RAFAEL SENE DE LIMA

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE MALHAS ADAPTATIVAS BLOCO-ESTRUTURADAS PARA COMPUTAÇÃO PARALELA EM MECÂNICA DOS FLUIDOS

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA MECÂNICA

2012

RAFAEL SENE DE LIMA

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE MALHAS ADAPTATIVAS BLOCO-ESTRUTURADAS PARA COMPUTAÇÃO PARALELA EM MECÂNICA DOS FLUIDOS

Tese apresentada ao Programa de Pós-graduação em Engenharia Mecânica da Universidade Federal de Uberândia, como parte dos requisitos para a obtenção do título de **DOUTOR EM ENGENHARIA MECÂNICA**.

Área de concentração:Transferência de Calor e Mecânica dos Fluidos.

Orientador: Prof. Dr. Aristeu da Silveira Neto
Coorientador: Prof. Dr. Alexandre Megiorin Roma

Uberlândia - MG
2012

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU , MG, Brasil

L732d 2012

Lima, Rafael Sene de, 1983-

Desenvolvimento e implementação de malhas adaptativas bloco-estruturadas para computação paralela em mecânica dos fluidos / Rafael Sene de Lima. - 2012.

112 f. : il.

Orientador: Aristeu da Silveira Neto.

Coorientador: Alexandre Megiorin Roma.

Tese (doutorado) – Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Mecânica.

Inclui bibliografia.

1. Engenharia mecânica - Teses. 2. Dinâmica dos fluidos - Simulação por computador - Teses. 3. Escoamentos - Simulação por computador - Teses. I. Silveira Neto, Aristeu da, 1955- II. Roma, Alexandre Megiorin. III. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Engenharia Mecânica. IV. Título.

CDU: 621

RAFAEL SENE DE LIMA

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE MALHAS ADAPTATIVAS BLOCO-ESTRUTURADAS PARA COMPUTAÇÃO PARALELA EM MECÂNICA DOS FLUIDOS

Tese **Aprovada** pelo Programa de Pós-graduação em Engenharia Mecânica da Universidade Federal de Uberândia.
Área de concentração: Transferência de Calor e Mecânica dos Fluidos.

Banca examinadora:

Prof. Dr. Aristeu da Silveira Neto - Orientador (FEMEC-UFU)

Prof. Dr. Alexandre Megiorin Roma - Coorientador (IME-USP)

Prof. Dr. Emanuel Rocha Woiski (FEIS-UNESP)

Dr. Ricardo Serfaty (CENPES/PETROBRAS)

Prof. Dr. João Marcelo Vedovoto (FEMEC-UFU)

Prof. Dr. Marcus Antonio Viana Duarte (FEMEC-UFU)

Uberlândia, 28 de Setembro de 2012

Aos meus pais Donatílio e Liberti, ao meu irmão Robison e ao meu sobrinho Guilherme.

Agradecimentos

a Deus,
por ter me dado força e inspiração para realizar este trabalho;

aos meus pais Donatílio e Liberti e ao meu irmão Robison,
pelos ensinamentos, pelo carinho e confiança;

Ao Prof. Dr. Aristeu e ao Prof. Dr. Alexandre,
pela orientação, confiança, paciência e apoio no decorrer do trabalho;

ao Marcos e ao Tiago,
pela amizade incondicional e pelo companheirismo nos momentos felizes e nos momentos difíceis;

à Millena e ao Márcio,
pela cooperação explícita neste trabalho;

aos colegas do MFLab
pelo apoio e cooperação neste trabalho;

aos funcionários do MFLab
pela incessante luta para manter o laboratório funcionando;

à Universidade Federal de Uberlândia e à Faculdade de Engenharia Mecânica
pela oportunidade de realizar este Curso;

ao MFLab,
pelo espaço físico, pelo suporte computacional, pela cooperação e pela oportunidade de crescimento;

ao CNPq #140874/2008 − 9 *e à PETROBRAS*,
pelo financiamento deste trabalho.

LIMA, R. S., 2012, "Desenvolvimento e Implementação de Malhas Adaptativas Bloco-Estruturadas para Computação Paralela em Mecânica dos Fluidos". Tese de Doutorado, Universidade Federal de Uberlândia, Uberlândia, MG, Brasil.

A simulação numérica de escoamentos envolvendo geometrias complexas é fortemente limitada pela resolução da malha espacial. Na grande maioria dos escoamentos, há pequenas regiões do domínio onde o fluido se movimenta de forma complexa gerando gradientes elevados, enquanto que no restante do domínio o escoamento é relativamente "calmo". O Refinamento Adaptativo de Malhas (*Adaptive Mesh Refinement - AMR*), possibilita que o refinamento da malha espacial seja mais apurado em regiões específicas, enquanto que nas demais regiões o refinamento pode ser mais grosseiro. O presente trabalho consiste no desenvolvimento de uma metodologia de paralelização para a solução das equações de Navier-Stokes em malhas adaptativas bloco-estruturadas (*Structured Adaptive Mesh Refinement - SAMR*) utilizando a interface *MPI* (*Message Passing Interface*) e o método de bisseção por coordenadas *RCB* (*Recursive Coordinate Bisection*) para o balanço de carga. Implementações de métodos *SAMR* em processamento paralelo oferecem a possibilidade de simulações precisas de escoamentos de elevada complexidade. No entanto, apresentam desafios interessantes quanto à dinamicidade na alocação e distribuição dos dados e no balanceamento de carga. Cabe ressaltar que a eficiência total das aplicações envolvendo métodos *SAMR* em processamento paralelo é fortemente dependente da qualidade do particionamento dinâmico de domínio, efetuado em tempo de execução, para que se garanta os menores custos de comunicação e sincronização possíveis, além de uma boa distribuição da carga computacional. Neste trabalho, utilizou-se o esquema semi-implícito proposto por Ceniceros *et al.* (2010) para avanço temporal. Todas as implementações foram efetuadas como uma extensão do código "AMR3D", proposto por Nós (2007). A eficiência e a robustez do método proposto são verificadas por meio do método das soluções manufaturadas. As validações foram feitas por meio da simulação do escoamento em uma cavidade com tampa deslizante e de um jato incompressível.

Palavras-chave: Malhas adaptativas, computação paralela, equações de Navier-Stokes, balanço dinâmico de carga.

LIMA, R. S., 2012, "Development and Implementation of Block-Structured Adaptive Mesh Refinement for Parallel Computations in Fluid Mechanics". Doctor Thesis, Universidade Federal de Uberlândia, Uberlândia, MG, Brazil.

Abstract

The numerical simulation of fluid flow involving complex geometries is greatly limited by the required spatial grid resolution. These flows often contain small regions with complex motions, while the remaining flow is relatively smooth. Adaptive mesh refinement (AMR) enables the spatial grid to be refined in local regions that require finer grids to resolve the flow. This work describes an approach to parallelization of a structured adaptive mesh refinement (SAMR) algorithm. This type of methodology is based on locally refined grids superimposed on coarser grids to achieve the desired resolution in numerical simulations. Parallel implementations of SAMR methods offer the potential for accurate simulations of high complexity fluid flows. However, they present interesting challenges in dynamic resource allocation, data-distribution and load-balancing. The overall efficiency of parallel SAMR applications is limited by the ability to partition the underlying grid hierarchies at run-time to expose all inherent parallelism, minimize communication and synchronization overheads, and balance load. The methodology is based on a message passing interface model (*MPI*) using the recursive coordinate bisection (RCB) for domain partition. For this work, a semi-implicit projection method has been implemented to solve the incompressible Navier Stokes equations. All numerical implementations are an extension of a sequential Fortran 90 code, called "AMR3D", developed in the work of Nós (2007) .The efficiency and robustness of the applied methodology are verified via convergence analysis using the method of manufactured solutions. Validations were performed by simulating an incompressible jet flow and a lid driven cavity flow.

Key Words: Adaptive Mesh Refinement, Navier-Stokes Equations, parallel computing, dynamic load balancing.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de símbolos

d_0	Diâmetro do bocal (m)
e	Valor exato do erro
$\mathbf{f}$	Vetor força (N/m^2)
$\mathbf{g}$	Aceleração gravitacional (m/s^2)
l	Nível de refinamento
L	Comprimento da cavidade (m)
M_x	Número de células na direção x
M_y	Número de células na direção y
M_z	Número de células na direção z
nc	Número de células em um bloco ou *patch*
$ncref$	Número de células em um bloco ou *patch* selecionada para refinamento
p	Pressão (Pa)
q	Correção de pressão
r	Razão de refinamento
r_0	Raio do bocal (m)
R	Resíduo
Re	Número de Reynolds
S	Tensor deformação
t	tempo (s)

$\mathbf{u}$ Vetor velocidade (m/s)

V tensor Vorticidade

Letras Gregas

α_i Constantes do método Gear

β_i Constantes do método Gear

Δx Espaçamento da malha computacional na direção x (m)

Δy Espaçamento da malha computacional na direção y (m)

Δz Espaçamento da malha computacional na direção z (m)

μ Viscosidade dinâmica $(N \times s/m^2)$

ρ Massa específica, densidade (kg/m^3)

ϕ Variável escalar genérica

ξ_{min} Eficiencia mínima do bloco ou *patch* gerado

Índices

n Passo no tempo

T Transposto

$\tilde{}$ Variável aproximada

bot Nível físico de base

top Nível físico mais fino

Operadores Matemáticos

∂ Derivada parcial

∇ gradiente

∇^2 Laplaciano

$\mathcal{P}$ Operador prolongamento

$\mathcal{R}$ Operador restrição

Capítulo 1

Introdução

A modelagem matemática e a solução numérica de equações diferenciais, aliadas ao crescente desempenho das ferramentas computacionais disponíveis, têm possibilitado o estudo detalhado de fenômenos físicos com elevado grau de complexidade, fazendo da simulação numérica uma ferramenta comum na solução de problemas de grande porte em engenharia. Porém, a análise de fenômenos físicos cada vez mais complexos alimenta a busca do homem por ferramentas cada vez mais robustas e eficientes.

A simulação computacional de escoamentos complexos em mecânica dos fluidos envolve a solução de equações diferenciais parciais, as quais apresentam solução suave na maioria do domínio de cálculo, exceto em regiões onde há gradientes elevados, choque, camadas limite ou descontinuidades, onde a solução é mais difícil de ser obtida (BERGER; OLIGER, 1984). Entretanto, se a presença de malhas mais refinadas é necessária somente em regiões específicas do escoamento, a utilização de uma malha refinada em todo o domínio tornaria a simulação ineficiente e cara computacionalmente. Uma solução para tal problema consiste na utilização da metodologia de malhas adaptativas, proposta nos trabalhos de Berger e Oliger (1984), Berger e Colella (1989) e Berger e Rigoutsos (1991). Tal metodologia consiste em utilizar um refinamento mais apurado em regiões específicas do domínio, para a captura dos fenômenos locais, sendo que nas demais regiões utiliza-se uma malha mais grosseira. A Fig. 1.1 mostra a utilização de malhas adaptativas localmente refinadas para um escoamento ao redor de uma esfera. A figura mostra as componentes do campo de velocidade u, v e w para o escoamento a $Re = 100$ sobrepostas pela malha composta.

Uma outra gama de escoamentos que justificam o uso de malhas adaptativas são aqueles que ocorrem no interior ou no exterior de geometrias complexas. A representação da fluido-dinâmica do escoamento no interior de geometrias complexas, requer um refinamento apurado da malha numérica nas regiões próximas à parede, principalmente a

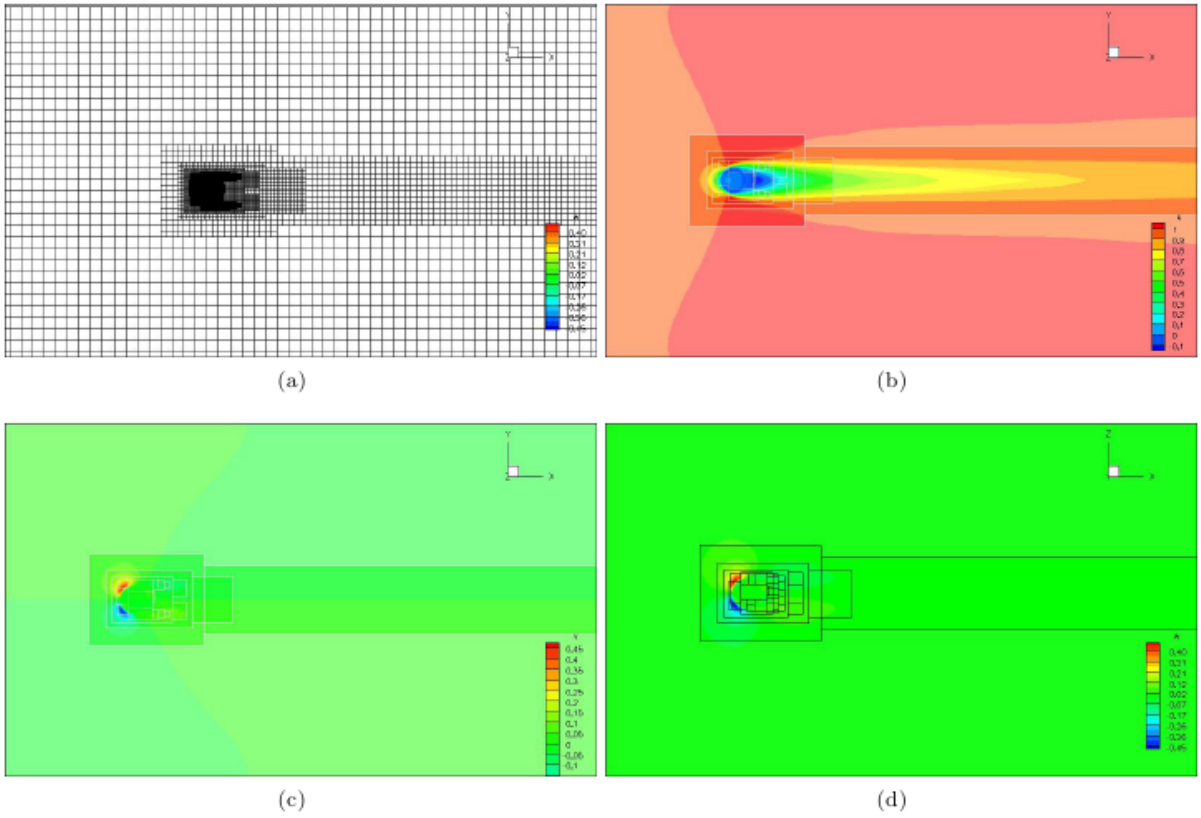
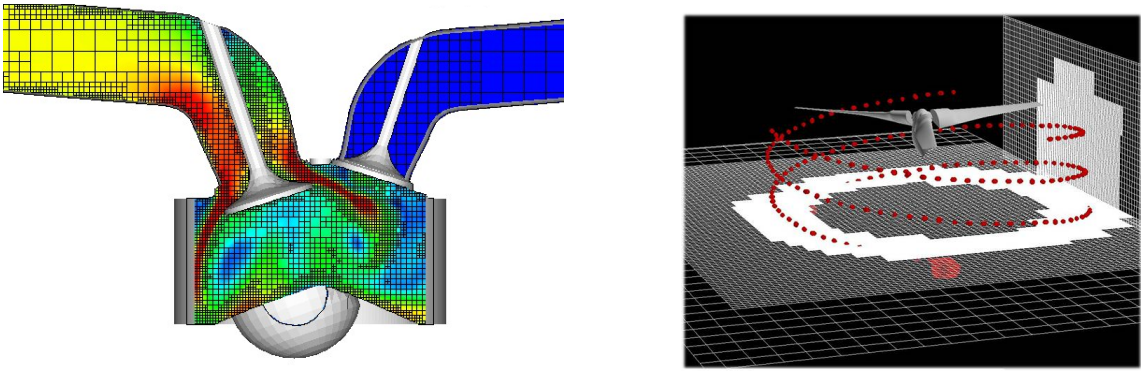


Figura 1.1: Campo de velocidade para escoamento ao redor de uma esfera. (a) Malha adaptativa refinada localmente; (b) velocidade u, (c) velocidade v; (d) velocidade w. (Neto *et al.* (2010))

números de Reynolds elevados. Porém, nas regiões mais afastadas tal refinamento não é necessário, pois o escoamento possui um comportamento mais brando, se comparado com as regiões próximas à parede.

A inserção de uma geometria qualquer no interior do domínio provoca perturbações no escoamento, gerando estruturas turbilhonares a jusante do escoamento. A simulação numérica de tais estruturas requer um refino apurado da malha numérica, tornando necessário tornando a simulação ainda mais complexa e cara computacionalmente. Para problemas em regime transiente, ainda existe a dificuldade de se detectar o local em que tal refinamento é necessário em cada instante de tempo. Exemplos de escoamento no interior e exterior de geometrias complexas são ilustrados pelas figuras 1.2(a) e 1.2(b). A Fig. 1.2(a) mostra a magnitude da velocidade em um plano de corte ao longo de uma das válvulas de admissão de um motor de combustão interna. A Fig. 1.2(b) mostra a utilização de refinamento adaptativo para a captura de vórtices na ponta das hélices de um motor V-22 isolado.

A metodologia de malhas adaptativas consiste na construção de malhas localmente refinadas nas regiões do problema em que é necessário um refinamento mais apurado, para



(a) (b)

Figura 1.2: Simulações numéricas de escoamentos utilizando malhas adaptativas: (a) Escoamento em um motor de combustão interna. (`http://convergecfd.com/applications/engine/sparkignited/`); (b) Ilustração do uso de malhas adaptativas para estudo de vortices formados na ponta da hélice de um motor V-22 (`https://computation.llnl.gov/casc/SAMRAI/SAMRAI_Applications.html`)

a captura de fenômenos locais, sendo que nas regiões mais estáveis utiliza-se uma malha mais grosseira. Existem várias técnicas para refinamento localizado, porém a escolha do método deve levar em conta a facilidade de lidar com a estrutura de dados, o tempo gasto para se percorrer toda a malha e custo de armazenamento dos dados. Segundo Villar (2007), malhas não estruturadas oferecem maior eficiência para refinamentos localizados, porém necessitam de uma complexa estrutura de dados para lidar com as informações necessárias para a descrição da malha. Partindo-se do princípio de que malhas cartesianas estruturadas são empregadas, no presente trabalho é utilizada a metodologia de malhas adaptativas bloco estruturadas, que consiste em um método particular para refinamento localizado, no qual a malha computacional é formada por uma hierarquia de blocos devidamente posicionados, separados em níveis de refinamento. Cada bloco é composto por uma malha cartesiana uniforme e o tamanho de cada célula computacional não varia em um mesmo nível.

O uso de malhas adaptativas visa não somente à redução do custo computacional, mas também a possibilidade de se resolver problemas em que a solução das equações governantes é proibitivamente cara usando-se malha uniforme. É importante observar que há problemas cuja solução demanda um alto custo, em termos de memória computacional (RAM), mesmo utilizando-se malhas adaptativas, como é o caso do uso de fluidodinâmica computacional para simulação de escoamentos bifásicos, o qual requer uma resolução muito apurada nas regiões onde há interface entre as fases. Escoamentos bifásicos no interior de dutos necessitam de um cuidado ainda maior devido à presença de parede, assim, a resolução da malha envolvida na solução destes problemas torna-o proibitivo computacionalmente, mesmo com malhas refinadas localmente. Uma solução para este

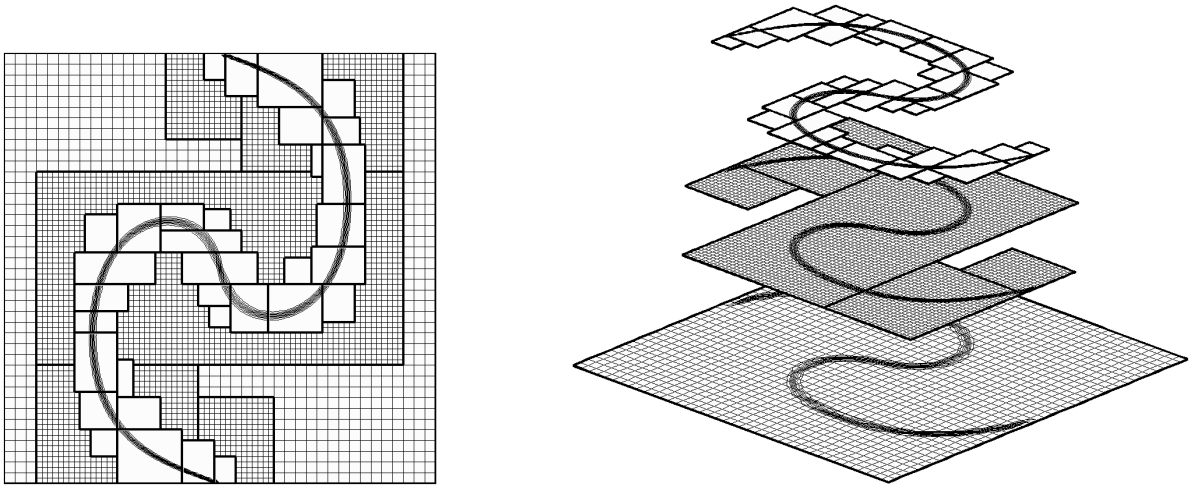


Figura 1.3: Exemplo de malha composta com três níveis de refinamento. (`http://math.boisestate.edu/~calhoun/www_personal/research/amr_software/index.html`)

problema seria a paralelização da ferramenta numérica utilizada na solução do problema.

