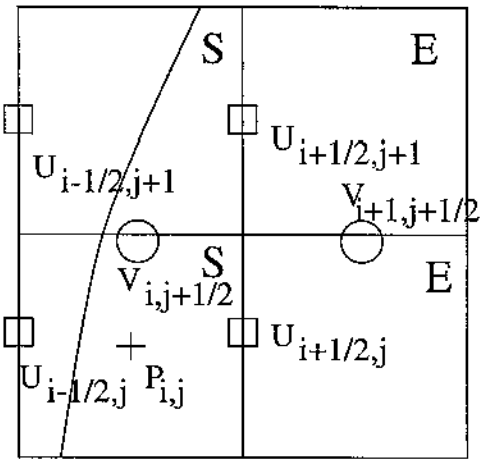


Figura 3.1: Células S com lado direito contíguo a uma célula E.

Neste caso, a aproximação de diferenças finitas para a tensão normal (3.20) é dada por:

$$p_{i,j} = \frac{2}{Re}\left(\frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{\delta x}\right),$$

enquanto que a tensão tangencial (3.21) é dada por:

$$v_{i+1,j+\frac{1}{2}} = v_{i,j+\frac{1}{2}} - \frac{\delta x}{\delta y}\left(u_{i+\frac{1}{2},j+1} - u_{i+\frac{1}{2},j}\right).$$

Outros tipos de células S com um lado contíguo a uma célula E são tratados de maneira similar.

ii) Células S com dois lados adjacentes a células E: para estas células supõe-se que a direção normal externa à superfície livre forme um ângulo de 45° entre os lados que são comuns com células E. Portanto, (3.18) e (3.19) se reduzem a:

$$p = \pm\frac{1}{Re}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{3.22}$$

e

$$\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} = 0. \tag{3.23}$$

O sinal em (3.22) é escolhido como sendo o sinal de $n_x n_y$.

Para exemplificar, considere a célula S mostrada na figura 3.2 a seguir:

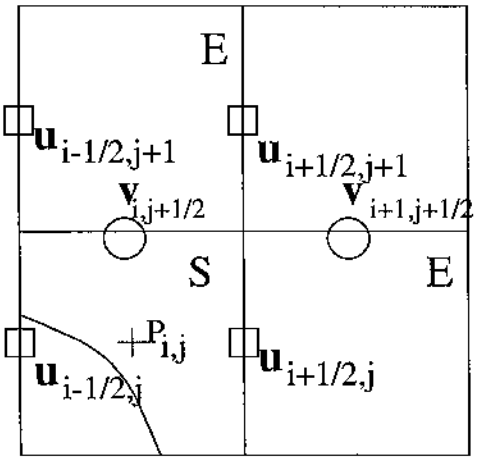


Figura 3.2: Célula S com dois lados adjacentes com células E

A aproximação para a tensão normal (3.22) é, então, dada por:

$$p_{i,j} = \frac{1}{2Re}\left[\frac{u_{i+\frac{1}{2},j} + u_{i-\frac{1}{2},j} - u_{i+\frac{1}{2},j-1} - u_{i-\frac{1}{2},j-1}}{\delta y} + \frac{v_{i,j+\frac{1}{2}} + v_{i,j-\frac{1}{2}} - v_{i-1,j+\frac{1}{2}} - v_{i-1,j-\frac{1}{2}}}{\delta x}\right].$$

Para a tensão tangencial (3.23) é exigido que $\frac{\partial u}{\partial x}$ e $\frac{\partial v}{\partial y}$ se anulem separadamente. A razão para se fazer tal exigência é que a equação da conservação de massa ($\nabla \cdot \mathrm{u} = 0$) seja também satisfeita para aquelas células. Assim, para células S que tenham dois lados comuns com células E, têm-se as seguintes aproximações:

$$u_{i+\frac{1}{2},j} = u_{i-\frac{1}{2},j} \quad \text{e} \quad v_{i,j+\frac{1}{2}} = v_{i,j-\frac{1}{2}}.$$

Outras configurações de células S com dois lados adjacentes contíguos com células E são tratados de maneira similar.

iii) Células S com três lados comuns a células E: estes tipos de células são muito raros e podem desaparecer facilmente com o refinamento da malha, eliminando-se, dessa forma, a alta curvatura local.

Em todo caso, se surgirem tais tipos de células, a pressão é tomada igual a zero e a velocidade é ajustada de modo que a condição $\nabla \cdot \mathrm{u} = 0$ seja também satisfeita por essas células. Assim, problemas envolvendo alta curvatura local podem, também, ser considerados dentro da metodologia GENSMAC.

3.1.4 O Movimento das Partículas

Em ambos os métodos SMAC e GENSMAC as partículas virtuais são usadas para representar o fluido. A tarefa essencial dessas partículas é fornecer o movimento da superfície livre de modo que a configuração das células que a compõem possam ser determinadas. A atualização de todas as células é realizada no final de cada passo de tempo calculado, fornecendo, desta forma, a dinâmica do movimento do fluido. As novas coordenadas da partícula são determinadas aproximando-se (3.14) pelo método de Euler. Assim, após serem atualizadas as velocidades, as partículas são movimentadas por:

$$x_p^{n+1} = x_p + u_p \delta t^{n+1} \quad \text{e} \quad y_p^{n+1} = y_p + u_p \delta t^{n+1},$$

onde (x_p, y_p) é a posição corrente da partícula, δt^{n+1} é o atual passo de tempo empregado e (x_p^{n+1}, y_p^{n+1}) é a posição atualizada da partícula.

As velocidades u_p e v_p são calculadas utilizando-se interpolação bilinear sobre as quatro velocidades, definidas sobre a malha diferenciada, que estão mais próximas ao ponto p. O procedimento é o mesmo adotado no método SMAC.

3.2 Tratamento do Contorno Irregular

A idéia básica da metodologia GENSMAC é decompor o contorno irregular em um outro que coincide com os segmentos definidos pela malha e, então, resolver as equações considerando esse pseudo-contorno.

Para um domínio retangular, não surgem problemas quando as equações de diferenças finitas (3.15) e (3.16) são aplicadas nos nós próximos às paredes do contorno, pois estes coincidem com as paredes das células. Mas, para um contorno irregular de um modo geral, não coincidirá e é, então, usado um procedimento de interpolação que coincide com a malha diferenciada. Isto é realizado primeiro determinando onde o contorno irregular corta as paredes das células e então marcando-as como sendo células B ou não. As células nas quais somente uma pequena parte é ocupada pelo contorno rígido são consideradas como células interiores, ou seja, F, S ou E, enquanto que para as outras células nas quais o contorno rígido ocupa uma parte maior, são consideradas como sendo células B. Assim, são analisados os diversos casos que podem ocorrer. Não entraremos em maiores detalhes, pois não é este o objetivo deste trabalho.

Após terem sido marcadas todas as células que compõem o contorno rígido é, então, definido o contorno virtual, que é formado pelas linhas da malha que são contínuas por partes.

A figura 3.3 a seguir mostra um exemplo de um contorno circular para um problema típico. Nesta figura são mostradas as células B, o contorno circular e o contorno virtual, formado por linhas paralelas ao eixo x e ao eixo y.

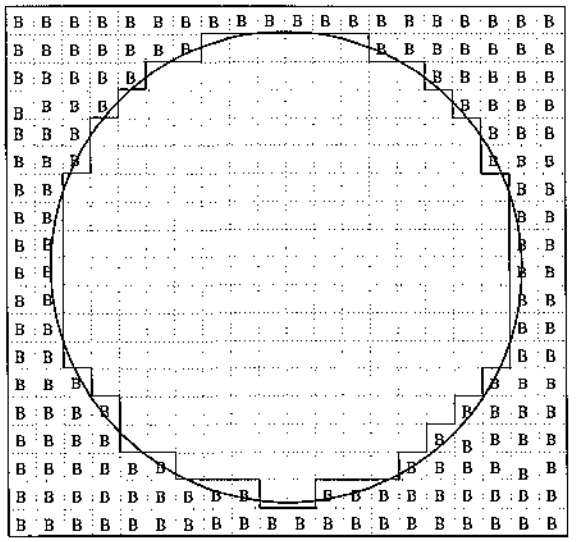


Figura 3.3: Contorno real para uma circunferência

A figura 3.4 a seguir mostra apenas o contorno virtual para o caso anterior.

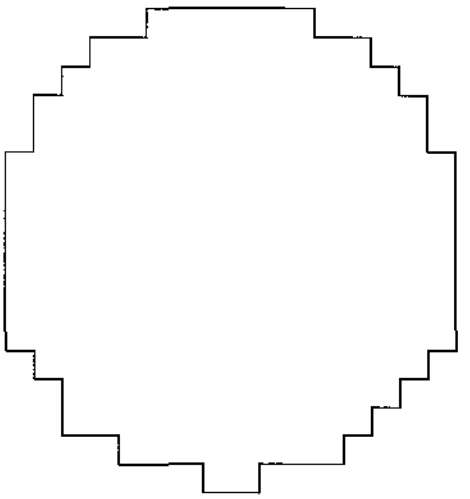


Figura 3.4: Contorno virtual para uma circunferência

Com essas duas últimas figuras pretendemos apenas dar uma idéia de como é definido o contorno virtual na metodologia GENSMAC. Neste caso, o contorno virtual é uma aproximação bastante grosseira para o contorno real, pois a malha utilizada na visualização é muito espessa.

3.2.1 Condições de Contorno sobre o Contorno Irregular

Como já dissemos anteriormente, existem vários tipos de condições de contorno que podem ser aplicadas: contorno de não escorregamento, contorno de escorregamento livre, contorno de entrada pré-estabelecido, contorno de saída pré-estabelecido e contorno de saída contínuo.

O tratamento dessas condições de contorno na metodologia GENSMAC é similar ao tratamento dispensado pelo método SMAC, exceto para a condição de contorno de não escorregamento sobre um contorno irregular, pois o método SMAC não trata esses casos.

Quando as equações discretas de Navier-Stokes são aplicadas nos nós adjacentes ao contorno virtual, os valores de u e v nas faces das células de contorno precisam ser calculados. Se são impostas condições de contorno de não escorregamento sobre o contorno irregular, então estes valores podem ser estimados em termos dos valores da função nos nós internos por interpolação linear ou bilinear.

As células do contorno podem ter um ou dois lados comuns a células interiores. Assim, se uma célula B tem um único lado comum a uma célula interior, então os valores de u e

v são calculados por interpolação linear, enquanto que para as células B que possuem dois lados comuns a células interiores, é usada interpolação bilinear.

Para exemplificar, considere a figura 3.5 a seguir, na qual temos uma célula B com um único lado comum com uma célula interior:

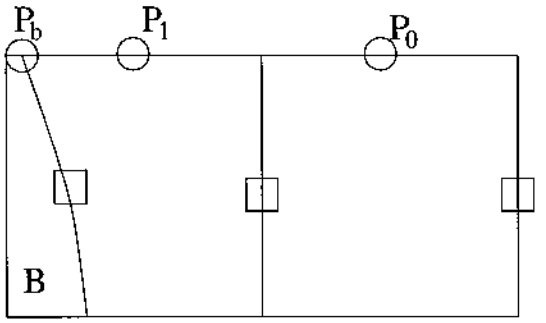


Figura 3.5: Célula B com um único lado contíguo a uma célula interior

Denotando os valores de x, y, u e v no ponto P_0, respectivamente, por x_0, y_0, u_0 e v_0, com notações similares para os pontos P_1 e P_b, o valor de v_1, quando calculado por interpolação de Lagrange, é dado por:

$$v_1 = \frac{(x_1 - x_b)}{(x_0 - x_b)} v_0.$$

O valor de x_b pode ser encontrado através da equação que descreve a trajetória do contorno:

$$f(x_b, y) = 0.$$

O valor de u_1 é determinado de maneira similar.

Para outras configurações de células B com um único lado comum com células interiores o procedimento é idêntico. No caso de interpolação bilinear, o raciocínio é semelhante. Mais detalhes dessas aproximações podem ser encontradas em Tomé [33].

3.2.2 Condições de Contorno para a Equação de Poisson

Para a equação de Poisson (3.11), a condição apropriada sobre o contorno rígido, segundo Amsden e Harlow [2], é:

$$\frac{\partial \psi}{\partial n} = 0.$$

Essa equação é aproximada, na metodologia GENSMAC, por diferenças finitas locais sem maiores problemas, uma vez que o contorno virtual coincide com as linhas da malha.

Por exemplo, para uma célula B com somente um lado contíguo a uma célula F, esta equação se torna

$$\frac{\partial \psi}{\partial x} = 0 \quad \text{ou} \quad \frac{\partial \psi}{\partial y} = 0,$$

dependendo se a linha de separação entre as duas células está na direção x ou na direção y. Para células B com dois lados contíguos a células F, então a condição de contorno é representada por

$$\frac{\partial \psi}{\partial x} = 0 \quad \text{e} \quad \frac{\partial \psi}{\partial y} = 0.$$

Como a função $\tilde{p}$ satisfaz à condição de pressão correta sobre a superfície livre, a condição de contorno apropriada para ψ sobre a superície livre é

$$\psi \;=\; 0.$$

3.3 Procedimento para Calcular o Passo no Tempo

O procedimento utilizado, na metodologia GENSMAC, para calcular automaticamente o espaçamento no tempo, foi o mesmo apresentado por Markham e Proctor [20], no qual foi imposta, além das restrições de estabilidade, a condição de que nenhuma partícula deve atravessar mais do que a região de uma célula em um determinado passo de tempo, ou seja:

$$|u|\,\delta t \;<\; \delta x,$$

$$|v|\,\delta t \;<\; \delta y.$$

O valor de δt é escolhido de forma tal que:

$$\delta t = \min(\delta t_{visc}, \delta t_u, \delta t_v) A, \tag{3.24}$$

onde:

$0 < A \leq 1$,

$\delta t_{visc} = \frac{A_1}{2\nu} \cdot \frac{\delta x^2 \delta y^2}{\delta x^2 + \delta y^2}$ (Condição de estabilidade),

$\delta t_u = A_2 \cdot \frac{\delta x}{2U_{\max}}$ (Condição de estabilidade na direção x),

$\delta t_v = A_2 \cdot \frac{\delta y}{2V_{\max}}$ (Condição de estabilidade na direção y),

ν é a viscosidade,

U_{max} e V_{max} são os valores máximos entre as velocidades provisórias,

com $0 \leq A_i \leq 1$.

3.4 O Método para a Equação de Poisson

Em cada iteração do ciclo de cálculo tem-se que resolver a equação de Poisson em um domínio geral sujeito a condições de contorno de Dirichlet e Neumann. Isto pode consumir grande parte do tempo de processamento para problemas grandes. É vital se fazer a escolha de um algoritmo robusto.

No caso da metodologia GENSMAC, a solução é procurada em uma região arbitrária, porém pré-estabelecida, usando-se coordenadas cartesianas. Como esta região consiste de linhas retas paralelas aos eixo x ou y, isto leva a um sistema de equações para a equação de Poisson cuja matriz é simétrica e definida positiva. Foi, então, implementado o método dos gradientes conjugados para resolver este sistema linear, resultando em um algoritmo que comprovou ser robusto e eficaz.

3.5 A Metodologia para Calcular a Superfície Livre

Tomé [33] apresentou, também, uma técnica para identificar a superfície livre. Essa técnica pode ser tanto utilizada dentro da metodologia GENSMAC quanto nas aproximações MAC de um modo geral. Além do mais, é relativamente fácil de implementar e pode ser utilizada para problemas envolvendo uma quantidade arbitrária de superfícies livres e envolvendo também vários contornos de entrada e saida.

Vários outros autores propuseram técnicas para identificação da superfície livre, porém essa parece ser a mais simples e eficiente dentre as técnicas que consideram a utilização de todo o conjunto de partículas virtuais para se representar o fluido em um processo de escoamento.

A proposta apresentada por Tomé é descrita a seguir.

3.5.1 A Identificação das Partículas sobre a Superfície Livre

Para utilizar um método de ajuste de curva para aproximar a superfície livre, primeiro tem-se que identificar todas as partículas que estão mais próximas da superfície livre. Isto é realizado inspecionando-se quais as partículas que estão sobre as células S.

A figura 3.6 a seguir mostra a configuração integral do fluido para um caso típico.

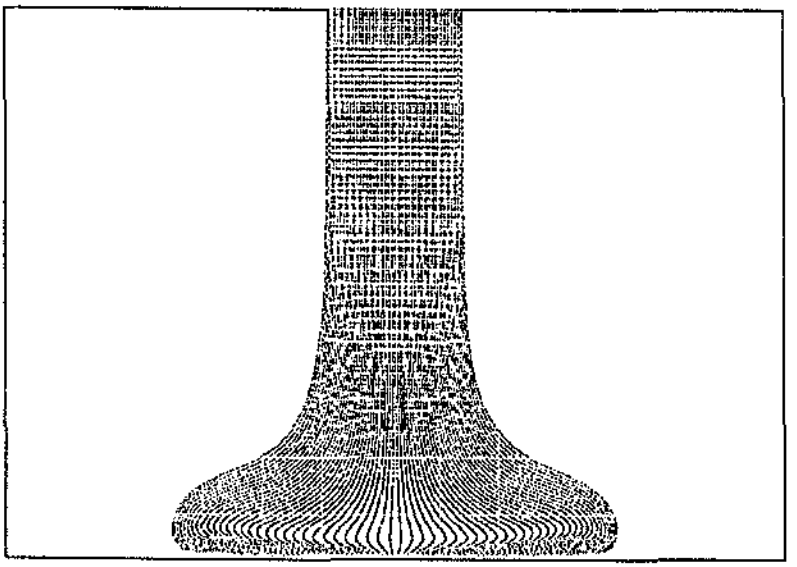


Figura 3.6: Configuração do fluido

Na sequência, mostramos as partículas que estão sobre as células "SURFACE" para a configuração do fluido apresentada anteriormente.

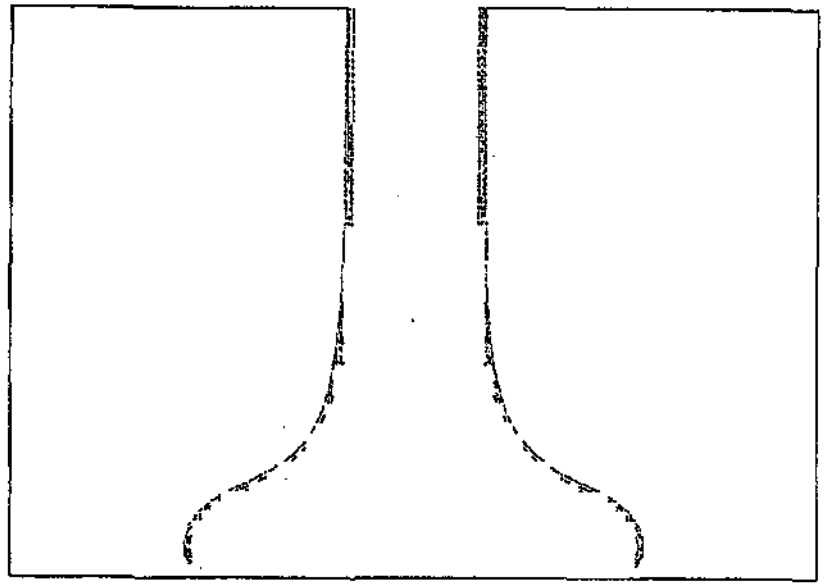


Figura 3.7: Partículas que estão sobre as células da superfície livre

Uma vez obtidas estas partículas, tem-se que identificar quais estão mais próximas à superfície livre, ou seja, quais são as partículas mais próximas a células E. Isto é realizado empregando-se um refinamento local da malha para cada célula S e identificando-se quais são as partículas mais próximas a células E. Desta forma, nos deparamos com uma cadeia de partículas que é adequada para ajustar uma curva para aproximar a superfície livre. A figura 3.8 mostra essa cadeia de partículas para a configuração de fluido anterior.

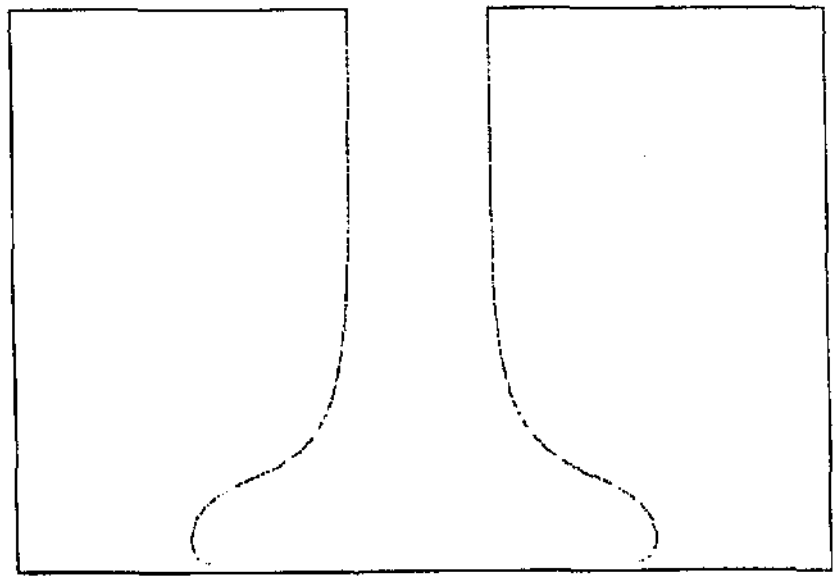


Figura 3.8: Partículas para ajustar uma curva para aproximar a superfície livre

Uma vez encontradas as partículas que descrevem com mais exatidão a superfície livre é, então, aplicado um método de ajuste de curva para determinar com mais regularidade e com uma quantidade menor de pontos, a cadeia de partículas que se ajusta de maneira satisfatória à superfície livre. O método utilizado para isso é um algoritmo de ajuste de mínimos quadrados, determinando, então, uma sequência de pontos regulares que aproximam a superfície livre.

A figura 3.9 ilustra o conjunto de pontos que descrevem uma aproximação para a superfície livre para um caso particular de configuração de fluido mostrada na figura 3.6.

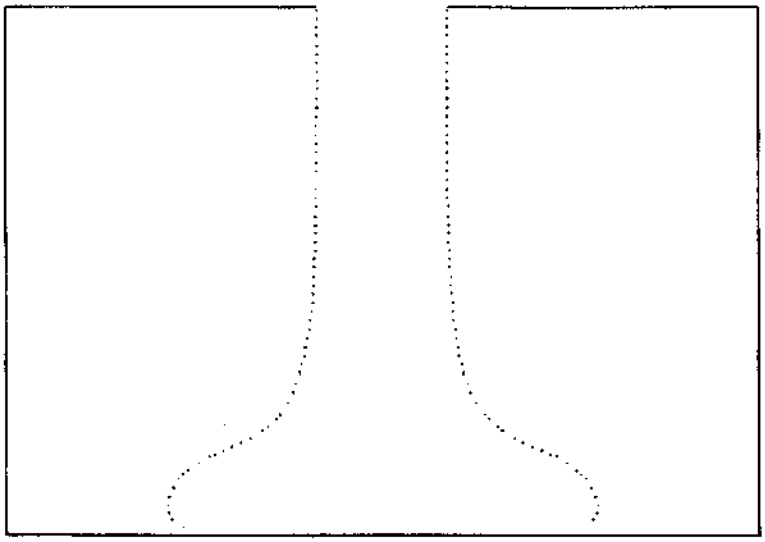


Figura 3.9: Conjunto de pontos que aproximam a superfície livre

3.6 Considerações Finais

Neste capítulo, apresentamos resumidamente os procedimentos de aproximações adotados na metodologia GENSMAC. Uma descrição mais completa deste trabalho pode ser encontrada em [33].

No próximo capítulo, apresentamos a técnica de decomposição de domínio que estamos propondo e, também, os aspectos da implementação paralela efetuada no código GENSMAC.

Capítulo 4

A Decomposição de Domínio e a Implementação Paralela

Descrevemos, nos parágrafos seguintes, nossa proposta de decomposição de domínio, a técnica de paralelização empregada e sua adequação à metodologia GENSMAC e, também, os aspectos da implementação paralela, tais como modelos de comunicação, a utilização do pacote PIM na resolução do sistema linear resultante da discretização da equação de Poisson.

Este capítulo está organizado da seguinte forma: no primeiro parágrafo descrevemos a proposta de decomposição de domínios, no segundo, a técnica de paralelização utilizada e, finalmente, na terceira e última seção mostramos os aspectos e dificuldades da implementação paralela.

4.1 A Decomposição de Domínio

A metodologia GENSMAC, como exposta no capítulo 3, é uma extensão da metodologia SMAC para resolver computacionalmente problemas de escoamentos de fluidos viscosos incompressíveis com superfícies livres em domínios mais gerais que os apresentados até então por outras metodologias similares. A implementação do método SMAC realizada no código GENSMAC constitui-se, então, de quatro partes principais:

1. o cálculo das velocidades baseado no esquema explícito de diferenças finitas, dadas pelas expressões (3.15) e (3.16) do capítulo anterior;

2. a solução da equação de Poisson baseada em diferenças finitas sobre cinco pontos, dada pela expressão (3.17). Neste caso, é suposto $\delta x = \delta y$;

3. atualização das velocidades $\mathrm{u}(x,t)$ e da pressão $\mathrm{p}(x,t)$:

$$\mathrm{u}(x,t) \;=\; \tilde{\mathrm{u}}(x,t) - \nabla\psi(x,t),$$

$$\mathrm{p}(x,t) \;=\; \tilde{\mathrm{p}}(x,t) + \psi(x,t)/\delta t;$$

4. o movimento das partículas:

$$x^{n+1} = x^n + u\delta t \quad \text{e} \quad y^{n+1} = y^n + v\delta t.$$

A estratégia de decomposição de domínio que propomos para efetuar a implementação paralela do código GENSMAC é dividir o domínio de solução em vários subdomínios de mesmo tamanho ou o mais próximo possível desta situação. A decomposição é realizada aplicando-se cortes verticais no domínio original de solução Ω, que é constituido por células quadradas de mesmo comprimento, pois no código GENSMAC supõe-se que $\delta x = \delta y$. Os cortes são posicionados de modo que cada subdomínio Ω_i contenha aproximadamente a mesma quantidade de células na direção x dos eixos coordenados.

A figura 4.1 a seguir retrata o domínio apresentado na figura 2.1, dividido em três subdomínios, sendo que cada subdomínio contém agora 8 células na direção x e 16 na direção y.

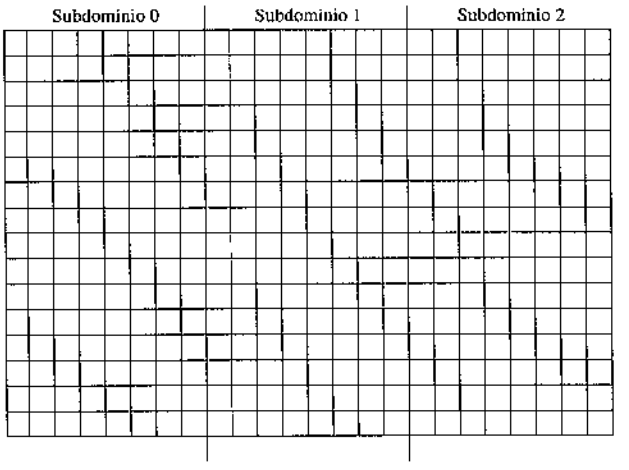


Figura 4.1: Decomposição do domínio de solução

Para exemplificar melhor, vamos considerar que em um dado tempo t o domínio de solução se pareça com o domínio mostrado na figura 4.2:

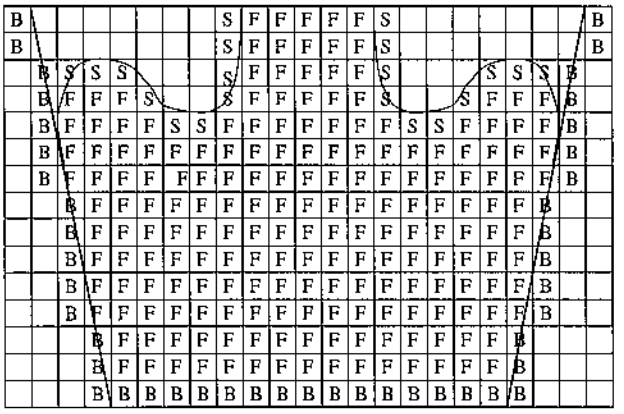


Figura 4.2: Domínio Ω e tipos de células no GENSMAC

A figura 4.2 retrata um recipiente que está sendo preenchido com um determinado fluido. Esse fluido está sendo injetado pela parte superior do recipiente, através de um contorno de entrada. Neste exemplo não está previsto nenhum contorno de saida. Como podemos observar nesta figura, o domínio de solução original é constituido por células quadradas de mesmo comprimento e marcadas com S, B ou F.

No código sequencial, o algoritmo descrito no início desta seção, nos passos de 1 a 4, é aplicado para todas as células marcadas com F, enquanto que o passo 4 é também aplicado para as células marcadas com S ou B. Para as células marcadas com S as velocidades são atualizadas utilizando-se as condições de tensão sobre a superfície livre descritas no capítulo 3. Sobre o contorno rígido, contorno de entrada e contorno de saida as velocidades são calculadas a partir de apropriadas condições de contornos aplicadas em cada caso.

Com a decomposição de domínio, a região contendo as células é dividida em um número fixo de subdomínios. A figura 4.3 ilustra essa situação para o caso particular de três subdomínios: o subdomínio 0 que contém a parte esquerda do domínio original, o subdomínio 1 que contém a parte central e o subdomínio 2 que contém a parte da direita.

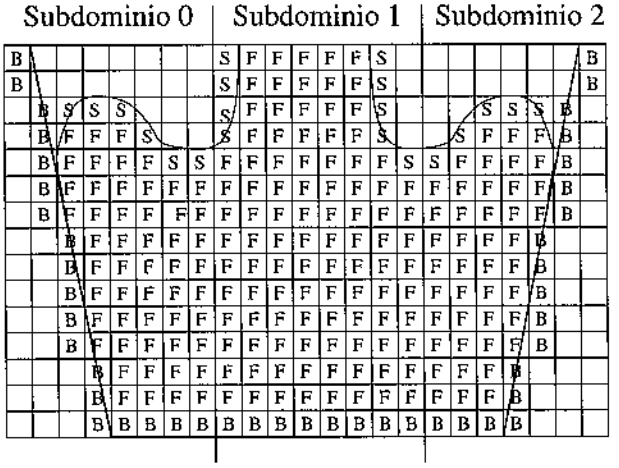


Figura 4.3: Subdomínios Ω_i e tipos de células no GENSMAC

Os dados referentes a cada subdomínio Ω_i devem ser processados por diferentes processadores. Cada processador deverá executar o mesmo algoritmo descrito anteriormente, porém para um conjunto diferente de dados, dentro de seu domínio de atuação.

Para o esquema de diferenças finitas explícito, os passos de 1 a 4 descritos no algoritmo anterior no início desta seção, não apresentam maiores dificuldades para o algoritmo paralelo. As dificuldades de implementação advindas de outras situações são apresentadas neste mesmo capítulo, nas próximas seções, ou mesmo em próximos capítulos, quando os problemas que surgem são relatados.

4.2 A Técnica de Paralelização

A computação paralela envolve o projeto, desenvolvimento e implementação de algoritmos para execução em computadores dotados de várias unidades de processamento, capazes de executar instruções em paralelo. E isso, na maioria das vezes, implica em uma reestruturação dos algoritmos, podendo-se, assim, obter um algoritmo paralelo nada equivalente ao algoritmo sequencial.

Os principais objetivos da programação paralela são:

i) a redução do tempo de execução: se um programa sequencial precisa de t unidades de tempo para executar em um único processador, então idealmente um programa paralelo equivalente seria completado em $\frac{t}{p}$ unidades de tempo, quando executado em p processadores;

ii) o aumento da quantidade de memória: se um programa precisa de w palavras de memória e cada processador tem $\frac{w}{p}$ palavras disponíveis, então p processadores ofereceriam, conjuntamente, a capacidade necessária.

4.2.1 Modelos de Programação Paralela

Existem vários modelos de programação paralela, entre eles podemos destacar o modelo de memória compartilhada e o de memória distribuída. O modelo mais adequado para a implementação da decomposição de domínio proposta na seção anterior é o de memória distribuída, pois permite maior flexibilidade tanto em programação quanto na utilização de equipamentos. Dentro deste contexto, um dos mecanismos de comunicação mais utilizados é o de troca de mensagens, que consiste em processos seqüenciais trocando informações

("dados") entre si através de uma mensagem.

Processo: é constituído do conjunto de instruções do programa em execução e suas posições de memória.

Mensagem: é um conjunto de dados (posições de memória) transferido de um processo para outro.

As mensagens podem ser síncronas ou assíncronas e podem ser de três tipos: um-para-um, um-para-muitos ("multicast") e um-para-todos ("broadcast").

Um-para-um: um processo enviando uma mensagem para outro.

Um-para-muitos: um processo enviando uma mensagem para outros, não todos.

Um-para-todos: um processo enviando uma mensagem para todos os outros.

Para realizar a comunicação entre os processos, ou seja, a troca de mensagens, utilizamos as rotinas do ambiente paralelo PVM ("Parallel Virtual Machine" [14]), "software" de domínio público e disponível no CENAPAD/SP-UNICAMP.

As principais características do PVM são:

- está disponível para diferentes tipos de máquinas: estações de trabalho, PC, computadores MPP (memória distribuida e compartilhada);

- possibilita a portabilidade do código;

- permite o uso transparente de diferentes arquiteturas na mesma aplicação;

- permite a utilização de rede de computadores dispersa geograficamente;

- possibilita o uso de diferentes linguagens (FORTRAN e C) numa mesma aplicação;

- possibilita processos sequenciais assíncronos, cooperando através da troca de mensagens;

- possibilita o particionamento de dados e de programas e, a ordem de execução de uma aplicação é explícita e manual.

Os principais modelos de algoritmos paralelos utilizando o ambiente PVM são: Mestre-escravo e SPMD ("Single-Program, Multiple-Data").

Mestre-escravo: também chamado de "farming", pode ser aplicado quando existem, no mínimo, dois tipos bem definidos de uma mesma aplicação, um deles chamado de "mestre" e o outro de "escravo". A cada um destes será atribuido um processo diferente. A aplicação paralela consistirá de um processo mestre que é responsável por atribuir as tarefas

a serem executadas pelos diversos processos escravos existentes.

SPMD: um único programa (mesmo conjunto de instruções) opera sobre diferentes dados.

O particionamento de dados entre os processos é necessário para se reduzir o tempo de execução da aplicação paralela quando comparado com uma aplicação sequencial equivalente. Como o particionamento de dados impõe o algoritmo paralelo a ser utilizado, o modelo que utilizamos é o SPMD, pois o mesmo programa deve ser executado em diferentes subdomínios.

4.2.2 O pacote PIM

Para resolver o sistema linear advindo da equação de Poisson, no passo 2 do algoritmo descrito na seção 4.1, utilizamos uma implementação paralela do método dos gradientes conjugados apresentada em Cunha [8] e Cunha e Hopkins [9] e distribuida no pacote de métodos iterativos PIM (Cunha e Hopkins [10]). A formação desse sistema é discutida na próxima seção. O PIM implementa todo o código necessário para as iterações do método dos gradientes conjugados, a menos da rotina que calcula o produto matriz-vetor, que deve ser fornecida pelo usuário.

O pacote PIM, em sua versão 2.2, é constituido de um conjunto de rotinas em FORTRAN 77 para a solução de sistemas de equações lineares usando métodos iterativos em arquiteturas paralelas. Além do método dos gradientes conjugados (CG), implementa também os seguintes métodos:

BI-CG, "Bi-Conjugate-Gradients",

CGS, "Conjugate-Gradients squared",

BI-CGSTAB, "The stabilized version of Bi-Conjugate-Gradients",

GMRES, "Generalized minimal residual",

GCR, "Generalized Conjugate residual",

CGNR, "Conjugate-Gradients for normal equations with minimisation of the residual norm",

CGNE, "Conjugate-Gradients for normal equations with minimisation of the error norm",

TFQMR, "The transpose-free quasi-minimal residual",

RBi-CGSTAB, "Restarted Bi-CGSTAB",

Chebyshev, "Aceleração de Chebyshev",

QMR, "Quasi-Minimal Residual".

Os sistemas podem ser reais ou complexos e podem ser resolvidos utilizando-se precisão simples ou dupla.

O modelo de programação utilizado é o SPMD, o mesmo que utilizamos na paralelização

do código GENSMAC. Esse conjunto de programas apresenta a seguinte estrutura:

. rotinas implementando os métodos iterativos,

. rotinas de suporte (critérios de parada, parâmetros),

. BLAS nível 1/nível 2 ("Basic Linear Algebra Subroutines", rotinas de domínio público),

. rotinas fornecidas pelo usuário.

4.3 A Implementação Paralela

A implementação paralela do código GENSMAC segue o mesmo roteiro de implementação utilizado na versão sequencial. Ambos os códigos foram elaborados em FORTRAN 77 e têm basicamente a mesma estrutura. Algumas rotinas existem somente no programa paralelo, outras foram adaptadas do código sequencial para o paralelo e outras, ainda, não sofreram qualquer alteração de um caso para outro. Descrevemos a seguir o roteiro de implementação do código paralelo, fazendo comparações, quando necessário, com o código sequencial. Dividimos essa narração em quatro partes: na primeira comentamos sobre a seção de marcação das células, na segunda comentamos sobre a decomposição de domínio e a distribuição de dados para os respectivos processos, na terceira comentamos sobre os modelos de comunicação utilizados e, finalmente, na quarta e última parte comentamos sobre o ciclo de cálculo.

A estrutura completa de ambas as implementações, sequencial e paralela, do código GENSMAC e uma breve descrição de cada rotina é apresentada no final deste trabalho, no apêndice A. Apresentamos aqui apenas os aspectos e dificuldades da implementação paralela.

4.3.1 A Marcação Inicial de Células

Lembramos que na metodologia GENSMAC o domínio de solução é dividido em células quadradas de mesmo tamanho e o contorno rígido é definido por segmentos de retas ou curvas. A partir dessas informações é possível identificar quais células pertencem ao contorno rígido e quais são interiores a ele. Essa tarefa é realizada na seção de marcação das células, identificando e marcando com B ("BOUNDARY") as células que fazem parte do contorno rígido e com E ("EMPTY") as células interiores, indicando que ainda não existe presença de fluido nessa região, naquele instante de tempo. A implementação dessa seção é exatamente a mesma para ambas as versões, sequencial e paralela, uma vez que essa tarefa é realizada para o domínio de solução original, ainda não decomposto em subdomínios.

4.3.2 A Decomposição de Domínio e a Distribuição dos Dados

Nesta seção, vamos discutir as rotinas que realizam a decomposição de domínio e a distribuição dos dados para os seus respectivos processos, uma vez que para cada subdomínio considerado na aplicação, será gerado um processo para o processamento computacional.

A partir da quantidade de subdomínios desejada e da quantidade de colunas de células existentes no domínio original de solução, a decomposição de domínio é realizada identificando-se regiões formadas por colunas de células. Cada região representa um subdomínio e deve conter a mesma quantidade de colunas de células que as demais. Como nem sempre é possível uma distribuição uniforme, as primeiras regiões podem ter uma coluna de células a mais que as demais.

Dando continuidade a este procedimento, vem a fase de identificação dos dados que serão distribuidos para os respectivos processos.

O primeiro passo a ser realizado é a identificação das células que compõem os contornos de entrada e de saida, quando estes fazem parte do problema que deverá ser resolvido. Essa tarefa é realizada separadamente em cada subdomínio considerado na aplicação. No caso do código sequencial, a identificação dessas células é realizada no domínio original de solução.

Está previsto, também, na metodologia GENSMAC, a existência de corpos de fluidos dentro do domínio de solução no instante inicial de cálculo. Cada corpo de fluido nesta situação pode ter a forma retangular ou circular. A seguir, são dados dois exemplos que ilustram esses casos. No primeiro exemplo, mostrado na figura 4.4, está ilustrado um domínio de solução constituido por um contorno rígido, sem contornos de entrada e de saida e contendo um corpo de fluido retangular em seu interior.

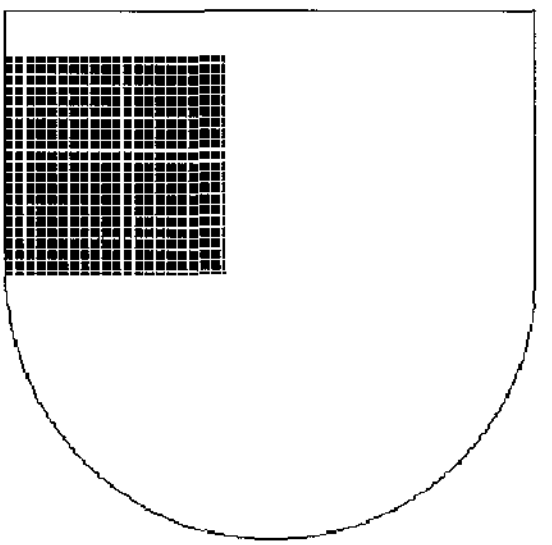


Figura 4.4: Domínio 1 com massa de fluido no seu interior

A figura 4.5 mostra um domínio de solução com contorno de entrada à direita e contorno de saida à esquerda e com um corpo de fluido, na forma circular, contida em seu interior.

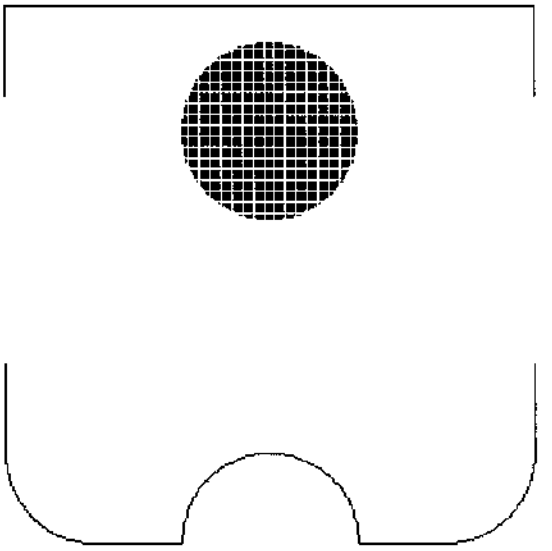


Figura 4.5: Domínio 2 com massa de fluido no seu interior

Na implementação paralela são identificados os subdomínios que possuem, no instante inicial, um corpo de fluido ou apenas uma parte dele. Essas informações são, então, repassadas para os seus respectivos processos.

Uma outra questão que surge na metodologia GENSMAC, é a questão do contorno rígido. Esse contorno é representado por segmentos de retas ou curvas, como mostra a figura 4.6 a seguir para um caso particular.

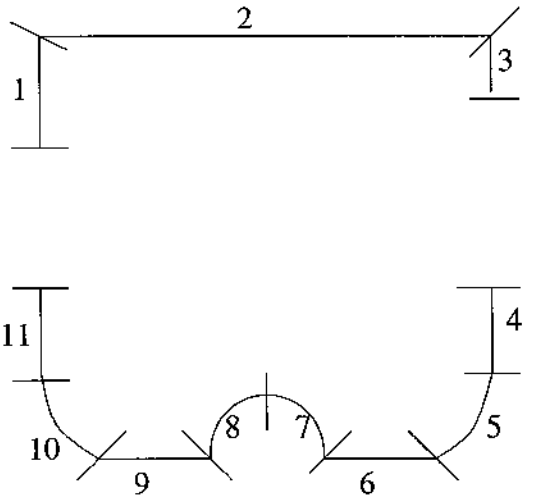


Figura 4.6: Segmentos de retas ou curvas que definem o contorno rígido para um caso típico

Neste caso, existem 11 segmentos definindo o contorno rígido, numerados de 1 a 11. Esses segmentos, além de definir o contorno rígido, são utilizados também no cálculo das condições de contorno, servindo de referência para a localização das células que representam o contorno rígido.

Na implementação paralela, esses segmentos são identificados e distribuidos para os seus respectivos subdomínios. Quando um segmento pertence a mais de um subdomínio, ele é "quebrado" em várias partes, de modo que cada subdomínio envolvido na decomposição contenha apenas a parte que está em sua região.

Para exemplificar, vamos considerar o mesmo conjunto de segmentos mostrado na figura 4.6. A figura 4.7 a seguir mostra como ficam esses segmentos com a decomposição do domínio original em três subdomínios.

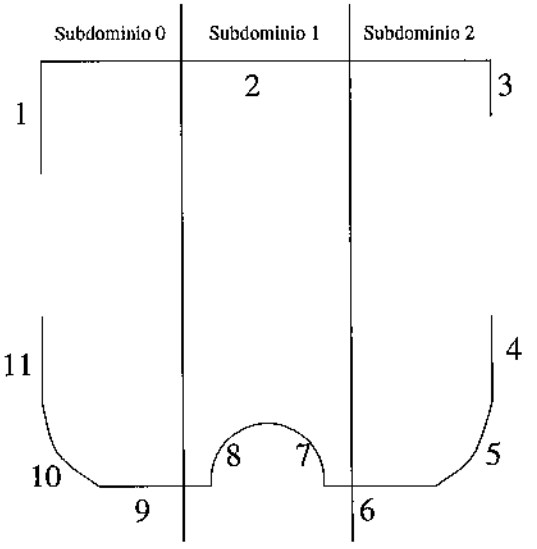


Figura 4.7: Segmentos que definem o contorno rígido distribuído por subdomínios

Neste exemplo, o segmento 2 é dividido em 3 partes: a parte da esquerda fica com o subdomínio 0, a parte central fica com o subdomínio 1 e a parte da direita fica com o subdomínio 2. Os segmentos 6 e 9 são divididos em duas partes, sendo que cada parte é atribuida ao seu respectivo subdomínio. Os demais segmentos são simplesmente distribuidos aos seus respectivos processos, não sofrendo qualquer alteração.

O término desta tarefa, finaliza a seção de decomposição de domínio e de distribuição de dados. Os dados referentes a cada subdomínio estão, assim, prontos para serem processados pelos diversos processadores que irão colaborar na solução do problema. Os processos referentes a cada subdomínio, armazenarão apenas as informações pertencentes ao seu domínio de atuação. Essas informações não serão conhecidas por processos referentes a outros subdomínios.

4.3.3 Modelos de Comunicação Utilizados

A última seção na implementação do código GENSMAC é composta pelas rotinas que executam o ciclo de cálculo. Antes, porém, de iniciar os comentários referentes a essa seção, vamos discutir os modelos de comunicação que são utilizados para troca de informações entre os processadores. O primeiro modelo é a comunicação entre os processadores vizinhos. Em determinados estágios do processamento, faz-se necessário que cada processador conheça informações armazenadas pelo processador da esquerda e/ou da direita. Um exemplo desta situação é a disposição das variáveis das equações de momento em uma malha diferenciada, como mostra a figura 4.8:

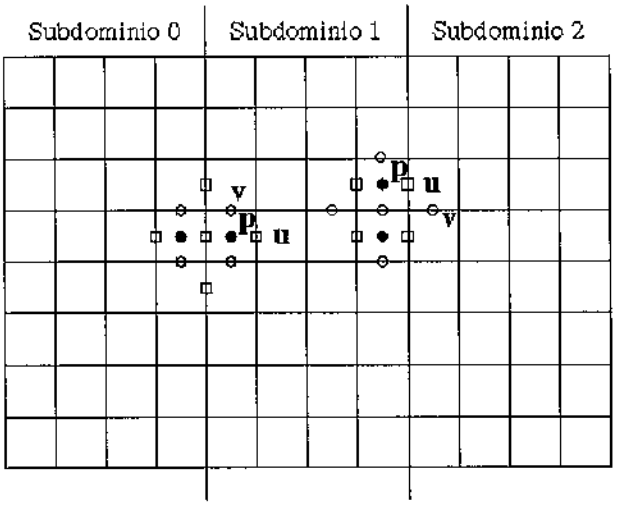


Figura 4.8: Distribuição das variáveis para as equações de momento.

Como podemos observar nesta figura, para o cálculo das velocidades e pressão nas vizinhanças dos subdomínios, cada processador deve conhecer as informações armazenadas pelos processadores vizinhos, não sendo necessário o conhecimento de informações armazenadas por mais de uma coluna de células. O processador 0 deve conhecer as informações contidas na primeira coluna de células do processador 1, o último processador deve conhecer as informações contidas na última coluna de células do penúltimo processador, o processador 1 deve conhecer as informações armazenadas pela última coluna de células do processador 0 e pela primeira coluna de células do processador 2 e, assim por diante. Este esquema é apresentado na figura 4.9:

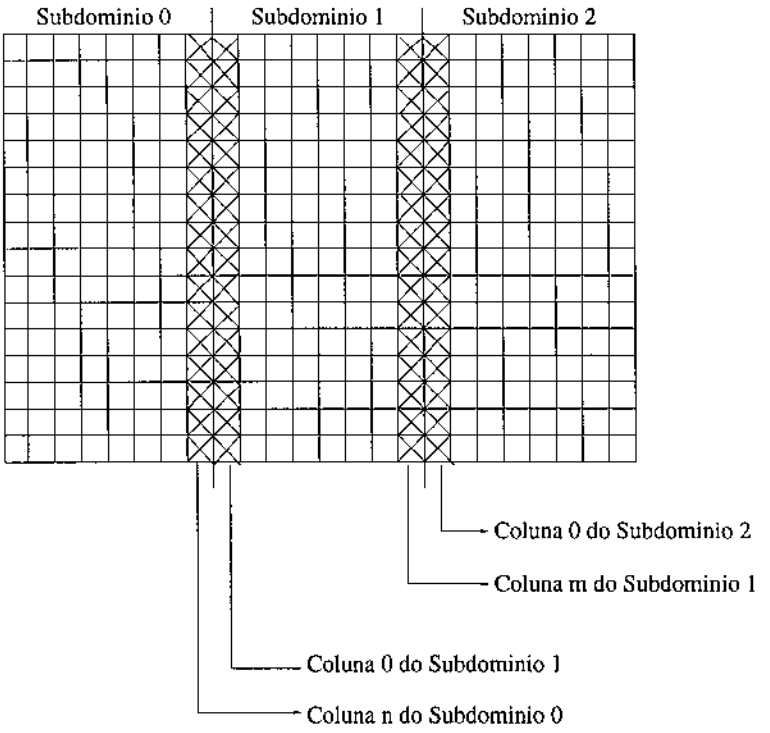


Figura 4.9: Comunicação entre processadores vizinhos

O próximo modelo de comunicação utilizado é o processo de redução. Esse processo é utilizado quando uma determinada informação tiver que ser conhecida por todos os processadores ou por um grupo deles.

Para auxiliar no processo de redução, utilizamos uma rotina contida no pacote PIM (descrito anteriormente), chamada TREE. Essa rotina constrói uma árvore binária balanceada de processadores. Por exemplo, no caso de termos oito processadores, teremos a seguinte árvore binária:

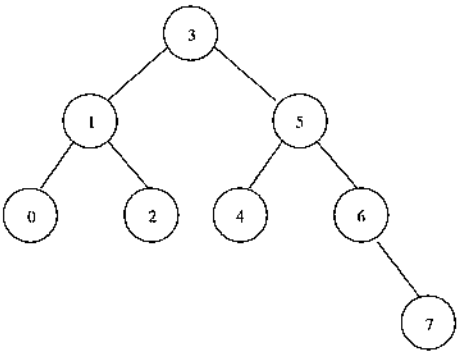


Figura 4.10: Árvore binária balanceada para 8 processos

Essa árvore é utilizada como referência para informar quais os processadores que devem trocar informações entre si. Para exemplificar, suponha que a operação a ser realizada seja

uma soma, na qual cada processador contribua com alguma parcela. O processador 7 envia sua contribuição para o processador 6, que calcula a soma parcial. Por sua vez os processadores 0 e 2 e 4 e 6 enviam suas constribuições, respectivamente, para os processadores 1 e 5, que executam as somas parciais e enviam esses dados para o processador 3, que finalmente conclui a soma global e retorna esse valor para os processadores 1 e 5. O processador 1 repassa esse valor para os processadores 0 e 2 e o processador 5 repassa para os processadores 4 e 6. Finalmente o processador 6 comunica a soma global para o processador 7, completando, desta forma, o processo de redução.

Na próxima seção tecemos comentários sobre o ciclo de cálculo definido pelo algoritmo descrito no início deste capítulo.

4.3.4 O Ciclo de Cálculo

A primeira tarefa a ser realizada dentro desse ciclo é o cálculo das condições de contorno. São calculados, nesta fase, as condições sobre o contorno rígido, sobre a superfície livre e sobre os contornos de entrada e de saida. A comunicação entre os processadores vizinhos, no caso do código paralelo, só acontece quando existem células "SURFACE" na vizinhança dos processadores. A geometria do problema faz o modelo de comunicação, nesse caso, muito mais complexo, pois a superfície livre tem forma arbitrária e está constantemente se modificando. É utilizado endereçamento indireto para que haja troca de informações apenas de elementos que estejam sobre a superfície livre, minimizando as comunicações envolvidas.

Após o cálculo das condições de contorno, é calculada a pressão sobre a superfície livre de acordo com a condição de tensão normal. No final desses cálculos é realizada a comunicação dos novos valores obtidos entre os processadores vizinhos.

Na sequência, cada processador fará o cálculo do tamanho do passo temporal. Como este cálculo depende das velocidades calculadas anteriormente, que podem ser diferentes em cada subdomínio, cada processador poderá obter um espaçamento diferente dos demais. Desta forma, é realizado um processo de redução envolvendo todos os processos para se determinar o menor valor e também para a distribuição dessa informação para todos os processadores envolvidos na aplicação.

Conhecidos o passo temporal, são calculadas as velocidades provisórias por uma fórmula de discretização explícita, não sendo necessárias comunicações entre os processadores nesse estágio.

O próximo passo é a construção e resolução do sistema linear resultante da discretização de cinco pontos da equação de Poisson. Calculados os coeficientes deste sistema, sua solução é, então, determinada através do pacote PIM.

O processo de escoamento de fluido é dinâmico por natureza. Com a técnica de decom-

posição de domínio que estamos utilizando, podem ocorrer situações em que determinados subdomínios recebam massa de fluido somente após decorrido algum tempo de processamento, ou mesmo não venham a receber fluido durante todo o processamento. Por exemplo, no início de uma aplicação, um problema com contorno de entrada não apresentará fluido em nenhum de seus subdomínios. Com a evolução do tempo vão surgindo, aos poucos, quantidades de massa de fluido nos subdomínios que apresentam contorno de entrada. Esse fluido vai se espalhando até atingir outros subdomínios envolvidos na aplicação.

Um processador contribuirá na solução do sistema linear em questão somente quando há fluido no interior de seu subdomínio. Desta forma, participam da resolução desse sistema, somente os processadores que contribuem para ele. Devido às particularidades do processo de comunicação cada processador precisa conhecer quais os outros que irão colaborar na execução dessa tarefa e quais não irão. Novamente é utilizado, neste estágio de processamento, o processo de redução para informar a todos os processadores quais são os que irão colaborar na solução do sistema linear em questão. A partir daí, só participam dessa operação os processadores envolvidos nesta etapa de processamento, e as comunicações exigidas nesta fase de processamento só acontecem entre os processadores participantes desse grupo.

Uma outra questão, relativa a implementação dessa fase, é que o PIM implementa todo o código necessário para as iterações do método dos gradientes conjugados, a menos das rotinas para os produtos matriz-vetor e redução de escalares para o cálculo dos produtos internos e normas de vetores, que devem ser fornecidas pelo usuário. Neste contexto, isso é uma das maiores dificuldades, pois o produto matriz-vetor é bastante complexo e requer uma implementação cuidadosa. Cada iteração do método dos gradientes conjugados requer o cálculo do produto matriz-vetor, $y = Ax$, onde, para cada ponto da malha (i, j), o correspondente $y_{i,j}$ é calculado da seguinte forma:

$$y_{i,j} = a_{i,j}x_{i,j} + a_{i+1,j}x_{i+1,j} + a_{i-1,j}x_{i-1,j} + a_{i,j+1}x_{i,j+1} + a_{i,j-1}x_{i,j-1},$$

onde alguns dos valores $a_{s,r}$ podem ser nulos ou não, de acordo com a sua localização na malha. Esses coeficientes são obtidos a partir da equação diferencial parcial e das condições de contorno.