A implementação em processamento sequencial de algoritmos envolvendo malhas adaptativas apresenta um grau de dificuldade elevado, principalmente devido à complexidade e dinamicidade da estrutura de dados. O desenvolvimento e implementação de algoritmos para paralelização de códigos envolvendo malhas adaptativas representa um desafio ainda maior, pois além da necessidade de se transferir dados entre regiões com diferentes resoluções de malha, a partição de domínio, o balanceamento da carga e distribuição dos dados devem ser feitos dinamicamente e com menor custo de comunicação possível, o que requer o uso de algoritmos altamente sofisticados.

No presente trabalho, propõe-se desenvolver uma metodologia de paralelização para malhas bloco estruturadas, utilizando o padrão MPI para comunicação de dados em paralelo. O balanço de carga e o particionamento de domínio são implementados utilizando-se o pacote Zoltan Devine *et al.* (2002) desenvolvido no laboratório Sandia (`http://www.cs.sandia.gov/Zoltan/`). Na solução das equações de Navier-Stokes, utiliza-se o mesmo método de projeção de Ceniceros, Nós e Roma (2010). Na discretização temporal das equações diferenciais parciais, emprega-se a mesma estratégia semi-implícita utilizada nos trabalhos de Villar (2007) e Ceniceros *et al.* (2010). Todos os desenvolvimento e implementações foram realizados como uma extensão do código "AMR3D", proposto por Nós (2007), o qual apresenta simulações tridimensionais utilizando malhas adaptativas e um modelo de campo de fase para escoamentos incompressíveis.

1.1 Organização do trabalho

A presente tese é estruturada a seguir. No Capítulo 1, são apresentadas motivações para o uso de malhas adaptativas em processamento sequencial e paralelo, assim como as dificuldades encontradas na utilização de malhas adaptativas em paralelo. No Capítulo 2, é feita uma revisão da literatura acerca dos temas relevantes para o desenvolvimento deste trabalho. São abordados temas envolvendo refinamento localizado em processamento sequencial e paralelo. Também é feita uma breve descrição do método Multinível-*Mulgrid*. A modelagem matemática e a metodologia numérica, incluindo método de projeção, discretização temporal e todo o processo de paralelização, são descritas no Capítulo 3. Os resultados obtidos por meio de simulação numérica são encontrados no Capítulo 4. Neste, avalia-se a qualidade da distribuição da carga computacional entre os processos e verifica-se e valida-se a metodologia proposta. No Capítulo 5, são feitas considerações finais sobre o trabalho, citanto as principais contribuições e perspectivas para trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

2.1 Refinamento Adaptativo

Técnicas de refinamento adaptativo podem ser utilizados tanto em malhas estruturadas como em malhas não estruturadas. A utilização de malhas estruturadas oferece vantagens sobre malhas não estruturadas, como por exemplo o fato de apresentarem uma simples implementação e serem convenientes para o uso de diferenças finitas ou volumes finitos, além de requerer menor capacidade de armazenamento. Além disso, a paralelização em malhas estruturadas é de mais fácil implementação, pois em malhas não estruturadas são necessários algoritmos sofisticados para particionamento de domínio.

Uma grande desvantagem na utilização de malhas estruturadas consiste na falta de flexibilidade em lidar com domínios mais complexos. Malhas não estruturadas se ajustam com grande facilidade a geometrias complexas, porém com um alto custo de armazenamento, pois é necessário armazenar dados como conetividade e aposição geométrica da célula. Dentre muitos trabalhos envolvendo malhas não estruturadas refinadas localmente encontra-se o de Walsh e Zingg (2001) sobre aerodinâmica e o de Huang e Tafti (2004) sobre sistemas dinâmicos não lineares.

Quando se resolve um problema em que a região de interesse é muito pequena em relação à área total de cálculo, é necessária a utilização de células com dimensão pequena o suficiente para a captura de fenômenos físicos locais. Porém a aplicação de malhas refinadas em todo o domínio de cálculo acarretaria em um aumento proibitivo do custo computacional envolvido na solução das equações diferenciais. Surge então a necessidade de se utilizar um refinamento localizado, afim de obter uma malha refinada somente na região de interesse.

Em malhas estruturadas, o refinamento pode ser célula a célula (*quad-tree/oct-tree*)

ou bloco-estruturado. No refinamento célula a célula, estas são marcadas e refinadas individualmente. Este método é eficiente, pois nenhuma célula é refinada desnecessariamente, porém é necessária uma complicada estrutura de dados envolvendo árvores binárias, quaternárias ou octonárias para uma, duas ou três dimensões respectivamente. Popinet (2003) propôs uma metodologia para solução das equações de Euler incompressíveis para geometrias complexas, utilizando malha *quad-tree*.

Em contraste, no presente trabalho utiliza-se uma técnica para refinamento localizado baseada na utilização de blocos (ou *patches*) retangulares contendo malhas estruturadas. Tais blocos são formados por pontos em que o erro da solução da malha mais grossa é elevado, devido a fenômenos locais. Esta técnica foi proposta por Berger e Oliger (1984) para a solução numérica de equações diferenciais parciais hiperbólicas. Berger e Colella (1989) estenderam este método para estudo de ondas de choque, obtendo ganho considerável em desempenho computacional. Esta técnica foi estendida para equações de Euler e Navier-Stokes incompressíveis por Almgren *et al.* (1993) e Almgren *et al.* (1994) respectivamente, porém somente para densidade constante. No trabalho de Almgren *et al.* (1998) foi proposto um método de projeção para solução das equações de Navier-Stokes para viscosidade constante e densidade variável. Roma, Peskin e Berger (1999) estendem a técnica de refinamento adaptativo ao redor de geometrias complexas usando o Método da Fronteira Imersa. Griffith *et al.* (2007) propuseram um método de segunda ordem para fronteira imersa em malhas adaptativas bloco-estruturadas, aplicando tal metodologia na simulação computacional do escoamento de sangue em um coração humano.

Seguindo esta mesma linha de pesquisa, Sussman *et al.* (1999) utilizaram o método *level-set* para a simulação de escoamentos bifásicos. Neste trabalho foi utilizado um refinamento localizado para capturar o movimento de bolhas ascendentes. Recentemente, Villar (2007) e Ceniceros *et al.* (2010) fizeram um estudo detalhado sobre escoamentos bifásicos bidimensionais utilizando o Método da Fronteira Imersa e malhas adaptativas. Nós (2007) e Ceniceros, Nós e Roma (2010) empregaram modelos de campo de fase e malhas adaptativas na simulação de escoamentos bifásicos tridimensionais.

2.2 Refinamento Adaptativo em Paralelo

A utilização eficiente de computadores em paralelo depende basicamente de dois objetivos concorrentes. O primeiro consiste em manter os processadores ocupados com trabalho útil e o segundo em minimizar a comunicação entre os processos. Para alguns problemas em computação científica, tais objetivos são alcançados com uma simples atribuição de tarefas que não se altera com o decorrer do tempo. Entretanto, a solução da

maioria dos problemas em engenharia requer ferramentas computacionais capazes de se adaptar às variações, com o tempo, do comportamento físico de cada problema. Para tais aplicações, é possível obter um bom desempenho em paralelo somente se a carga de trabalho for distribuída entre os processos de uma forma dinâmica.

Devido ao comportamento dinâmico de malhas adaptativas é necessária a implementação algoritmos para redistribuição dinâmica da carga computacional. A distribuição da carga em malhas bloco estruturadas pode ser feita de três formas: por partição de domínio (*domain based*), distribuição dos blocos (*patch based*) ou de forma híbrida. Nos métodos de partição de domínio somente o nível de base é particionado. O balanço de carga é alcançado projetando-se a carga dos níveis mais finos no nível de base. Dentre os trabalhos encontrados, o de Berger e Bokhari (1987) foi um dos primeiros a propor um método de partição de domínio para malhas bloco estruturadas. O método consistia em decompor o domínio em retângulos, utilizando decomposição binária, de modo que a carga computacional fosse bem distribuída. Entretanto o particionamento era estático, o que limitava drasticamente o desempenho em paralelo.

Parashar e Browne (1996) foram um dos primeiros a implementar uma técnica para partição dinâmica de domínio com balanço de carga em malhas adaptativas. Em seu trabalho é utilizada a técnica de *space filling curves*,Sagan (1994), para executar o balanço de carga. Seguindo esta linha de pesquisa, Chandra *et al.* (2001) afirmam que a escolha do método de particionamento depende do problema a ser resolvido e da arquitetura dos computadores a serem utilizados em paralelo. Em seu trabalho é proposto um método para escolha dinâmica do particionador de domínio. Outros trabalho importantes nesta linha de pesquisa são os de Steensland e Ray (2005) e Rantakokko (1998). Uma implementação deste método é encontrada no pacote GrACE (*Grid Adaptive Computational Engine*- `http://nsfcac.rutgers.edu/TASSL/Projects/GrACE/`), Parashar *et al.* (1997).

Para métodos de balanço de carga por distribuição de blocos, ou *patch based*, o balanço de carga é efetuado distribuindo-se individualmente os blocos entre os processos. Teoricamente este método retorna um balaço de carga perfeito, porém na prática ocorrem limitações devido aos tamanhos dos blocos (JOHANSSON; VAKILI, 2008). Thuné (1997) foi um dos pioneiros no estudo de métodos *patch based*. Em seu trabalho foi feita uma comparação entre métodos *patch based* e *domain based*, apontando algumas desvantagens de métodos *domain based*. Uma das desvantagens era o fato de métodos *domain based* não levarem em conta a comunicação entre os blocos no balanço de carga, pois em métodos *domain based* somente o nível de base é particionado. Seguindo esta linha de pesquisa situam-se os trabalhos de Rantakokko (2000), Lan, Taylor e Bryan (2002) e Balsara e Norton (2001). Este método de balanço de carga está implementado no *framework* SAMRAI *Structured Adaptive Mesh Refinement Application Infrastructure* (Wissink *et al.* (2001) e

Wissink, Hysom e Hornung (2003).

Métodos *patch based* e *domain based* se ajustam muito bem para configurações mais simples de malha, sem muitos níveis de refinamento (BHAVSAR; SHEE; PARASHAR, 1999). Infelizmente a aplicação de ambos os métodos pode se tornar inviável para hierarquias de malha complexas e com vários níveis de refinamento (BHAVSAR; SHEE; PARASHAR, 1999). Uma solução é a combinação de ambas as estratégias, formando um particionamento híbrido (JOHANSSON, 2004). Este método é composto por duas etapas. Na primeira etapa, utiliza-se o método *domain based* para gerar partições grosseiras, que devem ser distribuídas em grupos de processos. A segunda etapa usa uma combinação das técnicas *patch based* e *domain based* para otimizar a distribuição das partições grosseiras entre os processos. Nesta linha de pesquisa , situa-se o método *Nature+Fable* (*Natural Regions + Fractional Blocking and bi-level partitioning*), proposto por Steensland (2002). Este método é composto por uma família de métodos híbridos, oferecendo uma grande variedade de opções para particionamento de domínio em malhas bloco estruturadas.

Apesar da disponibilidade de métodos altamente sofisticados disponíveis na literatura, a complexidade de implementação e a necessidade de uma estrutura de dados altamente sofisticada, pode tornar a utilização de tais métodos menos atrativa. Métodos *patch based* não serão utilizados, pois demandam altos custos de comunicação em malhas com um elevado número de blocos (JOHANSSON; VAKILI, 2008). No presente trabalho, propõe-se uma técnica para paralelização com balanço dinâmico de carga utilizando um método *domain based* denominado *RCB* (*Recursive Coordinate Bisection*), proposto por Berger e Bokhari (1987) como um algoritmo para balanço de carga estático. Uma implementação do algoritmo *RCB* é encontrada no pacote *Zoltan* Devine *et al.* (2002), o qual é utilizado no presente trabalho.

2.3 Jatos

Jatos são formados pela injeção de quantidade de movimento em um bocal ou orifício, e descarga em um ambiente circundante, sem a presença de paredes. A Fig. 2.1 ilustra o processo de transição em um jato em diferentes fases: (1) bocal convergente; (2) núcleo de escoamento potencial; (3) primeiras instabilidades; (4) toroide de alta concentração de vorticidade ou instabilidades de Kelvin-Helmholtz; (5) geração de estruturas turbilhonares toroidais tridimensionais e emparelhamento; (6) oscilações tridimensionais sobre as estruturas toroidais; (7) degeneração em turbulência tridimensional; (8) reorganização da turbulência em estruturas coerentes compostas de múltiplas escalas.

A Figura 2.2 mostra esquematicamente as formas características dos perfis médios de

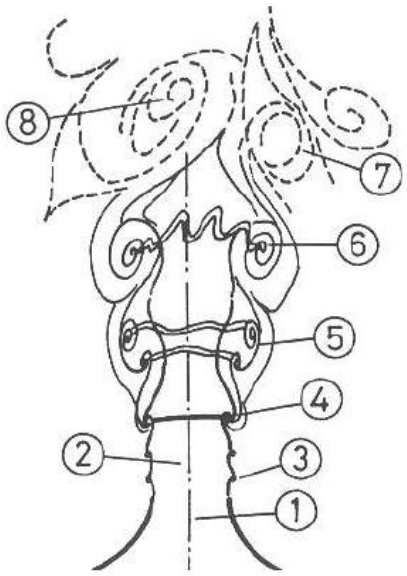


Figura 2.1: Esquema ilustrativo das estruturas turbilhonares encontradas na transição a turbulência de um jato (SILVEIRA-NETO, 2002).

velocidades de um jato em função da distância sobre a linha de simetria em relação ao bocal de saída. Inicialmente o jato se distribui como um perfil aproximadamente plano com velocidade de saída do bocal. O escoamento potencial desaparece rapidamente, a uma distância de cerca de quatro a sete diâmetros do bocal, onde o perfil de velocidade perde sua forma plana. A medida que se avança sobre a linha central do jato, o escoamento começa a se desenvolver dentro de uma forma gaussiana característica. Finalmente, a cerca de 20 diâmetros da saída do bocal, o perfil de velocidade atinge e mantém a forma autossimilar dependendo se o jato é plano ou axi-simétrico.

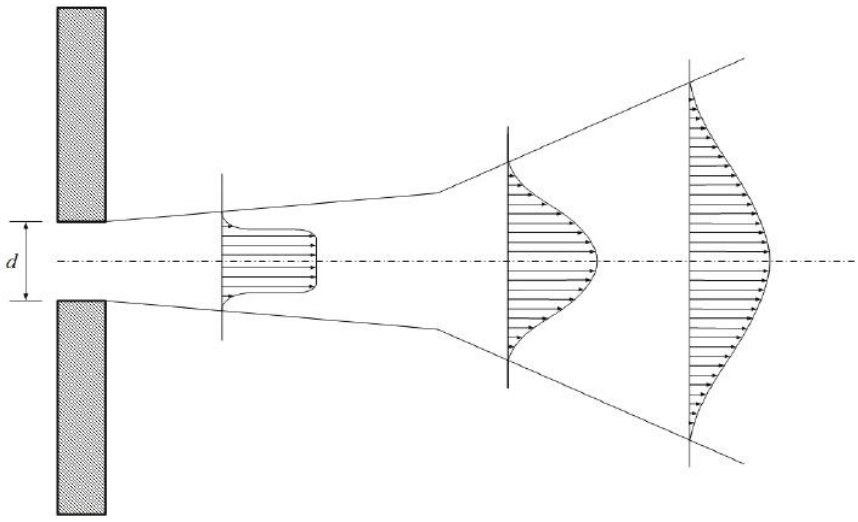


Figura 2.2: Esquema bidimensional contendo as formas características dos perfis médios de velocidades de um jato em função da distância sobre a linha de simetria em relação ao bocal de saída (MOREIRA, 2011).

É interessante destacar a influência do número de Reynolds neste tipo de escoamento. Com a variação deste parâmetro, alguns efeitos são notados nas estruturas coerentes do escoamento e no número de pares de vórtices. Aumentando o número de Reynolds, acelera-se mecanismo de transição à turbulência e torna-se mais difícil a identificação das estruturas coerentes, pois há uma intensificação do processo de fragmentação dos vórtices. Morris (1976) estudou a influência do número de Reynolds em instabilidades características e para perfis de velocidade pela análise da teoria de estabilidade linear. A influência da viscosidade é um parâmetro importante apenas para um números de Reynolds menores que $Re_d = 100$.

As primeiras investigações sobre jatos incluem os trabalhos de R. (1943), CORRSIN e UBEROI (1950a) e CORRSIN e UBEROI (1950b). As falhas das intensidades turbulentas em atingir a autossimilaridades nestes primeiros trabalhos motivaram os esforços de Wygnanski e Fiedler (1969). O trabalho destes autores constituiu um estudo completo da região de autossimilaridade do jato circular e tornou-se a referência para a descrição quantitativa para perfis de velocidade média e tensões turbulentas.

STANLEY S. A.; SARKAR (1999) estudaram o efeito do bocal de saída do jato sobre a evolução do perfil central da direção do escoamento. Eles estudaram a evolução do perfil na linha central do jato na direção do escoamento e a intensidade de flutuação na camada cisalhante do bocal, em regimes laminar e turbulento. Eles encontraram que, usando uma camada limite turbulenta, a evolução é insensível às variações do bocal de saída. Já com a camada limite laminar, há uma forte dependência do tipo de bocal.

STANLEY S. A.; SARKAR (1999) estudaram a influência de características do bocal no desenvolvimento do jato. Usando Simulação Numérica Direta, ou DNS (*Direct Numerical Simulation*), STANLEY S. A.; SARKAR (1999) analisaram os efeitos da intensidade de flutuações e da espessura da camada cisalhante no desenvolvimento do jato. Especial atenção é dada à taxa de espalhamento e à análise do decaimento da velocidade média na linha de centro do jato. A espessura da camada cisalhante tem uma forte influência sobre o aparecimento das estruturas cisalhantes que se formam próximo ao bocal. A medida que a espessura de cisalhamento diminui, a primeiras estruturas surgem mais próximas ao bocal. Uma estreita espessura de cisalhamento diminuiu o núcleo potencial, medido sobre o perfil de velocidade na linha central do jato.

Moreira (2011) desenvolveu uma metodologia que permite a simulação de um jato em desenvolvimento espacial utilizando o método Pseudo Espectral de Fourier, baseada no trabalho de Mariano (2011), acoplada ao método de fronteira imersa, proposto por LIMA-E-SILVA, SILVEIRA-NETO e DAMASCENO (2003). A principal característica do método Pseudo Espectral de Fourier é a alta precisão associada a um custo computacional relativamente pequeno, pois a solução da equação de Poisson para o acopla-

mento pressão-velocidade não é mais necessário (VEDOVOTO, 2011). Como o método Pseudo Espectral de Fourier é aplicado somente a problemas com condições de contorno periódicas, foi necessário o acoplamento deste método ao método de Fronteira Imersa, para tornar possível sua aplicação a problemas mais complexos.

2.4 Método *Multigrid*

A resolução de problemas de mecânica dos Fluidos e transferência de calor através de métodos numéricos acarreta um custo computacional elevado e muitas vezes proibitivo devido ao grande número de equações a serem resolvidas. Uma alternativa para diminuir o tempo computacional investido na simulação numérica destes problemas seria o uso de Métodos *Multigrid* (MG), que aceleram consideravelmente a solução numérica das equações diferenciais parciais envolvidas na modelagem matemática destes problemas (MALISKA, 2004), (MCCORMICK, 1987), (TROTTENBERG; SCHULLER, 2001). Para Hirsch (1988) e Tannehill, Anderson e Pletcher (1997), este método é considerado um dos mais eficientes e gerais desenvolvidos nos últimos anos.

Na década de 60 Fedorenko (1964) desenvolveu o primeiro método *Multigrid* para solução de uma equação de Poisson. Desde então alguns matemáticos, como por exemplo Bakhvalov (1966), estenderam a ideia de Fedorenko para problemas elípticos de valor de contorno com coeficientes vairiáveis. Entretanto, a eficiência do método *Multigrid* foi analisada somentoe nos trabalhos de Hackbusch (1977) e Brandt (1977), o qual introduziu os esquemas específicos *CS* (*Correction Storage*) e *FAS* (*Full Approximation Storage*). Segundo Brandt, para problemas lineares, recomenda-se o uso do *CS*, enquanto que o *FAS* é mais adequado para situações não lineares. A vantagem do algoritmo *Multigrid CS* frente ao *FAS* é que, na passagem de uma malha fia para uma malha mais grossa, requer-se apenas a manipulação dos resíduos das equações na malha fina, não se requerendo também a manipulação das grandezas. Computacionalmente, a restrição dos resíduos é bem mais simples do que a restrição das grandezas e o algoritmo então torna-se de fácil implementação (RABI, 1998). No presente trabalho o modo *CS* é adotado.

No esquema *CS* trabalha-se somente com o resíduo nas malhas grossas, onde resolve-se a equação residual. No esquema *FAS* trabalha-se com com aproximações completas da solução discreta na malha grossa, e não somente com o resíduo. Segundo Brandt, cada ciclo do esquema *FAS* era mais caro computacionalmente se comparado ao esquema *CS* devido à transferência de informações para as malhas mais grosseiras.