A principal dificuldade em executar o produto matriz-vetor está associada à irregularidade do domínio. Isso é realizado de maneira tal que somente os pontos que estão inseridos dentro da região que contém fluido (células "FULL") sejam incluidos nos cálculos. Para se alcançar isso, foi utilizado endereçamento indireto de modo a incluir as contribuições dos pontos mais distantes (da esquerda e da direita em nosso caso, pois estamos numerando as incógnitas por colunas) e, também, para minimizar a comunicação, pois não queremos

comunicar os coeficientes dos pontos que não serão utilizados no cálculo desse produto. Para pontos dentro da região Ω (marcadas com células "FULL") com vizinhos fora de Ω, os correspondentes coeficientes são considerados nulos. Para implementar as condições de contorno corretas é necessário, nesse estágio, modificar os coeficientes e/ou o lado direito do vetor apropriadamente. Note que os coeficientes da matriz são formados uma única vez e, então, mantidos fixos durante todas as iterações do método. A generalidade da geometria faz o modelo de comunicação com o vizinho mais próximo muito mais complexo que o caso de uma região quadrada, pois os tamanhos e posições dos vetores dependem agora da geometria do problema, que pode se alterar a cada ciclo de cálculo.

Após a determinação da solução do sistema linear, o programa paralelo exige a comunicação de elementos da solução que estejam nas vizinhanças de processadores.

Embora tenha sido implementada uma rotina específica para o código sequencial, para se determinar a solução do sistema linear pelo método dos gradientes conjugados, optamos por substituí-la pela implementação sequencial disponível no pacote PIM pois, assim, temos exatamente a mesma implementação, tanto no programa sequencial quanto no paralelo e, consequentente, podemos fazer uma melhor análise da performance do programa paralelo.

O próximo passo após a solução da equação de Poisson são os ajustes nas velocidades e pressão, ou seja, a execução do passo 3 do algoritmo apresentado no início desse capítulo. Esse passo não apresenta maiores dificuldades na implementação paralela, não sendo exigidas comunicações entre os processadores nesse estágio do processamento.

O passo seguinte, é a aplicação das condições de contorno novamente, já discutida anteriormente e, em seguida, a movimentação das partículas.

Conjuntamente com a movimentação das partículas, são realizados os processos de eliminação das partículas que ultrapassam os contornos de saida e de geração de novas partículas que são inseridas dentro do domínio de solução através de contornos de entrada.

Na movimentação das partículas, algumas podem migrar de um processador para outro. Neste caso, há a necessidade de transferências de partículas entre os processadores vizinhos. As partículas que serão transferidas de um processador para outro, são armazenadas e transferidas todas de uma única vez no final de processamento de cada ciclo de cálculo, evitando-se, com isso, um excesso de comunicação.

Nesse estágio de movimentação das partículas são também realizados ajustes nas velocidades u e v das células que estão sobre a superfície livre. No caso de uma partícula estar se transferindo de um processador para outro, esses ajustes são calculados em ambos os processadores, evitando-se dessa maneira que sejam realizadas comunicações na atualização dessas velocidades.

Quando um contorno de entrada (respectivamente, de saída) está dividido por mais de um processador, o processo de gerar (respectivamente, eliminar) partículas é executado parte

em um processador, parte em outro, cada processador gerando (ou eliminando) somente as partículas que estão em seu domínio de atuação.

A última etapa dentro do ciclo de cálculo, é a fase de atualização das células. Nesta fase, é feita uma atualização da matriz de células que armazena informações sobre cada célula dentro do domínio de solução.

Após a execução dessa rotina é necessário a comunicação entre os processadores vizinhos das colunas da matriz de células que estejam nas vizinhanças dos processadores. Pode também ocorrer o caso em que uma célula "FULL" ou "SURFACE" na divisa de algum processador, passar a ser "EMPTY". Neste caso, o valor da velocidade u, que era não nulo, passa a ser nulo e, então, é exigida a comunicação dessa posição de memória para o processador vizinho.

As únicas alterações dessa rotina do caso sequencial para o paralelo, são os controles das variáveis que devem ser comunicadas de um processador para outro e as comunicações que se fazem necessárias.

Com o término da seção de marcação das células se encerra o ciclo de cálculo, sendo realizadas então as comunicações das velocidades entre os processadores vizinhos. Neste estágio, todos os processadores conhecem todas as informações necessárias para iniciar novamente o ciclo de cálculo, que se encerrará quando a soma dos espaçamentos no tempo, calculado em cada ciclo, atingir o tempo previsto para o encerramento da aplicação.

Com o término do processamento dos dados de cada subdomínio, finaliza-se a execução do programa. A junção dos resultados obtidos por cada processador fornecerá o resultado global da aplicação.

4.4 Considerações Finais

Nos parágrafos anteriores, primeiro apresentamos a técnica de decomposição de domínio que propusemos e depois expusemos brevemente os aspectos da implementação paralela.

Com os comentários tecidos anteriormente sobre a implementação do código paralelo, nosso objetivo era apenas o dar uma idéia das dificuldades e do roteiro de implementação, e não entrar em maiores detalhes, pois se tornaria uma leitura extensa, cansativa e nada atrativa.

No próximo capítulo, expomos a técnica GENSMAC em coordenadas cilíndricas para a resolução de problemas tri-dimensionais eixo-simétricos e comentamos as implementações sequencial e paralela para este caso.

Capítulo 5

O Método SMAC em Coordenadas Cilíndricas

Como mencionamos nos capítulos anteriores, o código GENSMAC foi projetado para resolver problemas de escoamento de fluidos incompressíveis em coordenadas cartesianas em domínios bi-dimensionais gerais. Neste capítulo, vamos descrever as implementações sequencial e paralela em coordenadas cilíndricas do código GENSMAC, projetado para simular problemas de escoamentos de fluidos com superfícies livres que são eixo-simétricos, ou seja, que são simétricos em relação a um dos eixos coordenados. Tais problemas são frequentemente encontrados em muitas aplicações industriais, tais como indústrias química e alimentícia.

O método SMAC em coordenadas cilíndricas para solucionar problemas tri-dimensionais que são eixo-simétricos segue o mesmo procedimento descrito anteriormente.

5.1 As Equações Básicas

Consideremos escoamentos Newtonianos eixo-simétricos incompressíveis. As equações básicas são as equações adimensionais de massa e de momento que, na forma conservativa em coordenadas cilíndricas, são dadas por (Amsden e Harlow [2]):

$$\frac{1}{r}\frac{\partial(rv)}{\partial r} + \frac{\partial v}{\partial z} = 0, \tag{5.1}$$

$$\frac{\partial u}{\partial t} + \frac{1}{r}\frac{\partial(ru^2)}{\partial r} + \frac{\partial(uv)}{\partial z} = -\frac{\partial p}{\partial r} + \frac{1}{Re}\frac{\partial}{\partial z}\left(\frac{\partial u}{\partial z} - \frac{\partial v}{\partial r}\right) + \frac{1}{F_r^2}g_r, \tag{5.2}$$

$$\frac{\partial v}{\partial t}+\frac{1}{r}\frac{\partial(ruv)}{\partial r}+\frac{\partial(v^2)}{\partial z}=-\frac{\partial p}{\partial r}-\frac{1}{Re}\frac{1}{r}\frac{\partial}{\partial r}\left(r\left(\frac{\partial u}{\partial z}-\frac{\partial v}{\partial r}\right)\right)+\frac{1}{F_r^2}g_z, \tag{5.3}$$

onde $Re = UL/\nu$ e $Fr = -U/\sqrt{Lg}$ denotam, respectivamente, o número de Reynolds e o número de Froude. L e U representam, respectivamente, escala e velocidade típicas, ν é uma viscosidade de referência e $g = |\mathrm{g}|$ representa a aceleração da gravidade. Além do mais, $\mathrm{u} = (u, v)^T$ são as componentes de velocidade radial e vertical e p é a pressão adimensional.

5.2 Método de Solução

Para resolver as equações (5.1), (5.2) e (5.3) empregando a metodologia GENSMAC, vamos supor que em um dado tempo t_0, a velocidade $u(r, z, t_0)$ é conhecida e são dadas a pressão e as condições de contorno para a velocidade. A atualização da velocidade $u(r, z, t)$ em $t = t_0 + \delta t$ é, então, calculada da seguinte forma:

1. seja $\tilde{p}(r, z, t)$ uma pressão satisfazendo exatamente as condições de pressão da superfície livre. Esta pressão é calculada de acordo com as condições de tensão dadas na seção 5.3.1;

2. calcula-se a velocidade intermediária $\tilde{u}(r, z, t)$ por uma discretização explícita de diferenças finitas, a partir das seguintes equações:

$$\frac{\partial u}{\partial t}=\left[-\frac{1}{r}\frac{\partial(ru^2)}{\partial r}-\frac{\partial(uv)}{\partial z}-\frac{\partial\tilde{p}}{\partial r}+\frac{1}{Re}\frac{\partial}{\partial z}\left(\frac{\partial u}{\partial z}-\frac{\partial v}{\partial r}\right)+\frac{1}{F_r^2}g_r\right]_{t=t_0}, \tag{5.4}$$

$$\frac{\partial v}{\partial t}=\left[-\frac{1}{r}\frac{\partial(ruv)}{\partial r}-\frac{\partial(v^2)}{\partial z}-\frac{\partial\tilde{p}}{\partial r}-\frac{1}{Re}\frac{1}{r}\frac{\partial}{\partial r}\left(r\left(\frac{\partial u}{\partial z}-\frac{\partial v}{\partial r}\right)\right)+\frac{1}{F_r^2}g_z\right]_{t=t_0}, \tag{5.5}$$

com $\tilde{u}(r, z, t_0) = u(r, z, t_0)$, utilizando-se as condições de contorno corretas para $u(r, z, t_0)$. Foi mostrado por Tomé e co-autores [34], que $\tilde{u}(r, z, t)$ possui a vorticidade correta no tempo t. Contudo, $\tilde{u}(r, z, t)$ não satisfaz à equação (5.1).

Seja

$$u(r, z, t) = \tilde{u}(r, z, t) - \nabla\psi(r, z, t), \tag{5.6}$$

com

$$\nabla^2\psi(r, z, t) = \nabla \cdot \tilde{u}(r, z, t). \tag{5.7}$$

Agora $u(r,z,t)$ satisfaz à equação (5.1) e a vorticidade permanece inalterada. Desta forma, $u(r,z,t)$ é identificada como sendo a velocidade atualizada no tempo t;

3. resolve-se a equação de Poisson:

$$\nabla^2\psi(r,z,t) \;=\; \nabla\cdot\tilde{u}(r,z,t);$$

4. calcula-se a velocidade:

$$u(r,z,t) \;=\; \tilde{u}(r,z,t) - \nabla\psi(r,z,t);$$

5. calcula-se a pressão. Pode-se mostrar que essa pressão é dada por (Tomé e co-autores [34]):

$$p(r,z,t) \;=\; \tilde{p}(r,z,t) + \psi(r,z,t)/\delta t;$$

6. atualiza-se a posição das partículas.

O último passo é o cálculo da movimentação das partículas para suas novas posições. As coordenadas dessas partículas são armazenadas e atualizadas no final de cada ciclo de cálculo, resolvendo-se, pelo método de Euler, as seguintes equações diferenciais ordinárias:

$$\frac{dr}{dt} \;=\; u, \quad \frac{dz}{dt} \;=\; v.$$

Isso fornece as novas coordenadas de uma partícula, permitindo-nos determinar se ela moveu-se ou não para uma nova célula ou se deixou a região de solução através de um contorno de saida.

5.2.1 Condições de Contorno

As condições de contorno devem ser impostas sobre o contorno rígido e sobre a superfície livre. Sobre o contorno rígido, podem ser impostas condições de contorno de não escorregamento, de escorregamento livre, de entrada pré-estabelecido, de saida pré-estabelecido e de saida contínuo. Essa implementação não sofreu qualquer alteração em relação ao caso anterior. As condições de contorno apropriadas para a superfície livre são as nulidades das tensões normal e tangencial que, na ausência de tensão superficial, são dadas por (Batchelor [3]):

$$n \cdot \sigma \cdot n = 0, \tag{5.8}$$

$$m \cdot \sigma \cdot n = 0, \tag{5.9}$$

onde n e m são, respectivamente, os vetores unitários normal e tangencial local e σ é o tensor de tensões. Essas condições são aproximadas por diferenças finitas locais (ver Tomé [33]) e são descritas na seção (5.3.1).

As condições de contorno apropriadas para a equação de Poisson (5.7) são dadas por (Amsden e Harlow [2]):

$\frac{\partial \psi}{\partial n} = 0$ sobre o contorno fixo e $\psi = 0$ sobre a superfície livre.

5.3 As Equações de Diferenças Finitas

As equações descritas na seção 5.2 são aproximadas por diferenças finitas sobre uma malha diferenciada. As variáveis de pressão $\tilde{p}_{ij}$ e a velocidade potencial ψ_{ij} são posicionadas no centro da célula, enquanto que u_{ij} e v_{ij} são deslocados por uma translação, respectivamente, de $\delta r/2$ e de $\delta z/2$.

As equações de momento (5.4) e (5.5) são discretizadas e aplicadas, respectivamente, nos nós u e v. É utilizada uma fórmula de diferenças regressivas no tempo para as derivadas com relação ao tempo, enquanto que os termos espaciais lineares do lado direito são aproximados por diferenças centrais. Para os termos de fluxo (uv), são usadas médias simples, como por exemplo:

$$(uv)_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{u_{i+\frac{1}{2},j} + u_{i+\frac{1}{2},j+1}}{2} \cdot \frac{v_{i,j+\frac{1}{2}} + v_{i+1,j+\frac{1}{2}}}{2}.$$

Assim, segundo Harlow e Welch [17], as aproximações de diferenças finitas se tornam:

$$\begin{aligned} \tilde{u}_{i+\frac{1}{2},j} = u_{i+\frac{1}{2},j} + \delta t \Bigg[& \frac{u_{i+\frac{1}{2},j}}{r_{i+\frac{1}{2}} \delta r} \left(r_i u_{i-\frac{1}{2},j} - r_{i+1} u_{i+\frac{3}{2},j} \right) + \frac{\tilde{p}_{i,j} - \tilde{p}_{i+1,j}}{\delta r} \\ & + \frac{1}{F_r^2} g_r + \frac{1}{4} \Big(\left(u_{i+\frac{1}{2},j} + u_{i+\frac{1}{2},j-1} \right) \left(v_{i,j-\frac{1}{2}} + v_{i+1,j-\frac{1}{2}} \right) \\ & - \left(u_{i+\frac{1}{2},j} + u_{i+\frac{1}{2},j+1} \right) \left(v_{i,j+\frac{1}{2}} + v_{i+1,j+\frac{1}{2}} \right) \Big) \end{aligned}$$

$$+\frac{1}{Re}\left(\frac{u_{i+\frac{1}{2},j+1}+u_{i+\frac{1}{2},j-1}-2u_{i+\frac{1}{2},j}}{\partial z^2}\right.$$

$$\left.-\frac{v_{i+1,j+\frac{1}{2}}-v_{i+1,j-\frac{1}{2}}-v_{i,j+\frac{1}{2}}+v_{i+1,j-\frac{1}{2}}}{\partial r\partial z}\right)\Bigg], \tag{5.10}$$

$$\begin{aligned}\tilde{v}_{i,j+\frac{1}{2}} &= v_{i,j+\frac{1}{2}}+\delta t\left[\frac{v_{i,j+\frac{1}{2}}\left(v_{i,j-\frac{1}{2}}-v_{i,j+\frac{3}{2}}\right)}{\partial z}+\frac{\tilde{P}_{i,j}-\tilde{P}_{i,j+1}}{\delta z}+\frac{1}{F_r^2}g_z\right.\\
&+\frac{1}{4r_i\partial r}\left(r_{i-\frac{1}{2}}\left(u_{i-\frac{1}{2},j}+u_{i-\frac{1}{2},j+1}\right)\left(v_{i-1,j+\frac{1}{2}}+v_{i,j+\frac{1}{2}}\right)\right.\\
&\left.-r_{i+\frac{1}{2}}\left(u_{i+\frac{1}{2},j}+u_{i+\frac{1}{2},j+1}\right)\left(v_{i+1,j+\frac{1}{2}}+v_{i,j+\frac{1}{2}}\right)\right)\\
&-\frac{1}{Re}\frac{1}{r_i\partial r}\left(r_{i+\frac{1}{2}}\left(\frac{u_{i+\frac{1}{2},j+1}+u_{i+\frac{1}{2},j}}{\partial z}-\frac{v_{i+1,j+\frac{1}{2}}-v_{i,j+\frac{1}{2}}}{\partial r}\right)\right.\\
&\left.\left.-r_{i-\frac{1}{2}}\left(\frac{u_{i-\frac{1}{2},j+1}+u_{i-\frac{1}{2},j}}{\partial z}-\frac{v_{i,j+\frac{1}{2}}-v_{i-1,j+\frac{1}{2}}}{\partial r}\right)\right)\right].\end{aligned} \tag{5.11}$$

A equação (5.7), em coordenadas cilíndricas, se torna:

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial\psi}{\partial r}\right)+\frac{\partial^2\psi}{\partial z^2}=\tilde{D}, \tag{5.12}$$

onde

$$\tilde{D} = \frac{1}{r}\frac{\partial(r\tilde{u})}{\partial r}+\frac{\partial\tilde{v}}{\partial z}.$$

Observamos que a discretização direta de (5.12) leva a um sistema linear não simétrico, que obviamente não poderá ser resolvido pelo método dos gradientes conjugados. Contudo, reescrevendo a equação (5.12) na forma:

$$\frac{\partial}{\partial r}\left(r\frac{\partial\psi}{\partial r}\right)+r\frac{\partial^2\psi}{\partial z^2}=r\tilde{D} \tag{5.13}$$

e, discretizando a equação (5.13) no centro da célula, obtemos:

$$-r_i\psi_{i,j-1}-r_{i-\frac{1}{2}}\psi_{i-1,j}+4r_i\psi_{i,j}-r_{i+\frac{1}{2}}\psi_{i+1,j}-r_i\psi_{i,j+1}=-r_i\tilde{D}_{i,j}, \tag{5.14}$$

onde

$$\tilde{D}_{i,j}=\frac{1}{r_i}\left(\frac{r_{i+\frac{1}{2}}\tilde{u}_{i+\frac{1}{2},j}-r_{i-\frac{1}{2},j}}{\delta r}\right)+\left(\frac{\tilde{v}_{i,j+\frac{1}{2}}-\tilde{v}_{i,j-\frac{1}{2}}}{\delta z}\right).$$

Como pode ser observado, (5.14) nada mais é do que um sistema linear com matriz simétrica definida positiva. Desta forma, igualmente ao caso anterior, pode-se empregar o método dos gradientes conjugados. Vale ainda lembrar que os coeficientes de (5.14) não são mais constantes, como no caso cartesiano, porém variáveis.

5.3.1 Condições de Tensão sobre a Superfície Livre

A condição de tensão (5.8)-(5.9) pode ser escrita como (Tomé e co-autores [35]):

$$p - \frac{2}{Re}\left[\frac{\partial u}{\partial r}n_r^2 + \frac{\partial v}{\partial z}n_z^2 + \left(\frac{\partial u}{\partial z} + \frac{\partial v}{\partial r}\right)n_r n_z\right] = 0, \tag{5.15}$$

$$\frac{2}{Re}\left[\left(\frac{\partial u}{\partial r} - \frac{\partial v}{\partial z}\right)n_r n_z + \left(\frac{\partial u}{\partial z} + \frac{\partial v}{\partial r}\right)\left(n_r^2 - n_z^2\right)\right] = 0. \tag{5.16}$$

A aplicação destas condições são semelhantes às aproximações adotadas no caso anterior, ou seja, considere dois tipos de orientações de superfícies livre, como a seguir.

i) Superfícies verticais/horizontais.

Estas superfícies são identificadas por células da superfície tendo somente um lado contíguo a células E. Para estas células, supõe-se que o vetor normal esteja apontando para a célula E. Neste caso, $n = (n_r, 0)$ ou $n = (0, n_z)$. A escolha é feita de acordo com o lado que é contíguo à célula E. Por exemplo, se uma célula da superfície tem somente o topo contíguo a uma célula E, como mostra a figura 5.1, então toma-se $n = (1, 0)$.

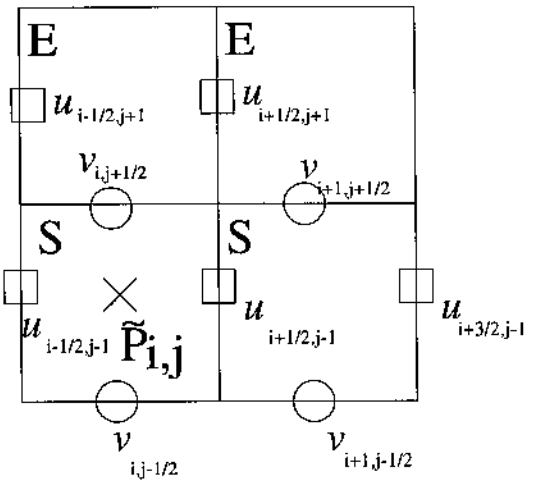


Figura 5.1: Células da superfície com o topo contíguo a uma célula E

As equações (5.15) e (5.16), nesse caso, reduzem-se a:

$$p - \frac{2}{Re}\left(\frac{\partial u}{\partial r}\right) = 0, \tag{5.17}$$

$$\left(\frac{\partial u}{\partial z} + \frac{\partial v}{\partial r}\right) = 0. \tag{5.18}$$

Observamos que quando as velocidades provisórias são calculadas pelas equações (5.11) e (5.12), são exigidos os valores da pressão $\tilde{p}_{ij}$, $u_{i+\frac{1}{2},j+1}$ e $v_{i,j+\frac{1}{2}}$ nas faces das células E, que por sua vez podem ser obtidas pelo emprego das equações (5.17) e (5.18) e a equação (5.1) de conservação de massa.

Primeiro, aplicando-se (5.1) no centro das células de superfície, obtém-se:

$$\frac{1}{r_i}\left(\frac{r_{i+\frac{1}{2}}u_{i+\frac{1}{2},j} - r_{i-\frac{1}{2}}u_{i-\frac{1}{2},j}}{\partial r}\right) + \frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{\partial z} = 0,$$

que resulta em:

$$v_{i,j+\frac{1}{2}} = v_{i,j-\frac{1}{2}} - \frac{\delta z}{\delta r}\frac{1}{r_i}\left(r_{i+\frac{1}{2}}u_{i+\frac{1}{2},j} - r_{i-\frac{1}{2}}u_{i-\frac{1}{2},j}\right).$$

Agora, ao discretizar a equação (5.18) na posição $(i+\frac{1}{2}, j+\frac{1}{2})$, obtém-se:

$$\frac{u_{i+\frac{1}{2},j+1} - u_{i+\frac{1}{2},j}}{\partial z} + \frac{v_{i+1,j+\frac{1}{2}} - v_{i,j+\frac{1}{2}}}{\partial r} = 0,$$

que pode ser escrito da forma:

$$u_{i+\frac{1}{2},j+1} = u_{i+\frac{1}{2},j} - \frac{\delta z}{\delta r}(v_{i+1,j+\frac{1}{2}} - v_{i,j+\frac{1}{2}}).$$

Uma vez que as velocidades foram calculadas, a pressão $\tilde{p}_{i,j}$ segue diretamente de (5.17) aplicado no centro da célula de superfície, ou seja:

$$\tilde{p}_{ij} = \frac{2}{Re}\left(\frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{\delta r}\right).$$

Os outros tipos de configuração de células de superfície tendo somente um lado contíguo com células E, são tratados de maneira similar.

ii) Superfícies com inclinação de 45^o.

Estas superfícies são identificadas por células de superfície tendo faces adjacentes contíguas a células E. Para estas células, supõe-se que o vetor normal faz um ângulo de 45^0 com

os eixos. Neste caso, toma-se $n = (\pm\frac{\sqrt{2}}{2}, \pm\frac{\sqrt{2}}{2})$. O sinal é escolhido de acordo com quais faces são contíguas a células E. A figura 5.2 nos mostra uma célula S contendo o lado inferior e o lado esquerdo contíguos a células E.

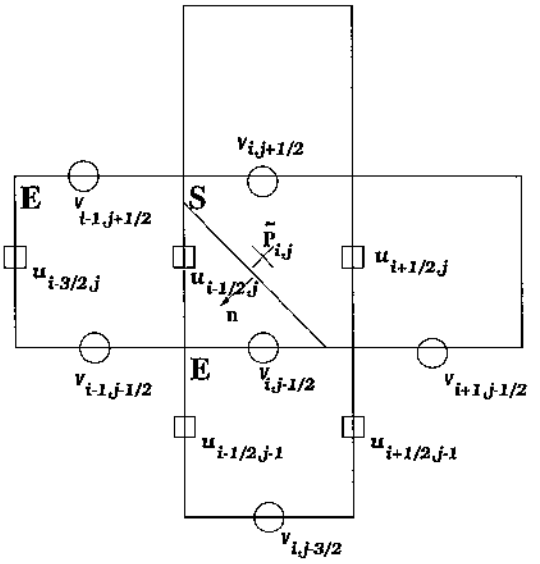


Figura 5.2: Células da superfície tendo o lado inferior e o lado esquerdo contíguos a células E

Para exemplificar, consideremos a célula de superfície mostrada na figura 5.2. Para esta célula, $n = \left(-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right)$ e, assim, (5.15) e (5.16), tornam-se:

$$p - \frac{1}{Re}\left[\frac{\partial u}{\partial r} + \frac{\partial v}{\partial z} + \left(\frac{\partial u}{\partial z} + \frac{\partial v}{\partial r}\right)\right] = 0, \tag{5.19}$$

$$\frac{\partial u}{\partial r} - \frac{\partial v}{\partial z} = 0. \tag{5.20}$$

Como pode ser visto na figura 5.2, são exigidos os valores de $u_{i+\frac{1}{2},j}$ e $v_{i,j+\frac{1}{2}}$, e estes, por sua vez, são obtidos pela equação (5.20) e pela equação de conservação de massa (5.1), aplicadas no centro das células de superfície, ou seja,

$$\frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{\delta r} + \frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{\delta z} = 0, \tag{5.21}$$

$$\frac{1}{r_i}\left(\frac{r_{i+\frac{1}{2}}u_{i+\frac{1}{2},j} - r_{i-\frac{1}{2}}u_{i-\frac{1}{2},j}}{\delta r}\right) + \frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{\delta z} = 0. \tag{5.22}$$

As equações (5.21) e (5.22) resultam em um sistema linear 2x2 para $u_{i+\frac{1}{2},j}$ e $v_{i,j+\frac{1}{2}}$, cuja solução é:

$$u_{i+\frac{1}{2},j} = \frac{r_i + r_{i-\frac{1}{2}}}{r_i + r_{i+\frac{1}{2}}} u_{i-\frac{1}{2},j}, \tag{5.23}$$

$$v_{i,j+\frac{1}{2}} = v_{i,j-\frac{1}{2}} + \frac{\delta z}{\delta r}(u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}). \tag{5.24}$$

Uma vez calculadas as velocidades nas faces das células E, a pressão segue diretamente de (5.19) aplicada no centro da célula de superfície, ou seja,

$$\begin{aligned} \tilde{p}_{i,j} \;=\; & \frac{1}{Re}\left[\frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{\delta r} + \frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{\delta z}\right. \\ & + \frac{1}{2}\left(\frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j} - u_{i+\frac{1}{2},j-1} - u_{i-\frac{1}{2},j-1}}{\delta z}\right. \\ & \left.\left. + \frac{v_{i,j+\frac{1}{2}} + v_{i,j-\frac{1}{2}} - v_{i-1,j+\frac{1}{2}} - v_{i-1,j-\frac{1}{2}}}{\delta r}\right)\right]. \end{aligned}$$

Para outras configurações de células de superfície tendo dois lados adjacentes contíguos a células E, os valores de u, v e $\tilde{p}$ são obtidos de maneira similar.

iii) Células de superfícies com dois ou três lados opostos contíguos a células E.

Estas células não fornecem informações suficientes para obtermos uma aproximação para o vetor unitário normal n. Se aparecem tais células durante os cálculos, a pressão é tomada como sendo igual a zero e as velocidades sobre as faces da célula E são ajustadas de modo que a massa seja conservada. Para evitar o aparecimento de tais células, é necessário o emprego de uma malha mais fina.

5.4 Movimento das Partículas

O movimento do fluido é obtido resolvendo-se as seguintes equações diferenciais ordinárias: $\dot{r} = u(r,z)$ e $\dot{z} = v(r,z)$, onde u e v são, respectivamente, as velocidades nas direções r e z.

5.5 Considerações Finais

Como pode ser visto no procedimento descrito na seção 5.2, a inclusão de coordenadas cilíndricas no código GENSMAC altera os passos 1 - 3, enquanto que o passo 4 permanece

inalterado, visto que as equações para a movimentação das partículas não se alteram no caso bi-dimensional para o caso eixo simétrico.

A implementação da metodologia GENSMAC em coordenadas cilíndricas segue o mesmo roteiro daquela em coordenadas cartesianas, sendo necessárias modificações somente na implementação das equações que são distintas de um caso para outro, não sofrendo, com isso, grandes alterações.

Optamos, neste capítulo, por uma descrição bastante breve e resumida sobre o assunto, em virtude de sua similaridade com a metodologia em coordenadas cartesianas.

No próximo capítulo, vamos discutir uma nova técnica para a representação da superfície livre, que pode ser empregada tanto para a metodologia GENSMAC em cordenadas cartesianas, quanto em coordenadas cilíndricas.

Capítulo 6

Uma Nova Técnica para Representação da Superfície Livre

Ao analisar o tempo de processamento computacional do código GENSMAC, Perez [28] verificou que a seção de movimentação de partículas consome mais tempo que as demais seções, podendo, inclusive consumir mais da metade do tempo total de processamento, dependendo, é claro, da quantidade de partículas envolvida no problema. Somado a isso, para malhas relativamente finas, a quantidade de partículas necessária para a representação do fluido é muito grande, exigindo um grande espaço de memória para armazená-las. Esses dois fatos representam, para problemas muito grandes e/ou malhas muito finas, uma séria restrição quanto ao uso do código GENSMAC.

Como já dissemos na seção 3.1.4 do capítulo 3, as partículas virtuais utilizadas na representação do fluido tem um movimento rígido, ou seja, cada uma delas é movimentada utilizando-se as quatro velocidades que estão mais próximas na malha de discretização. Assim, existem regiões nessa malha nas quais as tais partículas são movimentadas utilizando-se as mesmas quatro velocidades. Desta forma, bastaria mover apenas algumas partículas "chaves" para se ter toda a movimentação do fluido. Motivados por essa simples idéia, Tomé e co-autores [35], desenvolveram, recentemente, uma nova versão do código GENSMAC, na qual é implementada uma nova técnica para representar a superfície livre. Chamaremos essa nova versão de sem partículas. Ela permite o armazenamento e a movimentação apenas das partículas que estão sobre a superfície livre, contra a versão anterior, a qual chamaremos com partículas, onde as partículas são usadas para representar todo corpo do fluido, incluindo a superfície livre. Além do mais, nesta nova versão, as partículas são armazenadas dinamicamente, fato que não ocorre na versão anterior.

6.1 A Representação da Superfície Livre

Na implementação sequencial do código sem partículas cada corpo de fluido é representado por uma lista orientada do tipo fechada, na qual são armazenadas as posições das partículas que estão sobre a superfície livre, proporcionando, dessa forma, a localização do fluido e também a localização da superfície livre. A orientação dessa lista é crucial para se determinar a posição do fluido. A figura 6.1 a seguir ilustra um exemplo de tais listas.

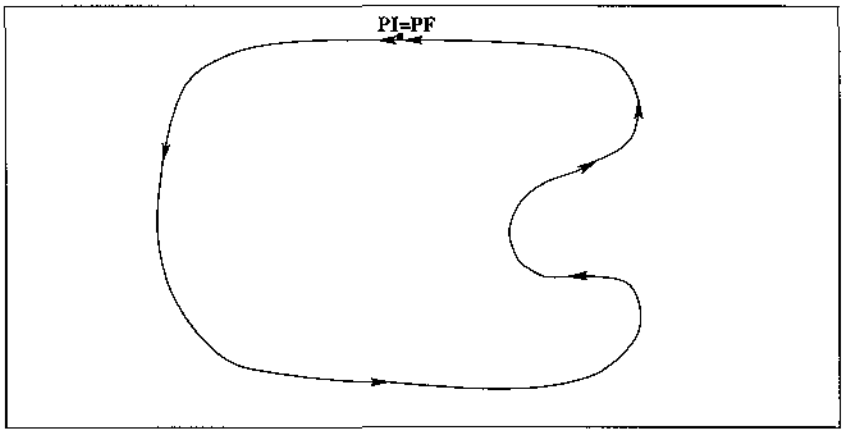


Figura 6.1: Representação da superfície livre por uma lista oriendada do tipo fechada

Como podemos observar na figura 6.1, o ponto inicial PI coincide com o ponto final PF. As setas indicam a orientação da lista, que por sua vez determina a posição do fluido. O fluido está sempre à esquerda seguindo a orientação das setas.

Em uma lista cada um de seus nós armazena informações referentes à posição da partícula, ao tipo de célula e ao tipo de movimento que pode ser realizado. Existem três tipos de nós: "INFLOW", "SURFACE" e "OUTFLOW". A inserção de fluido no interior do domínio de solução se dá através de nós do tipo "INFLOW", a eliminação da porção do fluido que ultrapassa os contornos de saida é realizada através dos nós do tipo "OUTFLOW" e os nós do tipo "SURFACE" representam o contorno do corpo do fluido dentro do domínio de solução, ou seja, representam a superfície livre.

Suponha que, em um determinado problema, temos a representação dada a seguir de um contorno de entrada:

B	B	I	I	I	I	I	I	B	B

No código com partículas, a inserção de fluido é realizada inserindo-se partículas virtuais no interior do domínio de solução, através de todas as células marcadas com I. Com o decorrer

do tempo novas partículas vão sendo inseridas e movimentadas de acordo com as velocidades de cada célula, representando dessa maneira, o corpo do fluido no processo de escoamento.

No código sem partículas, é gerada uma lista orientada com o primeiro ponto na primeira célula I (mais à esquerda) e o último ponto na última célula I (mais à direita). A partir daí são inseridos novos nós, de acordo com as necessidades de entrada de fluido, mas somente após o primeiro e o último pontos desta lista, que estão fixos. Esses pontos servem de referência para a inserção de novos nós e não podem jamais ser alterados. Os demais pontos são movimentados de acordo com as velocidades de cada célula.

O processo de eliminação de partículas, através de um contorno de saida é realizado de maneira semelhante. Para exemplificar, vamos considerar o seguinte contorno de saida de um determinado problema:

B	B	O	O	O	O	O	O	B	B

No código com partículas, toda vez que uma determinada partícula atinge, através do processo de movimentação, uma célula O, ela é eliminada. De maneira semelhante, no código sem partículas, toda vez que um nó de uma determinada lista que representa um corpo de fluido atinge uma célula O do contorno de saida, esse nó é eliminado da lista.

Há, também, a necessidade de eliminação ou de inserção de novos nós em uma lista, no caso do código sem partículas, em outras situações. Quando existem regiões da lista com excesso de nós acumulados, alguns deles são eliminados, evitando-se, desta forma, um acúmulo desnecessário de nós em uma mesma lista. Em contra-partida, quando existem regiões da lista em que os nós estão distantes uns dos outros, são inseridos novos nós intermediários aos mais distantes, de modo a dar o equilíbrio necessário, evitando, desta forma, desestabilizar o processo.

Cabe-nos ainda ressaltar que a implementação computacional dessa nova versão do código GENSMAC, exigiu um grau de complexidade muito maior que para a versão anterior. Mais complexa ainda é a implementação paralela dessa nova técnica de representação da superfície livre, pois a estrutura de dados apresentada anteriormente não pode ser transportada facilmente para a implementação paralela, uma vez que a decomposição de domínio pode dividir cada corpo de fluido em partes menores que serão atribuidas a diferentes processadores. Cada uma dessas partes, possivelmente terá continuação em um processador vizinho. Assim, um corpo de fluido não pode mais ser representado por uma lista do tipo fechada. Decidimos, então, representar cada corpo de fluido envolvido no problema por uma lista orientada do tipo aberta e decomposta em várias componentes. Essas componentes podem estar distribuidas pelos vários subdomínios considerados na decomposição. A figura 6.2 mostra um exemplo de tal disposição.

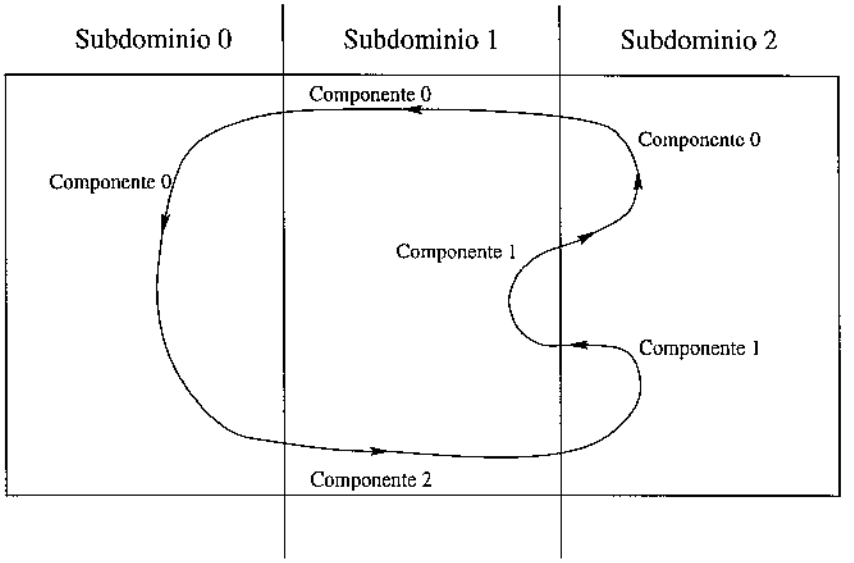


Figura 6.2: Lista dividida em componentes

A figura 6.2 ilustra a situação de um escoamento com um único corpo de fluido e, portanto, uma única lista orientada, distribuido por três subdomínios. Note que o processador 0 armazena informações de uma única componente, chamada componente 0, que inicia na componente 0 do processador 1 e termina na componente 2, também do processador 1. O processador 1 armazena informações de três componentes: da componente 0, que inicia na componente 0 do processador 2 e termina na componente 0 do processador 0; da componente 1, que inicia na componente 1 e termina na componente 0, ambas do processador 2 e, da componente 2, que inicia na componente 0 do processador 0 e termina na componente 1 do processador 2.

Com a evolução do tempo, a geometria do fluido pode variar consideravelmente. Assim, o início, o final, algumas partes centrais ou mesmo toda uma componente pode transferir-se do domínio de um processador para o domínio de outro. Quando isso ocorre, são trocadas informações entre os processadores envolvidos, de modo que a nova situação seja atualizada em cada processador e um novo ciclo de cálculo possa ser iniciado. Por exemplo, quando o início de uma componente migra para um processador vizinho, as informações desse trecho de componente são acrescentadas ao final de uma outra, que corresponde a sua continuação natural no processador de destino. Quando uma parte central de uma componente tiver que ser transferida para um outro processador, então é gerada uma nova componente no processador de destino, enquanto que a componente "doadora", no processador de origem, é desmembrada em outras duas. Pode acontecer, também, a situação em que várias componentes de um mesmo processador venham a se agrupar, formando uma única componente. As figuras mostradas a seguir ilustram algumas situações que ocorrem com frequência.

Para exemplificar, considere que em um determinado tempo t_1, temos a situação mostrada na figura 6.2, onde as componentes de cada subdomínio se encontram todas atualizadas e,

em um tempo $t_2 > t_1$, ocorra a situação exemplificada na figura 6.3, onde após o processo de movimentação de partículas, o final da componente 0 e o início da componente 2, ambas pertencentes ao subdomínio 1, invadem o subdomínio 0 e, também, o início e o final das componentes 0 e 1, ambas pertencentes ao subdomínio 2, invadem o subdomínio 1.

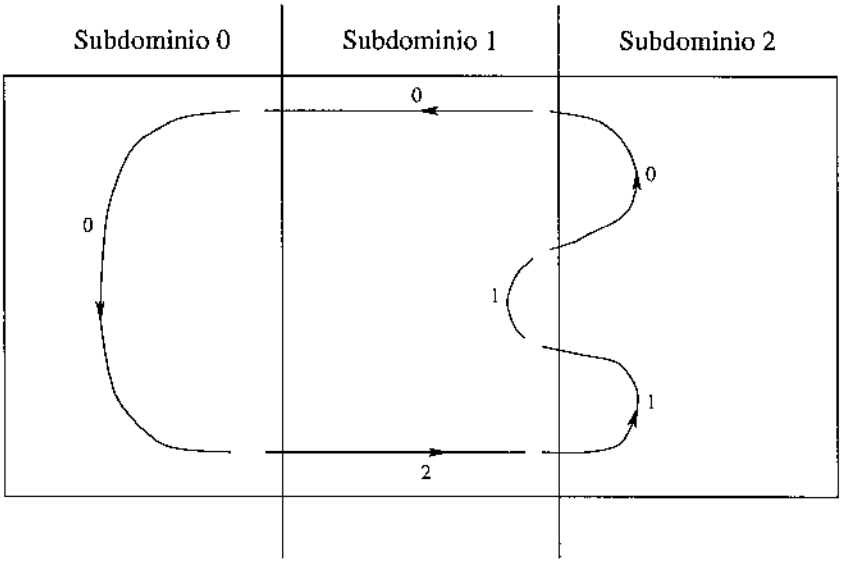


Figura 6.3: Movimentação de partículas no tempo t_2

Os espaços em branco mostrados na figura 6.3 entre uma componente e outra e, também, nas figuras subsequentes, são deixados propositalmente para que se possa distinguir as componentes de cada subdomínio. Quando uma componente que está em um determinado subdomínio tem, após o processo de movimentação das partículas, uma parte invadindo um outro subdomínio, esse trecho de componente deve ser transferido para o subdomínio apropriado. Para o exemplo considerado, a atualização de cada subdomínio é realizada da seguinte maneira: o final da componente 0 do processador 1 deve ser transferido para o início da componente 0 do processador 0; o início da componente 2 do processador 1 deve ser transferido para o final da componente 0 do processador 0. O final e o início da componente 0 do processador 2 devem ser transferidos, respectivamente, para o início da componente 0 e para o final da componente 1, ambas pertencentes ao processador 1. E, finalmente, o início e o final da componente 1 do processador 2 devem ser transferidos, respectivamente, para o final da componente 2 e início da componente 1, que estão no processador 1.

Considere, agora, que no tempo $t_3 > t_2$, ocorra uma situação semelhante à ilustrada na figura 6.4, na qual, além das transferências de partes do início e partes do final das componentes, também uma parte central de cada uma das componentes 0 e 1 do processador 2 devem ser transferidas para o processador 1.

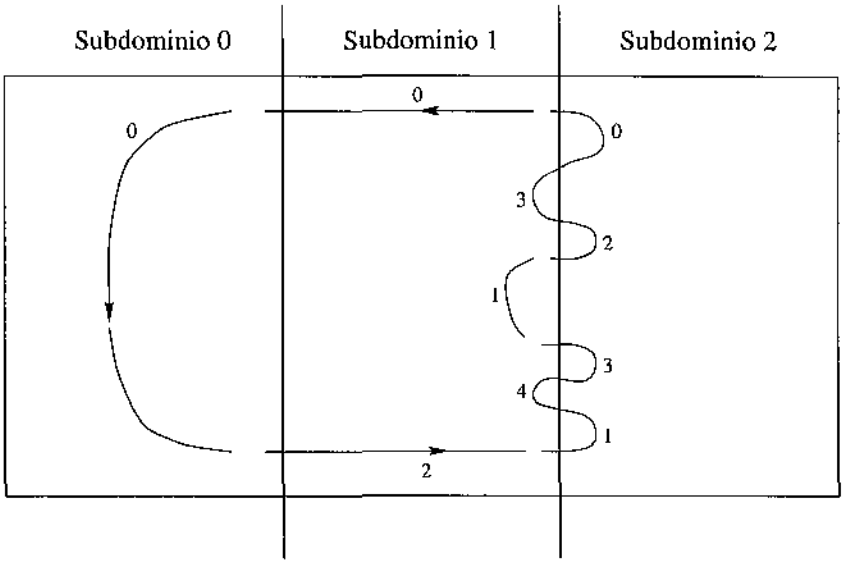


Figura 6.4: Movimentação de partículas no tempo t_3

Após realizadas as atualizações necessárias em cada subdomínio, o resultado final é o seguinte:

o processador 0 permanece com uma única componente, ou seja, a componente 0;

o processador 1 passa a ter as componentes 0, 1, 2, 3 e 4;

o processador 2 fica com as componentes 0, 1, 2 e 3.

Finalmente, considere que em um tempo $t_4 > t_3$, as componentes 0, 1, 2 e 3, todas pertencentes ao processador 2, devem ser transferidas para o processador 1, como ilustram as figuras a seguir:

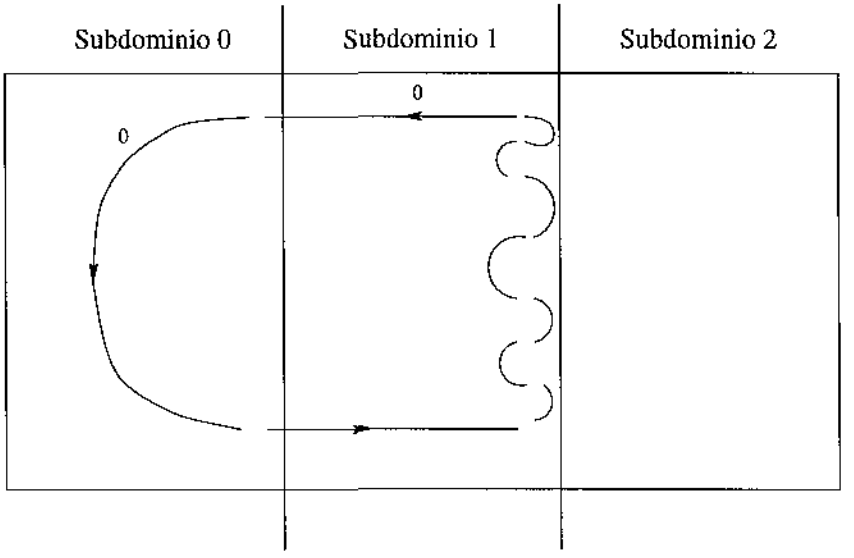


Figura 6.5: Movimentação de partículas no tempo t_3

Neste estágio, as componentes 0, 1, 2 e 3 do processador 2 devem ser transferidas para o processador 1, que por sua vez armazena informações de 5 componentes. A junção de todas essas componentes resulta em uma única, como mostra figura 6.6:

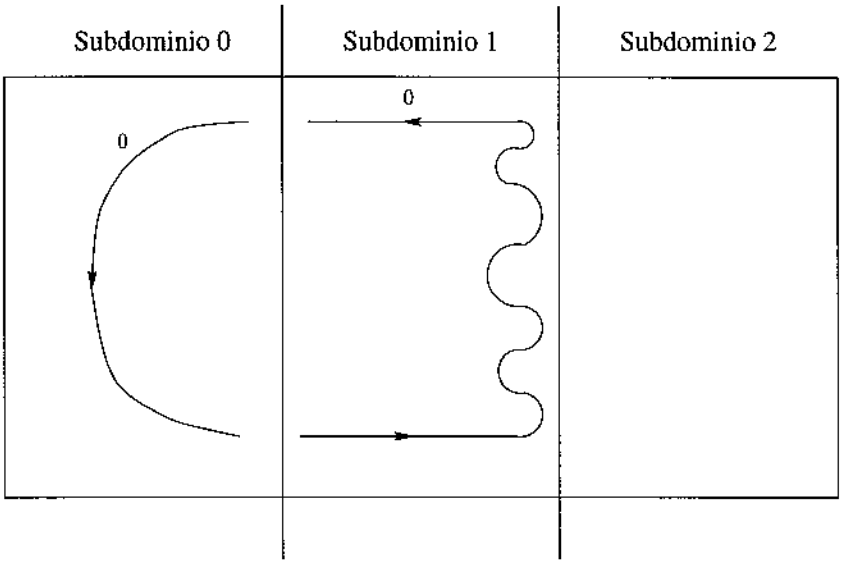


Figura 6.6: Agrupamento de componentes no tempo t_3

Nesse estágio do processamento, o processador 0 permanece com uma única componente, ou seja, a componente 0. O processador 1 passa a conter apenas a componente 0, enquanto que o processador 2 não possui mais nenhuma componente.

Essas situações descritas anteriormente podem ocorrer com frequência na simulação de um processo de escoamento, utilizando esta técnica para a representação da superfície livre. Podem ocorrer, também, situações em que vários trechos centrais de uma mesma componente devam ser transferidos todos ao mesmo tempo de um processador para outro.

A seguir mostramos outros aspectos do código GENSMAC que tiveram que ser alterados com essa nova técnica para representação da superfície livre.

Como vimos anteriormente no capítulo 4, a implementação do código GENSMAC permite, também, a solução de problemas que já contenham fluido no instante inicial de tempo. Esse é o caso, também, dessa nova versão. O fluido inserido no interior do domínio de solução no instante inicial de tempo pode ter a forma retangular ou circular, como já descrito previamente.

No código paralelo, com a decomposição de domínio, esse corpo de fluido pode estar distribuido por mais de um subdomínio. As figuras 6.7 e 6.8 ilustram a representação de um mesmo corpo dividido por dois subdomínios. Neste caso, as listas que representam o corpo de fluido são divididas de modo que cada subdomínio contenha a sua parte. Essa tarefa é realizada antes do ciclo de cálculo e cada processador se encarrega de detectar qual é a sua parte, passando esta a ser armazenada como uma componente que compõe a lista, conforme

a estrutura de dados descrita anteriormente. As figuras 6.7 e 6.8 ilustram essa situação. A figura 6.7 mostra o caso de um corpo de fluido na forma retangular, no instante inicial de tempo, distribuido por dois subdomínios.

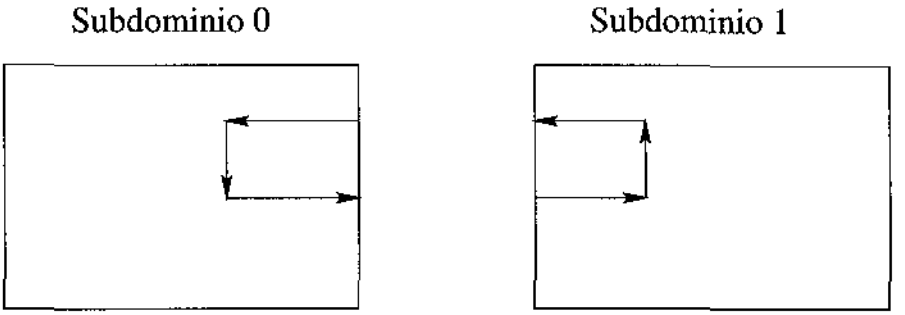


Figura 6.7: Corpo de fluido retangular decomposto em subdomínios

A figura 6.8 mostra um corpo de fluido na forma circular dividido, também, entre dois subdomínios no instante inicial de tempo.

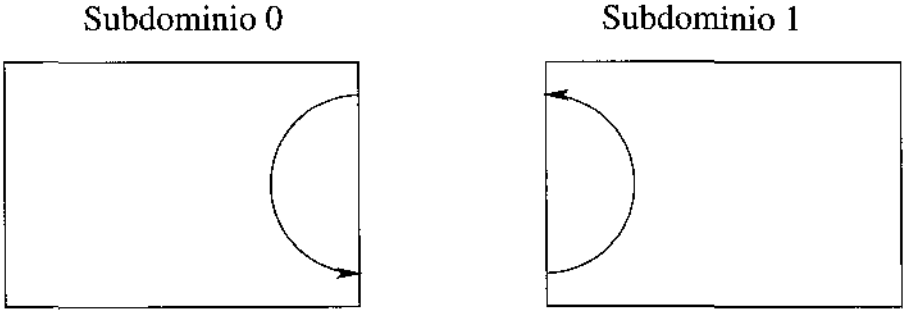


Figura 6.8: Corpo de fluido circular decomposto em subdomínios

A seção de marcação das células e a seção de movimentação de partículas, também sofreram drásticas modificações da implementação do código com partículas para a implementação do código sem partículas.

Na seção de marcação das células, do código com partículas, são utilizadas todas as partículas virtuais para a atualização das células que compõem o domínio de solução, enquanto que na versão sem partículas, são utilizadas apenas as listas orientadas que definem a superfície livre, diminuindo significativamente a quantidade de trabalho que deve ser realizada nesse estágio de processamento.

A seção de movimentação de partículas é o estágio de processamento mais beneficiado na redução da quantidade de trabalho e, consequentemente, na redução do tempo de processamento, com a utilização dessa nova técnica de representação da superfície livre. Na implementação do código com partículas, todas as partículas que representam o corpo do fluido participam desse processo de movimentação, enquanto que na implementação do código

sem partículas, são movimentadas somente as partículas que representam a superfície livre. Além do mais, na versão do código com partículas, a quantidade de partículas virtuais que podem ser armazenadas é limitada, enquanto que na versão sem partículas, não há uma limitação pré-estabelecida, pois é utilizada uma estrutura de dados com alocação dinâmica de posição de memória, contra a versão anterior, na qual essas informações são armazenadas utilizando a estrutura de matriz para a representação desses dados.

O primeiro passo realizado para a paralelização dessa estrutura de listas ordenadas foi densenvolver um código sequencial utilizando a decomposição de domínios e simulando vários processadores em um único processo. Nesse código, desenvolvemos uma estrutura na qual tínhamos o controle total das componentes que formam uma dessas listas.

Para cada componente eram armazenadas as seguintes informações:

- identificador da componente;
- processador a que pertence;
- processador inicial, ou seja, processador vizinho ao ponto inicial;
- processador final, ou seja, processador vizinho ao último ponto da componente;
- componente inicial, ou seja, rótulo da componente no processador vizinho da qual a componente se origina;
- componente final, ou seja, rótulo da componente no processador vizinho, na qual a componente se encerra.