Vários trabalhos têm identificado as principais vantagens do método *Multigrid*, por meio de análises teóricas ou numéricas. Hirsch (1988) e Ferziger e Peric (2001) afirmaram

que sua propriedade mais importante é que o número de iterações na malha mais fina, necessária para alcançar a convergência, é independente do número de pontos da malha. Nessa mesma linha de raciocínio, Wesseling (1984), Hortmann, Perić e Scheuerer (1990) e Pavlov et al. (2001) afirmaram que o tempo de CPU necessário para obter a solução de um problema com o uso do *Multigrid* é diretamente proporcional ao número de variáveis.

Métodos *Multigrid* estão entre os métodos numéricos mais rápidos para resolver sistemas esparsos de equações lineares. Mas sua eficiência paralela é comprometida pela razão entre computação e comunicação nas malhas mais grossas (HÜLSEMANN *et al.*,). Nos níveis mais grossos, ou seja, com um número menor de elementos por malha, a comunicação pode inclusive dominar o tempo total de execução e, consequentemente, comprometer completamente a eficiência paralela do código. Além disso, a partir de um certo nível de refinamento, pode ocorrer do número de elementos da malha ser menor que o número de processos, tornando inviável o particionamento.

Devido aos altos custos de comunicação envolvidos na paralelização do método *Multigrid*. Trottenberg e Schuller (2001) propuseram algumas modificações nos relaxadores para evitar a sobrecarga de comunicação via rede. De acordo com Trottenberg e Schuller (2001), algumas variações do método *Multigrid* foram apresentadas por vários autores, em particular nota-se algumas variantes do método *Multigrid* propostas por Bramble, Pasciak e Xu (1990) e Greenbaum (1986) e o método *Multigrid* paralelo superconvergente (*parallel superconvergent multigrid method*) proposto por Foerster, Stüben e Trottenberg (1980). No presente trabalho, usa-se uma estratégia de paralelização baseada no particionado físico do domínio computacional, onde o métdodo *Multigrid* é executado separadamente em cada processador, fazendo-se a comunicação de dados via *MPI* quando for necessário.

Capítulo 3

Metodologia

Neste Capítulo apresentam-se os detalhes referentes ao desenvolvimento de uma metodologia de paralelização para a solução numérica das equações de Navier-Stokes para escoamentos incompressíveis utilizando malhas adaptativas refinadas localmente. Serão abordadas as discretizações temporal semi-implícita e espacial, por diferenças finitas, assim como a metodologia de passos fracionados utilizada no tratamento do acoplamento pressão-velocidade.

Quanto à metodologia de paralelização proposta, inicialmente é feita uma descrição do método empregado para particionamento domínio com balanço de carga. Em seguida, como toda transferência de dados entre os blocos de malhas é feita por meio de células fantasmas, faz-se uma descrição detalhada do procedimento de paralelização das mesmas. Por fim, toda a metodologia empregada na paralelização do método *Multigrid* utilizado no presente trabalho é descrita neste Capítulo.

3.1 Equações de Navier-Stokes

A modelagem de escoamentos é feita por meio das equações de Navier-Stokes e pela equação da continuidade, as quais são compostas por um sistema acoplado de equações diferenciais parciais não lineares. Para escoamentos em regime transiente, adiabáticos e incompressíveis, as equações de Navier-Stokes e da continuidade são dadas respectivamente por,

$$\begin{aligned} \rho\big(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\big) &= -\nabla p + \nabla\cdot\big[\mu\big(\nabla\mathbf{u} + \nabla\mathbf{u}^T\big)\big] + \rho\mathbf{g}, \\ \frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) &= 0, \end{aligned} \tag{3.1}$$

nas quais ρ e μ representam respectivamente a massa específica e a viscosidade dinâmica, propriedades físicas do fluido. As características do escoamento são dadas por: p campo de pressão, $\mathbf{u} = (u, v, w)$ vetor velocidade e $\mathbf{g}$ o vetor aceleração gravitacional.

3.2 Discretização temporal

Vários métodos numéricos têm sido propostos para resolver modelos dados por equações diferenciais parciais discretizadas no tempo e no espaço. Para a discretização temporal das equações diferenciais parciais, emprega-se a mesma estratégia semi-implícita utilizada por Villar (2007) e Ceniceros *et al.* (2010), baseada no trabalho de Badalassi, Ceniceros e Banerjee (2003). Para ilustrar a ideia de sua discretização temporal, Ceniceros *et al.* (2010) parte de uma equação difusiva com coeficiente variável, dada por

$$\frac{\partial \phi}{\partial t} = \nabla \cdot (\chi \nabla \phi), \quad \chi > 0. \tag{3.2}$$

A equação 3.2 pode ser reescrita como

$$\frac{\partial \phi}{\partial t} = C_\chi \nabla^2 \phi + h(\phi), \tag{3.3}$$

na qual C_χ é uma constante no espaço e no tempo e

$$h(\phi) = \nabla \cdot (\chi \nabla \phi) - C_\chi \nabla^2 \phi. \tag{3.4}$$

Na discretização temporal da Eq. 3.3, Ceniceros *et al.* (2010) parte do método de Gear tratando o termo $C_\chi \nabla^2 \phi$ implicitamente e o termo $h(\phi)$ explicitamente. O método de Gear para uma dada função ϕ, é dado por

$$\frac{\alpha_2 \phi^{n+1} + \alpha_1 \phi^n + \alpha_0 \phi^{n-1}}{\Delta t} = f(x^{n+1}, y^{n+1}, z^{n+1}), \tag{3.5}$$

na qual o índice $n+1$ define tempo atual e os índices n e $n-1$ definem os tempos precedentes. Partindo da Eq. 3.5, a discretização temporal da Eq. 3.3 é dada por

$$\frac{\alpha_2 \phi^{n+1} + \alpha_1 \phi^n + \alpha_0 \phi^{n-1}}{\Delta t} = \lambda \nabla^2 \phi^{n+1} + \beta_1 g_1(\phi^n) + \beta_0 g_0(\phi^{n-1}). \tag{3.6}$$

Os índices 0, 1 e 2 referem-se aos tempos $n-1$, n, $n+1$, respectivamente. Segundo

Ceniceros *et al.* (2010), os coeficientes α e β são dados por

$$\alpha_0 = \frac{\Delta t^2}{\Delta t_0 \Delta t_1}, \tag{3.7}$$

$$\alpha_1 = -\frac{\Delta t_1}{\Delta t_0}, \tag{3.8}$$

$$\alpha_2 = \frac{\Delta t_0 + 2\Delta t}{\Delta t_1}, \tag{3.9}$$

$$\beta_0 = -\frac{\Delta t}{\Delta t_0}, \tag{3.10}$$

$$\beta_1 = \frac{\Delta t_1}{\Delta t_0}, \tag{3.11}$$

no qual $\Delta t = t^{n+1} - t^n$, $\Delta t_0 = t^n - t^{n-1}$ e $\Delta t_1 = \Delta t_0 + \Delta t$.

Segundo Villar (2007) e Ceniceros *et al.* (2010) ,ao aplicar-se o mesmo método da Eq. 3.6 na Eq.3.1, tem-se a discretização temporal de segunda ordem para as equações de Navier-Stokes, tal que

$$\begin{aligned} &\frac{\rho^{n+1}}{\Delta t}\left(\alpha_2 u^{n+1} + \alpha_1 u^n + \alpha_0 u^{n-1}\right) = \\ &\quad \lambda \nabla^2 u^{n+1} + \beta_1 g_1(u^n, \mu^n) + \beta_0 g_0(u^{n-1}, \mu^{n-1}) - \nabla p^{n+1} + \rho^{n+1} g, \end{aligned} \tag{3.12}$$

$$\nabla \cdot (\rho \mathbf{u}) = 0, \tag{3.13}$$

onde

$$\mathbf{g} = -\lambda \nabla^2 \mathbf{u} + \nabla \cdot \left[\mu\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)\right] - \mathbf{u} \cdot \nabla \mathbf{u} \tag{3.14}$$

$$\lambda = C_\lambda \parallel \mu \parallel_\infty . \tag{3.15}$$

Dadas as condições iniciais para as velocidades e propriedades físicas do fluido no primeiro passo de tempo, é necessário calculá-las no segundo passo de tempo. Como o método de Gear trabalha com três passos no tempo, recorre-se ao método de Euler para resolver as equações de Navier-Stokes no segundo passo de tempo. Para que isto seja possível, considera-se $\alpha_2 = 1$, $\alpha_1 = -1$, $\alpha_0 = 0$, $\beta_1 = 1$ e $\beta_0 = 0$. A partir do terceiro passo no tempo, o método de Gear é aplicado normalmente. No presente trabalho, adota-se $C_\lambda = 2$. Esta escolha é baseada no trabalho de Ceniceros *et al.* (2010), os quais foram guiados por suas próprias experimentações numéricas e pelo trabalho de Xu e Tang (2006).

3.3 Acoplamento pressão - velocidade

Métodos de projeção tiveram sua origem nos trabalhos de Chorin (1968) e Temam (1968). Nesse método, velocidade e pressão são determinadas em dois passos. No primeiro, um campo de velocidades auxiliar $\tilde{\mathbf{u}}$ é calculado com a equação da quantidade de movimento, desprezando-se a condição de incompressibilidade. No segundo, o campo de velocidades auxiliar $\tilde{\mathbf{u}}$ é projetado no espaço dos campos vetoriais com divergente nulo para calcular a pressão ou a correção de pressão, o que possibilita a atualização do campo de velocidades. Segundo Villar (2007), a projeção que define o segundo passo é fundamentada no teorema de decomposição de Helmholtz-Hodge, cuja demonstração pode ser vista em Chorin e Marsden (1993), o qual afirma que qualquer campo vetorial $\mathbf{w}$, em um espaço D com fronteira suave ∂D, pode ser decomposto em

$$\mathbf{w} = \mathbf{u} + \gamma \nabla q, \tag{3.16}$$

na qual $\mathbf{u}$ tem divergente nulo e é paralelo a ∂D, ou seja, $\mathbf{u} \cdot \mathbf{n} = 0$ em ∂D, γ é uma constante e q um campo escalar qualquer. Existem várias discretizações para o método de projeção, as quais diferem na forma de calcular o campo de velocidades auxiliar $\tilde{\mathbf{u}}$ e o passo de projeção. No presente trabalho adota-se a estratégia semi-implícita utilizada nos trabalhos de Villar (2007), Ceniceros *et al.* (2010), Nós (2007) e Ceniceros, Nós e Roma (2010).

Considera-se um campo de velocidade auxiliar $\tilde{\mathbf{u}} = (\tilde{u},\ \tilde{v},\ \tilde{w})$ dado por

$$\tilde{\mathbf{u}}^{n+1} = \mathbf{u}^{n+1} + \frac{\Delta t \nabla q^{n+1}}{\alpha_2 \rho}, \tag{3.17}$$

na qual $q^{n+1} = p^{n+1} - p^n$ representa a correção de pressão.

Num método de passos fracionados, utilizam-se os campos de velocidade e pressão nos tempos precedentes para calcular uma estimativa para a velocidade no tempo atual, $\tilde{\mathbf{u}}^{n+1}$. A Eq. 3.12 é reescrita com esse campo auxiliar $\tilde{\mathbf{u}}^{n+1}$, ou seja,

$$\begin{aligned} &\frac{\rho^{n+1}}{\Delta t}\left(\alpha_2 \tilde{\mathbf{u}}^{n+1} + \alpha_1 \mathbf{u}^n + \alpha_0 \mathbf{u}^{n-1}\right) = \\ &\quad \lambda \nabla^2 \tilde{\mathbf{u}}^{n+1} + \beta_1 g_1(\mathbf{u}^n, \mu) + \beta_0 g_0(\mathbf{u}^{n-1}, \mu) - \nabla p^n + \rho^{n+1} g. \end{aligned} \tag{3.18}$$

Substituindo a Eq. 3.17 na Eq. 3.18 tem-se

$$\frac{\rho^{n+1}}{\Delta t}\left(\alpha_2\mathbf{u}^{n+1}+\alpha_1\mathbf{u}^{n}+\alpha_0 u^{n-1}\right)=$$
$$\lambda\nabla^2\mathbf{u}^{n+1}-\nabla q^{n+1}+\frac{\lambda\Delta t}{\alpha_2}\nabla^2\left(\frac{1}{\rho}\nabla q^{n+1}\right)+\beta_1 g_1(\mathbf{u}^n,\mu^n)+\beta_0 g_0(\mathbf{u}^{n-1},\mu^{n-a})$$
$$-\nabla p^n+\rho^{n+1}g. \tag{3.19}$$

Subtraindo a Eq. 3.19 da Eq. 3.12, tem-se

$$\nabla p^{n+1}=\nabla q^{n+1}+\nabla p^n-\nabla\left[\lambda\nabla\left(\frac{\Delta t}{\alpha_2\rho}\nabla q^{n+1}\right)\right]. \tag{3.20}$$

De acordo com Nós (2007), ao desprezar o ultimo termo da Eq. 3.20, obtém-se uma aproximação de primeira ordem para a pressão. Portanto a pressão é corrigida por meio da expressão

$$p^{n+1}=q^{n+1}+p^n. \tag{3.21}$$

Para a solução da Eq. 3.12 falta ainda definir a equação para a correção da pressão q. Aplicando o operador divergente na Eq. 3.17, tem-se

$$\nabla\cdot\tilde{\mathbf{u}}^{n+1}=\nabla\cdot\mathbf{u}^{n+1}+\frac{\Delta t}{\alpha_2}\nabla\cdot\left(\frac{1}{\rho^{n+1}}\nabla q^{n+1}\right). \tag{3.22}$$

Considerando a condição de incompressibilidade ($\nabla\cdot\mathbf{u}^{n+1}=0$) e substituindo na Eq. 3.22, obtém-se a equação diferencial elíptica ou equação de Poisson para a correção da pressão,

$$\nabla\cdot\left(\frac{1}{\rho^{n+1}}\nabla q^{n+1}\right)=\frac{\alpha_2}{\Delta t}\nabla\cdot\tilde{\mathbf{u}}^{n+1}. \tag{3.23}$$

Portanto, o método dos passos fracionados pode ser resumido na seguinte sequência de cálculo:

1. Entrar com as condições de contorno e condições iniciais;

2. Calcular o campo de velocidades estimado por meio da equação:

$$\frac{\rho}{\Delta t}\left(\alpha_2\tilde{\mathbf{u}}^{n+1}+\alpha_1\mathbf{u}^{n}+\alpha_0\mathbf{u}^{n-1}\right)=$$
$$\lambda\nabla^2\tilde{\mathbf{u}}^{n+1}+\beta_1 g_1(\tilde{\mathbf{u}}^n,\mu^n)+\beta_0 g_0(\tilde{\mathbf{u}}^{n-1},\mu^{n-1})-\nabla p^n+\rho\mathbf{g};$$

3. Com o campo de velocidade estimado, calcular a correção de pressão:

$$\nabla \cdot \left(\frac{1}{\rho^{n+1}} \nabla q^{n+1} \right) = \frac{\alpha_2}{\Delta t} \nabla \cdot \tilde{\mathbf{u}}^{n+1};$$

4. Corrigir o campo de velocidades, utilizando:

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}}^{n+1} - \frac{\Delta t \nabla q^{n+1}}{\alpha_2 \rho^{n+1}};$$

5. Corrigir o campo de pressão, utilizando:

$$p^{n+1} = p^n + q^{n+1};$$

6. Verificar a conservação da massa dentro da tolerância especificada;

7. Avançar para o próximo passo no tempo.

3.4 Discretização espacial

O domínio computacional do presente trabalho é dado por um retângulo de dimensão $[a1, b1] \times [a2, b2] \times [a3, b3]$. Inicialmente, este domínio é discretizado em uma malha uniforme com $M_x \times M_y \times M_z$ células computacionais e espaçamentos $\Delta x = \dfrac{b1 - a1}{M_x}$, $\Delta y = \dfrac{b2 - a2}{M_y}$ e $\Delta z = \dfrac{b3 - a3}{M_z}$ nas direções x, y e z respectivamente.

No presente trabalho, o centro de cada célula é dado por

$$\mathbf{x}_{i,j,k} = (x_i, y_j, z_k) = \left[a_1 + \left(i - \frac{1}{2} \right) \Delta x,\ a_2 + \left(j - \frac{1}{2} \right) \Delta y,\ a_3 + \left(k - \frac{1}{2} \right) \Delta z \right], \quad (3.24)$$

para $1 \leq i \leq M_x$, $1 \leq j \leq M_y$ e $1 \leq k \leq M_z$.

Para comportar as condições de contorno emprega-se uma camada adicional de células, ou mais, ao redor de todo o domínio. A essa camada de células dá-se o nome de células fantasmas. A figura 3.1 representa uma malha computacional com uma camada de células fantasmas.

Uma vez estabelecido o domínio computacional com seu conjunto de células, é necessário definir a posição das variáveis $\mathbf{u} = (u, v, w)$, p e $\mathbf{g}$, assim como das propriedades físicas do fluido ρ e μ. No presente trabalho, as variáveis escalares, tais como campo de

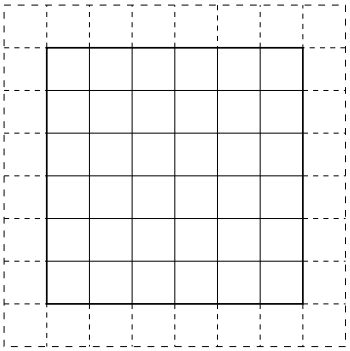


Figura 3.1: Esquema representativo de uma malha computacional regular com uma camada de células fantasmas, representada pelas linhas tracejadas.

pressão, propriedades físicas do fluido e divergentes são definidos no centro das células. Variáveis vetoriais, tais como velocidades e gradiente de pressão, são definidas no centros das faces da célula elementar, conforme a Fig. 3.2

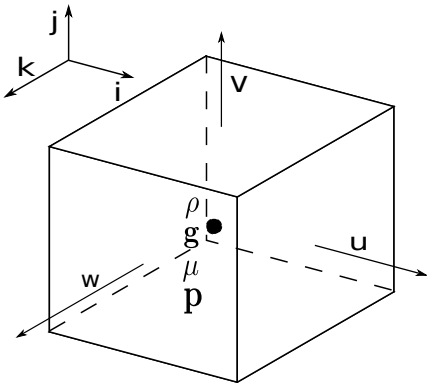


Figura 3.2: Esquema representativo de uma célula computacional elementar contendo as posições das variáveis segundo o esquema MAC

Segundo Roma (1996), citado por Villar (2007), a principal razão dessa escolha está no fato de que é possível definir convenientemente aproximações de segunda ordem para os operadores divergente e gradiente, de forma que a discretização da projeção apresente excelentes propriedades numéricas, permitindo uma fácil aplicação dos métodos *Multigrid* na solução da equação de Poisson.

3.5 Refinamento local adaptativo

Desenvolvida inicialmente por Berger e Oliger (1984), a metodologia de malhas bloco-estruturadas refinadas localmente permite obter a solução de equações diferenciais parciais substituindo-se a discretização do domínio físico em malha cartesiana uniforme por uma

hierarquia de malhas devidamente agrupadas. Esta sequência hierárquica de malhas forma níveis de refinamento sucessivamente refinados que vão de $l = l_{bot} + 1, ..., l_{top}$, onde l_{top} define o nível mais fino. As malhas estão contidas em blocos alinhados com os eixos de coordenadas e são formadas basicamente por pontos onde o erro da solução da malha mais grossa é elevado, devido a fenômenos localizados, como alta turbulência, alta vorticidade ou presença de interface.

Dinamicamente, malhas mais refinadas são aplicadas em regiões de interesse, tais como regiões isoladas que apresentam elevados gradientes, choque ou descontinuidades. Dessa forma, a solução em cada malha pode ser aproximada por diferenças finitas, como em malhas uniformes. Porém é necessário tomar cuidado no processo de comunicação entre as malhas compostas, ou seja, no cálculo das células fantasmas. Detalhes sobre o processo de cálculo das células fantasmas na malha composta são apresentados na seção 3.6. A Fig. 3.3 representa esquematicamente uma malha composta com três níveis de refinamento, sendo Ω^1 a malha de base (l_{bot}) que cobre todo o domínio e Ω^2 e Ω^3 as regiões representando os demais níveis de refinamento.