Para exemplificar, considere a situação ilustrada na figura 6.9:

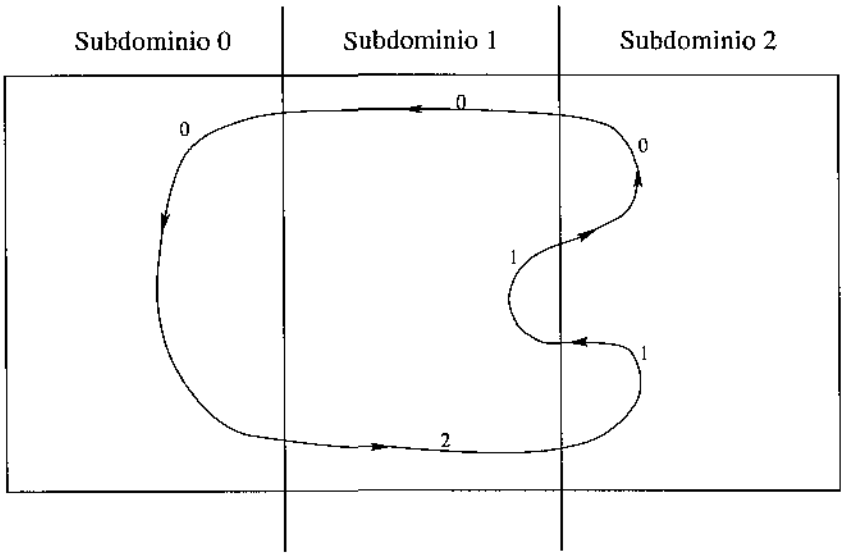


Figura 6.9: Identificação de componentes

Nesta situação, são armazenadas as seguintes informações para cada componente:

1) componente 0 do processador 0:
   identificador: 0
   processador inicial: 1
   processador final: 1
   componente inicial: 0
   componente final: 2

2) componente 0 do processador 1:
   identificador: 0
   processador inicial: 2
   processador final: 0
   componente inicial: 0
   componente final: 0

3) componente 1 do processador 1:
   identificador: 1
   processador inicial: 2
   processador final: 2
   componente inicial: 1
   componente final: 0

4) componente 2 do processador 1:
   identificador: 2
   processador inicial: 0
   processador final: 2
   componente inicial: 0
   componente final: 1

5) componente 0 do processador 2:
   identificador: 0
   processador inicial: 1
   processador final: 1
   componente inicial: 1
   componente final: 0

6) componente 1 do processador 2:
   identificador: 1
   processador inicial: 1
   processador final: 1

componente inicial: 2
componente final: 1

Com a evolução do tempo, essa situação pode se alterar completamente. Considere, por exemplo, que após um determinado tempo uma parte central da componente 2 do processador 1 deva ser transferida para o processador 2, o que, obviamente, provoca uma renumeração dos identificadores das componentes tanto no processador 1 quanto no processador 2. Algo semelhante ocorre, também, com a componente 0 do processador 1. A nova forma da componente 2 do processador 1 é mostrada na figura 6.10, antes de fazer as alterações necessárias:

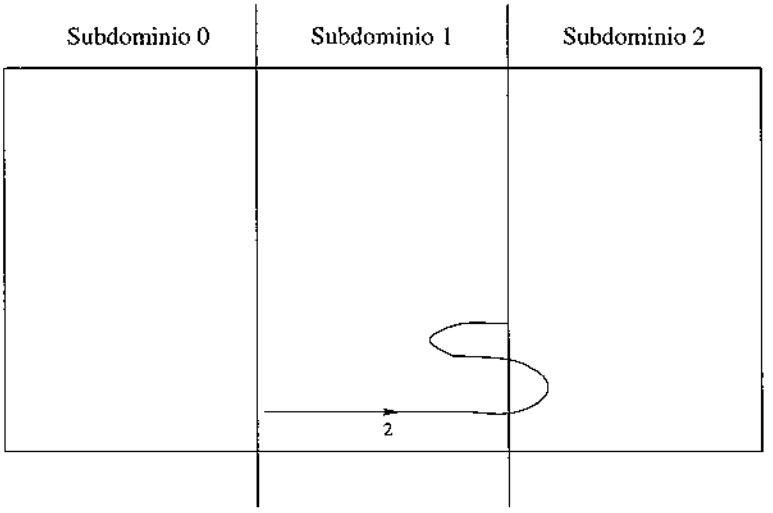


Figura 6.10: Movimentação da componente 2

Após realizadas as atualizações necessárias, temos a seguinte situação:

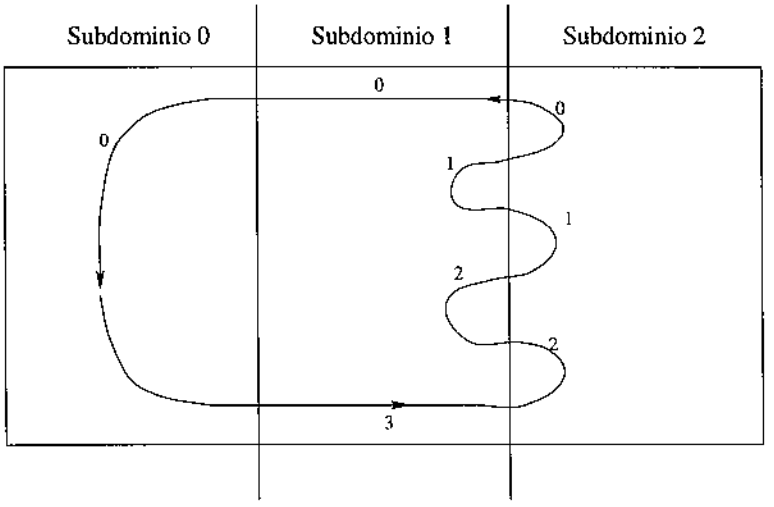


Figura 6.11: Movimentação de partículas

Observe que a componente 2 do processador 1 se dividiu em outras duas, respectivamente, componentes 2 e 3. Na implementação que realizamos anteriormente, é sempre o final da componente que mantém o antigo rótulo. Com isso, alteram-se as informações das componentes 1 do processador 2, componente 2 do processador 1 e, até mesmo, da componente 0 do processador 0, pois antes essa componente terminava na componente 2 do processador 1 e, agora, termina na componente 3 do processador 1. Observe, aqui, que a componente 0 do processador 0 não deveria sofrer qualquer alteração em virtude de não estar diretamente envolvida com as trocas efetuadas entre as demais componentes.

Com essa estrutura tínhamos um controle completo de cada componente, mas, por outro lado, como a dinâmica do fluido pode estar constantemente se alterando, paralelizar essa estrutura não seria uma tarefa muito fácil e, além do mais, geraria um excesso de trocas de informações e sincronismos entre os processadores, sem se levar em consideração que tais fatos poderiam estar ocorrendo simultaneamente em processadores distantes e envolvendo processadores vizinhos comuns que não estão envolvidos diretamente com essas trocas e serem, nesse caso, obrigados a sofrerem alterações. No caso de mantermos o início da componente com o rótulo antigo ou mesmo de atribuirmos um rótulo totalmente diferente para cada componente, os problemas que surgiriam seriam os mesmos.

Desta forma, no programa paralelo atual, decidimos armazenar apenas a informação de identificação da componente e esquecer todas as outras informações de controle. Para a implementação computacional dessa técnica, adotamos a seguinte estratégia de programação: quando estamos transferindo o início de uma componente para um outro processador, para identificar em qual componente deve ser juntado esse trecho, procuramos sempre pela componente cuja distância entre o seu último elemento e o primeiro da subcomponente que está sendo transferida é a menor possível. Quando estamos transferindo o final de uma componente, o procedimento é exatamente o mesmo, considerando que se deve procurar a componente, no outro processador, que tenha a distância mínima com relação ao primeiro elemento. No caso de estarmos transferindo partes internas de uma componente, são geradas novas componentes nos processadores de destino. Quando é transferida uma componente inteira de um processador para outro, deve ser encontrado o final de uma componente para ser ligado ao início da componente considerada e o início de uma outra componente para ser ligada ao seu final, sempre pela distância mínima. Caso não sejam encontradas tais componentes, é gerada uma nova componente no processador de destino.

As trocas de informações entre os processadores é efetuada, conforme a necessidade, após a realização de todo o processo de movimentação das partículas que representam a superfície livre.

A implementação dessa técnica das distâncias mínimas, que apesar de não ter um controle total da situação, se mostrou bastante eficiente, pois diminui a quantidade de troca de

informações entre os processadores, eliminando muitos pontos de sincronismos e, também, não apresentou problemas para os exemplos testados.

6.2 Considerações Finais

Reservamos este capítulo para a discussão de uma nova técnica para a representação da superfície livre e, também, para discutir os aspectos das implementações de ambas as versões: sequencial e paralela. Cabe-nos aqui ressaltar que embora a implementação sequencial não tenha sido de nossa autoria, em nenhum outro trabalho sua implementação tinha sido discutida com tantos detalhes. A implementação da estrutura de dados para essa nova técnica para representação da superfície livre foi realizada em linguagem C.

Lembramos, ainda, que com essa nova maneira para se representar a superfície livre, não há mais a necessidade de se identificar a cadeia de partículas que representa a superfície livre, como no caso da versão anterior. No final do capítulo 3 deste trabalho, foi relatada uma técnica, proposta por Tomé, para se determinar a superfície livre na versão anterior do código GENSMAC.

Na implementação paralela dessa técnica, os contornos de entrada e de saida podem, também, estar divididos por vários subdomínios. Essa implementação foi muito mais trabalhosa que no caso da implementação paralela da versão do código em que são consideradas todas as partículas.

Essa estrutura de dados foi implementada para o código GENSMAC em coordenadas cartesianas e, também, em coordenadas cilíndricas.

No próximo capítulo, mostramos alguns problemas testes que utilizamos para testar o código paralelo durante o seu desenvolvimento e, também, problemas que foram utilizados na análise da performance.

Capítulo 7

Problemas Testes

O código sequencial foi extensivamente testado ao longo do tempo, incluindo a validação através de testes experimentais (Ver Tomé [33] e Tomé e McKee [36]). O código paralelo foi testado através de vários problemas testes. Os testes foram realizados utilizando-se de 2 a 8 processadores. A seguir mostramos alguns deles. Muitos dos problemas descritos aqui aparecem com frequência na literatura especializada e são, por isso, bastante estudados, fornecendo muitos dados para comparação.

7.1 Descrição dos Problemas e Visualização dos Resultados

Vamos considerar, aqui, oito problemas testes. A descrição de cada problema e, também, a visualização de alguns resultados são mostrados a seguir.

Problema 1: "Sloshing"

Vamos considerar, neste exemplo, um bloco de fluido na forma retangular em queda livre sob a influência da gravidade, em uma região circular, como mostram as figuras a seguir. São impostas condições de contorno do tipo de não escorregamento sobre a parede com contorno circular e do tipo de escorregamento livre sobre as outras paredes (linhas retas). Os parâmetros utilizados neste exemplo são: $Re = 6.666$ e $Fr = 1.0$.

As figuras mostradas a seguir foram geradas com os resultados fornecidos pela utilização do código com partículas.

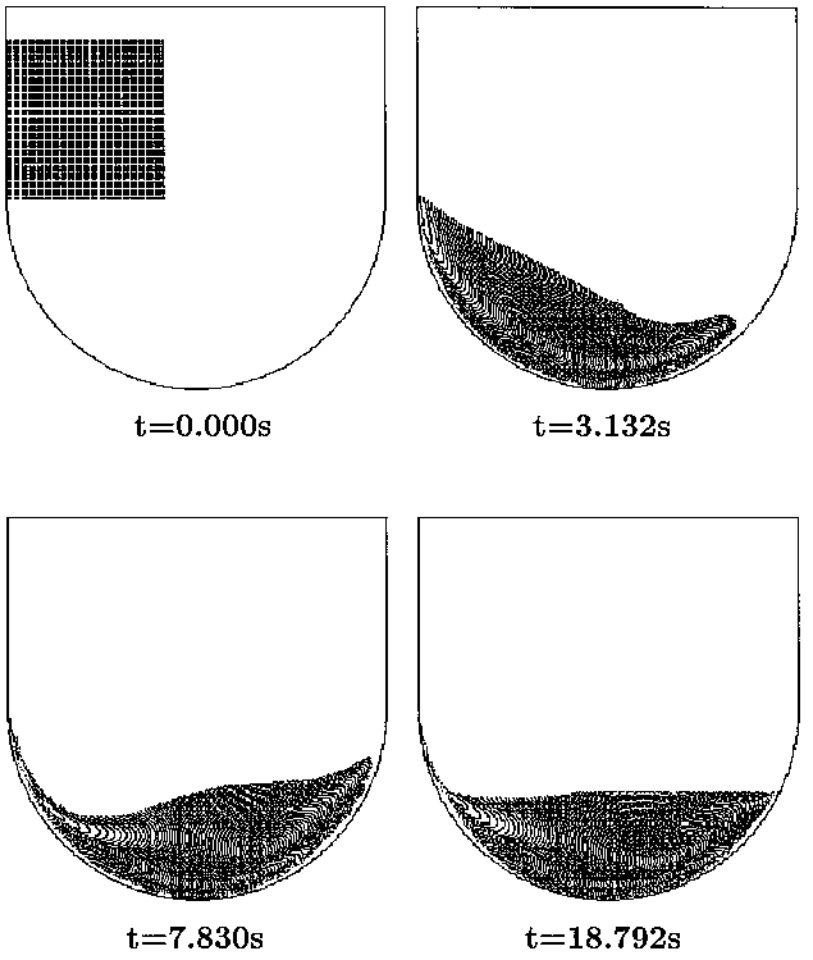
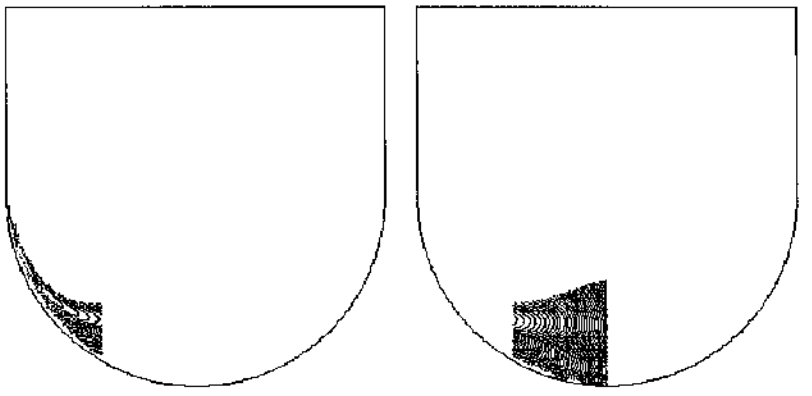


Figura 7.1: Partículas plotadas para o problema 1 em diferentes tempos

As figuras seguintes mostram como ficam as configurações dos fluidos em cada um dos processadores, considerando-se 4 processadores. Essas figuras refletem o tempo $t = 7.83s$.



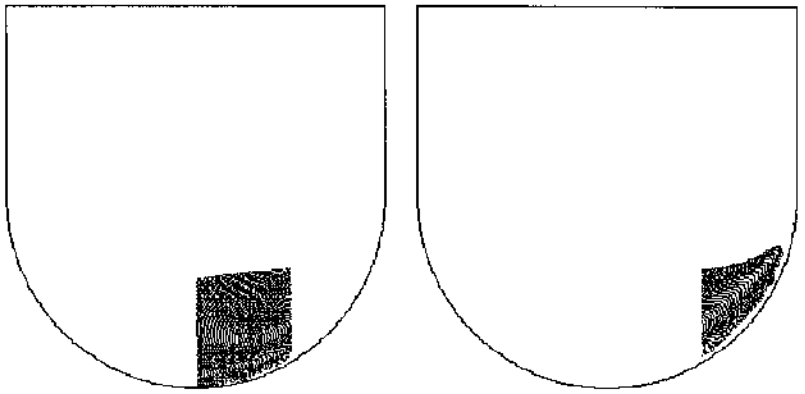


Figura 7.2: Partículas plotadas para o problema 1 no tempo $t = 7.83s$

As figuras a seguir foram geradas com os resultados fornecidos pela utilização do código sem partículas.

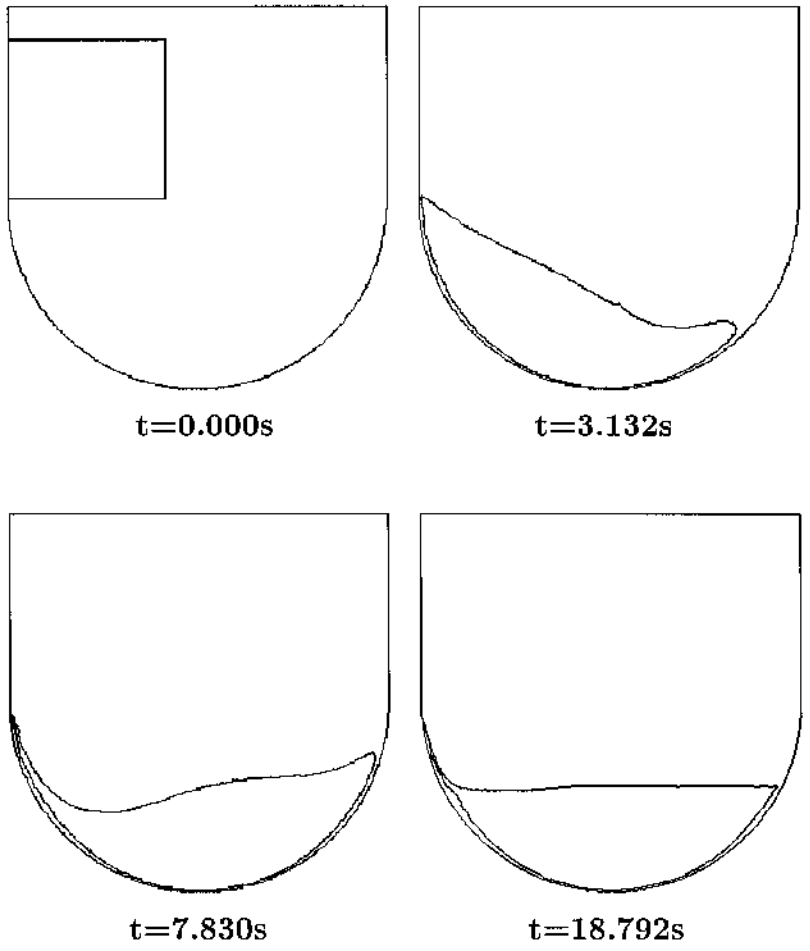


Figura 7.3: Visualização do problema 1 em diferentes tempos

Problema 2: "Escoamento de um fluido sobre um degrau circular"

Este exemplo simula o escoamento de um fluido através de um recipiente com contorno irregular e na presença de contornos de entrada e de saída. O fluido entra no sistema no tempo t_0, cai sob a influência da gravidade sobre o degrau com contorno circular e é escoado para fora através de um contorno de saída contínuo. As condições sobre as paredes desse recipiente e sobre o degrau são condições de contorno de não escorregamento. Os pontos plotados mostram a configuração das partículas para diferentes tempos. Os parâmetros utilizados com este exemplo são: $(1/Re = 0.1)$ e $(1/Fr^2 = 1.0)$.

As figuras apresentadas abaixo foram geradas com os resultados fornecidos pela utilização do código com partículas.

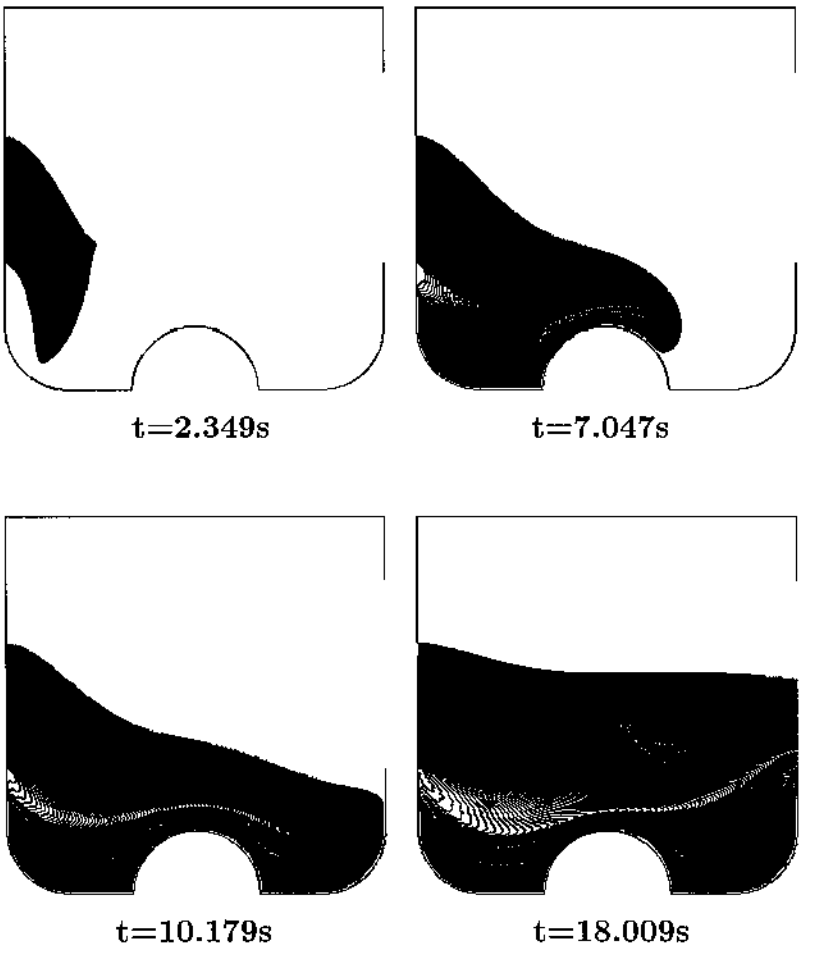


Figura 7.4: Partículas plotadas para o problema 2 em diferentes tempos

As figuras a seguir foram geradas com os resultados fornecidos pela utilização do código

sem partículas.

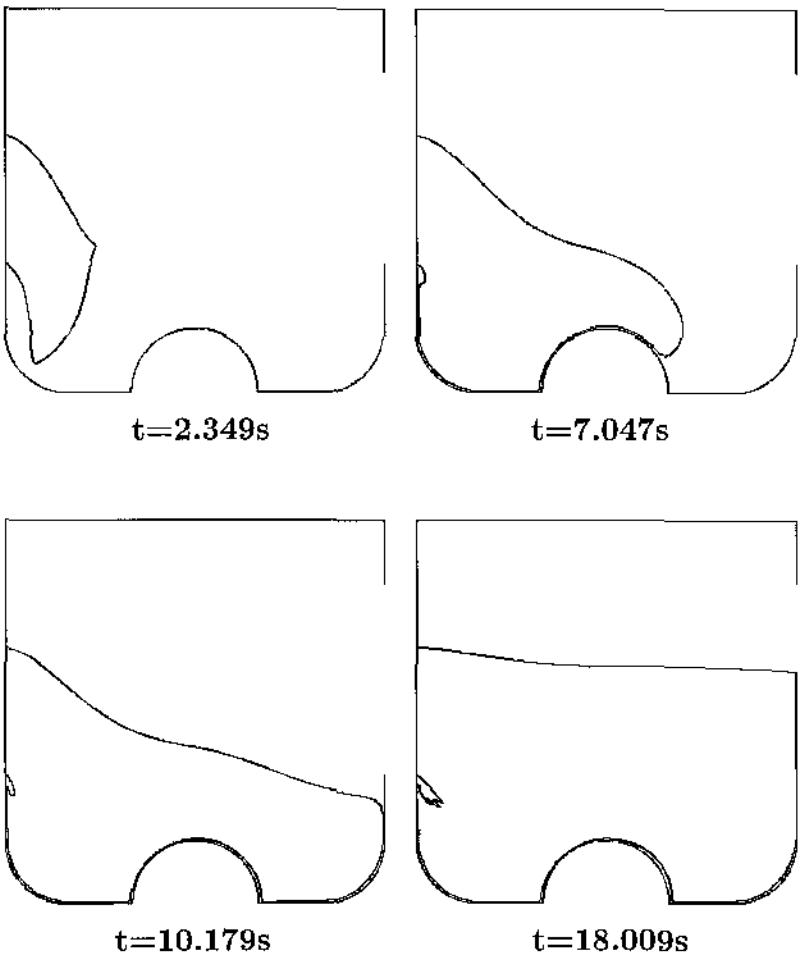


Figura 7.5: Visualização dos resultados do problema 2 em diferentes tempos

Problema 3: "Escoamento de um fluido sobre um degrau retangular"

Este exemplo simula o escoamento de um fluido caindo em um degrau retangular na presença de um contorno de entrada na parede lateral esquerda e de um contorno de saida na parade lateral direita. O fluido entra no sistema em t_0, cai sob a influência da gravidade sobre o degrau e é escoado para fora através de um contorno de saida contínuo. As condições sobre as paredes do recipiente e sobre o degrau são condições de contorno de não escorregamento. Os pontos plotados mostram a configuração das partículas para diferentes tempos. Os parâmetros utilizados para este exemplo são: $Re = 10.0$ e $Fr = 1.0$.

As figuras a seguir foram geradas com os resultados fornecidos pela utilização do código com partículas.

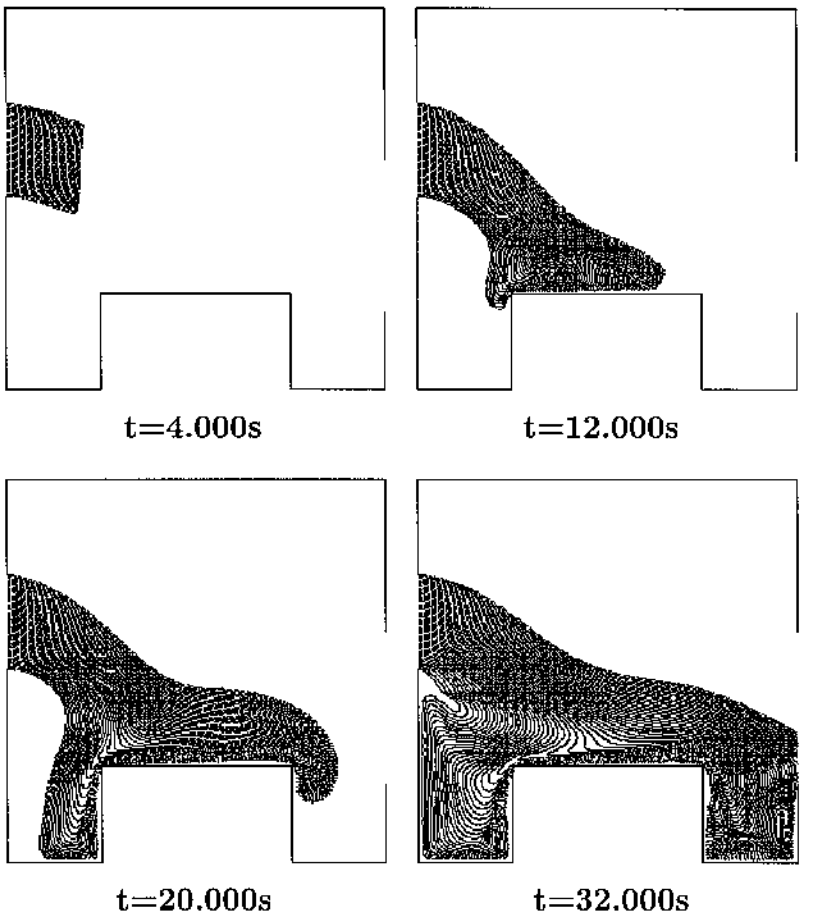


Figura 7.6: Partículas plotadas para o problema 3 em diferentes tempos

Problema 4: "Tubo circular"

Neste exemplo, vamos considerar um tubo circular, no qual um determinado fluido é injetado pela parte superior, como mostra a figura 7.7a. O escoamento, neste caso, é eixo-simétrico e o domínio de solução tem a forma mostrada na figura 7.7b

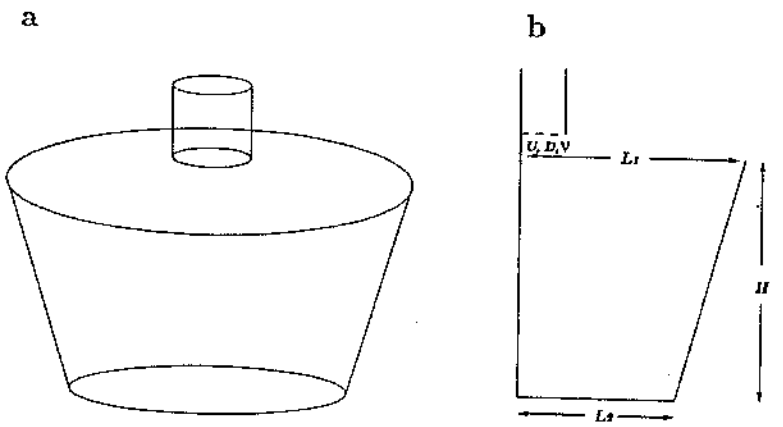


Figura 7.7: Definição do problema em duas dimensões.

As figuras a seguir, foram geradas com os resultados fornecidos pela utilização do código sem partículas e são mostradas abertas, ou seja, são mostradas como se as configurações dos resultados em 3D tivessem sofridos um corte na horizontal. A seguir, mostramos a configuração na região desse corte.

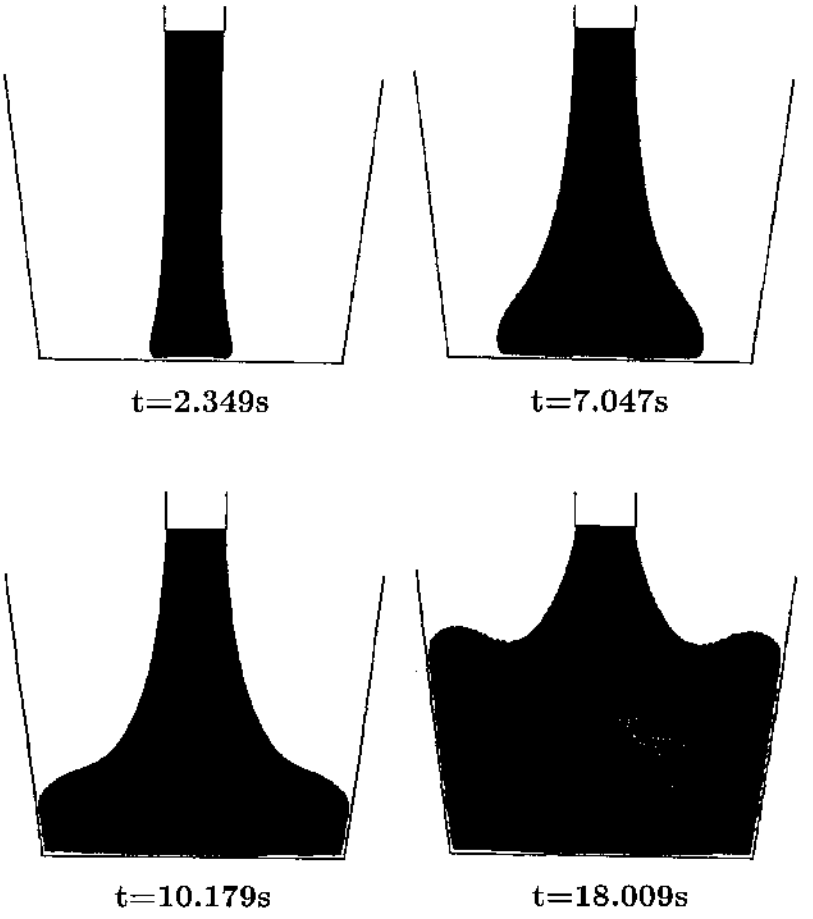


Figura 7.8: Visualização do problema 4 em diferentes tempos.