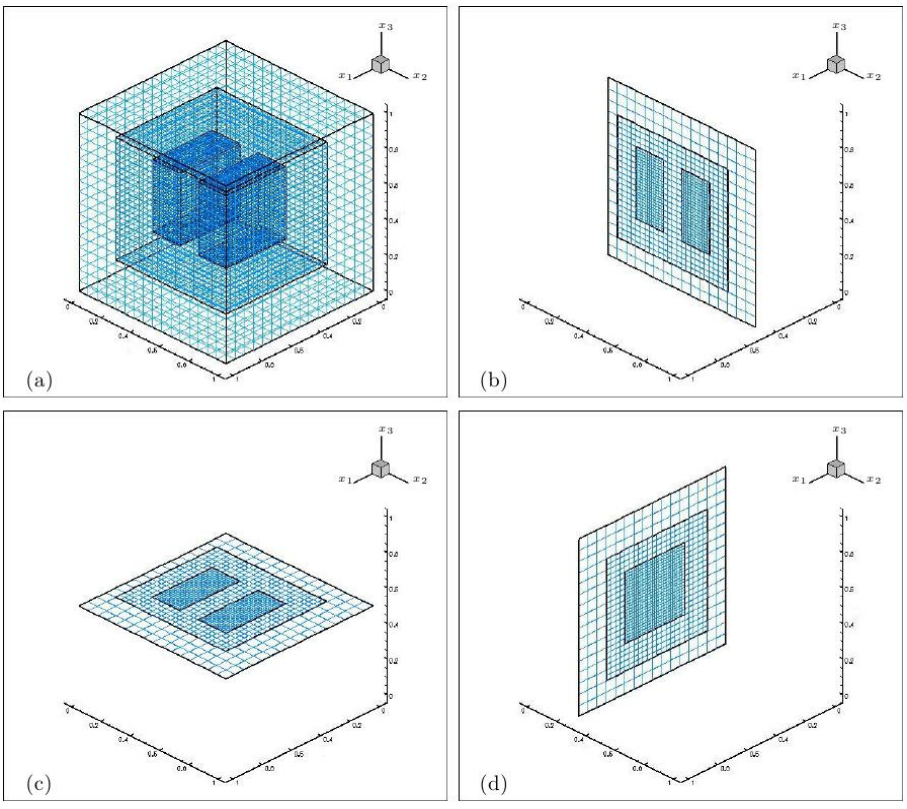


Figura 3.3: Esquema representativo de uma hierarquia de malhas com três níveis de refinamento.(NóS, 2007)

Uma característica importante de malhas estruturadas refinadas localmente, é razão entre os espaçamentos de malhas de nível l e $l + 1$, denominada razão de refinamento

r. Dessa forma, o espaçamento de uma malha de nível $l+1$ deve ser menor que o de nível l, relacionados por meio de um fator inteiro $r > 1$, ou seja $\Delta x_{l+1} = \frac{\Delta x_l}{r}$. Escolhas típicas para essa razão de refinamento são geralmente 2 ou 4, no presente trabalho será utilizada uma razão de refinamento igual a 2. Como na malha uniforme, o centro de cada célula nas malhas cartesianas em um dado nível l é dado pelos pontos $\mathbf{x_{i,j,k}} = \left(a_1 + \left(i + \frac{1}{2}\right)\Delta x_l,\ a_2 + \left(j + \frac{1}{2}\right)\Delta y_l,\ a_3 + \left(k + \frac{1}{2}\right)\Delta z_l\right)$. É importante notar que o nível base $l = l_{bot}$ é completamente coberto por toda a hierarquia de malhas.

A Figura 3.4 representa esquematicamente uma hierarquia de malhas refinadas localmente a uma razão igual a 2. A malha é composta por três níveis de refinamento, sendo que a malha base $l = l_{bot}$ cobre todo o domínio. O segundo nível $l = 2$ é composto por um bloco e o terceiro nível é composto por dois blocos.

As malhas para diferentes níveis em uma hierarquia precisam estar agrupadas corretamente, ou seja, devem satisfazer às duas propriedades abaixo:

1. Os cantos de uma malha mais fina coincidem com os cantos das células pertencentes a um nível imediatamente abaixo;

2. Malhas mais finas devem estar no interior da união das malhas do nível abaixo, exceto quando tocam as bordas do nível físico.

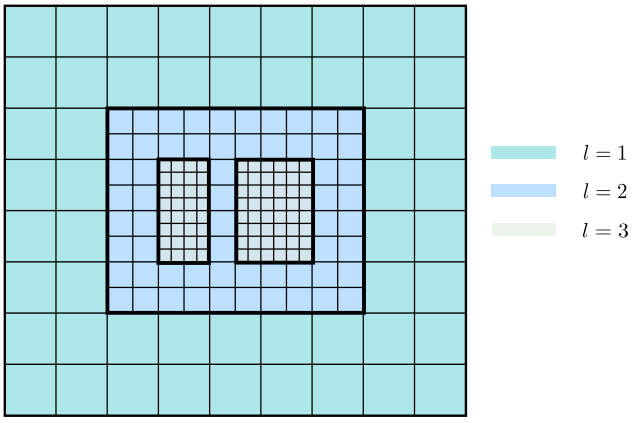


Figura 3.4: Esquema representativo de uma hierarquia de malhas com três níveis de refinamento.

A Figura 3.5 exemplifica duas malhas compostas que não estão agrupadas corretamente. Em cada um dos casos somente uma propriedade é violada. Para o caso da Fig. 3.5 (a), as extremidades das malhas do nível mais fino não tocam as extremidades das células do nível imediatamente abaixo. No caso da Fig. 3.5 (b), o agrupamento é dito não apropriado pois as células pertencentes ao terceiro nível estão diretamente adjacentes

às células pertencentes ao segundo nível. Vale observar que estas propriedades não estão restritas a apenas um bloco, ou seja, todas os blocos em qualquer nível devem obedecer tais propriedades. A Fig. 3.4 apresenta um conjunto de malhas agrupadas corretamente.

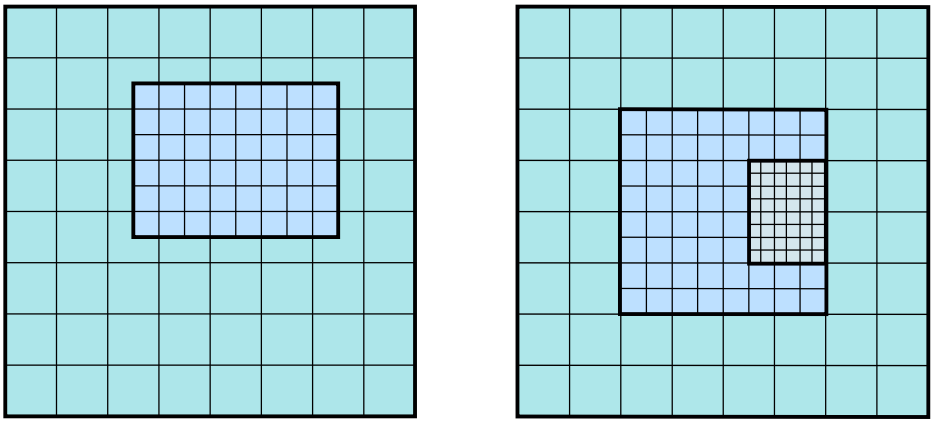


Figura 3.5: Malhas não corretamente agrupadas: propriedade 1 é violada (a) e propriedade 2 é violada (b).

Uma importante diferença entre o presente trabalho e o de Berger e Colella (1989), é que não há refinamento do passo de integração Δt. Isto significa que o passo de integração Δt, em todos os níveis de refinamento, é determinado pelo espaçamento $(\Delta x, \Delta y, \Delta z)$ da malha mais fina.

3.5.1 Geração de uma malha bloco-estruturada

No início do processo de construção da hierarquia de malhas, somente malha base e o número máximo de níveis são especificados pelo usuário. O número de níveis passa então a ser limitado somente pela memória do computador que está sendo usado. Dado um conjunto de células computacionais de um nível l marcadas para refinamento, utiliza-se um método proposto por Berger e Rigoutsos (1991), que agrupa tais células em blocos de nível l de tal forma que: o conjunto de células marcadas seja totalmente recoberto; o volume do bloco que não precisa de refinamento seja o menor possível; o número de blocos seja o menor possível. Uma representação esquemática da seleção de células para a geração de um bloco de nível l é ilustrada na Fig. 3.6. Para avaliar a eficiência do bloco gerado, Berger e Rigoutsos (1991) definem um parâmetro ξ_{min}, o qual relaciona o número total de células em um bloco nc com o número de células marcadas para refino no mesmo bloco $ncref$. Dessa forma tem-se:

$$\xi_{min} \leq \frac{ncref}{nc}, \tag{3.25}$$

na qual $0 \geq \xi_{min} \leq 1$. Assim, se $\xi_{min} = 0,90$, significa que 90% das células de um bloco de nível l foram marcadas para refinamento. Resumidamente, o processo de geração de malhas bloco estruturadas consiste é dado pelo algoritmo a seguir:

1. selecionar as células em que o refinamento é necessário;
2. agrupar as células selecionadas;
3. gerar os blocos para cada agrupamento;
4. avaliar a eficiência, repetir o processo caso a eficiência não seja atingida.

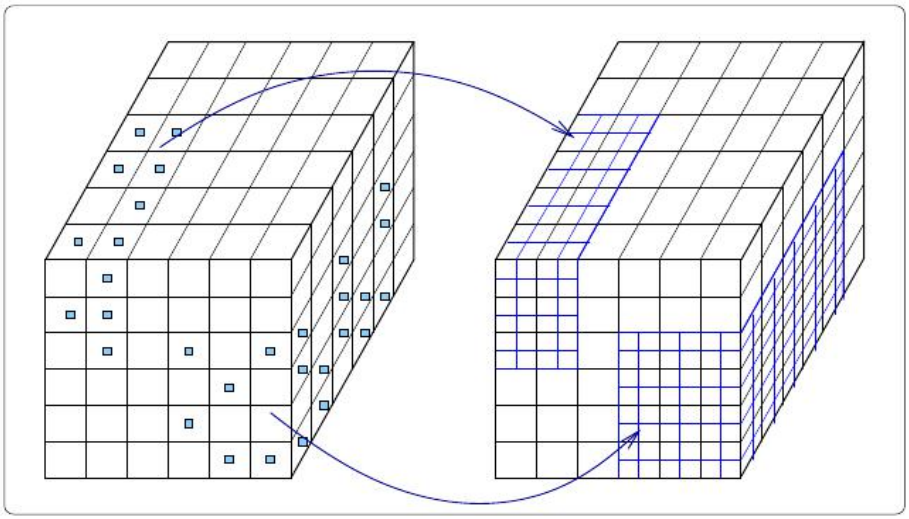


Figura 3.6: Células computacionais marcadas para o refinamento localizado(NóS, 2007)
.

3.6 Células fantasmas

As células fantasmas permitem armazenar os valores para as condições de contorno de cada bloco, de forma a evitar que os operadores diferenciais sejam redefinidos nas suas bordas. Dessa forma, o mesmo estêncil usado no interior das células de uma malha pode também ser usado nas células localizadas nas bordas das malhas. O número de células fantasmas considerado depende do tamanho do estêncil necessário para a discretização do operador diferencial, sendo necessária uma camada de células fantasma ao redor de cada malha, inclusive no nível base l_{bot}.

O cálculo das células fantasmas é feito nível a nível, sendo l o nível de refinamento atual. O procedimento de cálculo é composto por cinco passos, representados esquematicamente nas figuras 3.7 a 3.10, os quais são:

1. extrapolação cúbica utilizando valores do próprio bloco em l (Fig. 3.7);

2. interpolação cúbica, utilizando valores de l, para células de *l-1* que estão no contorno dos blocos de l (Fig. 3.8);

3. interpolação quadrática entre valores de l e *l-1* (Fig. 3.9);

4. preenchimento da célula fantasma de blocos no nível l utilizando valores de células do mesmo nível, pertencente a outro bloco, pelo processo de injeção (Fig. 3.10);

5. preenchimento da célula fantasma de l utilizando condições de contorno físicas.

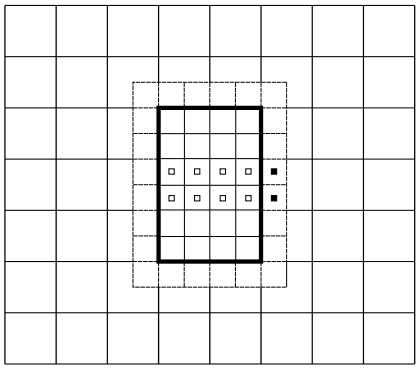


Figura 3.7: Representação esquemática do primeiro passo, utilizando extrapolação cúbica, no cálculo das células fantasmas.

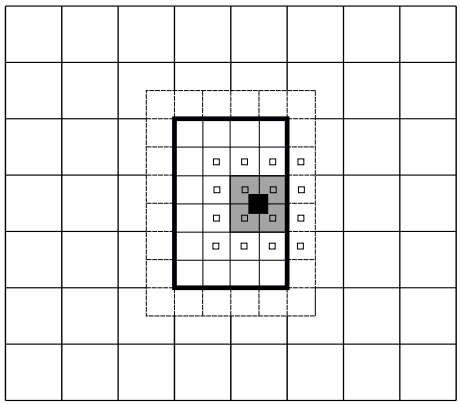


Figura 3.8: Representação esquemática do segundo passo, utilizando interpolação cúbica. Os valores são interpolados das células de nível l para as células de nível *l-1* que tocam as bordas dos blocos de nível l.

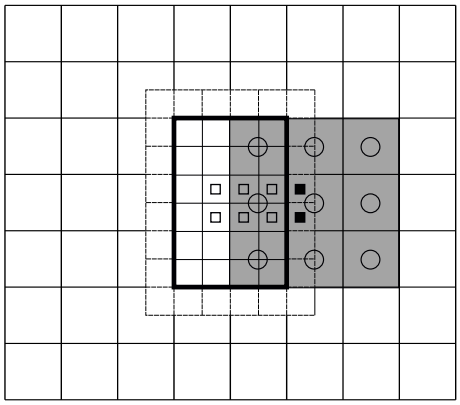


Figura 3.9: Representação esquemática do terceiro passo, utilizando interpolação quadrática no cálculo das células fantasmas.

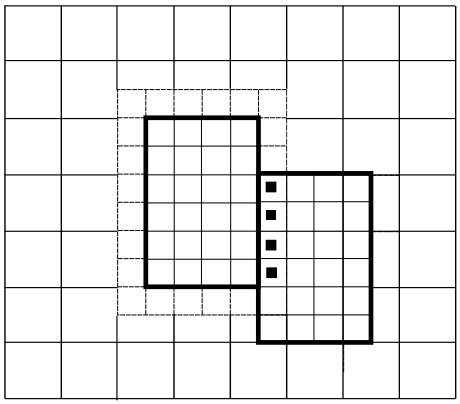


Figura 3.10: Representação esquemática do processo de injeção entre malhas irmãs, utilizada no cálculo das células fantasmas.

3.7 Discretização das equações de Navier-Stokes

De acordo com o que foi descrito anteriormente, não é necessária a redefinição dos operadores diferenciais nas interfaces entre níveis, devido à presença de células fantasmas. Sendo assim, as discretizações espaciais podem ser tratadas igualmente tanto em malhas uniformes como em malhas compostas. Para a discretização espacial das equações do movimento no domínio euleriano, é utilizado o método das diferenças finitas centradas, de segunda ordem, em malhas deslocadas. Para a discretização do termo temporal, utiliza-se Euler no segundo instante de tempo e Gear para os demais instantes de tempo. Segue abaixo a discretização de cada um dos termos das equações de Navier-Stokes e da continuidade para a componente x

- Equação da continuidade:

$$\nabla \cdot \mathbf{u} = 0,$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0,$$

$$\frac{u_{i+1,j,k} - u_{i,j,k}}{\Delta x} + \frac{v_{i,j+1,k} - v_{i,j,k}}{\Delta y} + \frac{w_{i,j,k+1} - w_{i,j,k}}{\Delta z} \cong 0. \tag{3.26}$$

- Termo temporal:

$$\rho \frac{\partial \mathbf{u}}{t},$$

$$\rho \frac{\partial u}{t}.$$

1. Euler:

$$\frac{\rho^{n+1}_{i,j,k} + \rho^{n+1}_{i-1,j,k}}{2} \frac{\tilde{u}^{n+1}_{i,j,k} - u^{n}_{i,j,k}}{\Delta t}. \tag{3.27}$$

2. Gear:

$$\frac{\rho^{n+}_{i,j,k} + \rho^{n+1}_{i-1,j,k}}{2} \frac{\alpha_2 \tilde{u}^{n+1}_{i,j,k} - \alpha_1 u^{n}_{i,j,k} + \alpha_0 u^{n-1}_{i,j,k}}{\Delta t}. \tag{3.28}$$

- Termo advectivo:

$$\rho(\mathbf{u} \cdot \nabla u),$$

$$\rho \left(u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} \right),$$

$$\frac{\rho^{n}_{i,j,k} - \rho^{n}_{i-1,j,k}}{2} \left[u_{i,j,k} \left(\frac{u_{i+1,j,k} - u_{i-1,j,k}}{2\Delta x} \right) + \left(\frac{v_{i,j+1,k} + v_{i-1,j+1,k} + v_{i,j,k} + v_{i-1,j,k}}{4} \right) \left(\frac{u_{i,j+1,k} - u_{i,j-1,k}}{2\Delta y} \right) + \left(\frac{w_{i,j,k+1} + w_{i-1,j,k+1} + w_{i,j,k} + w_{i-1,j,k}}{4} \right) \left(\frac{u_{i,j,k+1} - u_{i,j,k-1}}{2\Delta z} \right) \right]. \tag{3.29}$$

- Gradiente de pressão:

$$\nabla p,$$

$$\frac{\partial p}{\partial x},$$

$$\frac{p^{n}_{i,j,k} - p^{n}_{i-1,j,k}}{\Delta x}. \tag{3.30}$$

- Termo difusivo:

$$\nabla\cdot\Big[\mu(\nabla\mathbf{u}+\nabla\mathbf{u}^T)\Big],$$

$$\mu\cdot\nabla^2 u+\mu\frac{\partial}{\partial x}\Big(\nabla\cdot\mathbf{u}\Big)+2\frac{\partial\mu}{\partial x}\frac{\partial u}{\partial x}+\frac{\partial\mu}{\partial y}\Big(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\Big)+\frac{\partial\mu}{\partial z}\Big(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\Big),$$

$$\begin{aligned}
\mu\cdot\nabla^2 u &= \frac{\mu_{i,j,k}+\mu_{i-1,j,k}}{2}\Big(\frac{u_{i+1,j,k}-2u_{i,j,k}+u_{i-1,j,k}}{\Delta x^2}+\\
&\quad \frac{u_{i,j+1,k}-2u_{i,j,k}+u_{i,j-1,k}}{\Delta y^2}+\frac{u_{i,j,k+1}-2u_{i,j,k}+u_{i,j,k-1}}{\Delta z^2}\Big),\\
2\frac{\partial\mu}{\partial x}\frac{\partial u}{\partial x} &= 2\Big(\frac{\mu_{i,j,k}-\mu_{i-1,j,k}}{\Delta x}\frac{u_{i+1,j,k}-u_{i-1,j,k}}{2\Delta x}\Big),\\
\frac{\partial\mu}{\partial y}\Big(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\Big) &= \Big(\frac{\mu_{i,j+1,k}-\mu_{i,j-1,k}+\mu_{i-1,j+1,k}-\mu_{i-1,j-1,k}}{4\Delta y}\Big)\\
&\quad \Big[\frac{u_{i,j+1,k}-u_{i,j-1,k}}{2\Delta y}+\frac{v_{i,j+1,k}-v_{i-1,j+1,k}+v_{i,j,k}-v_{i-1,j,k}}{4\Delta x}\Big],\\
\frac{\partial\mu}{\partial z}\Big(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\Big) &= \Big(\frac{\mu_{i,j,k+1}-\mu_{i,j,k-1}+\mu_{i-1,j,k+1}-\mu_{i-1,j,k-1}}{4\Delta z}\Big)\\
&\quad \Big[\frac{u_{i,j,k+1}-u_{i,j,k-1}}{2\Delta z}+\frac{w_{i,j,k+1}-w_{i-1,j,k+1}+w_{i,j,k}-w_{i-1,j,k}}{4\Delta x}\Big],\\
\mu\frac{\partial}{\partial x}\Big(\nabla\cdot\mathbf{u}\Big) &= 0. \qquad (3.31)
\end{aligned}$$

É importante destacar que os índices temporais não são aqui apresentados para os termos difusivo e advectivo pois, conforme as equações 3.12 a 3.15, tais termos são discretizados no tempo n e $n-1$. O termo de gradiente de pressão é discretizado apenas no tempo n e a presença do termo gravitacional ocorre somente na direção y onde g_y é a aceleração gravitacional na direção y, a qual é uma constante. Além disso, na Eq. 3.31, o termo $\mu\frac{\partial}{\partial x}\Big(\nabla\cdot\mathbf{u}\Big)=0$, pois pela condição de incompressibilidade temos que $\nabla\cdot\mathbf{u}=\mathbf{0}$.

3.7.1 Discretização das equações para o cálculo das estimativas das velocidades

O cálculo das velocidades estimadas é efetuado por meio da equação 3.18. Agrupando os termos das velocidades estimadas que estão no tempo $n+1$, tem-se

$$\begin{aligned}
&\frac{\rho}{\Delta t}\alpha_2\tilde{\mathbf{u}}^{n+1}-\lambda\nabla^2\tilde{\mathbf{u}}^{n+1}=\\
&\quad -\frac{\rho}{\Delta t}\left(\alpha_1\mathbf{u}^n+\alpha_0\mathbf{u}^{n-1}\right)+\beta_1 g_1(\mathbf{u}^n,\mu)+\beta_0 g_0(\mathbf{u}^{n-1},\mu)-\nabla p^n+\rho\mathbf{g}. \qquad (3.32)
\end{aligned}$$

Definindo o lado direito da Eq.3.32 como Ψ,

$$\Psi = \beta_1 g_1(\mathbf{u}^n, \mu) + \beta_0 g_0(\mathbf{u}^{n-1}, \mu) - \nabla p^n + \rho g - \frac{\rho}{\Delta t}\alpha_1 \mathbf{u}^n - \frac{\rho}{\Delta t}\alpha_0 \mathbf{u}^{n-1}, \tag{3.33}$$

$$g(\mathbf{u}, \mu) = -\lambda \nabla^2 \mathbf{u} + \mu \nabla^2 \mathbf{u} - \mathbf{u} \cdot \nabla \mathbf{u}, \tag{3.34}$$

e utilizando o método das diferenças finitas de segunda ordem para a discretização do operador laplaciano, tem-se:

$$\begin{aligned} \frac{\rho}{\Delta t}\alpha_2 \tilde{u}^{n+1} - \lambda \Bigg(& \frac{\tilde{u}^{n+1}_{i+1,j,k} - 2\tilde{u}^{n+1}_{i,j,k} + \tilde{u}^{n+1}_{i-1,j,k}}{\Delta x^2} + \frac{\tilde{u}^{n+1}_{i,j+1,k} - 2\tilde{u}^{n+1}_{i,j,k} + \tilde{u}^{n+1}_{i,j-1,k}}{\Delta y^2} \\ & + \frac{\tilde{u}^{n+1}_{i,j,k+1} - 2\tilde{u}^{n+1}_{i,j,k} + \tilde{u}^{n+1}_{i,j,k-1}}{\Delta z^2} \Bigg) = \Psi_{i,j,k}. \end{aligned} \tag{3.35}$$

Reescrevendo a Eq. 3.35 em função dos coeficientes a_e,a_w, a_n, a_s, a_t, a_b, temos

$$a_e \tilde{u}_{i+1,j,k} + a_w \tilde{u}_{i-1,j,k} + a_n \tilde{u}_{i,j+1,k} + a_s \tilde{u}_{i,j-1,k} + a_t \tilde{u}_{i,j,k+1} + a_b \tilde{u}_{i,j,k-1} + a_p \tilde{u}_{i,j,k} = \Psi_{i,j,k}, \tag{3.36}$$

nos quais

$$\begin{aligned} a_e &= -\frac{\lambda}{\Delta x^2}, \\ a_w &= -\frac{\lambda}{\Delta x^2}, \\ a_n &= -\frac{\lambda}{\Delta y^2}, \\ a_s &= -\frac{\lambda}{\Delta y^2}, \\ a_t &= -\frac{\lambda}{\Delta z^2}, \\ a_b &= -\frac{\lambda}{\Delta z^2}, \\ a_p &= \rho\frac{\alpha_2}{\Delta t} + 2\frac{\lambda}{\Delta x^2} + 2\frac{\lambda}{\Delta y^2} + 2\frac{\lambda}{\Delta z^2}, \\ \lambda &= 2\mu. \end{aligned}$$

A solução da Eq. 3.36 é feita por meio do método Multinível-*Multigrid*, o qual será descrito na seção 3.8.

3.7.2 Discretização da equação para a correção de pressão e velocidades corrigidas

A discretização da Eq. 3.23 é apresentada em seguida, sendo que todos os termos para a correção da pressão estão no tempo $n+1$. É necessário observar que a correção de pressão q está situada no centro da célula computacional, a qual dá origem ao sistema linear representado por:

$$\nabla \cdot \left(\frac{1}{\rho^n} \nabla q^{n+1} \right) = \frac{\alpha_2}{\Delta t} \nabla \cdot \tilde{\mathbf{u}}^{n+1},$$

$$\frac{\partial}{\partial y}\left(\frac{1}{\rho}\frac{\partial q}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{1}{\rho}\frac{\partial q}{\partial z}\right) = \frac{\alpha_2}{\Delta t}\left(\frac{\partial \tilde{u}}{\partial x} + \frac{\partial \tilde{v}}{\partial y} + \frac{\partial \tilde{w}}{\partial z}\right),$$

$$\frac{\partial}{\partial x}\left(\frac{1}{\rho}\frac{\partial q}{\partial x}\right) = \frac{1}{2}\left(\frac{1}{\rho^n_{i+1,j,k}} + \frac{1}{\rho^n_{i,j,k}}\right)\left(\frac{q^{n+1}_{i+1,j,k} - q^{n+1}_{i,j,k}}{\Delta x^2}\right) - \frac{1}{2}\left(\frac{1}{\rho^n_{i,j,k}} + \frac{1}{\rho^n_{i-1,j,k}}\right)\left(\frac{q^{n+1}_{i,j,k} - q^{n+1}_{i-1,j,k}}{\Delta x^2}\right),$$

$$\frac{\partial}{\partial y}\left(\frac{1}{\rho}\frac{\partial q}{\partial y}\right) = \frac{1}{2}\left(\frac{1}{\rho^n_{i,j+1,k}} + \frac{1}{\rho^n_{i,j,k}}\right)\left(\frac{q_{i,j+1,k} - q_{i,j,k}}{\Delta y^2}\right) - \frac{1}{2}\left(\frac{1}{\rho^n_{i,j,k}} + \frac{1}{\rho^n_{i,j-1,k}}\right)\left(\frac{q_{i,j,k} - q_{i,j-1,k}}{\Delta y^2}\right),$$

$$\frac{\partial}{\partial z}\left(\frac{1}{\rho}\frac{\partial q}{\partial z}\right) = \frac{1}{2}\left(\frac{1}{\rho^n_{i,j,k+1}} + \frac{1}{\rho^n_{i,j,k}}\right)\left(\frac{q^{n+1}_{i,j,k+1} - q^{n+1}_{i,j,k}}{\Delta z^2}\right) - \frac{1}{2}\left(\frac{1}{\rho^n_{i,j,k}} + \frac{1}{\rho^n_{i,j,k-1}}\right)\left(\frac{q^{n+1}_{i,j,k} - q^{n+1}_{i,j,k-1}}{\Delta z^2}\right),$$

$$\frac{\alpha_2}{\Delta t}\nabla \cdot \tilde{\mathbf{u}}^{n+1} = \frac{\alpha_2}{\Delta t}\left(\frac{\tilde{u}_{i+1,j,k} - \tilde{u}_{i,j,k}}{\Delta x} + \frac{\tilde{v}_{i,j+1,k} - \tilde{v}_{i,j,k}}{\Delta y} + \frac{\tilde{w}_{i,j,k+1} - \tilde{w}_{i,j,k}}{\Delta z}\right). \tag{3.37}$$

Denominando o lado direito da Eq. 3.37 de $\Theta_{i,j,k}$,o sistema linear para a correção de pressão a ser resolvido pode ser representado por

$$a_n q_{i,j+1,k} + a_s q_{i,j-1,k} + a_e q_{i+1,j,k} + a_e q_{i-1,j,k} + a_t q_{i,j,k+1} + a_b q_{i,j,k-1} = \Theta_{i,j,k}, \tag{3.38}$$

no qual,

$$a_e = \frac{\rho_e}{\Delta x^2} = \frac{1}{2\Delta x^2}\left(\frac{1}{\rho^n_{i+1,j,k}} + \frac{1}{\rho^n_{i,j,k}}\right),$$

$$a_w = \frac{\rho_w}{\Delta x^2} = \frac{1}{2\Delta x^2}\left(\frac{1}{\rho^n_{i,j,k}} + \frac{1}{\rho^n_{i-1,j,k}}\right),$$

$$a_n = \frac{\rho_n}{\Delta y^2} = \frac{1}{2\Delta y^2}\left(\frac{1}{\rho^n_{i,j+1,k}} + \frac{1}{\rho^n_{i,j,k}}\right),$$

$$a_s = \frac{\rho_s}{\Delta y^2} = \frac{1}{2\Delta y^2}\left(\frac{1}{\rho^n_{i,j,k}} + \frac{1}{\rho^n_{i,j-1,k}}\right),$$

$$a_t = \frac{\rho_t}{\Delta z^2} = \frac{1}{2\Delta z^2}\left(\frac{1}{\rho^n_{i,j,k+1}} + \frac{1}{\rho^n_{i,j,k}}\right),$$

$$a_b = \frac{\rho_b}{\Delta z^2} = \frac{1}{2\Delta z^2}\left(\frac{1}{\rho^n_{i,j,k}} + \frac{1}{\rho^n_{i,j,k-1}}\right),$$

$$a_p = -(a_e + a_w + a_n + a_s + a_t + a_b).$$

Para a solução dos sistema linear dado pela Eq. 3.38, utiliza-se o método Multinível-*Multigrid*- descrito na seção 3.8.

A atualização ou correção das velocidades é dada pela seguinte discretização, apresentada aqui na mesma direção da componente de velocidades u.

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}}^{n+1} - \frac{1}{\rho^n}\frac{\Delta t \nabla q^{n+1}}{\alpha_2},$$

$$u^{n+1}_{i,j,k} = \tilde{u}^{n+1}_{i,j,k} - \frac{1}{2}\left(\frac{1}{\rho^n_{i+1,j,k}} + \frac{1}{\rho^n_{i,j,k}}\right)\frac{\Delta t}{\alpha_2}\left(\frac{q^{n+1}_{i,j,k} - q^{n+1}_{i-1,j,k}}{\Delta x}\right). \tag{3.39}$$

3.8 Método *Multigrid*

Uma alternativa para diminuir o tempo necessário para a solução numérica de problemas envolvendo mecânica dos fluidos e transferência de calor seria o uso de métodos *Multigrid* para a resolução dos sistemas lineares envolvidos na solução numérica destes problemas (BRIGGS; EMDEN; MCCORMICK, 2000), (TROTTENBERG; SCHULLER, 2001). Provavelmente, a situação mais comum na qual o (MG) é empregado é na solução da equação de Poisson 3.23. A equação, discretizada para um dado ponto nodal no centro da célula, é dada pela Eq. 3.38, a qual pode ser escrita na forma

$$A\phi = B, \tag{3.40}$$

onde A é a matriz de coeficientes, ϕ é o vetor das incógnitas e B é o vetor que acomoda os termos fontes.

Segundo Trottenberg e Schuller (2001) o método *Multigrid* é composto por dois princípios básicos. O primeiro baseia-se na propriedade de suavização do erro de métodos iterativos clássicos para solução de sistemas lineares, entre eles o método Gauss Seidel (GS). O segundo afirma que uma quantidade suavizada em uma determinada malha pode ser aproximada em uma malha mais grosseira. Em outras palavras, se o erro for suavizado em um determinado número de iterações, é possível aproximar tal erro em uma malha mais grosseira. Para que as componentes do erro de longos comprimentos de onda possam ser suavizadas com eficiência, o método *Multigrid* procura trabalhar não com uma única malha, mas com uma sequência de malhas cada vez mais grosseiras. Assim, comprimentos de onda que são longos em malhas finas, são transformados em curtos em malhas grossas, onde o erro pode ser rapidamente suavizado (MCCORMICK, 1987).

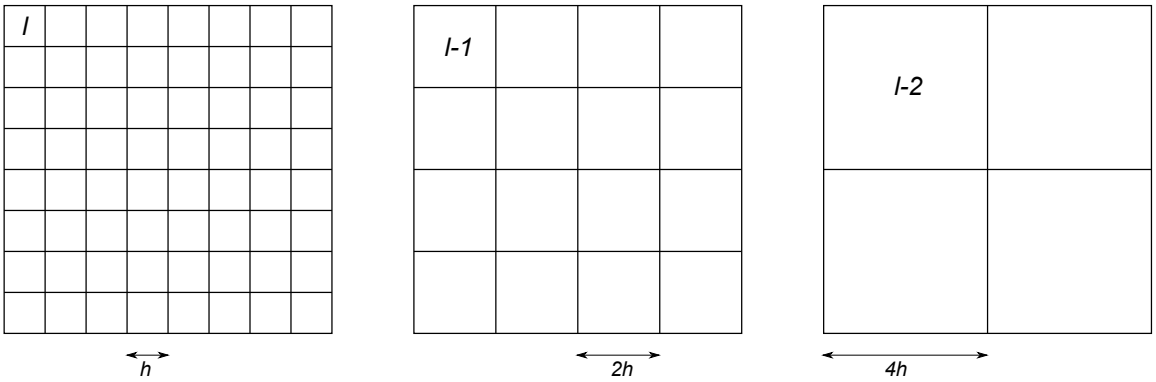


Figura 3.11: Exemplo de sequência de malhas onde as componentes do erro são reduzidas pelo método *Multigrid*.

3.8.1 Método *Multigrid* em malhas uniformes

Ao considerar que $\bar{\phi}$ é uma aproximação da solução exata ϕ e que $e = \phi - \bar{\phi}$, é o desvio da aproximação $\bar{\phi}$ em relação à solução exata ϕ, pode-se escrever a seguinte equação

$$A(e + \bar{\phi}) = B. \tag{3.41}$$

Como o valor exato do erro é tão inacessível quanto valor exato da solução, é necessário determinar uma estimativa do erro, no caso, o resíduo R, ou seja,

$$R = B - A\bar{\phi} \tag{3.42}$$

O processo iterativo é interrompido quando o valor de R for suficientemente pequeno, da ordem de $0,5min(\Delta x^2, \Delta y^2, \Delta z^2)$. Reorganizando a Eq.3.42 tem-se,

$$A\bar{\phi} = B - R, \tag{3.43}$$

e efetuando-se a subtração entre as equações 3.40 e 3.43 tem-se

$$A(\phi - \bar{\phi}) = R, \tag{3.44}$$

que, com a inserção da relação $e = \phi - \bar{\phi}$, temos a equação, dada por

$$Ae = R. \tag{3.45}$$

No nível mais grosso $l = 1$ recomenda-se que a Eq. 3.45 seja resolvida exatamente ou que nela seja aplicado um número bem maior de iterações. Por envolver um número expressivamente menor de células e, portanto, corresponder ao menor sistema de equações, tal procedimento não compromete muito o esforço computacional (LEBRÓN, 2001). Com a definição da equação do resíduo busca-se obter a solução na malha fina, empregando-se os demais níveis apenas como esquemas de correção.

3.8.2 Operadores para transferência entre malhas

Elementos essenciais do método *Multigrid* são os operadores de transferência, ou seja, os operadores de restrição $\mathcal{R}$ e prolongamento $\mathcal{P}$. Quando a transferência é no sentido fina-grossa ($l \rightarrow l-1$) utiliza-se o operador de restrição $\mathcal{R}$. Quando a transferência se dá no sentido oposto ($l \leftarrow l-1$) o operador empregado é o prolongamento $\mathcal{P}$. Normalmente, usa-se médias aritméticas para operações de restrição e polinômios bilineares ou trilineares para as operações de prolongamento. No presente trabalhos são empregadas as mesmas interpolações trilineares do trabalho de Nós (2007).

A sequência de como os procedimentos de mudança de malha são realizados caracteriza os chamados *Ciclo-V* e *Ciclo-W*. A Fig. 3.12 mostra a sequência de operações em cada ciclo durante uma iteração completa do método *Multigrid*.

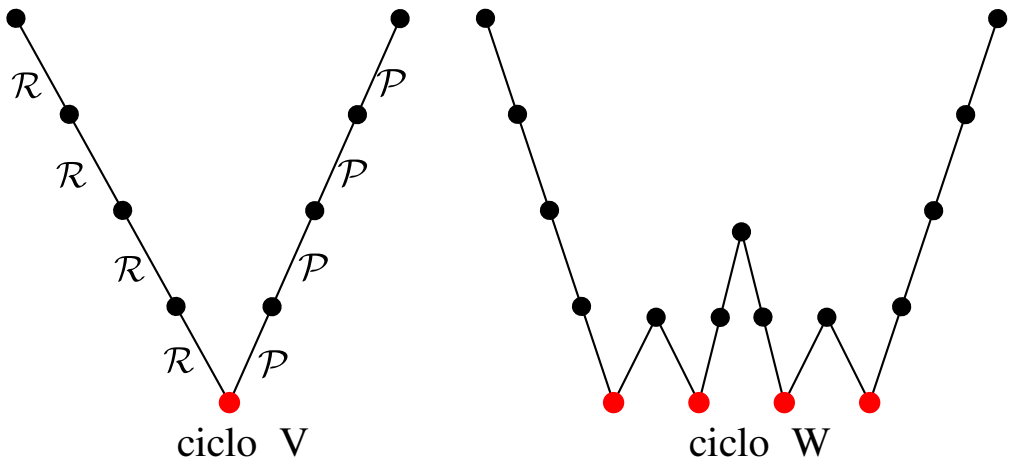


Figura 3.12: Esquema representativo dos Ciclos V e W em uma malha uniforme com 4 níveis sequencialmente mais grossos.

O processo iterativo do método *Multigrid* é composto pelos seguintes passos:

1. relaxar η vezes $A^l\bar{\phi}^l = B^l$ na malha do nível fino l^{top}, partindo-se de um valor inicial $\bar{\phi}_0$ e obtendo-se uma estimativa $\bar{\phi}^l_\eta$, a qual será utilizada para avaliar o resíduo;
2. calcular o resíduo $R^l = B^l - A^l\bar{\phi}^l$;
3. restringir o resíduo R^l para a malha pertencente ao nível imediatamente abaixo $l-1$, empregando interpolações trilineares, determinando-se R^{l-1};
4. relaxar algumas vezes a equação do resíduo $A^{l-1}e^{l-1} = R^{l-1}$, com valor inicial $e^{l-1} = 0$, obtendo-se os valores e^{l-1}. Observa-se que o termo B não é conhecido para as malhas $l \leq l_{bot}$;
5. transferir os valores deste erro e^{l-1} para uma malha mais fina, por meio de interpolações trilineares, determinando-se e^l;
6. corrigir a aproximação da malha fina com $\bar{\phi}^l \rightarrow \bar{\phi}^l + e^l$.

3.8.3 Método Multinível-*Multigrid*

Uma variante do método *Multigrid* é representada pelo método Multinível-*Multigrid*, o qual estende tal metodologia para malhas refinadas localmente. Considerando que malhas estruturadas refinadas localmente são utilizadas neste trabalho, a escolha do método Multinível-*Multigrid* se ajusta bem dentro do contexto de malhas compostas. No método entanto, verifica - se um tratamento diferenciado na implementação do método *Multigrid* nas malhas compostas.

Devido à utilização do método Multinível-*Multigrid* para a solução de sistemas lineares, é necessário dividir os níveis de refinamento em dois grupos: níveis físicos e níveis virtuais. A Fig. 3.13 ilustra de forma esquemática os níveis físicos e virtuais em uma malha composta para um Ciclo-V. Nesta figura, observa-se que, para uma malha com níveis de que vão de $l = 1, ..., l_{top}$, os níveis virtuais são formados pelas malhas $1, ..., l_{bot} - 1$ e os níveis físicos pelas malhas $l_{bot}, ..., l_{top}$. Dessa forma, conclui-se que a hierarquia de malhas adaptativas é composta por células visíveis e não visíveis. As células não visíveis são formadas pelos níveis virtuais e pelas células fantasmas de cada bloco. O conjunto de células visíveis é formado por toda a malha numérica, com exceção das células não visíveis.

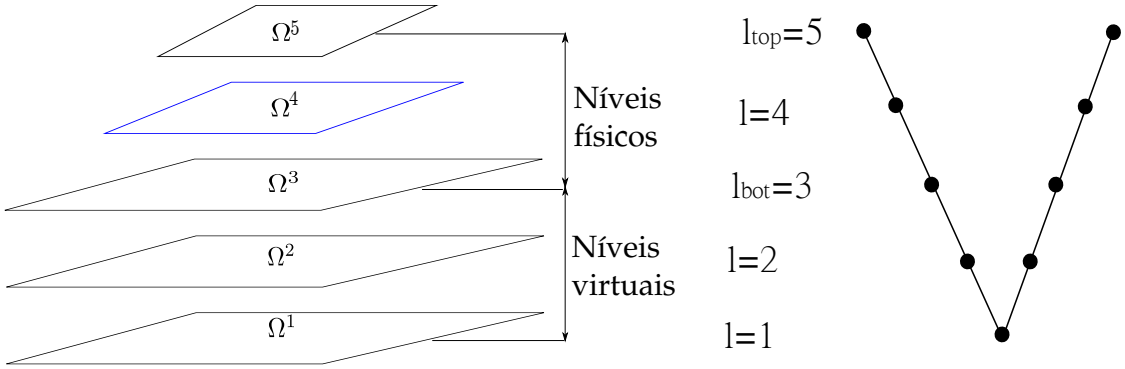


Figura 3.13: Representação esquemática dos níveis físicos e virtuais para uma malha refinada localmente.

Espera-se que os valores calculados nas malhas mais finas apresentem maior precisão que nas malhas grossas. Isto leva ao conceito de duas regiões: uma região válida , a qual não está coberta por malhas mais finas e uma região composta, a qual está coberta por malhas mais finas. Ao final do processo, se uma determinada célula pertence a uma região composta então ela é recoberta pelos valores da malha fina através de médias aritméticas simples.

Como citado anteriormente, a equação que se deseja resolver é dada por

$$A\phi = B \tag{3.46}$$

Para cada nível de refinamento l, os valores de ϕ^l, do resíduo R e da correção e^l, devem ser armazenados em todo o domínio. O valor de B^l é definido somente na região em que a malha grossa não está coberta pela malha fina, ou seja, em $\Omega^l - \Omega^{l-1}$. Da mesma forma que na malha uniforme, é necessário também um operador de relaxação o qual realiza η relaxações na equação de Poisson. Assim define-se $GSRB(A^l, e^l, R^l)$ em Ω^l , o qual realiza η relaxações com o método *Gauss-Seidel Red-Black* ($GSRB$) em um dado nível l.