Problema 5: "Escoamento eixo-simétrico"

Neste problema consideramos, também, um escoamento eixo simétrico. Consideramos uma caixa quadrada de 2,5cm x 5cm contendo um fluido. Acima do recipiente foi colocado um bocal com um contorno de entrada de 5 mm de diâmetro do qual um jato colorido cai verticalmente na caixa com uma velocidade pré-estabelecida de $1m/s$. A viscosidade considerada é $\nu = 0.005$ e, assim os parâmetros utilizados são $(1/Re = 0.1)$ e $(1/Fr^2 = 20.38)$. Neste exemplo, são mostrados os resultados considerando a versão do código na qual as partículas virtuais são utilizadas apenas na representação da superfície livre.

As figuras que apresentamos a seguir foram geradas com os resultados fornecidos pela utilização do código sem partículas. As figuras, aqui, também são mostradas abertas.

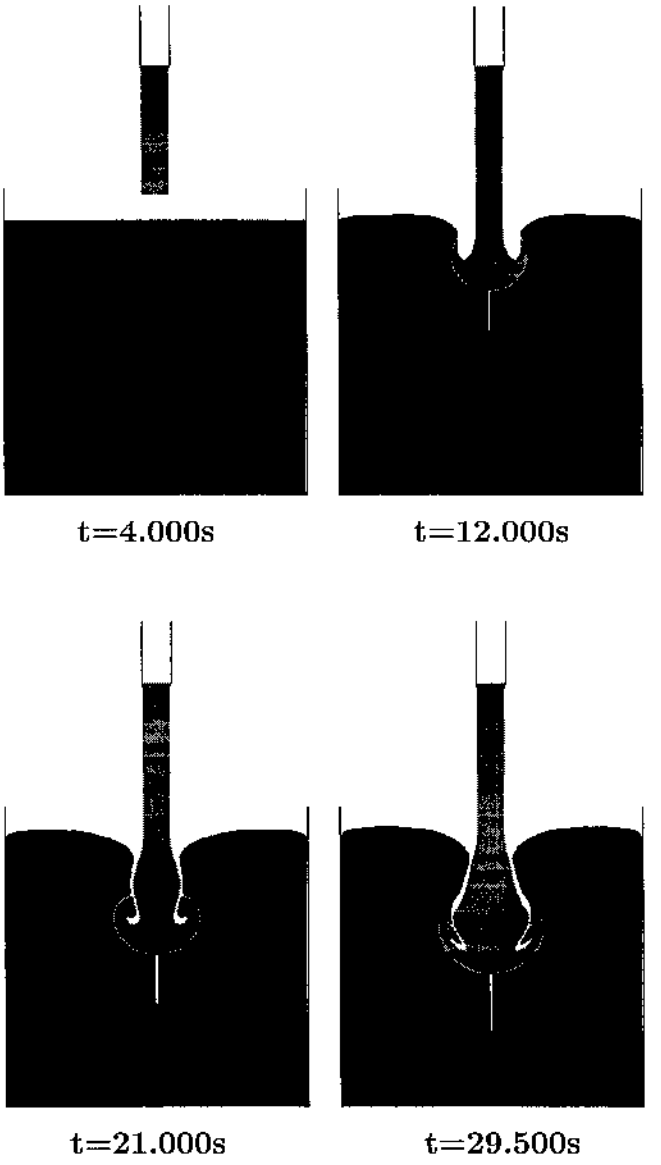
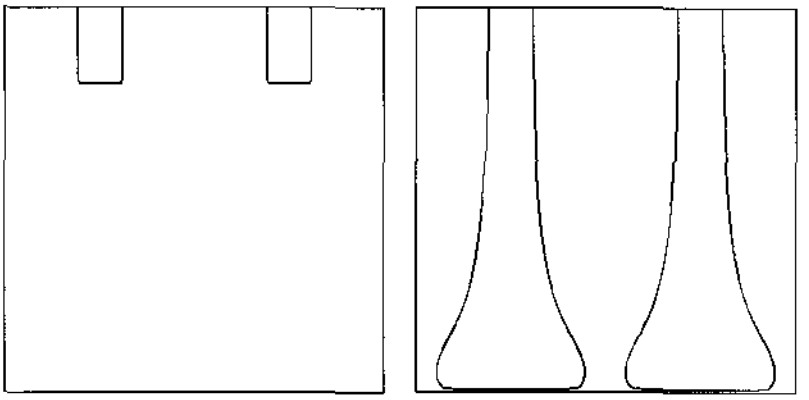


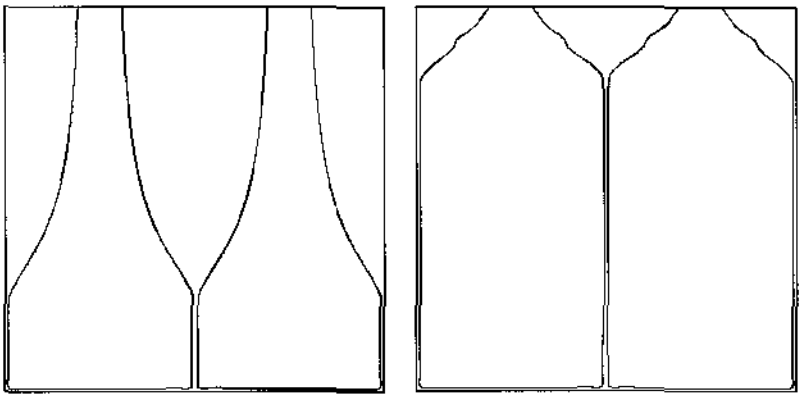
Figura 7.9: Visualização dos resultados para o problema 5 em diferentes tempos

Ploblema 6: "Preenchimento de molde com 2 contornos de entrada"

Vamos considerar, aqui, uma cavidade retangular com 5cm de largura por 5cm de altura e com duas entradas de fluido no topo com 0.6cm de largura cada uma. A velocidade de entrada do fluido considerada foi de $1m/s$ e a viscosidade cinemática é igual a $\nu = 0.005m^2/s$. Os números de Reynolds e de Froude são dados, respectivamente, por $Re = 1.2$ e $Fr = 4.12$.



t=1.667s t=13.333s

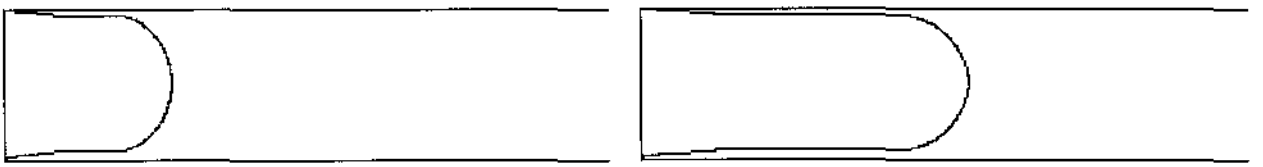


t=20.000s t=31.667s

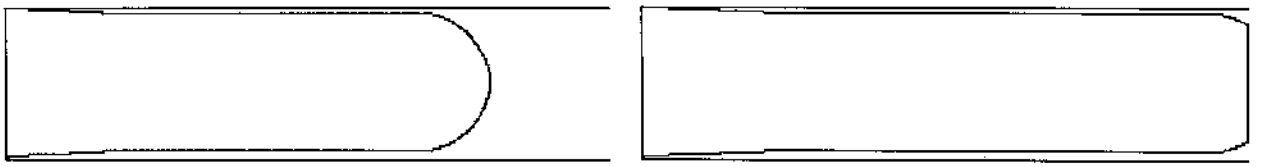
Figura 7.10: Visualização dos resultados para o problema 6 em diferentes tempos

Problema 7: "Tubo"

Vamos considerar, aqui, um tubo retangular com 20cm de comprimento por 5cm de largura, no qual um determinado fluido é injetado através de um contorno de entrada à esquerda, percorre toda a extensão do tubo e é escoado para fora através de um contorno de saida à direita. A velocidade de entrada do fluido considerada foi de $1m/s$ e a viscosidade cinemática é igual a $\nu = 1.000m^2/s$. Os números de Reynolds e de Froude são dados, respectivamente, por $Re = 1.0$ e $Fr = 0.32$.



t=5.000s t=10.000s

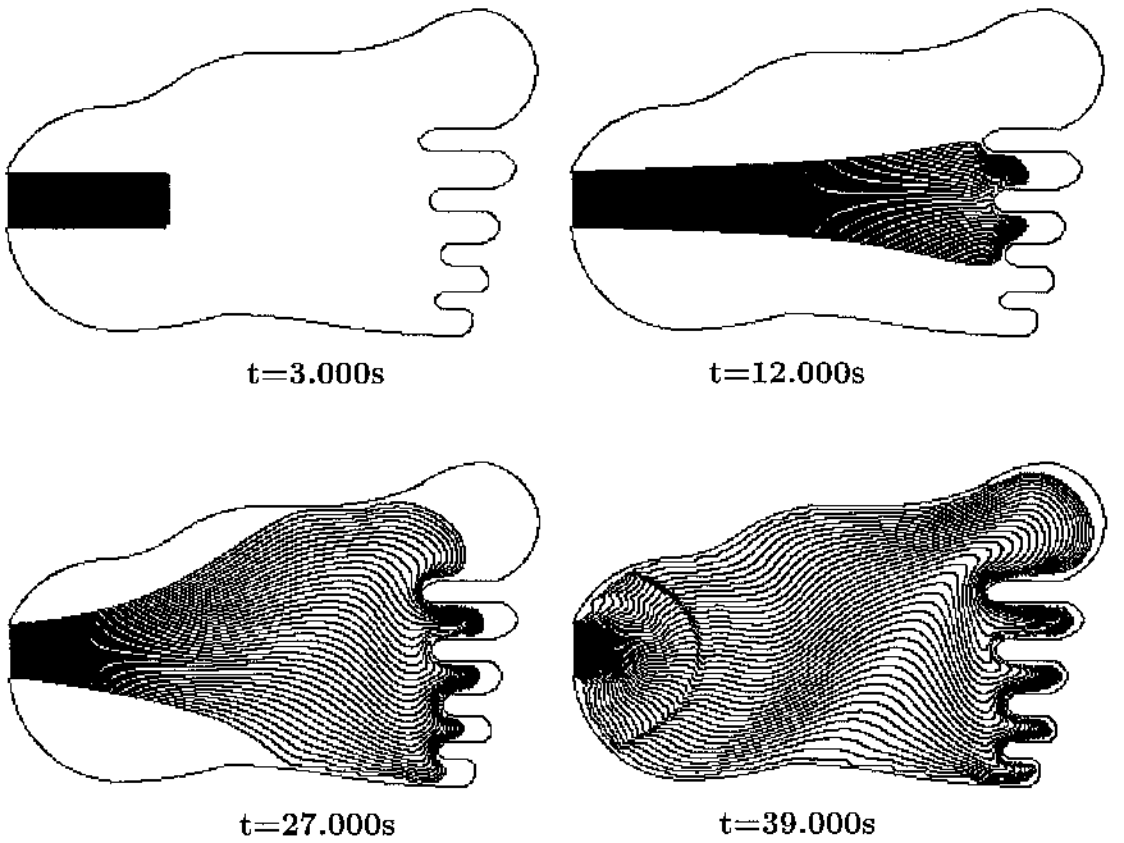


t=15.000s **t=20.000s**

Figura 11: Visualização dos resultados para o problema 7 em diferentes tempos

Problema 8: "Pézinho"

Para mostrar como o código GENSMAC pode lidar com moldes envolvendo altas curvaturas locais vamos considerar o problema do "pézinho", ou seja, vamos considerar um molde na forma de "pézinho" com um contorno de entrada na parede lateral esquerda. As condições de contorno sobre o contorno rígido são do tipo de não escorregamento. Os parâmetros utilizados são: $Re = 1.0$ e $Fr = 50.0$.



t=3.000s **t=12.000s**

t=27.000s **t=39.000s**

Figura 12: Partículas plotadas para o exemplo 8 em diferentes tempos

7.2 Considerações Finais

Alguns dos problemas descritos neste capítulo são utilizados na análise da performance do código paralelo, que é apresentada no próximo capítulo. Os demais problemas são apresentados para ilustrar como o código GENSMAC pode lidar com moldes altamente complexos, como por exemplo, o problema do "pézinho".

Capítulo 8

Análise do Desempenho

Com a utilização de técnicas paralelas é possível se atingir um alto desempenho computacional, mas infelizmente isso nem sempre ocorre. Muitas técnicas podem causar um aumento do tempo de execução de um programa ao invés de diminuí-lo. Assim, quando uma nova técnica é proposta, se faz necessário uma avaliação adequada de seu desempenho.

Neste capítulo, vamos apresentar alguns conceitos necessários na avaliação do paralelismo, tais como: balanceamento de carga, eficiência, escalabilidade e granularidade da decomposição, bem como uma análise do desempenho de alguns problemas propostos. Muitos dos conceitos que são aqui desenvovidos podem ser encontrados em Amêndola [1], Cunha [11] e [12] e Peña [27].

8.1 Balanceamento de Carga

O balanceamento de carga é um aspecto bastante importante dentro da computação paralela, pois ele é necessário para se obter uma redução realmente eficaz do tempo de execução quando comparado com o programa sequencial.

O ideal é fazer com que os processadores que cooperam na execução de uma determinada tarefa executem aproximadamente o mesmo número de operações. O balanceamento de carga pode ser estático ou dinâmico.

Estático: quando se sabe de antemão o número total de operações a serem efetuadas. Pode ser obtido, em determinadas situações, até mesmo embutindo tal informação no programa.

Dinâmico: quando o número de operações depende das atividades de cada processador. O programa necessita de controle dinâmico das atividades atribuidas a cada processador.

A dinâmica do balanceamento de carga apresenta uma grande dificuldade para problemas

com superfícies livres, pois o domínio muda com o tempo. A superfície livre pode tomar geometrias arbitrárias e alterar-se substancialmente com o tempo. Em determinados problemas, na fase inicial, a região ocupada com fluido pode ser pequena e, assim, o é a computação envolvida. Com o passar do tempo essa região pode crescer e, com isso, aumentar a carga de trabalho de cada processador. É nessa fase do processamento que se pode ganhar tempo na execução do programa utilizando vários processadores.

Essa técnica de decompor o domínio em vários subdomínios já no início dos cálculos e atribuir cada um deles a um processador, faz com que os processadores com parte do domínio que não contenham fluido permaneçam inativos durante algum tempo, ou seja, até que fluidos apareçam naquela região. Entretando, isso não nos parece restritivo para o tipo de problema envolvido, visto que este problema ocorrerá independentemente da técnica de paralelização considerada.

8.2 Avaliação do Desempenho

São dois os principais fatores que degradam a eficiência de um algoritmo paralelo: o tempo de comunicação, que é o tempo necessário para a troca de mensagens entre os processos; e o tempo de sincronização, que é o tempo de espera desses processos até que determinadas tarefas sejam concluidas a fim de dar continuidade às suas execuções.

Medir o desempenho de um algoritmo paralelo é uma tarefa bastante complexa devido à diversos fatores, tais como: o pouco desenvolvimento de máquinas e programas no tocante à questão do paralelismo e a necessidade de ambientes propícios para a realização de testes de avaliação.

As medidas mais aceitas para avaliar o desempenho computacional de programas paralelos são a eficiência e o "speed-up".

A eficiência é uma medida de quão bem o algoritmo paralelo utiliza o tempo nos vários processadores e é dada pela seguinte expressão:

$$E(p,n) = \frac{T_s}{pT_p},$$

onde

p é o número de processadores envolvidos na aplicação,

n é a dimensão do problema considerado,

T_s é o tempo de execução do melhor algoritmo sequencial,

T_p é o tempo de execução do algoritmo paralelo, considerando-se p processadores.

O "speed-up" mede o ganho do processamento paralelo sobre o sequencial e é dado pela seguinte expressão:

$$\text{"Speed-up"} = \frac{T_s}{T_p},$$

Este é o chamado "speed-up" absoluto, ou seja, tomando para comparação o melhor algoritmo sequencial, aquele que executa uma determinada tarefa no menor intervalo de tempo possível. Mas, infelizmente, não há um critério para se determinar qual é o melhor algoritmo sequencial.

Uma outra métrica é o "speed-up" relativo, que utiliza o tempo de execução do mesmo algoritmo em um único processador como sendo o tempo do algoritmo sequencial.

A eficiência de um algoritmo varia com o número de processadores e o "speed-up" relativo mede essa variação.

Existem vários comandos para se medir o tempo de execução de um programa, tais como: "mclock", "dtime", "timex" e "time", disponíveis em ambientes UNIX. Mas, infelizmente reconhece-se que a precisão desses "medidores de tempo" não é muito satisfatória.

8.3 Granularidade da Decomposição

A granularidade de um algoritmo paralelo refere-se ao balanço entre duas cargas: computação e comunicação.

Ao escrever um algoritmo paralelo, a distribuição da computação usualmente implica em comunicação entre os processadores.

Um bom "speed-up" só será alcançado se a granularidade for alta, ou seja, a carga de comunicação for muito inferior à de computação.

A granularidade depende de características da máquina, tais como: velocidade de processamento e velocidade de comunicação (latência e largura de banda).

8.4 Escalabilidade

Algoritmos escaláveis são aqueles cuja eficiência paralela $E(p, n)$ permanece uniformemente limitada com relação a n quando o número de processos tende ao infinito para problemas de dimensões fixas, ou seja;

$$0 < E_0 < E(p, n).$$

Se a eficiência do programa paralelo permanecer uniformemente limitada quando o número de processos varia no intervalo

$$P_{min} < P < P_{max},$$

então, o algoritmo é dito "quase-escalável" e tal intervalo é chamado de zona de escalabilidade. Mas a escalabilidade de um algoritmo não tem grande interesse, uma vez que não revela o tempo de execução.

Na próxima seção, expomos os resultados obtidos durante a execução do código paralelo para vários problemas e empregamos os conceitos e definições apresentados anteriormente na análise desses resultados.

8.5 Performance do Código Paralelo

Nesta seção, apresentamos uma análise da performance do código paralelo para vários problemas. Os testes para essa análise foram realizados em uma máquina IBM/SP2 com oito processadores, disponível no Centro Nacional de Computação de Alto Desempenho em São Paulo (CENAPAD/SP), localizado na UNICAMP, em Campinas-SP. As principais características desse equipamento são:

- processadores RISC/6000 mod. 370,
- arquitetura POWER2,
- sistema operacional AIX, versão: 4.1.4,
- 256MB de memória RAM,
- disco local de 2 GB por nó,
- desempenho local de 200 MFlops,
- velocidade de clock 66Hz,
- compiladores: IBM AIX xlf FORTRAN versão 3.2.5 e IBM AIX xlc C++ verão 3.1.4.

Realizamos testes variando as dimensões da malha e também a quantidade de processadores. Foram utilizados 1, 2, 4, 6 e 8 processadores em cada aplicação. Durante a realização desses testes, o computador não estava inteiramente dedicado ao nosso uso e a rede também foi compartilhada com outras aplicações. Na sequência descrevemos cada problema, bem

como, os resultados obtidos. Essa análise é baseada principalmente no balanceamento de carga e nas definições de eficiência e "speed-up" descritas anteriormente.

Para realizar a análise do desempenho, consideramos quatro problemas com características diferentes um do outro.

No problema 1, consideramos um domínio de solução contendo um contorno de entrada na parede lateral esquerda e um contorno de saida na parede lateral direita.

No problema 2, consideramos um escoamento eixo-simétrico.

No exemplo 3, consideramos o problema de preencher, com um determinado fluido, um molde de formato retangular com dois contornos de entrada em sua parte superior.

Finalmente, no exemplo 4, consideramos um escoamento através de um tubo retangular com contorno de entrada à esquerda e contorno de saída à direita.

A seguir apresentamos cada um desses exemplos.

Exemplo 1:

Este exemplo refere-se ao problema 2 apresentado no capítulo 7, no qual consideramos um domínio de solução contendo um contorno de entrada na parede lateral esquerda e um contorno de saida na parede lateral direita. Um determinado fluido é injetado dentro desse domínio e é escoado para fora atraves do contorno de saida. Como pode ser observado, neste caso, a distribuição das tarefas entre os processadores que participam na solução desse problema não é uniforme no início da aplicação, vindo a atingir essa uniformidade somente nos estágios finais de processamento. A seguir, mostramos o molde do domínio de solução utilizado com a finalidade específica de situar o problema.

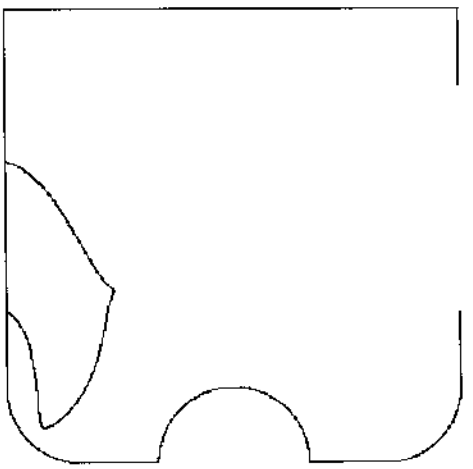


Figura 8.1: Molde para o exemplo 1.

Para analisar a performance do código paralelo, para esse exemplo, utilizamos ambas as

versões do código, ou seja, a versão com partículas e a versão sem partículas e, também, variando as dimensões do problema. Dividimos essa seção em três partes. Na primeira parte (Parte A), utilizamos a versão do código com partículas e o problema com dimensões fixas. Na segunda parte (Parte B), consideramos o problema com dimensões fixas, porém, utilizamos a versão do código sem partículas. Na terceira e última parte (Parte C), utilizamos, também, a versão do código sem partículas, entretanto, variamos as dimensões da malha.

Parte A

Utilizamos, aqui, a versão do código com partículas e a malha com dimensões fixas, ou seja, 153x153. A tabela 8.1 a seguir apresenta os tempos, em segundos, obtidos por cada processador. A nomenclatura utilizada é a seguinte:

T_s é o tempo de execução do código sequencial,

T_i é o tempo de execução de cada processador, considerando-se que foram utilizados i processadores na aplicação,

Pi é o rótulo do processador i,

MT é o máximo dos tempos obtidos em cada aplicação, com a utilização de j processadores.

	T_s	T_2	T_4	T_6	T_8
P1	198018	121740	75009	55302	41555
P2		75977	50129	48417	40650
P3			41258	29669	32311
P4			38042	27638	22897
P5				30812	20586
P6				25261	25078
P7					23907
P8					18656
MT	198018	197717	204438	217099	225650

Tabela 8.1: Tempos, em segundos, obtidos por cada processador para o exemplo 1

Note que o tempo necessário para a obtenção da solução deste problema caiu de 198018 segundos, com a utilização de 1 processador, para 41555, com a utilização de 8 processadores, ou seja, houve uma redução que corresponde a quase 5 vezes. Se a solução de um problema, nestas condições, levasse 5 dias para ser encontrada com a simulação em um único processador, a mesma poderia ser obtida em apenas 1 dia com a simulação em 8. Embora não tenhamos obtido a situação ideal, que seria a resolução do problema em 1/8 do tempo,

obtivemos uma diminuição realmente significativa. Este fato pode ser melhor observado na tabela 8.2 a seguir.

A tabela 8.2 mostra a eficiência e o "speed-up" para o mesmo problema, considerando, para o tempo do código paralelo, o tempo do processador mais lento, neste caso, o primeiro.

A nomenclatura utilizada para essa tabela é a seguinte:

NPROC é a quantidade de processadores considerados na aplicação.

eficiência é a eficiência alcançada utilizando-se NPROC processadores na aplicação,

"speed-up" é o "speed-up" alcançado utilizando-se NPROC processadores na aplicação.

NPROC	Eficiência	"speed-up"
2	0.81	1.62
4	0.66	2.64
6	0.60	3.60
8	0.60	4.80

Tabela 8.2: Eficiência e "speed-up" para o exemplo 1.A.

Parte B

Neste caso, as dimensões consideradas para a malha foram, também, 153x153, porém, utilizamos a versão do código sem partículas.

Na sequência, apresentamos uma tabela com os tempos em segundos obtidos por cada processador na resolução desse problema.

	T_s	T_2	T_4	T_6	T_8
P1	65708	44505	25886	19318	15908
P2		34066	22450	19585	17119
P3			20167	13668	14525
P4			18381	14021	11977
P5				13466	11448
P6				11731	13477
P7					11789
P8					9469
MT	65708	78571	86884	91789	105712

Tabela 8.3: Tempos, em segundos, obtidos por cada processador para o exemplo 1.B.

Na tabela 8.3 podemos observar que os tempos obtidos em segundos são muito inferiores aos tempos apresentados na tabela 8.1. Isso nos confirma que o processo de movimentação das partículas é realmente dispendioso, que exige um grande esforço computacional e pode consumir mais tempo de processamento que o necessário para o processamento de todas as outras tarefas. Entretanto a taxa computação/comunicação diminui com a utilização dessa versão do código, fazendo com que a eficiência seja menor, como pode ser observado quando comparamos as tabelas 8.4 a seguir, com a tabela 8.2 apresentada anteriormente.

A diminuição da granularidade da decomposição, de quando utilizamos a versão do código com partículas para quando utilizamos a versão sem partículas, é mais evidente quando aumentamos a quantidade de processadores que irão colaborar na solução do problema, como pode ser observado nos gráficos 8.1 e 8.2 referentes, respectivamente, à eficiência e ao "speed-up" . Os resultados obtidos com a utilização de ambas as versões do código são mostrados nos mesmos gráficos, facilitando a comparação dos resultados.

A tabela 8.4 mostra a eficiência e o "speed-up" alcançados nessa aplicação.

NPROC	Eficiência	"speed-up"
2	0.73	1.46
4	0.63	2.52
6	0.56	3.36
8	0.48	3.84

Tabela 8.4: Eficiência e "speed-up" do exemplo 1,B

Os gráficos 8.1 e 8.2 apresentados a seguir são referentes à eficiência e ao "speed-up" alcançados na solução desse problema, considerando as duas versões do código, ou seja, a versão com partículas e a versão sem partículas. As linhas tracejadas mostradas em ambos os gráficos referem-se à utilização do código com partículas, enquanto que as linhas contínuas referem-se à utilização da versão sem partículas.

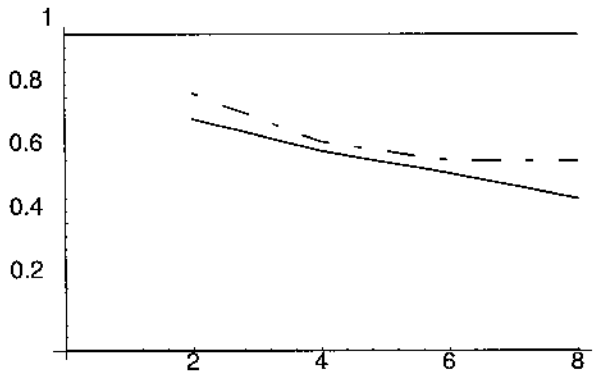


Gráfico 8.1: Eficiência para o problema 1

A semi-reta paralela ao eixo dos x que inicia no ponto $(0,1)$, no gráfico 8.1, representa a curva ideal para a eficiência, ou seja eficiência igual a 1. A semi-reta $x = y$ que inicia no ponto $(0,0)$, no gráfico 8.2, representa a curva ideal para o "speed-up".