O algoritmo para o método Multinível-*Multigrid* utilizado no presente trabalho é dado por:

para $l = l_{top}$ até 1 **faça**
  se $l = l_{top}$ **faça**
    relaxar $A^{l_{top}}\bar{\phi}^{l_{top}} = B^{l_{top}}$
    calcular $R^{l_{top}} = B^{l_{top}} - A^{l_{top}}\bar{\phi}^{l_{top}}$
    relaxar $A^{l_{top}}e^{l_{top}} = R^{l_{top}}$
  senão
    relaxar $A^l\bar{\phi}^l = B^l$
    na região de l coberta por $l+1$
      calcular $R^l = \mathcal{R}_l^{l+1}(R^{l+1} - A^{l+1}e^{l+1})$
    na região de l não coberta por $l+1$
      calcular $R^l = B^l - A^l\bar{\phi}^l$
    relaxar $A^le^l = R^l$
  se $l = 1$ **faça**
    calcular $A^1e^1 = R^1$
para $l = 2$ até l_{top} **faça**
  corrigir $e^l = e^l + \mathcal{P}_l^{l-1}e^{l-1}$
  relaxar $A^le^l = R^l$
  corrigir $\bar{\phi}^l = \bar{\phi}^l + e^l$

Observa-se que cada nível de refinamento, o resíduo possui duas componentes. A primeira é determinada apenas em regiões de l cobertas por $l+1$, por meio do processo de restrição do resíduo de $l+1$ para l. A outra componente é definida na região não coberta pelo nível mais fino, ou seja, $\Omega^l - \Omega^{l+1}$. Baseando-se nisto, tem-se os seguintes cálculos de resíduo:

- No nível mais fino, para $l = l_{top}$

$$R^{l_{top}} \leftarrow B^{l_{top}} - A^{l_{top}}\bar{\phi}^{l_{top}};$$

- Na região do nível l coberta pelo nível $l+1$

$$R^l \leftarrow \mathcal{R}_l^{l+1}(R^{l+1} - A^{l+1}(e^{l+1}));$$

- Na região do nível l que não está coberta pelo nível $l+1$

$$R^l \leftarrow B^{l-1} - A^l \bar{\phi}^l.$$

O algoritmo para o método Multinível-*Multigrid* para uma quantidade máxima de níveis igual a l_{top}, é estruturado da mesma forma que o método *Multigrid* para malha uniforme, onde o processo se inicia no nível mais fino $l = l_{top}$. Nas malhas progressivamente mais grossas realiza-se a relaxação, resolve-se a equação do resíduo até o nível $l = 1$ e finalmente em sentido contrário interpola e relaxa novamente. A diferença é que para este caso, a forma como os operadores são aplicados não é a mesma para todo o nível l. Os processos de interpolação e atualização da correção também são idênticos ao aplicado na malha uniforme.

3.9 Particionamento de domínio com balanço de carga

Para se obter um uso eficiente de computadores em paralelo, dois objetivos devem ser alcançados. O primeiro consiste em manter os processadores ocupados com trabalho útil e o segundo em minimizar a comunicação entre os processos. Para alguns problemas em computação científica, tais objetivos são alcançados com uma simples atribuição de tarefas que não se altera com o decorrer do tempo. Entretanto, o comportamento dinâmico da hierarquia de malhas apresentada no presente trabalho requer que a distribuição de tarefas seja dinâmica.

Como dito anteriormente, no contexto de malhas adaptativas bloco estruturadas a distribuição de tarefas pode ser feita por partição de domínio, distribuição de blocos ou híbrida. No presente trabalho utiliza-se o método da bisseção recursiva por coordenadas (RCB - *Recursive Coordenate Bisection*) para partição de domínio. A escolha deste método levou em conta o baixo custo para balanço de carga e fato de retornar balanços de carga satisfatórios, além de trabalhar com domínios retilineares.

O método RCB é um algoritmo simples e intuitivo. O algoritmo escolhe uma direção de corte para a malha, fazendo um corte perpendicular apropriado. O domínio deve ser cortado de tal forma que a soma das cargas atribuídas a cada célula seja a mesma em cada domínio. Os sub domínios são então cortados utilizando-se recursivamente o mesmo algoritmo até que o número de partições seja igual ao número de processadores.

A implementação do balanço de carga foi feita utilizando a biblioteca Zoltan Devine *et al.* (2002), o qual é constituído de um conjunto de ferramentas para balanço de carga e gerenciamento de dados em paralelo. Para utilização do pacote são necessárias chamadas

de funções, que determinam o conjunto de dados necessários para que se possa chamar a função responsável pelo balanço de carga. Este conjunto de dados é composto por:

- Número de elementos em todo o domínio e em cada pré-partição;
- vetor contendo os pesos (carga computacional) de cada elementos;
- vetor contendo a posição geométrica de cada elemento;
- vetor contendo os índices globais e locais (de cada processo).

Após a construção do conjunto de dados mencionado acima, deve-se escolher o algoritmo desejado para balanço de carga, por meio da seleção de um parâmetro. Após todas as configurações, chama-se a função *LB_Partition* da biblioteca Zoltan, a qual retornará a nova partição.

Deve-se dar atenção especial para a construção do vetor contendo os pesos de cada célula do nível base $l = l_{bot}$, o qual é responsável pelo balanço propriamente dito. No presente trabalho, este vetor é construído somando-se recursivamente as malhas do nível mais refinado até o nível mais grosso. Uma representação esquemática do método utilizado segue abaixo.

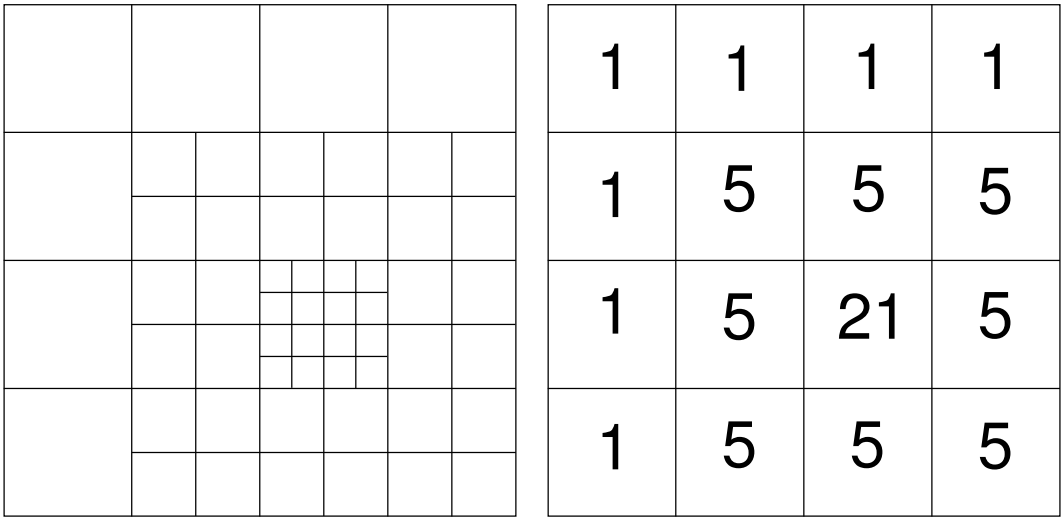


Figura 3.14: Representação esquemática do método utilizado para balanço de carga. a) malha adaptativa com 3 níveis. b) Malha computacional do nível base $l = l_{bot}$ com os pesos de cada célula.

O primeiro passo do algoritmo proposto para balanço de carga consiste em particionar o domínio sem balanço de carga, o que pode ser feito distribuindo-se as partições

uniformemente entre os processos utilizados. A Fig. 3.15 apresenta uma malha composta bidimensional com nível base $l = l_{bot}$ 12×12, dividida em quatro processos, onde cada processo possui uma malha com nível base $l = l_{bot}$ 6×6. Após a determinação da partição de domínio inicial, Fig. 3.15, é necessário construir o conjunto de dados necessários para a utilização da biblioteca Zoltan. A partir desses dados, determina-se uma nova partição, com balanço de carga, e reconstrói-se a hierarquia de malhas.

Segue abaixo o algoritmo proposto para partição de domínio com balanço de carga utilizado no presente trabalho.

1. Geração da malha em paralelo sem balanço de carga;
2. construção do conjunto de dados necessários para utilização da biblioteca Zoltan, usando dados da partição gerada inicialmente;
3. determinação da nova partição utilizando a função LB_Partition do pacote Zoltan;
4. reconstrução da hierarquia de malhas ou corte e redistribuição dos blocos entre os processos

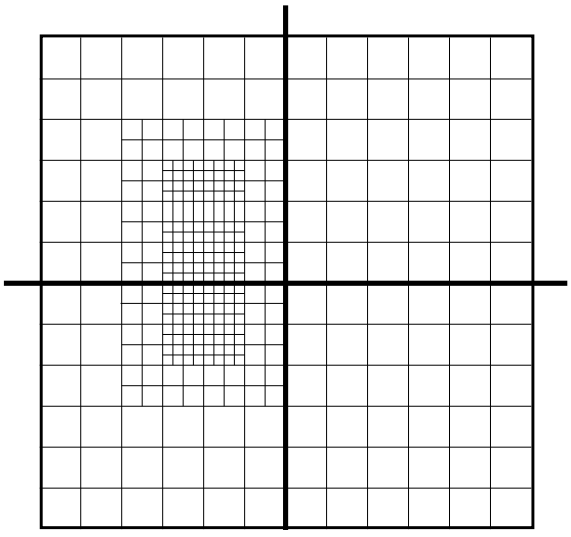


Figura 3.15: Exemplo de malha adaptativa com três níveis de refinamento particionada em quatro processadores sem balanço de carga. sem balanço de carga.

3.9.1 Particionamento dinâmico de domínio

Ao se empregar malhas adaptativas para a captura e representação satisfatórias de fenômenos físicos envolvidos em um escoamento, é necessário que as regiões de interesse do

domínio sejam cobertas pelas malhas dos níveis mais finos. Como as equações diferenciais parciais utilizadas são dependentes do tempo, as regiões de interesse variam com o tempo. Desta forma as malhas precisam se adaptar ao longo do tempo à região de interesse, de forma que a malha do nível base $l = l_{bot}$ permaneça fixa e malhas de outros níveis $l = lbot + 1...l_{top}$ acompanhem o desenvolvimento do fenômeno em estudo.

O refinamento local adaptativo permite a adição ou a remoção de malha onde se faz necessário, recorrendo apenas a interpolações trilineares para o transporte da solução entre as diferentes configurações de malhas. A Fig. 3.16 mostra esquematicamente o processo de remalhamento com particionamento dinâmico de domínio para uma malha composta dividida em quatro processadores. As figuras 3.16(a) e 3.16(d) representam, respectivamente, uma malha composta antes e após o remalhamento.

O procedimento de remalhamento com particionamento dinâmico de domínio é descrito a seguir, com base na Fig. 3.16. Inicialmente, marca-se em cada processador as células para refino, conforme a Fig. 3.16(b). Em seguida, cria-se em processamento paralelo uma hierarquia de malhas auxiliar a partir dos pontos selecionados, contendo apenas seus dados geométricos, conforme a Fig. 3.16(c). Efetua-se uma nova partição de domínio levando em conta a malha auxiliar recém criada, gerando os processos $1', 2', 3'$ e $4'$, conforme a Fig. 3.16(d). Para a utilização da biblioteca Zoltan no particionamento de domínio que gera os processos $1', 2', 3'$ e $4'$, utiliza-se o particionamento da Fig. 3.16(a) como particionamento de domínio inicial. Deve-se ressaltar que existem, em um mesmo passo de tempo, duas configurações de malha e dois particionamentos de domínio: 1 a 4 para a malha inicial e $1'$ a $4'$ para a malha auxiliar recém criada. Portanto este ponto é considerado crítico em termos de consumo de memória virtual.

As variáveis presentes na nova malha, Fig. 3.16(d), são obtidas por meio de interpolações trilineares provindas da malha antiga, Fig. 3.16(a), exceto nas regiões onde as malhas são coincidentes. Para a atualização em paralelo da nova malha é necessário, para cada processo da Fig. 3.16(d), percorrer todos os processos da Fig. 3.16(a) em busca de células coincidentes para serem simplesmente "injetadas" da malha nova, ou em busca de células para o preenchimento do estêncil da interpolação.

É importante ressaltar que todo o processo de construção da hierarquia de malhas descrito na subseção 3.5.1 é refeito no processo de remalhamento, assim como todo o precedimento utilizado para o particionamento do domínio e balanceamento da carga computacional, conforme a seção 3.9. No presente trabalho, o remalhamento é feito após uma determinada quantidade de passos no tempo, a ser determinada pelo problema a ser resolvido.

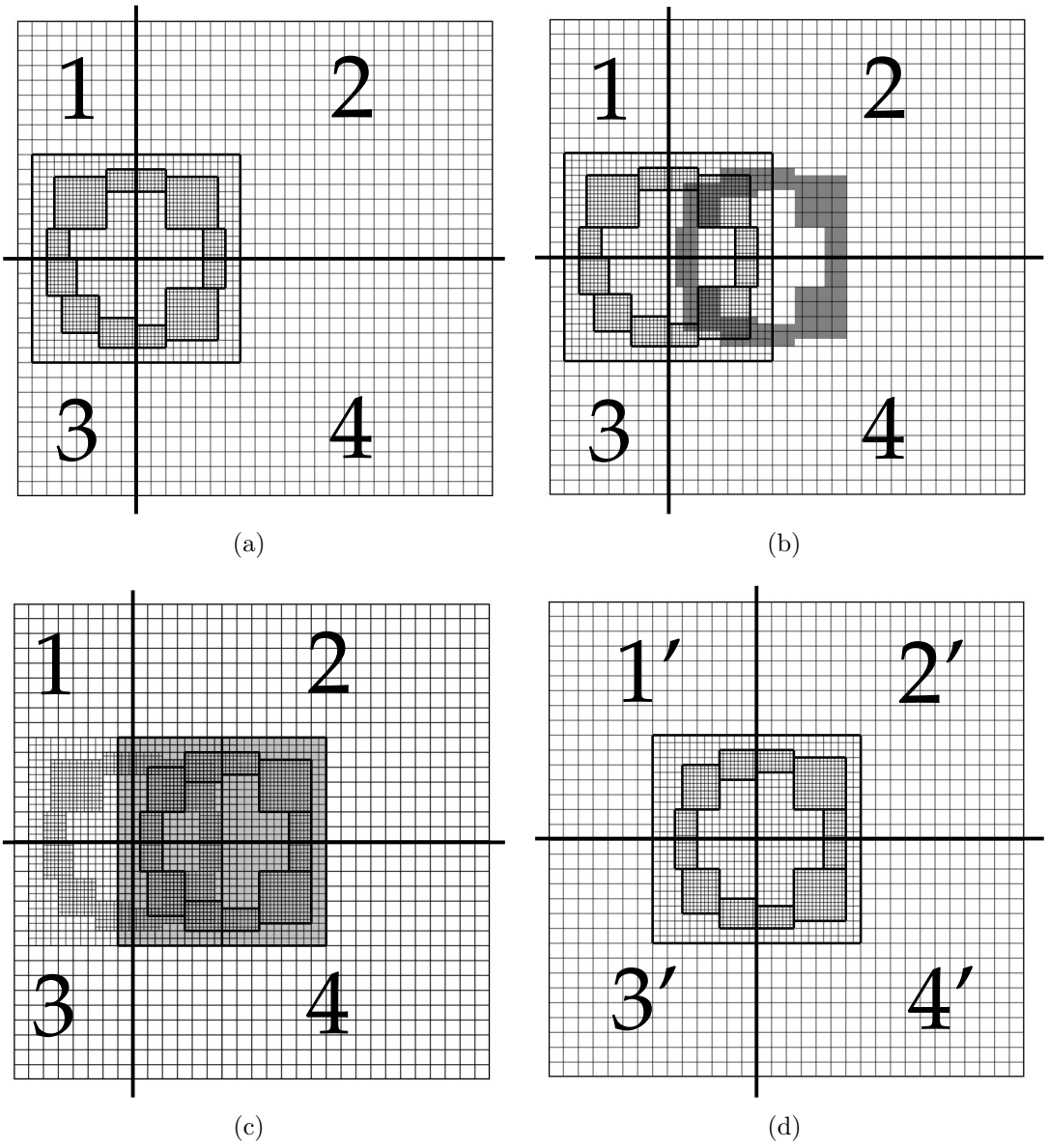


Figura 3.16: Malha composta dividida em quatro processadores representando esquematicamente o processo de partição dinâmica de domínio. (a) Malha composta antes do remalhamento, (b) seleção de células para geração da nova malha, (c) Geração da nova malha, (d) malha composta após o remalhamento.

3.10 Cálculo das células fantasmas em paralelo

Para o cálculo das células fantasmas em paralelo, é necessário dar atenção aos passos 3 e 4 da seção 3.6, os quais necessitam de células pertencentes a blocos adjacentes. Primeiro é necessário armazenar os dados geométricos dos blocos que tocam as bordas de cada domínio local em vetores auxiliares. Em seguida, estes dados são enviados para os processos vizinhos via MPI_SEND/MPI_RECV.

Visando evitar o envio desnecessário de dados pela rede, somente as células que tocam as fronteiras dos processos são comunicadas via MPI_SEND/MPI_RECV. Como no presente trabalho usa-se somente uma camada de células fantasmas e devido às interpolações quadráticas entre malhas fina/grossa, veja Fig. 3.9, é necessária a comunicação de uma camada contendo duas células em cada direção. A Fig. 3.17 representa esquematicamente tal situação. Observa-se que caso sejam necessárias duas ou mais camadas de células fantasmas, o estêncil de comunicação deve ser aumentado quando a partição de domínio é efetuada.

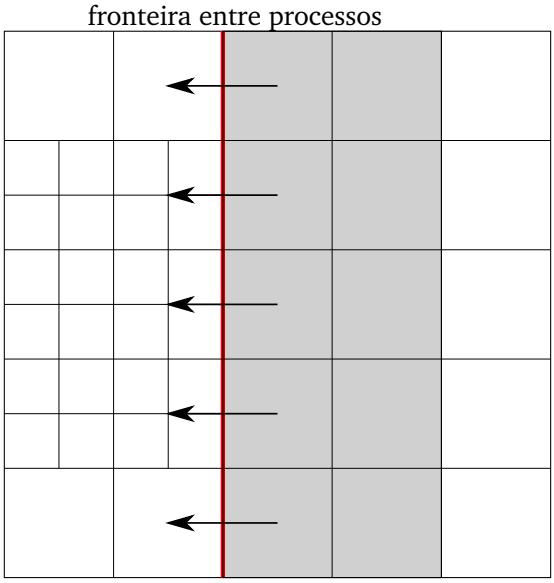


Figura 3.17: Representação esquemática simplificada da comunicação de dados via MPI para interpolação fina-grossa

Como o cálculo das células fantasmas é feito nível a nível, e sendo necessário trabalhar com os níveis l e *l-1*, ao calcular as fantasmas do nível l, é necessário comunicar os valores de l e *l-1* em uma única mensagem (MPI_SEND), evitando assim o desperdício de comunicação via rede. O algoritmo detalhado do processo de paralelização do cálculo das células fantasmas é dado por:

1. "percorre-se" os blocos à procura dos que tocam as fronteiras locais de cada processo;
2. "envia-se" os dados geométricos dos blocos selecionados para os processos vizinhos via MPI_SEND;
3. "percorre-se" os blocos selecionados no passo 1 à procura das células que tocam as fronteiras. Lembrando que sempre é necessário uma camada extra de células devido às interpolações quadráticas no cálculo das células fantasmas. Caso seja necessária

a utilização de mais de uma camada de células fantasmas, aumenta-se a camada de células a ser transferida;

4. "envia-se" dos dados das contidos nas células para os respectivos vizinhos via MPI_SEND;

5. aplicação dos mesmos algoritmos sequenciais, agora utilizando os dados recebidos da vizinhança via MPI_RECV.

3.11 Método Multinível-*Multigrid* em paralelo

Métodos *Multigrid* estão entre os métodos numéricos mais rápidos para resolver sistemas esparsos de equações lineares. Porém sua paralelização pode ser comprometida pela razão entre computação e comunicação nas malhas mais grosseiras, onde o tempo de comunicação pode ser tão elevado a ponto de comprometer completamente a eficiência paralela do código. Além disso, a partir de um certo nível de refinamento, o número de células da malha pode ser menor que o número de processos, tornando inviável o particionamento.

Uma das abordagens encontradas na literatura para a solução de sistemas de equações em paralelo consiste no chamado método de decomposição de domínio (MDD). O MDD é baseado no particionamento do domínio computacional em subdomínios, de modo que a solução global do problema é obtida pela combinação apropriada das soluções obtidas em cada um dos subdomínios. No presente trabalho, a paralelização do método *Multigrid* segue o princípio básico do MDD. O domínio é particionado geometricamente e os ciclos são executados paralelamente em cada subdomínio.

A paralelização do método *Multigrid* envole as três componentes básicas do método: relaxação, prolongamento e restrição. No relaxador *Gauss-Seidel Red-Black* (*GSRB*), utilizado no presente trabalho, cada iteração é composta por dois passos. No primeiro passo todos os pontos vermelhos são tratados independentemente. No segundo passo os pontos pretos são tratados utilizando os valores atualizados dos pontos vermelhos. De acordo com Trottenberg e Schuller (2001) esta característica favorece a paralelização do método *GSRB*.

A Figura 3.18 representa esquematicamente uma iteração do relaxador *GSRB* para um domínio particionado em quatro subdomínios, cada subdomínio em um processador. Observa-se que para a atualização da célula vermelha em destaque, são necessárias duas células pretas que estão em outros processos: uma acima e outra à direita. Neste caso, os valores das células pretas são armazenados em células fantasmas, as quais já foram atualizadas na iteração anterior. Dessa forma, observa-se que, no presente trabalho, a

paralelização do relaxador recai na paralelização do cálculo das células fantasmas, a qual é discutida na seção 3.10.