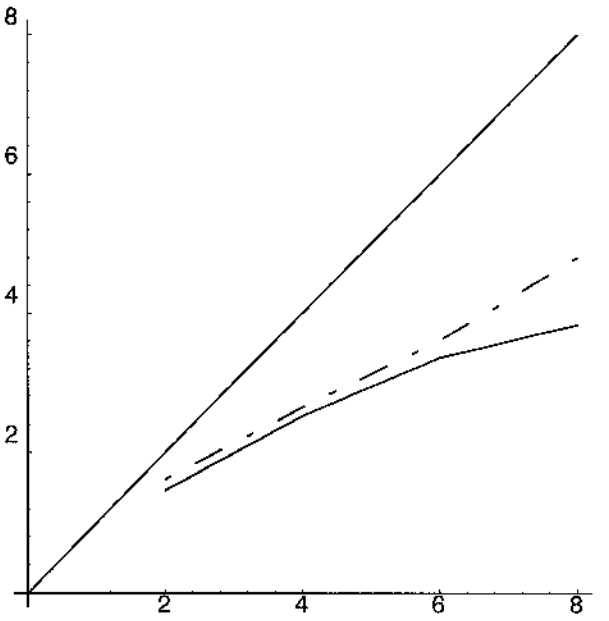


Gráfico 8.2: "Speed-up" para o problema 1

Observando os resultados obtidos para este exemplo nas partes A e B mostrados nas tabelas 8.1, 8.2, 8.3 e 8.4 e nos gráficos 8.1 e 8.2, observamos que a performance do código com partículas é melhor do que o caso sem partículas, podendo transparecer em uma análise superficial que o primeiro código apresenta melhor desempenho que o segundo. Neste caso, temos que a quantidade de cálculos, quando utilizamos a versão do código com partículas, é muito maior que quando utilizamos a versão do código sem partículas, sendo que a quantidade de comunicação não aumenta na mesma proporção na utilização de uma versão para outra e, portanto, é realmente esperado uma eficiência muito melhor no primeiro caso que no segundo. Por outro lado, comparando-se os tempos em segundo obtidos com cada versão, observamos que o tempo do código sem partículas é aproximadamente 1/3 do tempo daquela da versão com partículas.

Parte C

Nesta última parte, mostramos duas tabelas contendo a eficiência e o "speed-up" para o exemplo 1, considerando 2, 4, 6 e 8 processadores e variando o tamanho do problema. Consideramos, aqui, a versão do código sem partículas. A notação utilizada nessas tabelas é a seguinte:

δx é o espaçamento utilizado na malha,

Pi é o rótulo do processador,

NEQ denota a quantidade de equações envolvidas no problema, ou seja, a quantidade de células contendo fluido.

Malha denota as dimensões da malha.

δx	P2	P4	P6	P8	NEQ	Malha
0.020	0.73	0.63	0.56	0.48	0-12000	153x153
0.025	0.66	0.57	0.42	0.35	0-8000	123x123
0.040	0.64	0.44	0.30	0.24	0-3000	78x78
0.050	0.59	0.38	0.24	0.20	0-750	63x63
0.100	0.39	0.19	0.12	0.08	0-500	33x33

Tabela 8.5: Eficiência para o exemplo 1 com malha variável

δx	P2	P4	P6	P8	NEQ	Malha
0.020	1.46	2.52	3.36	3.84	0-12000	153x153
0.025	1.32	2.28	2.52	2.80	0-8000	123x123
0.040	1.28	1.76	1.80	1.92	0-3000	78x78
0.050	1.18	1.52	1.44	1.60	0-750	63x63
0.100	0.78	0.76	0.72	0.64	0-500	33x33

Tabela 8.6: "Speed-up" para o exemplo 1 com malha variável

Dos resultados mostrados anteriormente, podemos observar que, quando mantemos o problema com dimensões fixas e aumentamos a quantidade de processadores, a performance do código paralelo cai. Isto ocorre porque, quando mantemos as dimensões fixas e aumentamos a quantidade de processadores, fazemos com que a carga de trabalho de cada processador seja diminuida, enquanto a quantidade de comunicação aumenta, ou seja, a granularidade diminui. Em outras palavras, a taxa computação/comunicação diminui. Por outro lado, quando mantemos o número de processadores fixos e aumentamos as dimensões do problema, a escalabilidade aumenta e, naturalmente, há um aumento na eficiência, como pode ser verificado nas tabelas 8.5 e 8.6. Pode ser observado, também, nessas mesmas tabelas, que os resultados para a malha 33x33 foram piores no caso paralelo que no caso sequencial. As dimensões do problema, neste caso, são muito pequenas e não justifica o uso de processamento paralelo, pois a granularidade da decomposição é muito baixa.

Os resultados de performance, neste exemplo, ficam prejudicados por não termos uma distribuição uniforme das tarefas entre os processadores envolvidos na aplicação; fato que não ocorre com o próximo exemplo.

Exemplo 2:

Este exemplo refere-se ao escoamento eixo-simétrico apresentado no problema 5 do capítulo 7, no qual o domínio de solução, em coordenadas cilíndricas, tem o formato de uma caixa retangular com 2,5cm de base por uma altura de 5,0cm, com um bocal contendo um contorno de entrada localizado na parte superior esquerda. Consideramos, também, neste exemplo a existência de um determinado fluido contido no interior deste recepiente, no instante inicial de tempo. Desta forma, o problema apresenta uma distribuição uniforme das tarefas entre os processadores que participam na resolução deste problema desde o início da aplicação, ou seja, apesar da dinâmica do balanceamento de carga, há uma distribuição equilibrada da carga de trabalho entre os processadores envolvidos. A seguir, é mostrado o molde do domínio de solução utilizado no processamento deste problema.

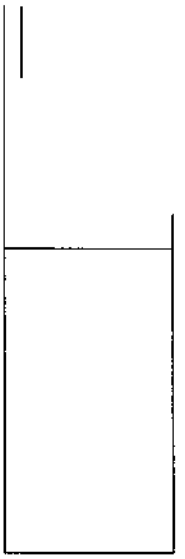


Figura 8.2: Molde para o exemplo 2

As tabelas 8.7 e 8.8 a seguir mostram a eficiência e o "speed-up" para dois tamanhos da malha, com aproximadamente 4.000 e 16.000 células contendo fluido.

Parte A) Com aproximadamente 4.000 células contendo fluido.

NPROC	Eficiência	"speed-up"
2	0.78	1.56
4	0.58	2.32
6	0.48	2.88
8	0.45	3.60

Tabela 8.7: Eficiência e "speed-up" do exemplo 2.A

Parte B) Com aproximadamente 16.000 células contendo fluido.

NPROC	Eficiência	"speed-up"
2	0.83	1.66
4	0.73	2.92
6	0.70	4.20
8	0.65	5.20

Tabela 8.8: Eficiência e "speed-up" do exemplo 2.B.

Mais uma vez, observa-se neste exemplo que, quando mantemos fixa a quantidade de processadores utilizados na aplicação e variamos as dimensões do problema, a performance do código melhora para dimensões maiores e piora para dimensões menores. Este fato pode ser confirmado pelos gráficos da eficiência (gráfico 8.3) e do "speed-up" (gráfico 8.4) mostrados a seguir. Observamos ainda que, quando fixamos as dimensões do problema e variamos a quantidade de processadores, a performance do código melhora com a diminuição da quantidade de processadores e piora com o aumento do número de processadores. Observamos também que este é um exemplo em que temos uma distribuição uniforme da carga de trabalho entre os processadores e, assim, tem-se também uma melhor performance do código. A injeção do jato de fluido faz com que a quantidade de trabalho de cada processador aumente com o tempo, entretando esse aumento é muito lento e a distribuição da carga de trabalho entre os processadores é uniforme. Desta forma, o balanceamento de carga é bastante equilibrado.

As linhas tracejadas nos próximos dois gráficos referem-se aos resultados da parte A, enquanto que as linhas contínuas referem-se à parte B.

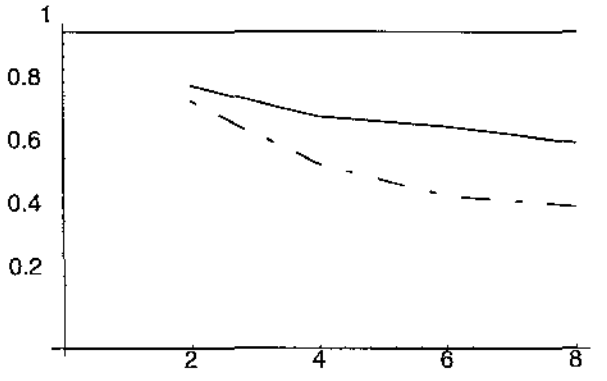


Gráfico 8.3: Eficiência para o problema 2

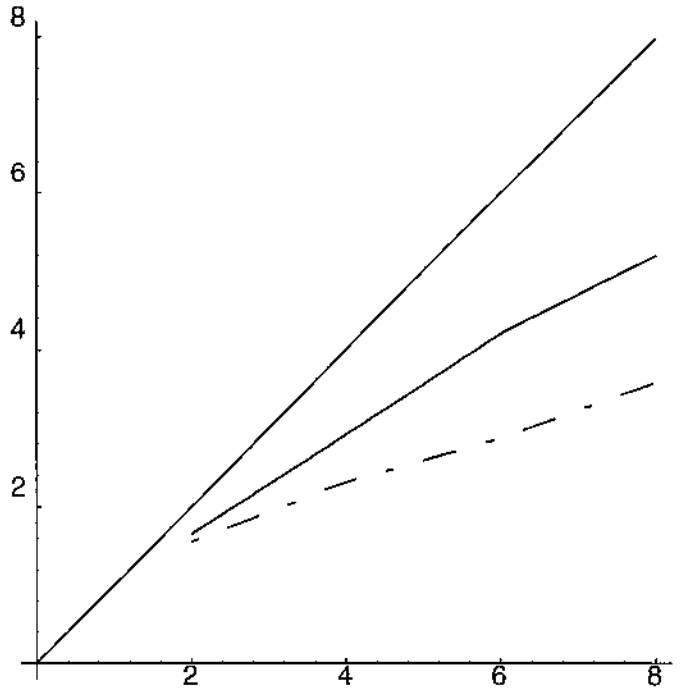


Gráfico 8.4: "Speed-up" para o problema 2

Como pode ser observado nesses gráficos, o código paralelo apresenta realmente uma boa performance, principalmente para o segundo caso, quando a quantidade de cálculos envolvida no problema é grande e justifica o uso de computação paralela. A queda na eficiência, com o aumento da quantidade de processadores, já é esperada em virtude de estarmos diminuindo a quantidade de trabalho de cada processador e aumentando as trocas de informações entre os processos envolvidos na decomposição de domínio, em outras palavras, estamos simplesmente diminuindo a granularidade da decomposição.

Há, porém, uma desvantagem na utilização de grandes quantidades de processadores neste caso. Como o domínio de solução deste problema é retangular com base muito menor que a sua altura, a divisão do domínio em painéis verticais acentua essa característica, ou seja, cada subdomínio será constituido por uma caixa retangular com uma base muito inferior a altura. Desta forma, as áreas de interação entre os processadores, as quais necessitam de trocas de informações, são muito grandes, exigindo a necessidade de troca de muitos dados entre os processadores em cada ponto de sincronismo. Temos, portanto, uma boa distribuição da carga de trabalho entre os processadores, porém, com necessidade de trocar muitos dados entre eles. Este fato impede uma melhora significativa na performance do código.

No próximo exemplo, podemos observar que a distribuição de trabalho é ideal quando consideramos 2 processadores, mesmo tendo um balanceamento de carga dinâmico. No caso de termos 4 processadores, a distribuição da carga de trabalho entre os processadores continua sendo boa e quando utilizamos 6 ou 8 processadores, o balanceamento de carga é prejudicado.

Exemplo 3:

Vamos considerar, aqui, uma cavidade retangular com 10cm de largura por 5cm de altura e com duas entradas de fluido no topo com 0.6cm de largura cada uma. Pode ser observado que, apesar deste problema estar sujeito a contornos de entrada, é possível determinar a quantidade de processadores que devem cooperar em sua solução, de modo que a distribuição dos trabalhos entre os processadores seja uniforme desde o início da aplicação. Neste caso, podemos considerar 2 ou 4 processadores que obteremos esse equilíbrio. Este exemplo refere-se ao problema 6, mostrado no capítulo 7, porém, o molde utilizado aqui apresenta uma base maior que naquele caso, como mostra a figura 8.3.

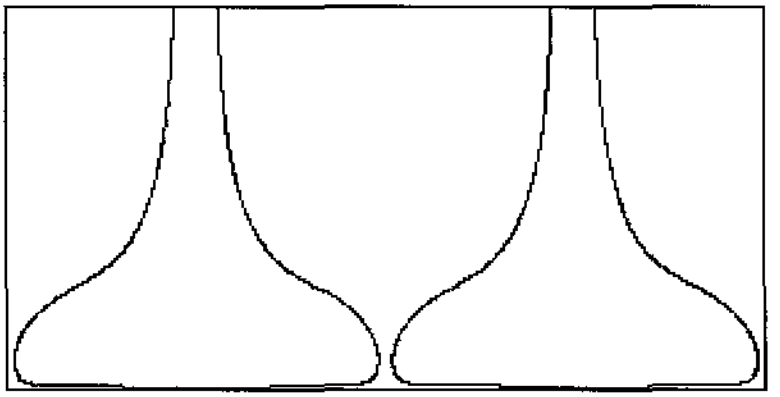


Figura 8.3: Molde para o exemplo 3

Neste caso, consideramos as dimensões do problema fixas e variamos a quantidade de processadores. Note que com 2 processadores temos a situação ideal, pois no início dos cálculos, quando a carga de trabalho de cada processador ainda é pequena, praticamente não se tem comunicação entre os processadores. As trocas de informações vão sendo necessárias conforme o fluido vai preenchendo o recipiente. No caso de quatro processadores, temos também uma distribuição equilibrada da carga de trabalho entre os processadores, porém as comunicações são necessárias desde o início dos cálculos. Para 6 e 8 processadores o balanceamento de carga fica prejudicado em função da geometria do problema.

A tabela 8.11 mostra a eficiência e o "speed-up" para o problema considerado.

NPROC	Eficiência	"speed-up"
2	0.97	1.94
4	0.76	3.04
6	0.65	3.60
8	0.57	4.56

Tabela 8.11: Eficiência e "speed-up" do exemplo 3

Quando utilizamos dois processadores na solução deste problema temos uma distribuição ideal da quantidade de trabalho e ainda temos a vantagem de praticamente não ter comunicação no início dos cálculos. As comunicações necessárias vão aumentando conforme vão aumentando os cálculos, fazendo com que a granularidade da decomposição permaneça alta em todas as fases do processamento, não prejudicando, desta forma, a performance do código, que neste exemplo atingiu seu ponto máximo. Para 4 processadores, a distribuição da carga de trabalho entre os processadores ainda é satisfatória, porém, os resultados ficam um pouco prejudicados, pois há a necessidade de troca de informações entre os processadores desde o início dos cálculos, quando a granularidade da decomposição é baixa. A granularidade vai aumentando com a evolução do tempo. Os resultados neste caso são influenciados também por mantermos as dimensões do problema fixas e aumentarmos a quantidade de processadores. Para 6 ou 8 processadores, a geometria do problema faz com que as tarefas não sejam bem distribuidas no início dos cálculos, prejudicando com isso os resultados em sua fase inicial. Porém, após transcorrido um certo período de tempo esse efeito é suavizado.

Como pode ser observado nas tabelas 8.6 do exemplo anterior e 8.11 do exemplo atual, os resultados neste caso para 2 e 4 processadores são melhores que para o caso anterior, apesar das dimensões desse problema serem menores. A explicação para isso se deve à quantidade de troca de informações exigidas em cada caso. No exemplo 2, é exigida a troca de informações em uma região muito grande e no exemplo em questão a região de troca de informações é dinâmica, cresce conforme a necessidade, porém, não chega a atingir as dimensões anteriores. Este fato pode ser confirmado comparando-se a base de cada subdomínio com sua altura, em ambos os casos.

Exemplo 4:

Este problema refere-se ao problema 7 do capítulo anterior, ou seja, consideramos um escoamento através de um tubo retangular com contorno de entrada à esquerda e contorno de saída à direita. Este exemplo, de certa forma, é semelhante ao exemplo 1. Entretanto, neste caso, consideramos duas medidas de tempo. A primeira, considerando o tempo total desde o início da aplicação (quando ainda não existe uma distribuição uniforme das tarefas entre os processadores) até o final (quando todos os processadores contém a mesma carga de trabalho). Na segunda medição dos tempos, consideramos apenas os tempos em que cada processador já continha a mesma carga de trabalho que os demais. Seu molde é mostrado na figura 8.4. Mais uma vez, utilizamos a versão do código sem partículas, tanto na visualização como na análise dos resultados.

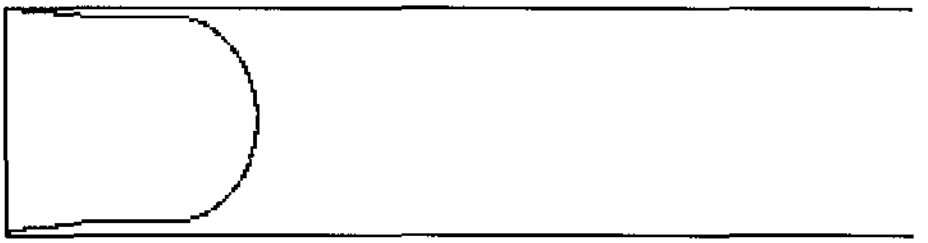


Figura 8.4: Molde para o exemplo 4

Dividimos essa apresentação em duas partes.

Parte A

Nestaa primeira parte, medimos o tempo de processamento necessário para simular 30 segundos de escoamento, partindo do tempo inicial $t = 0s$ até atingir o tempo final $t = 30s$.

A tabela 8.12 a seguir mostra a eficiência e o "speed-up" para o problema considerado. Para a resolução desse problema foi utilizada uma malha 200x50 e a quantidade de equações envolvidas no problema variou de 0 a 9400.

NPROC	Eficiência	"speed-up"
2	0.65	1.29
4	0.50	2.00
6	0.42	2.52
8	0.37	2.98

Tabela 8.12: Eficiência e "speed-up" do exemplo 4.A

Parte B

Nesta segunda parte, consideramos o problema com as mesmas dimensões utilizadas anteriormente, porém, medimos os tempos de processamento computacional apenas no intervalo de tempo entre 20 e 30 segundos de escoamento, ou seja, quando a carga de trabalho entre os processadores atinge uma distribuição uniforme.

NPROC	Eficiência	"speed-up"
2	0.79	1.69
4	0.61	2.45
6	0.54	3.25
8	0.53	4.28

Tabela 8.13: Eficiência e "speed-up" do exemplo 4.B

Observe que nos primeiros 20 segundos de escoamento, a carga de trabalho não é uniformemente distribuida entre os processadores. Somente após transcorridos esse tempo, a distribuição se torna equilibrada. Na Parte B, mostramos os resultados para o escoamento no intervalo de tempo de 20 a 30 segundos, quando carga de trabalho é bem distribuida entre os processadores e, mais uma vez, pode ser observado que o balanceamento de carga é crucial para se ter um bom desempenho computacional.

Os resultados obtidos na primeira parte ficam bastante comprometidos em função do balanceamento de carga. Os resultados da parte B só não são melhores por se tratar de um problema pequeno onde não se justifica a utilização do paralelismo com uma quantidade grande de processadores.

8.6 Considerações Finais

Na análise que realizamos, consideramos problemas de dimensões relativamente pequenas, onde não se justifica o uso do paralelismo para uma quantidade grande de processadores. A razão para isso é não termos disponível máquinas para utilização sem limitante de horário para processamento. Nas máquinas que utilizamos tínhamos autorização para processamento por no máximo 24 horas, quantidade de tempo que não é suficiente para resolver problemas de grande porte, principalmente para medir o tempo do programa sequencial e do programa paralelo com a utilização de dois processadores que são os mais demorados.

Vale ainda lembrar que todos esses exemplos referem-se a problemas com balanço de carga dinâmico, nem sempre permitindo um balanceamento de carga equilibrado no início dos cálculos.

Capítulo 9

Conclusões e Trabalhos Futuros

9.1 Conclusões

Propomos, neste trabalho, uma técnica de decomposição de domínios para simular problemas de escoamento de fluidos em domínios gerais. Tais problemas são, na maioria das vezes, resolvidos numericamente e de solução bastante trabalhosa, exigindo uma grande quantidade de cálculos matemáticos e, consequentemente, um grande esforço computacional, principalmente quando são utilizadas malhas muito finas que dão maior estabilidade e precisão ao método utilizado.

A técnica de decomposição de domínio que propomos constitui-se em dividir o domínio de solução em vários subdomínios com aproximadamente as mesmas dimensões e, então, atribuir o conjunto de dados de cada um deles a um processador que se encarrega de aplicar o mesmo algoritmo de solução. A junção dos resultados parciais de cada subdomínio fornece a solução geral procurada do problema.

Como estamos tratando de problemas de escoamento de fluidos em domínios gerais sujeitos, na maioria das vezes, a contornos de entrada e/ou de saida, a carga de trabalho dos algoritmos se concentra nas regiões onde se localizam os corpos de fluido e essas regiões podem se alterar substancialmente com a evolução do tempo. Assim, não podemos esperar distribuir sempre uniformemente a carga de trabalho entre os subdomínios que irão constituir a aplicação.

O objetivo principal da técnica de decomposição de domínio que propomos é a determinação da solução, em arquiteturas paralelas, de problemas de escoamento de fluidos com superfícies livres, a fim de que o tempo necessário para o processamento computacional seja diminuido.

Para viabilizar a técnica de decomposição de domínio proposta, apresentamos, também, uma implementação paralela do método SMAC. Esta implementação é baseada no código

GENSMAC que foi testado com bastante sucesso para uma variedade de problemas e se mostrou um algoritmo eficaz e robusto na solução desse tipo de problema.

A implementação paralela que apresentamos é baseada na "computação paralela" por troca de mensagens. A implementação das trocas de informações entre um processador e outro foi realizada utilizando o "software" PVM, que permite o uso de um conjunto de estações de trabalho ou mesmo de microcomputadores como sendo uma máquina paralela, que hoje em dia é de custo relativamente baixo.

Os testes de eficiência foram realizados em uma máquina paralela IBM/SP com 8 processadores, porém as trocas de informações entre os processadores foram realizadas utilizando a rede local, que foi compartilhada com outras aplicações. Infelizmente os "switchs" próprios para comunicação entre os processadores dessa máquina não estavam disponibilizados para o nosso uso, prejudicando, desta forma, os resultados. Acreditamos que se fossem utilizados esses "switchs" os resultados poderiam ser bem melhores. Por outro lado, este fato nos mostrou que o algoritmo paralelo que propusemos pode ser executado em uma rede de máquinas como sendo um computador paralelo e apresentar bons resultados mesmo sendo a rede local compartilhada com outras aplicações.

Por se tratar de um problema bastante complexo e o domínio de solução variar consideravelmente em função do tempo, podendo começar ocupando uma região bastante pequena que vai ganhando espaço com a evolução do tempo, ou mesmo o processo inverso, o domínio de solução pode ser grande no início e ir diminuindo com o tempo (isto é uma característica inerente ao tipo de problema que estamos trabalhando), concluimos, baseado nos testes de performance realizados, que os resultados apresentados são satisfatórios.

Devido às características do tipo de problema que estamos resolvendo, tais como a dinâmica da movimentação do fluido e o tipo de molde utilizado, deve-se procurar associar o problema que se tem interesse em resolver à quantidade de processadores que irão colaborar em sua solução. Associar também as dimensões do problema a essa quantidade, pois quanto maior o problema maior é a quantidade de processadores que se pode utilizar sem afetar a performance do código. Essa combinação é perfeitamente exeqüível em alguns casos, dependendo, é claro, do tipo de problema que estamos interessados em resolver.

Se não podemos resolver com boa performance todos os tipos de problemas que são resolvidos pela metodologia GENSMAC, pelo menos uma boa parte deles pode ser resolvida com sucesso pelo código paralelo que estamos propondo.

As reais necessidades de propostas de técnicas eficientes para solucionar problemas de escoamentos com superfícies livres são para a classe de problemas tri-dimensionais gerais que, com os recursos disponíveis nos dias de hoje, são de solução excessivamente dispendiosas, principalmente no tocante à questão do tempo necessário para o processamento computacional. O computador é hoje uma ferramenta sem a qual, com as técnicas disponíveis, a

solução de tais problemas seria completamente inviabilizada. Desta forma, a técnica de decomposição de domínios e a implementação paralela que propusemos neste trabalho representam um primeiro teste e serve de base para o desenvolvimento de técnicas de solução similares para a solução de problemas tri-dimensionais gerais, que é o propósito central de todo o trabalho que desenvolvemos até o momento.

Apesar de termos consciência de que a extensão pura e simples de um modelo matemático em uma determinada dimensão para outra superior é, na maioria das vezes, uma tarefa árdua e com um grau de dificuldade muito maior, fica aqui lançada a primeira "sementinha" para que no futuro, esperamos não tão distante, que essa idéia possa "florescer" com sucesso e a solução bastante trabalhosa nos dias de hoje para essa classe de problemas, possa ser viabilizada sem maiores dificuldades, nos dias de amanhã.

9.2 Trabalhos Futuros

Como já dissemos na seção anterior, o propósito essencial do trabalho que aqui estamos apresentando é a extensão de uma técnica de decomposição de domínios, similar a esta, para a solução de problemas de escoamento com superfícies livres tri-dimensionais gerais, que devido ao grande período de tempo necessário para o processamento computacional envolvido na aplicação e a grande quantidade de informações que devem ser armazenadas, a implementação paralela é uma necessidade real. Propomos, então, como um trabalho futuro tal extensão.

Ainda no caso bi-dimensional podem ser realizadas outras tarefas, que podem auxiliar no desenvolvimento de algoritmos paralelos para o caso tri-dimensional, como por exemplo o tratamento dispensado à aplicação das condições de contorno sobre a superfície livre.

A aplicação das condições de contorno sobre a superfície livre, na metodologia GENSMAC, é análoga à utilizada na metodologia SMAC. A aplicação dessas condições no caso da implementação paralela é bastante complexa, pois a arbitrariedade da geometria associada ao problema faz com que ocorram situações imprevisíveis, situações estas que podem ocorrer na fronteira dos subdomínios envolvidos na aplicação exigindo, com isso, muitas trocas de informações entre os processadores e, portanto, aumentando os pontos de sincronismos. São duas as rotinas que implementam a aplicação dessas condições e como cada uma delas é chamada duas vezes dentro do ciclo de cálculo, os pontos de sincronismos gerados nesses estágios de processamento são realmente muitos. Desta forma, propomos a busca de técnicas de aplicações dessas condições de forma tal que esses pontos de sincronismos sejam minimizados ou preferencialmente eliminados no caso da implementação paralela.

Observamos, aqui, que a procura de novas técnicas de aproximações das condições de contorno sobre a superfície livre beneficiará também a implementação computacional paralela da metodologia GENSMAC para problemas tri-dimensionais gerais, pois com a grande quantidade de informação que deve ser trocada entre os processadores, que deve se acentuar ainda mais no caso tri-dimensional, a diminuição de pontos de sincronismos é muito importante para que se possa realmente atingir um bom desempenho computacional.

Uma outra tarefa, ainda, está relacionada ao armazenamento das informações das componentes que representam a superfície livre, como está descrito no capítulo 6. Toda vez que é exigida a troca de informações entre duas componentes, os trechos que são transferidos de um processador para outro são atualizados em cada processador procurando sua posição pela distância mínima. Embora não tenhamos detectado nenhum problema com a utilização dessa técnica para os exemplos testados, propomos uma outra maneira de implementação, que nos parece mais precisa. Entretanto sua implementação ainda não foi viabilizada por não ser tão imediata. Acreditamos ser ela viável.

Como descrevemos no capítulo 6, a técnica proposta inicialmente na qual todas as componentes eram numeradas, ou seja, continham um rótulo, poderia gerar muitos pontos de sincronismos e sua implementação seria demasiadamente complicada. A técnica que estamos propondo atualmente é que cada componente contenha dois rótulos, um deles se referindo ao início da componente e o outro se referindo ao seu final. Para exemplificar, vamos considerar a situação mostrada na figura 9.1:

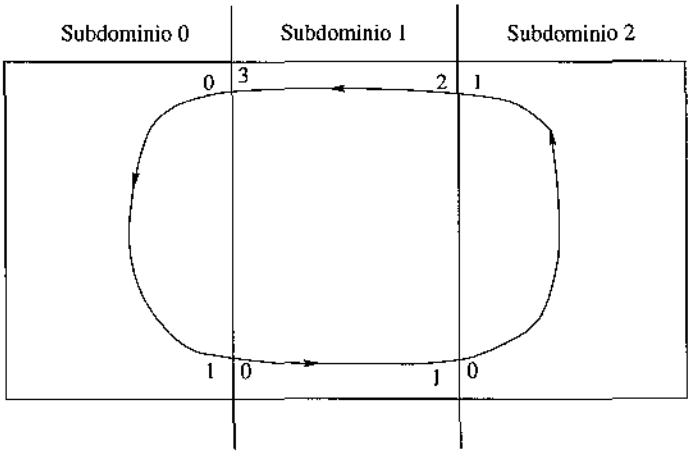


Figura 9.1: Lista com dois rótulos

Como pode ser observado nesta figura, temos um mesmo corpo de fluido distribuido por três subdomínios. As várias componentes, distribuidas geograficamente entre os processadores, armazenam informações que referem-se ao seu início e ao seu final. Por exemplo, a componente armazenada pelo processador 0, que tem o rótulo 0 no início e 1 no final, é a

continuação natural da componente com rótulo final 3, pertencente ao processador 1. Essa mesma componente, pertencente ao subdomínio 0, tem continuação natural na componente com rótulo inicial 0 no processador 1. Assim, toda vez que é exigida troca de informações entre duas componentes tem-se a localização exata de onde essas informações devem ser armazenadas. Por exemplo, toda vez que o final da componente com rótulo 3 no processador 1 envia informações para o processador 0, sabe-se de antemão que essas informações devem ser recebidas pelo início da componente com rótulo inicial 0.

Caso haja a necessidade de desmembramento de uma componente, como mostra a figura 9.2, os rótulos de início e final devem ser mantidos e devem ser gerados novos rótulos somente para as partes envolvidas nos desmembramentos. Para exemplificar, vamos supor que dois trechos centrais da componente armazenada pelo processador 2 devam ser transferidos para o processador 1. A figura 9.2 nos dá uma idéia de como ficam as componentes após realizadas as transferências necessárias.

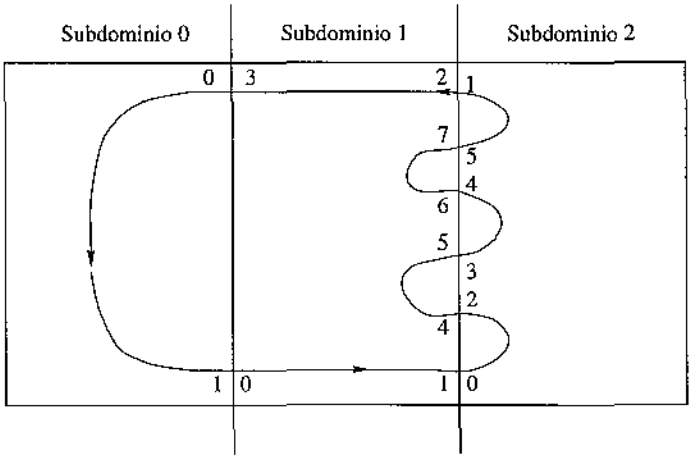


Figura 9.2: Desmembramentos de componentes

Observe, nesta figura, que os desmembramentos ocorridos com a componente com rótulo inicial 0 e rótulo final 1 pertencente ao processador 2, não alteram as informações nos demais processadores, pois a componente com rótulo final 1 pertencente ao processador 1 continua tendo sua continuação natural na componente com rótulo inicial 0 do processador 2. Da mesma forma, a componente com rótulo inicial 2 pertencente ao processador 1 continua sendo a continuação natural da componente com rótulo final 1 do processador 2.

A organização das componentes dessa forma faz com que não haja enganos nas trocas de informações entre os processadores. Há, porém, a necessidade de um ponto de sincronismo: toda vez que ocorrer desmembramentos de componentes, os processadores envolvidos nessa operação devem trocar informações para que haja um perfeito ajuste dos rótulos de cada componente. Por exemplo, o processador 1 deve receber, no caso da situação mostrada

na figura 9.2, duas novas componentes que recebem, respectivamente, rótulos iniciais 4 e 6 e rótulos finais 5 e 7. A única componente existente anteriormente no processador 2 foi desmembrada em três componentes que passam a conter os seguintes rótulos:

primeira componente: rótulo inicial 0, rótulo final 2;

segunda componente: rótulo inicial 3, rótulo final 4;

terceira componente: rótulo inicial 5, rótulo final 1.

Os processadores 1 e 2 devem trocar informações para atualizar os dados referentes a cada componente, ou seja, qual é a continuação natural de cada componente no outro processador. Esse ponto de sincronismo não existe na implementação atual.

Caso todas as componentes pertencentes ao subdomínio 2 devam ser transferidas para o subdomínio 1 e, neste caso, devam ser agrupadas às componentes já existentes nesse subdomínio vindo a se transformar em uma única componente, o resultado final será, então, uma componente com rótulo inicial 0 e rótulo final 3, não prejudicando, desta forma, as informações armazenadas pelo processador 0.

9.3 Considerações Finais

Algumas técnicas de solução, na tentativa de solucionar determinados problemas, podem surgir e desaparecer em um pequeno espaço de tempo, devido ao surgimento de outras técnicas muitas vezes mais simples, menos árduas e até mesmo mais abrangentes.

O termo "computação paralela" refere-se, de maneira geral, à utilização de vários recursos computacionais cooperando na execução de uma determinada tarefa. Por exemplo, a operação em caixas bancárias automáticas enquadram-se nesse tipo de computação.

A utilização da computação paralela nos ramos da ciência, tais como em aplicações físicas e problemas das engenharias, surgiu da necessidade de aceleração do tempo de processamento computacional e, também, da necessidade de armazenamento de grande quantidade de informações, viabilizando, dessa maneira, a solução de problemas até então inviáveis.

Tenho consciência de que, com o surgimento de máquinas cada vez mais poderosas capazes de processamento de grande quantidade de dados em um pequeno espaço de tempo e com grande capacidade de armazenamento de informações, a "computação paralela científica" pode vir a cair em desuso. Dependendo, entretanto, do curso da história, pode vir a ser mais necessária do que é hoje, em virtude de problemas que podem ser deflagrados pelas necessidades da vida moderna. Permita-me, aqui, transcrever na íntegra uma frase de Rudnei Dias da Cunha, encontrada em [12] e que, em minha opinião, expressa de maneira bastante

clara esse pensamento:

" No futuro (não tão distante), já vislumbramos a utilização de computadores óticos, com uma velocidade de operação ordens de magnitude mais rápida do que os mais potentes computadores hoje existentes. Se, por um lado, para muitos problemas essa velocidade será suficientemente grande que não necessitará o uso de programação paralela (a nível de aplicação), sabemos que as necessidades da humanidade sempre ultrapassarão os recursos disponíveis. Acreditamos, portanto, que o uso de arquiteturas e programação paralelas permanecerá sendo de grande valia".

Referências Bibliográficas

[1] Amêndola, M. - Resolução Numérica de um Problema de Fronteira Livre: Cavitação na Lubrificação Hidrodinâmica de Mancais. Tese de Doutorado, IMECC-UNICAMP, Campinas-SP, 1996.

[2] Amsden, A.A.; Harlow, F.H. - The SMAC method: A numerical technique for calculating incompressible fluid flows - Los Alamos Scientific Laboratory, Report LA-4370, 1971.

[3] Batchelor, G.K., An Introduction to Fluid Dynamics, Cambridge University Press, Cambridge-UK, 1967.

[4] Boaventura, M. et. al. - Parallel Solution of Free Surface Flows, in: VI Congresso Brasileiro de Engenharia e Ciências Térmicas, VI Congreso Latinoamericano de Transferencia de Calor y Materia. Proceedings, Florianópolis, **6** pp. 583-588, 1996.

[5] Boaventura, M. et. al. - Simulation of Free Surface Flows in a Distributed Memory Environment - in Applied Computational Mathematical Topics in Partial Differential Equations - V. Pereyra and J. Castillo, Editors, Birkhauser, pp. 141-160, 1998.

[6] Chan, R.K.C., A Generalized Arbitrary Lagrangian-Eulerian Method for Incompressible Flows with Sharp Interfaces, J. Comp. Phys., **17**, 311-331, 1975.

[7] Chan, R.K.C. e Street, R.L., A Computer Study of Finite Amplitude Water Waves, J. Comp. Phys., **6**,68-94, 1971.

[8] Cunha, R.D. - A Study of Iterative Methods for the Solution of Systems of Linear Equations on Transputer Networks - Ph. D. Thesis - University of Kent at Canterbury U.K., 1992.

[9] Cunha, R.D.; Hopkins, T. - The parallel solution of linear equations using iterative methods on transputer networks - Technical Report 16/92 - University of Kent at Canterbury U.K., 1992.

[10] Cunha, R.D.; Hopkins, T. - PIM 1.1 - The parallel iterative package for systems of linear equation - User's guide - Technical Report - University of Kent at Canterbury U.K., 1993.

[11] Cunha, R.D. - Paralelização de operadores de álgebra linear e sua aplicação em métodos numéricos. XIII Curso de verão - ICMSC-USP, São Carlos - SP, 1996.

[12] Cunha, R.D. - Computação Paralela: uma breve introdução. XX CNMAC, Gramado-RS, 1997.

[13] Deville, M. O. - Numerical Experiments on the MAC Code for a Slow Flow - Journal of Computational Physics, 15, **13** 362-374, 1974.

[14] Geist, A. et. al. - PVM3 user 's Guide and Reference Manual - Oak Ridge - National Laboratory, Oak Ridge - Tennessee, 1994.

[15] Golafshani, M., A Simple Numerical Technique for Transient Creep Flows with Free Surfaces, Int. J. for Numer. Meth. Fluids, **8**, 897-912.

[16] Graham, S. J. et. al., Fluid Flow in thin films using finite elements, Mathematical Engineering in Industry, **3**, 4, 229-246, 1992.

[17] Harlow, F.; Welch, J.E. - Numerical calculation of time-dependent viscous incompressible flow of fluid with free surfaces - Phys. Fluids **8** pp. 2182-2189, 1965.

[18] Hirt, C.W. e Shannon, J.P., Free-Surface Stress Conditions for Incompressible-Flow Calculations, J. Comp. Phys.,**2**, 403-411, 1968.

[19] Hirt, C.W. et. al., A Lagrangian Method for calculation the dynamics of an incompressible fluid with a free surface, J. Comp. Phys., **5**, 103-124, 1970.

[20] Markham, G. e Proctor, M.V., Outline Description of a Recently Implemented Fluid Flow Code Zuni, C.E.G.B. report LM/PHYS/258, 1981.

[21] McQueen, J.F. e Rutter, P., Modifications to the Two-Dimensional Incompressible Fluid Flow Code ZUNI to Provide Enhanced Performance, C.E.G.B. report TPRD/L/0063/M82, 1983.

[22] Melo, S.T. e Moura Neto, F. - Mecânica dos Fluidos e Equações Diferenciais. 18o. Colóquio Brasileiro de Matemática, IMPA-CNPq, Rio de Janeiro-RT, 1988.

[23] Miyata, H. e Nishimura, S., Finite-Diference Simulation of Nonlinear Waves Generated by Ships of Arbitrary Three-Dimensional Configuration, J. Comp. Phys., **60**, 391-436, 1985.

[24] Miyata, H. e Nishimura, S., Finite-difference simulations of nonlinear ship waves, J. Fluid Mech., **157**, 327-357, 1985.

[25] Nichols, B.D. e Hirt, C.W., Improved Free Surface Boundary conditions for Numerical Incompressible Flow Calculations, J. Comp. Phys., **8**, 434, 1971.

[26] Nickell, R.E. at. al., The solution of viscous incompressible jet and free surface flows using finite element methods, J. Fluid Mech., **65**, 189-206, 1974.

[27] Peña, M.E.C. - Um algoritmo de Alta Precisão para o Sistema Acoplado da Termoelastodinâmica. Tese de Doutorado, IM-UFRJ, Rio de Janeiro-RJ, 1996.

[28] Perez, P.A.S. - Um ambiente de auxílio a paralelização de aplicações FORTRAN. Dissertação de Mestrado, ICMSC-USP, São Carlos-SP, 1996.

[29] Pracht, W.E., A Numerical Methods for Calculating Transient Creeping Flows, J. Comp. Phys., **7**, 46-60, 1971.

[30] Pracht, W.E., Calculating Three-Dimensional Fluid Flows at All Speeds with an Eulerian-Lagrangian Computing Mesh, J. Comp. Phys., **17**, 132-159, 1975.

[31] Tomé, M. F. et al. - GENSMAC3D: Implementation of the Navier-Stokes Equations and Boundary Conditions for 3D Free Surface Flows, Notas do ICMSC, Nº 29, [1996].

[32] Tome, M.F.; McKee, S. - GENSMAC: A computational marker and cell method for free surface flows in general domains - J. Comp. Physics **110** pp. 171-186, 1994.

[33] Tome, M. F. - GENSMAC: A Multiple Free Surface Fluid Flow Solver - Ph. D. Thesis University of Strathclyde - Glasgow - Scotland, 1993.

[34] Tomé, M. F. et al. - Numerical Simulation of Axisymetric Free Surface Flows, Notas do ICMSC-USP, Nº 30, São Carlos, 1996.

[35] Tomé, M. F. et al. - A Marker-and-Cell Thecnique for Solving Axysimetric Free Surface Flows. in: VI Congresso Brasileiro de Engenharia e Ciências Térmicas, VI Congreso Latinoamericano de Transferencia de Calor y Materia. Proceedings, Florianópolis, **6**, 1996.

[36] Tomé, M. F. e McKee, S. - Freeflow - A multiple free surface Fluid flow code, A user's Manual.

[37] Viecelli, J. A. - A Method for including arbitrary external boundaries in the MAC incompressible fluid computing technique - J. Comp. Phys. **4** pp. 543-551, 1969.

[38] Viecelli, J. A., A Computing Method for Incompressible Flows Bounded by Moving Walls, J. Comp. Phys., 8, 119-143, 1971.

[39] Welch, J.E. at. al., The MAC Method, Los Alamos Scientific Lab. Report LA-3425, Los Alamos, 1966.

Apêndice A

Esboço do Código

Neste apêndice apresentamos um esboço do código GENSMAC.

A.1 A Estrutura do Código GENSMAC com Partículas

A versão do código GENSMAC com partículas tem a seguinte estrutura: o programa principal, chamado por GENSMAC, faz chamada à rotina GSMAC, onde são realizados todos os cálculos da metodologia SMAC. A rotina GSMAC é composta por duas partes: na primeira são chamadas algumas rotinas que fazem a preparação dos dados e, na outra parte, são chamadas as rotinas que compõem o ciclo de cálculo descrito nos passos de 1 a 4 na seção 4.1 do capítulo 4.

```
PROGRAM GENSMAC
início
   . CALL GSMAC
fim.


   ROTINA GSMAC
* início
* Chamadas às rotinas da primeira parte.
   . Leitura de dados.
   . CALL FLAGGING
```

```
   . CALL INTERVALOS
   . CALL INFLOW
   . CALL PART
   . CALL CORTE
   . CALL BCOND
   . CALL TANVEL
   . CALL TREE
* Rotinas da segunda parte (ciclo de cálculo)
   . DO WHILE (T.LE.TWFIN.AND.IERROR.EQ.0)
* T variável que incrementa o tempo
* TWFIN variável que armazena o tempo máximo permitido para processamento
* IERROR variável com o valor zero se não ocorrer nenhum problema
        . IF (T.GT.TCLOSEINF)
            . CALL STOPINF
        . END IF
        . CALL REFLAG
* rotinas que implementam o passo 1 descrito anteriormente
        . CALL SUREMP
        . CALL TANOUT
        . CALL BCOND
        . CALL TANVEL
        . CALL PSEUDPRESS
        . CALL DTSET
        . CALL TILDEV
* rotina que implementa o passo 2
        . CALL CGSSOLVER
* rotinas que implementam o passo 3
        . CALL FINVEL
        . CALL SUREMP
        . CALL TANOUT
        . CALL BCOND
        . CALL TANVEL
* rotina que implementa o passo 4
        . CALL PACKIN
   . END DO
* fim da rotina GSMAC.
```

Descrevemos a seguir, resumidamente, cada uma dessas rotinas.

Rotina STOPINF

Rotina que encerra a entrada de fluido. Se t (tempo decorrido até o estágio atual de processamento) for maior que TCLOSEINF (tempo permitido para a entrada de fluido), o programa faz uma chamada a essa rotina que faz com que seja cessada a entrada de fluido.

Rotina CGSSOLVER

Esta rotina constrói o sistema linear resultante das discretizações de diferenças finitas de cinco pontos da equação de Poisson. A solução deste sistema é, então, determinada pelo pacote PIM.

Após a determinação da solução do sistema linear, o programa paralelo exige a comunicação de elementos da solução que estejam vizinhos a processadores.

A implementação do produto matriz-vetor, exigida pelo pacote PIM, foi realizada na rotina PDMVPDE.

Rotina FLAGGING

Esta rotina identifica e marca com B as células que fazem parte do contorno rígido e com E, as demais células. Não sofreu qualquer alteração no programa paralelo.

Rotina INTERVALOS

Esta rotina divide o domínio de solução em painéis verticais de igual largura ou o mais próximo possível desta situação. Cada painel representará o domínio de solução de cada processador. O número desses painéis deve ser igual ao número de processadores disponíveis para a aplicação paralela. Rotina não utilizada no caso sequencial.

Rotina INFLOW

Esta rotina marca as células dos contornos de entrada e de saida e é a mesma para os casos sequencial e paralelo.

Rotina PART

Esta rotina gera as partículas, quando existe fluido no interior do domínio no instante de tempo inicial. No caso sequencial, gera todas as partículas necessárias, enquanto que no caso paralelo gera as partículas que estão no domínio de cada processador, quando for o caso. Os corpos de fluidos podem assumir uma forma retangular ou circular.

Rotina CORTE

Rotina que "quebra" e distribui os segmentos de retas ou curvas que definem o contorno rígido para os respectivos processos considerados na decomposição de domínio. Esta rotina existe somente no código paralelo.

Rotina BCOND

Esta rotina calcula as velocidades nas células que estão sobre o contorno rígido. É a mesma tanto no caso sequencial como no paralelo.

Rotina TANVEL

Calcula as velocidades tangenciais nos contornos de entrada e de saida. Esta rotina não sofreu qualquer alteração do caso sequencial para o paralelo.

Rotina TREE

Esta rotina faz parte do pacote PIM. Além de sua utilização na solução do sistema linear, utilizamos também essa rotina em nosso algoritmo. Ela contrói uma árvore binária balanceada de processos, que é, então, utilizada quando precisamos fazer alguma comunicação entre os processadores através do processo de redução.

Essa rotina foi utilizada em várias situações do programa paralelo, não fazendo parte do programa sequencial.

Rotina REFLAG

Nesta rotina é feita uma atualização, em cada passo de tempo, da matriz F, que armazena os tipos de células ("FULL", "SURFACE" ou "EMPTY") que compõem o domínio de solução.

Após a execução dessa rotina é necessária a comunicação entre os processadores vizinhos das colunas da matriz F que são vizinhas a algum processador. As alterações dessa rotina do caso sequencial para o paralelo são os controles das variáveis que devem ser comunicadas de um processador para outro e as comunicações que se fazem necessárias.

Rotina SUREMP

Esta rotina calcula as velocidades u e v sobre as células da superfície livre, células "SURFACE" que são contíguas a células "EMPTY" quando

$$\nabla.u = 0.$$

Após a execução dessa rotina é necessária a comunicação entre os processadores vizinhos dos novos valores dessas velocidades, porém apenas quando ocorrem células "SURFACE" nas vizinhanças dos processos.

No código paralelo é exigida a comunicação das velocidades atualizadas por essa rotina. Para não se comunicar toda a coluna de valores da matriz que armazena esses dados, utilizamos endereçamento indireto de modo a comunicar somente os valores que interessam.

Rotina TANOUT

Nessa rotina são calculadas as velocidades sobre as células "SURFACE" quando as condições sobre a tensão tangencial são satisfeitas.

Exige-se comunicação semelhante ao caso da rotina SUREMP.

Rotina PSEUDPRESS

Rotina que calcula a pressão sobre a superfície livre de acordo com a condição sobre a tensão normal.

Não sofreu alterações do caso sequencial para o caso paralelo. Após a execução dessa rotina, o caso paralelo exige a comunicação entre os processadores vizinhos, dos valores da pressão calculada recentemente.

Rotina DTSET

Esta rotina calcula o espaçamento no tempo em cada ciclo de cálculo. Uma vez que em seu cálculo envolve os valores máximos das velocidades u e v, e como esses máximos locais podem diferir do máximo global, é necessária comunicação, feita através do processo de redução, para que cada processador seja informado do menor valor do espaçamento no tempo calculado, afim de que as condições de estabilidade sejam satisfeitas de maneira global. A única diferença nessa rotina, entre os casos sequencial e o paralelo, é o processo de redução exigido pelo programa paralelo.

Rotina TILDEV

Esta rotina calcula as velocidades provisórias. Estas velocidades são calculadas por uma fórmula de discretização explícita. A rotina é a mesma em ambos os casos: paralelo e sequencial.

Rotina FINVEL

Rotina que faz os ajustes das velocidades finais ($u = \tilde{u} + \nabla\psi$).

Após a execução dessa rotina é exigida a comunicação das velocidades u e v nas vizinhanças dos processadores. Esta rotina não sofreu alterações do caso sequencial para o paralelo.

Rotina PACKIN

Esta rotina é responsável pelo movimento das partículas, pela eliminação das partículas que ultrapassam os contornos de saida e, também, pela geração de novas partículas nos contornos de entrada.

Ao movimentar uma partícula, que estava em um determinado processador, pode ocor-

rer dela ter migrado para um outro processador vizinho. Nesta situação essa partícula é transferida de um processador para outro. Nesse processo de movimentação das partículas é, também, realizado um ajuste nas velocidades u e v. No caso de uma partícula estar se transferindo de um processador para outro, esse ajuste é calculado em ambos os processadores, evitando dessa maneira comunicação para se atualizar as velocidades calculadas nas vizinhanças dos processadores. As partículas que serão transferidas de um processador para outro são armazenadas e transferidas todas de uma única vez em cada ciclo de cálculo, evitando, desta forma, um excesso de comunicação.

Quando um contorno de entrada (respectivamente, de saida) está dividido por mais de um processador, o processo de gerar (respectivamente, eliminar) partículas é executado parte em um processador, parte em outro, cada processador gerando (ou eliminando) somente a parte que lhe cabe.

A estrutura desta rotina é a mesma em ambos os casos: sequencial e paralelo. Entretanto, no caso paralelo há a necessidade de troca de informações entre os processadores para que seja realizada a atualização das partículas nos subdomínios envolvidos na decomposição.

A.2 A Estrutura do Código em Coordenadas Cilíndricas

Como pode ser visto no procedimento descrito na seção 5.2, a inclusão de coordenadas cilíndricas no código GENSMAC afetará os passos de 1 a 3, enquanto que o passo 4 permanece inalterado, visto que as equações para a movimentação das partículas não se alteram do caso bi-dimensional para o caso eixo-simétrico. Apenas quatro rotinas sofreram alterações. São elas: PSEUDPRESS, TILDEV, CGSSOLVER e SUREMP. Assim, a estrutura do programa é exatamente a mesma, sendo feitas somente as substituições das rotinas acima pelas correspondentes rotinas em coordenadas cilíndricas.

A rotina PSEUDPRESS, no caso bi-dimensional, que calcula a pressão $\tilde{p}(x, y, t)$ de acordo com as equações para a condição de tensão, foi adequada para calcular a pressão $\tilde{p}(r, z, t)$ conforme as condições de tensão apresentadas na seção 5.3.1 e passou a se chama PTILDE1AXYS.

A rotina TILDEV, que calcula as velocidades provisórias, passou a se chamar TILDEVAXYS e implementa as equações (5.10) e (5.11) apresentadas na secção 5.3.

A rotina CGSSOLVER, que prepara a matriz resultante da equação de Poisson para que o sistema linear possa ser resolvido pelo método dos gradientes conjugados, foi então adequada para preparar a matriz para a equação de Poisson discretizada (5.14) apresentada na secção 5.3 e passou a se chamar CGSSOLVERAXYS.

Finalmente, a rotina SUREMP, relativa às condições de contorno para $u(x, y, t)$, que calcula as velocidades sobre as faces das células E, foi reescrita de modo a implementar as equações descritas na secção 5.3.1 e passou a se chamar SUREMPAXYS.

As equações para calcular as velocidades tangenciais na superfície livre, bem como as equações para calcular as condições sobre o contorno rígido, não sofreram qualquer alteração.

As comunicações envolvidas nessas rotinas, para o código paralelo, são similares ao caso anterior.

A.3 A Implementação da Nova Técnica para a Representação da Superfície Livre

Com a introdução desta nova técnica as rotinas que sofreram alterações são as seguintes: PART, INFLOW, STOPINF, REFLAG, PACKIN.

A rotina PART que gera as partículas quando existe fluido no interior do domínio de solução no instante inicial de tempo, foi transformada em duas outras: SETBOX e SETBALL.

Rotina SETBOX: utilizada para gerar uma lista formando um corpo de fluido no tempo inicial $t = 0$, em formato retangular.

Rotina SETBALL: utilizada para gerar uma lista formando um corpo de fluido no tempo inicial $t = 0$, na forma circular.

A rotina INFLOW passou a se chamar SETINF nesta nova versão. Essas rotinas são utilizadas para marcar os contornos de entrada e de saida. As células marcadas com contorno de entrada (ou de saida) pela rotina INFLOW controlam a inserção (ou eliminação) de partículas que estão entrando ou saindo do domínio de solução, enquanto que as células marcadas pela rotina SETINF controlam a inserção ou (eliminação) de partículas apenas nas listas que representam a superfície livre. Há a necessidade de inserção de novos nós em uma determinada lista, quando existem contornos entrada e/ou quando os nós vizinhos estão distantes uns dos outros. Em contrapartida, há a necessidade de eliminação de nós de uma determinada lista quando existem contornos de saida e/ou quando os nós estão muito próximos uns dos outros, evitando, desta forma, uma acúmulo desnecessário de nós em uma mesma lista.

A rotina STOPINF passou a se chamar FREEINFLOW. Essa rotina é utilizada para encerrar a entrada de fluido no interior do domínio de solução após um determinado tempo

definido pelo usuário.

A rotina REFLAG, agora denominada REFLAGC, é utilizada para atualizar a matriz de células em cada passo de tempo. A rotina REFLAG utiliza todas as partículas virtuais para realizar essa tarefa, enquanto que a rotina REFLAGC utiliza apenas as listas orientadas que definem a superfície livre.

A rotina PACKIN passou a se chamar MOVEFREESURFACE. Na rotina PACKIN é realizado todo o processo de movimentação das partículas envolvidas no problema, enquanto que na rotina MOVEFREESURFACE são movimentadas apenas as partículas que representam a superfície livre.

Com a substituição da rotina PACKIN pela rotina MOVEFREESURFACE, houve a necessidade de algumas rotinas auxiliares. A seguir descrevemos cada uma delas:

Rotina INSERTNODECOMPONENT: utilizada para inserir um novo nó em uma determinada componente.

Rotina DELETENODECOMPONENT: utilizada para eliminar um nó em uma determinada componente.

Rotina DISPOSECOMPONENT: utilizada para eliminar uma componente.

Rotina FINDBEGINCOMPONENT: utilizada para procurar uma componente para ser ligada a uma outra, pelo seu início.

Rotina FINDENDCOMPONENT: utilizada para procurar uma componente para ser ligada a uma outra, pelo seu final.

Rotina GLUEBEGINCOMPONENT: utilizada para ligar o início de uma componente ao final de uma outra.

Rotina GLUEENDCOMPONENT: utilizada para ligar o final de uma componente ao início de uma outra.

Rotina CREATECOMPONENT: utilizada para gerar uma nova componente.

Rotina GLUETWOCOMPONENT: utilizada para ligar duas componentes. Dada uma componente, será ligada uma em seu início e uma outra em seu final.

Rotina SPLITBEGINLISTLEFT: utilizada para dividir uma componente em duas partes: a inicial, que será transferida para o processador da esquerda e a final, que permacerá no mesmo processador.

Rotina SPLITBEGINLISTRIGH: utilizada para dividir uma componente em duas partes: a inicial, que será transferida para o processador da direita e a final, que permacerá no mesmo processador.

Rotina SPLITENDLISTLEFT: utilizada para dividir uma componente em duas partes: a inicial e a final que será transferida para o processador da esquerda.

Rotina SPLITENDLISTRIGH: utilizada para dividir uma componente em duas partes: a inicial e a final que será transferida para o processador da direira.

Rotina SPLITMIDLELISTLEFT: utilizada para dividir uma componente em três partes: inicial, final e a parte do meio que será transferida para o processador da esquerda.

Rotina SPLITMIDLELISTRIGH: utilizada para dividir uma componente em três partes: inicial, final e a parte do meio, que será transferida para o processador da direita.