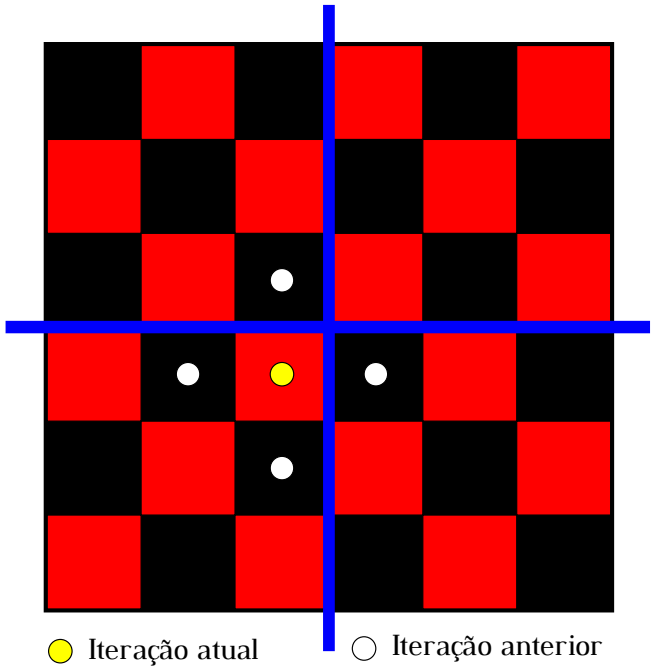


Figura 3.18: Representação esquemática de uma iteração do relaxador *GSRB* em quatro processos.

No processo de restrição são utilizadas médias aritméticas, utilizando somente células do nível fino l contidas em uma célula de nível mais grosso $l-1$. A Fig. 3.19 representa um esquema bidimensional do processo de restrição para dois subdomínios, cada subdomínio em um processador. Observa-se que não é necessária a utilização de células adjacentes para o cálculo, tornando desnecessária a comunicação de dados via MPI caso a célula do nível $l-1$ esteja na fronteira do processo.

No processo de prolongamento são utilizados polinômios trilineares, interpolando valores contidos em células do nível grosso $l-1$ para o nível fino l. A Fig. 3.20 representa um esquema bidimensional do prolongamento para dois subdomínios, cada subdomínio em um processador. Observa-se que é necessária a utilização de células adjacentes para a interpolação, tornando necessária a comunicação de dados via MPI caso a célula do nível $l-1$ esteja na fronteira do processo.

O procedimento empregado na paralelização do prolongamento é semelhante ao do cálculo em paralelo das células fantasmas. As células que estão nas fronteiras dos processos são selecionadas e armazenadas em vetores auxiliares. Em seguida enviam-se os dados armazenados nos vetores auxiliares para os respectivos processos vizinhos via MPI_SEND. Após o recebimento dos dados, via MPI_RECV, estes são armazenados em vetores auxili-

ares ou zona de *buffer*. Um esquema representativo deste processo é descrito na Fig.3.21. Observa-se que os dados do *processador B* são enviados à zona de *buffer* do processador *processador A*.

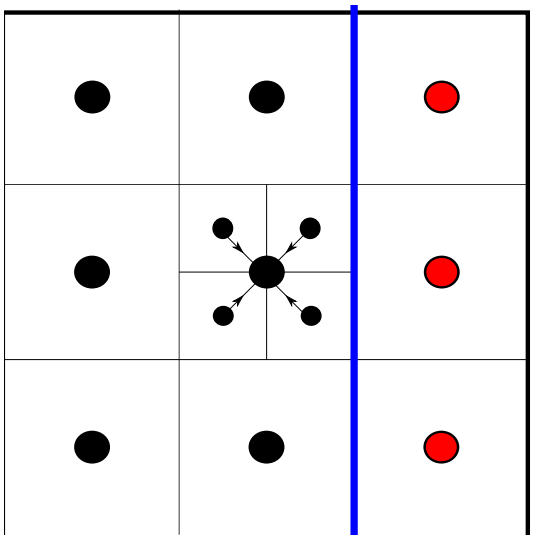


Figura 3.19: Representação esquemática bidimensional do processo de restrição para dois subdomínios, separados por uma linha vertical.

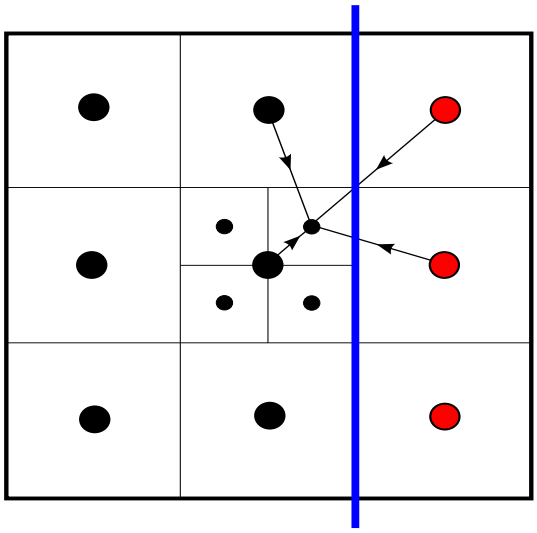


Figura 3.20: Representação esquemática bidimensional do processo de prolongamento para dois subdomínios, separados por uma linha vertical.

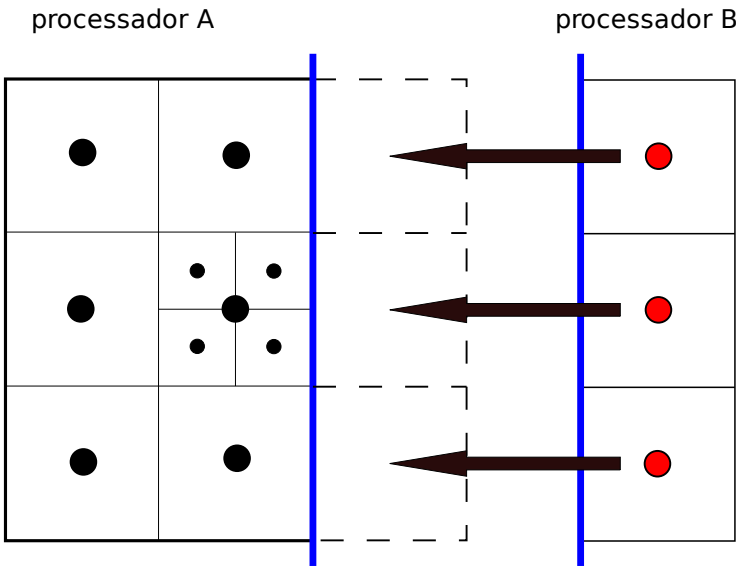


Figura 3.21: Representação esquemática bidimensional do procedimento empregado na paralelização do prolongamento.

Capítulo 4

Resultados

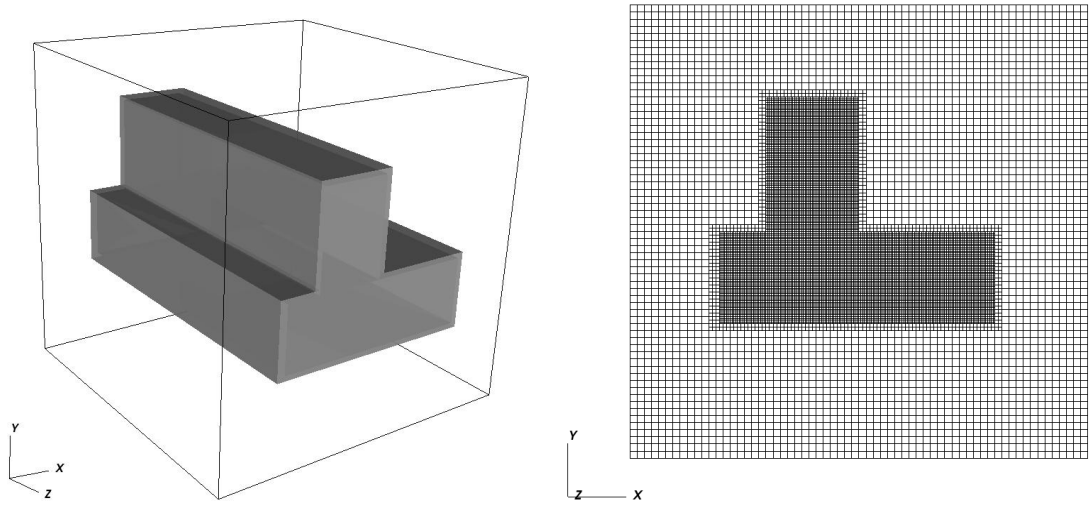
Os resultados apresentados neste capítulo estão divididos em três partes:

1. testes para verificação do balanço de carga;
2. testes de convergência numérica por refinamento de malha em processamento paralelo;
3. análise do desempenho em paralelo da metodologia de paralelização;
4. testes de validação da metodologia proposta em processamento paralelo;

Inicialmente, verifica-se a metodologia empregada na paralelização do método Multinível-*Multigrid* por meio de testes de convergência para a solução numérica de uma equação de Poisson. Posteriormente, toda a metodologia empregada na solução numérica das Equações de Navier-Stokes também é verificada por meio de testes de ordem de convergência.

Os testes para verificação do balanço de carga são realizados para malha com nível base $l_{base} = 32^3$ com quatro níveis de refinamento ($32L4$), para quatro e oito processos e com domínio computacional de dimensão unitária. O objetivo destes testes é avaliar a distribuição das células computacionais entre os processos e o ganho obtido, em termos de tempo computacional, com o balaço de carga.

Todos os testes de convergência (seções 4.1.1 e 4.1.2) são realizados em uma malha com dois níveis de refinamento, para quatro processos. A malha composta utilizada é apresentada na Fig. 4.1.



(a) (b)

Figura 4.1: Malha composta ($32L3$), utilizada nos testes de convergência numérica por refinamento de malha. (a) Vista em perspectiva, (b) secção transversal em $z = 0,5$.

4.1 Balanço de Carga para Partição de Domínio

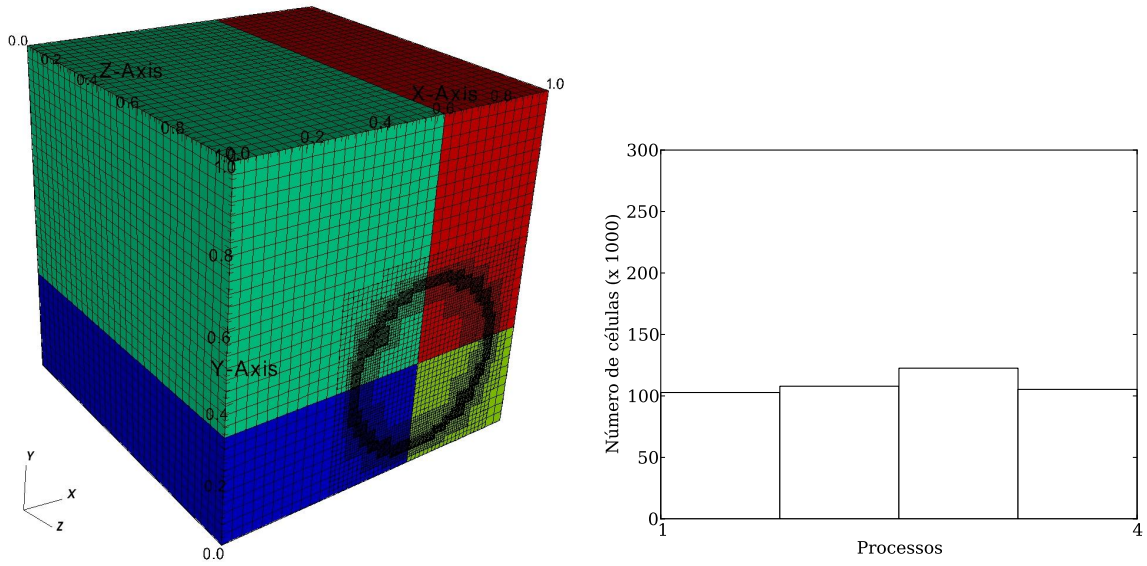
Os testes de balanço de carga foram efetuados para malhas estáticas com nível base $l_{base} = 32^3$ mais três níveis de refinamento, geradas de acordo com a seção 3.5.1. As configurações de malha para a realização dos testes são apresentadas a seguir:

1. Malha bloco-estruturada que recobre uma esfera com centro em $(x_{ct}, y_{ct}, z_{ct}) = (0,6; 0,4; 1,0)$ e raio $0,125$;

2. Malha bloco-estruturada que recobre o plano $y = \alpha x$ para $z = [0,0 : 1,0]$.

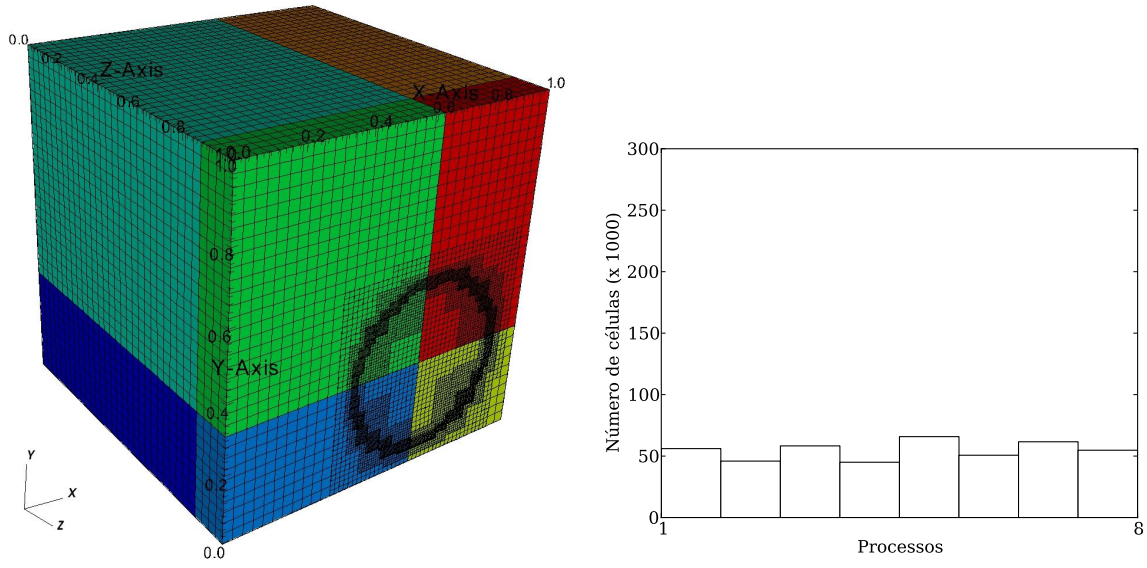
A carga computacional é representada pelo número de células computacionais contidas em cada processo. Nas figuras 4.2 a 4.5, cada cor representa uma partição de domínio ou processo. Observa-se nos resultados apresentados pelas figuras 4.2 a 4.5, que a distribuição da carga computacional por processador apresenta uma boa uniformidade. Observa-se uma variação da carga computacional entre 10% e 30% para as malhas apresentadas, o que reflete em um ganho em eficiência de paralelização significativo para a presente tese.

A figuras 4.6 e 4.7 mostram as distribuições das células computacionais entre quatro



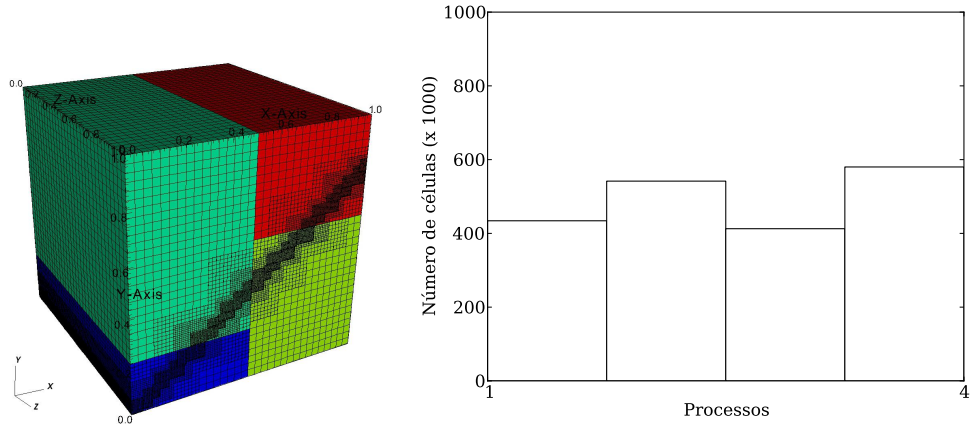
(a) (b)

Figura 4.2: Malha bloco-estruturada $32L4$ que recobre uma esfera esfera em quatro processos: a) visualização; b) distribuição da carga entre os processos.



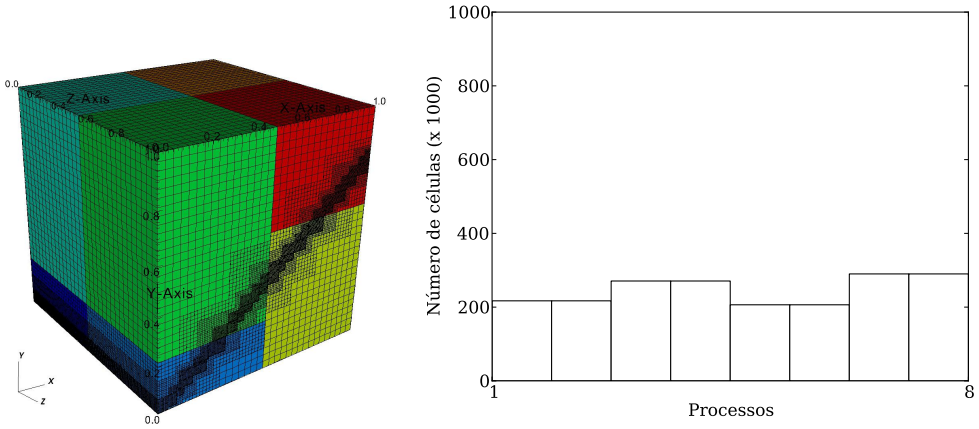
(a) (b)

Figura 4.3: Malha bloco-estruturada $32L4$ que recobre uma esfera em oito processos: (a) visualização;(b) distribuição da carga entre os processos.



(a) (b)

Figura 4.4: Malha bloco-estruturada $32L4$ que recobre um plano em quatro processos: a) visualização; b) distribuição da carga entre os processos.



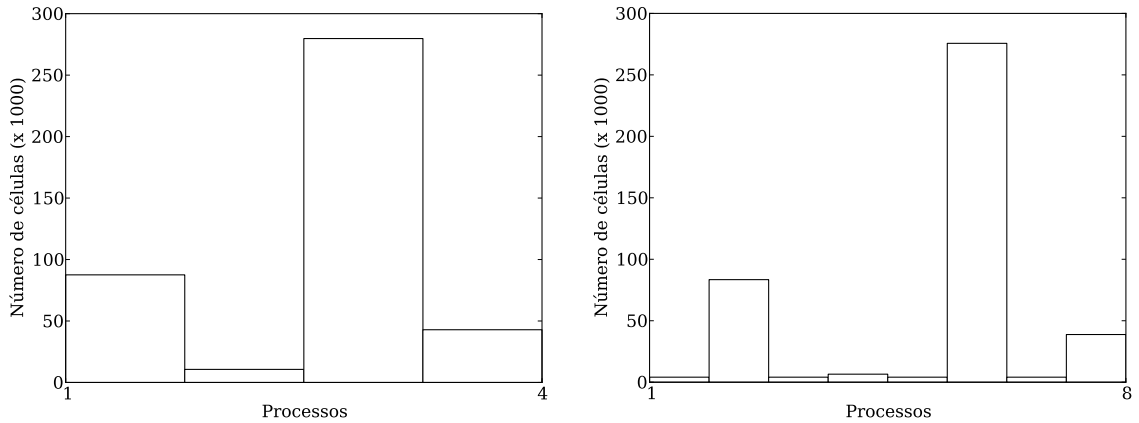
(a) (b)

Figura 4.5: Malha bloco-estruturada $32L4$ que recobre um plano em oito processos: a) visualização; b) distribuição da carga entre os processos.

e oito processos sem balanço de carga. Os resultados das figuras 4.6(a) e 4.6(b) foram obtidos com as mesmas malhas das figuras 4.2(a) 4.3(a). Os resultados das figuras 4.7(a) e 4.7(b) foram obtidos com as mesmas malhas das figuras 4.4(a) 4.5(a).

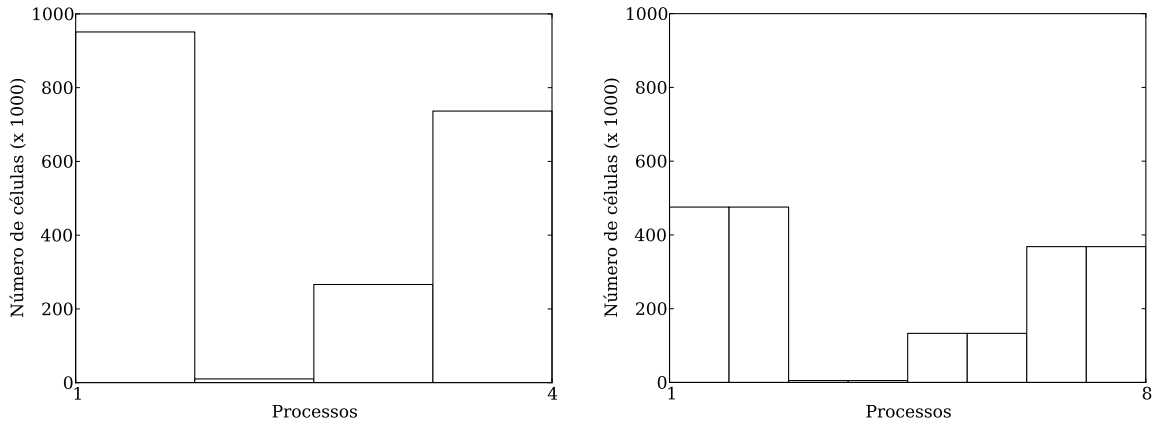
Observa-se uma distribuição irregular das células computacionais, o que provoca sobrecarga em alguns processadores, afetando drasticamente o desempenho em paralelo. Na configuração de malha da Fig. 4.2(a), em um $PC\ IntelCored2 - quad$ utilizando quatro processos, o tempo computacional gasto em um passo de integração no tempo, para a solução das equações de Navier - Stokes, foi de $67s$ em serial, $48s$ em paralelo sem balanço de carga e $29s$ em paralelo com balanço de carga. Tal resultado reflete em um ganho de

tempo computacional significativo para a presente tese.



(a) (b)

Figura 4.6: Distribuição das células computacionais em uma malha $32L4$ que recobre uma esfera em: (a) quatro processo; (b) oito processos.



(a) (b)

Figura 4.7: Distribuição das células computacionais em uma malha $32L4$ que recobre um plano em: (a) quatro processo; (b) oito processos.

4.1.1 Testes de Convergência para Equação de Poisson

Nesta subseção, é apresentada uma análise da convergência espacial para a solução de uma equação de Poisson em processamento paralelo, dada por

$$\nabla \cdot \left(\frac{1}{\rho} \nabla \phi \right) = f, \tag{4.1}$$

na qual ρ é a densidade, ϕ é a função a ser determinada e f é chamado de termo forçante, pois "força"a solução da equação 4.1 para a solução exata, ϕ_e, dada por:

$$\phi_e = cos(2\pi x + 2\pi y + 2\pi z), \tag{4.2}$$

e ρ assume valor constante $\rho = 1$. A malha composta utilizada é representada na Fig. 4.1.

As tabelas 4.1 a 4.4 apresentam resultados para a razão de convergência (r_e), baseados na norma L_2, bem como o número de ciclos (NC) gastos pelo método Multinível-*Multigrid* para a solução da Eq. 4.1, considerando condições de contorno de Dirichlet (valor imposto no contorno) e Neumman (derivada imposta no contorno). As tabelas. 4.1 e 4.3 apresentam resultados em processamento sequencial e as tabelas 4.2 e 4.4 apresentam resultados em processamento paralelo. Percebe-se dois fatos importantes: primeiro o número de iterações não muda com o número de processadores e segundo a ordem de convergência também não se altera significativamente.

Tabela 4.1: Teste de convergência para malha composta utilizando condições de contorno de Dirichlet nas direções x, y, z e massa específica constante $\rho = 1$ em um domínio $\Omega[0; 1]^3$

Equação Elíptica - Serial				
Malha base (l_{base})	Número de Ciclos	$\|\phi - \phi_e\|_2$	r_e	Resíduo
16 x 16 x 16L2	14	1,1686E-6	-	2,007E-4
32 x 32 x 32L2	15	3,6788E-7	3,17	6,3452E-5
64 x 64 x 64L2	16	1,0873E-7	3,38	1,8818E-5
128 x 128 x 128L2	17	3,1903E-8	3,41	5,5340E-6

Tabela 4.2: Teste de convergência em paralelo para malha composta utilizando condições de contorno de Dirichlet nas direções x, y, z e massa específica constante $\rho = 1$ em um domínio $\Omega[0; 1]^3$.

Equação Elíptica - Paralelo				
Malha base	Número de ciclos	$\|\phi - \phi_e\|_2$	r_e	Resíduo
16 x 16 x 16L2	14	1,1684E-6	-	2,0071E-4
32 x 32 x 32L2	15	3,6788E-7	3,18	6,3452E-5
64 x 64 x 64L2	16	1,0873E-7	3,38	1,8818E-5
128 x 128 x 128L2	17	3,1903E-8	3,41	5,5339E-6

Tabela 4.3: Teste de convergência em serial para malha composta utilizando condições de contorno de Neumann nas direções x, y, z e massa específica constante $\rho = 1$ em um domínio $\Omega[0;1]^3$

Equação Elíptica - Serial				
Malha base (l_{base})	Número de Ciclos	$\|\phi - \phi_e\|_2$	r_e	Resíduo
16 x 16 x 16L2	11	2,9395E-4	-	2,7208E-4
32 x 32 x 32L2	14	1,0102E-4	2,91	3,9948E-5
64 x 64 x 64L2	17	2,5559E-5	3,95	8,8495E-6
128 x 128 x 128L2	19	6,2328E-6	4,1	6,3389E-6

Tabela 4.4: Teste de convergência em paralelo para malha composta utilizando condições de contorno de Neumann nas direções x, y, z e massa específica constante $\rho = 1$ em um domínio $\Omega[0;1]^3$.

Equação Elíptica - Paralelo				
Malha base	Número de ciclos	$\|\phi - \phi_e\|_2$	r_e	Resíduo
16 x 16 x 16L2	11	2,9391E-4	-	2,7198E-4
32 x 32 x 32L2	14	1,0105E-4	2,91	3,9980E-5
64 x 64 x 64L2	17	2,5559E-5	3,95	8,8031E-6
128 x 128 x 128L2	19	6,2343E-6	4,01	6,3134E-6

4.1.2 Testes de Convergência para Equações de Navier - Stokes

Nesta subseção, verifica-se o esquema numérico empregado na solução das equações de Navier-Stokes para um escoamento incompressível em processamento paralelo. Para tanto, as equações de Navier-Stokes são acrescidas de um termo forçante $\mathbf{f}$, calculado por meio das soluções exatas para as velocidades u, v, w, para o campo de pressão p, para a densidade ρ e para a viscosidade μ. As equações de Navier-Stokes para um escoamento incompressível são dadas por

$$\rho\big(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}\big) = -\nabla p + \nabla \cdot \big[\mu\big(\nabla \mathbf{u} + \nabla \mathbf{u}^T\big)\big] + \mathbf{f}, \tag{4.3}$$

na qual o termo forçante $\mathbf{f}$ é dado por

$$\mathbf{f} = \rho_e \frac{\partial \mathbf{u_e}}{\partial t} + \rho_e \cdot \mathbf{u_e}(\nabla \mathbf{u_e}) + \nabla p_e - \nabla \cdot (\mu_e \nabla \mathbf{u_e}). \tag{4.4}$$

As soluções exatas para o campo de velocidade $\mathbf{u}_e$, para o campo de pressão p_e, para a densidade ρ_e e para a viscosidade μ_e são dadas por

$$\begin{aligned} u_e &= sen^2(2\pi x + 2\pi y + 2\pi z + t), \\ v_e &= -cos^2(2\pi x + 2\pi y + 2\pi z + t), \\ w_e &= 2cos^2(2\pi x + 2\pi y + 2\pi z + t), \\ p_e &= cos(2\pi x + 2\pi y + 2\pi z + t) \\ \rho_e &= 1 + sen^2(2\pi x), \\ \mu_e &= 1 + 0,2cos^2(2\pi x + 2\pi y + 2\pi z + t), \end{aligned} \tag{4.5}$$

na qual as soluções exatas para u_e, v_e, w_e devem ser escolhidas de forma a garantir que $\nabla \mathbf{u} = 0$. A Tabela 4.5 apresenta resultados para a razão de convergência (r_e), baseados

Tabela 4.5: Teste de convergência em paralelo para malha composta utilizando condições de contorno de Dirichlet para a pressão e Neumman para as velocidades nas direções x, y, z em um domínio $\Omega[0;1]^3$.

Malha	p		w		v		w	
	$\|\phi - \phi_e\|_2$	r_e	$\|\phi - \phi_e\|_2$	r_e	$\|\phi - \phi_e\|_2$	r_e	$\|\phi - \phi_e\|_2$	r_e
16^3L2	0,581	-	1,57E-2	-	1,50E-2	-	2,50E-2	-
32^3L2	0,195	2,98	4,05E-3	3,88	4,01E-3	3,74	6,92E-3	3,61
64^3L2	0,064	3,04	1,05E-3	3,85	1,03E-3	3,89	1,84E-3	3,75

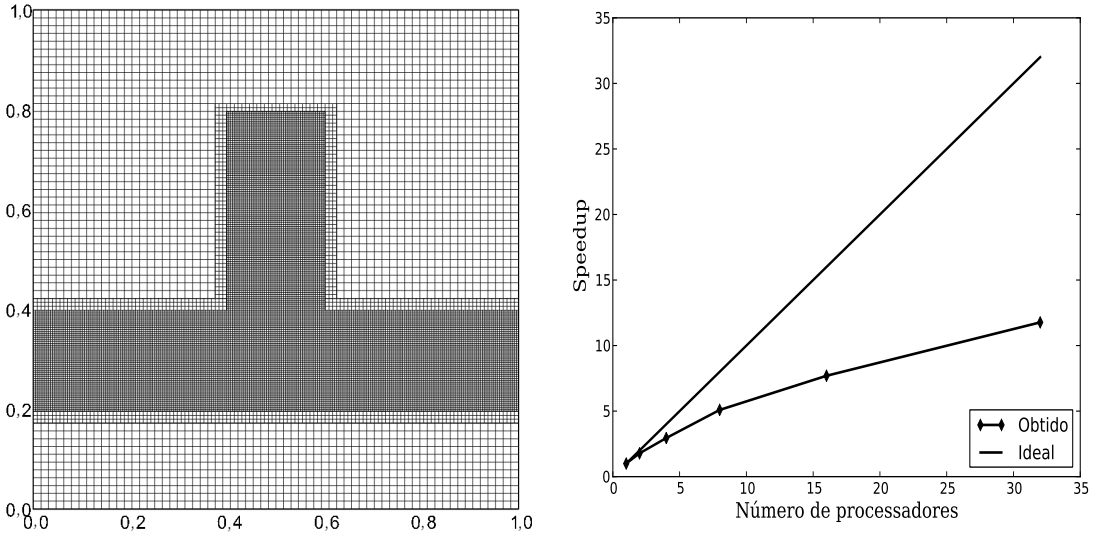
na norma L_2, para a velocidade nas direções x, y, z e para a pressão. Os testes são efetuados utilizando-se condições de contorno de Dirichlet para a pressão e Neumman para as velocidades, para um tempo de simulação $t = 1.0s$. Observa-se que todas as velocidades apresentam segunda ordem para convergência espacial, enquanto que a pressão apresenta no mínimo primeira ordem.

4.2 Análise do desempenho em paralelo

Nesta seção, faz-se uma análise do desempenho da metodologia de paralelização proposta. São apresentados testes para duas configurações de malhas estáticas, mostradas nas figuras 4.8(a), 4.11(a), e para uma configuração de malhas dinâmicas, mostrada na Fig.4.14. Nos testes apresentados, verifica-se a influência do particionamento de domínio em parâmetros como número de células e de *patches*. Os testes de *speedup* são realizados em 20 passos de tempo (ct), avaliando-se a norma L2 para as velocidades e para a pressão. No cálculo da norma, utiliza-se os campos de velocidades e pressão analíticos, dados pela Eq. 4.5.

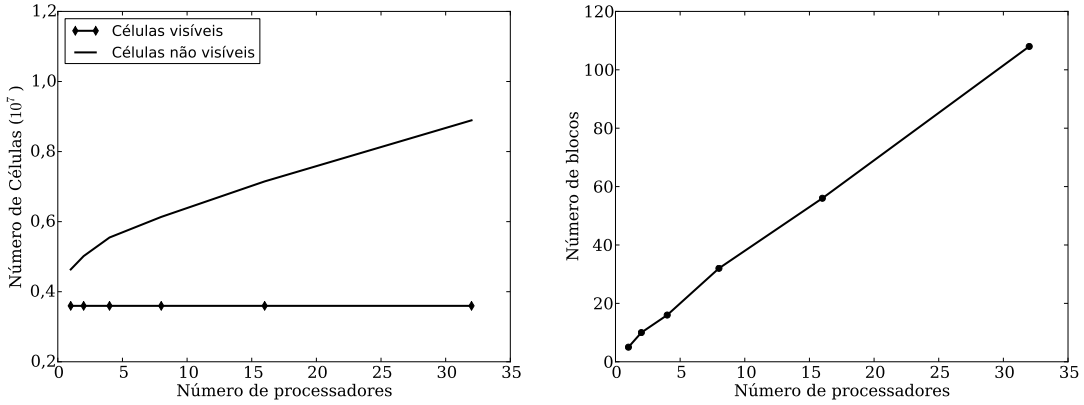
Nas figuras 4.8(a) e 4.8(b), são apresentados respectivamente a malha utilizada e o *speedup* obtido. Nota-se que o desempenho em paralelo diminui mais acentuadamente para um número maior que 16 processos. Apesar de o desempenho apresentar uma perda em torno de 65 % em relação ao ideal, há uma diminuição de 12 vezes no tempo de cálculo com 32 processos.

A norma L2 para pressão e para as velocidades u, v, w, avaliada durante 20 passos no tempo é apresentada nas figuras 4.10(a) a 4.10(d). Cada gráfico é composto por oito curvas, que representam o valor da norma para 1, 2, 4, 8, 16 e 32 processos. Observa-se nas figuras. 4.10(a) a 4.10(d) que as curvas se sobrepõem, mostrando que o valor da norma, para uma mesma malha, se apresenta independente do número de processos.



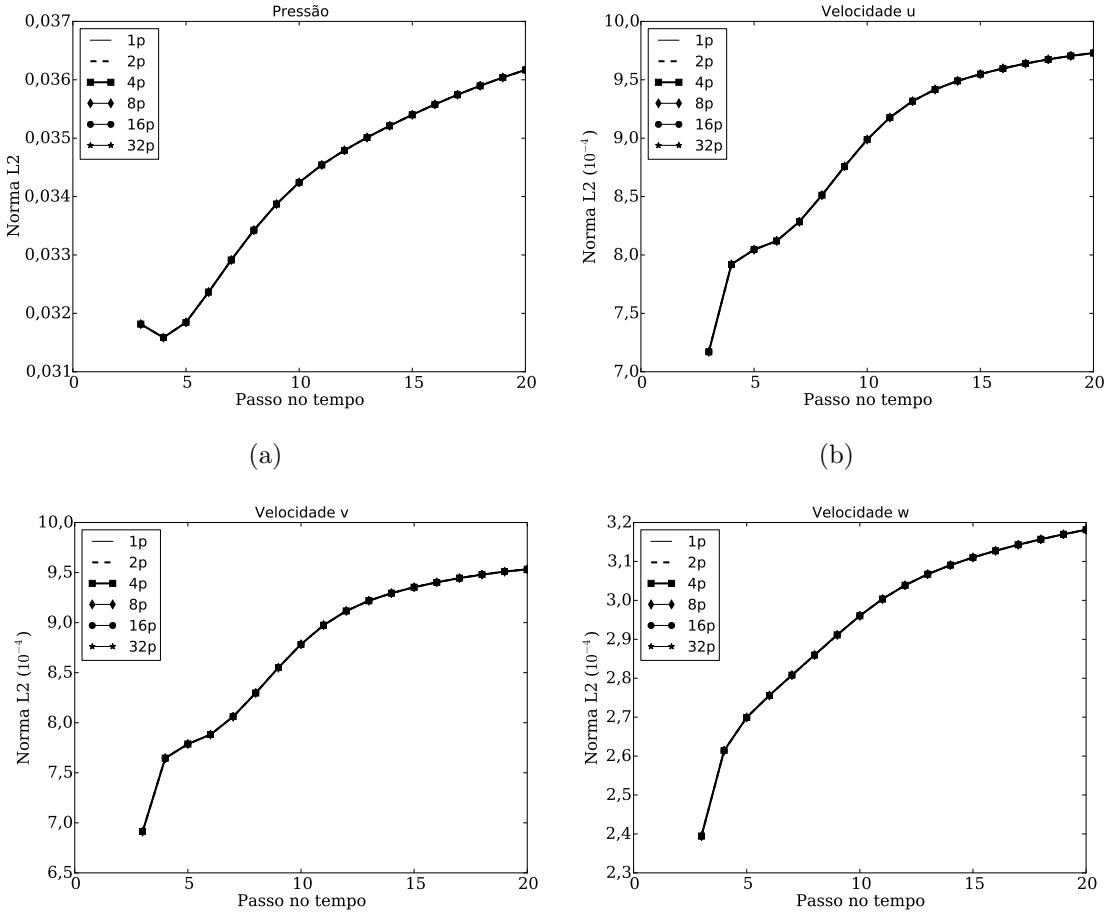
(a) (b)

Figura 4.8: Malha 64^3 com três níveis de refinamento e *speedup* obtido. (a) Secção transversal em $z = 0,5$, (b) *speedup*.



(a) (b)

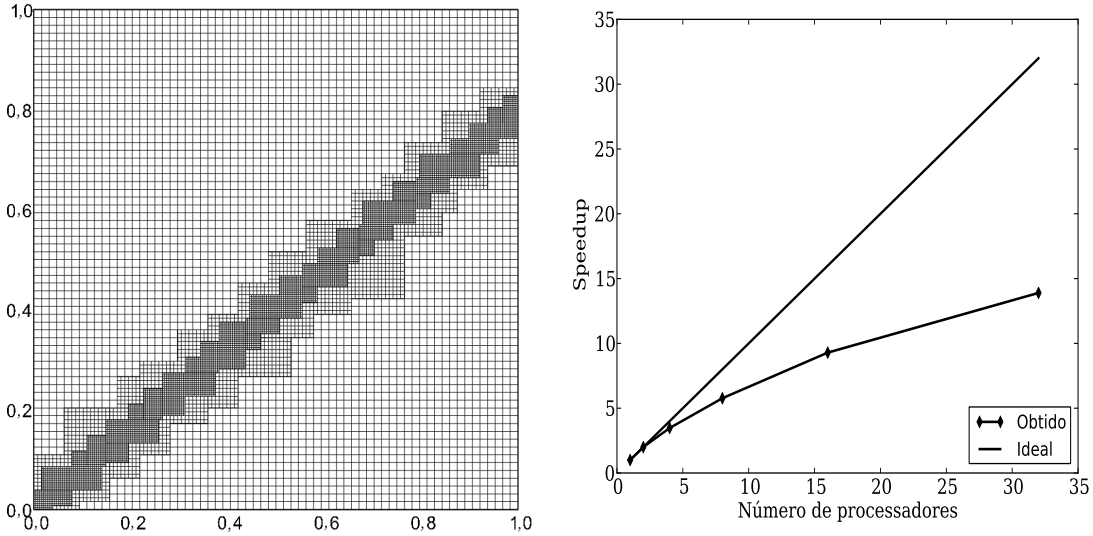
Figura 4.9: Influência da número de processos na quantidade de células e *patches*. (a) Número de células ao longo do tempo, (b) número de *patches* ao longo do tempo.



(a) (b)

(c) (d)

Figura 4.10: Norma L2 ao longo de 20 passos no tampo para: (a) pressão, (b) velocidade u, (c) velocidade v, (d) velocidade w



(a) (b)

Figura 4.11: Malha 64^3 com três níveis de refinamento e *speedup* obtido. (a) Secção transversal em $z = 0,5$, (b) *speedup*.

A Figura 4.9 mostra a influência da número de processos na quantidade de células e *patches*. Os gráficos das figuras 4.9(a) e 4.9(b) mostram que para a malha utilizada, Fig. 4.8(a), não há alteração do número de células visíveis quando se aumenta o número de processos. Porém, o número total de células e de *patches* apresenta um aumento significativo. O aumento no número de *patches* ocorre porque ao particiona-lo acrescenta-se um *patch* à estrutura de dados. Já o aumento do número total de células ocorre pois é necessário recobrir as faces particionadas com uma camada de células fantasmas. Na Fig. 4.9(a), o número total de células é dado pelo número de células visíveis e não visíveis, definidos na seção 3.8.3.

A malha e a respectiva curva de *speedup* para o segundo teste são apresentados nas figuras 4.11(a) e 4.11(b). Para esta configuração de malha, observa-se na curva de *speedup* apresentada na Fig.4.11(b), que o desempenho apresenta um ganho em torno de 15% em relação à curva mostrada na Fig.4.8(b). Este ganho pode ser explicado ao analisar as figuras4.13(b) e figuras4.9(b), onde observa-se que o aumento do número de *patches* é menos acentuado no caso da Fig.4.13(b). Uma quantidade menor de *patches* requer menos comunicação em paralelo, resultando em um melhor desempenho.

A norma L2 para pressão p e para as velocidades u, v, w, avaliada durante 20 passos no tempo é apresentada nas figuras 4.12(a) a 4.12(d). Cada gráfico é composto por oito curvas, que representam o valor da norma para 1, 2, 4, 8, 16 e 32 processos. Observa-se nas figuras 4.12(a) a 4.12(d) que, ao contrário do que foi mostrado nas figuras 4.10(a) a 4.10(d), as curvas não se sobrepõem. pois analisando-se a Fig. 4.13(a) nota-se que há

uma alteração no número de células visíveis ao aumentar o número de processos. Dessa forma, o valor da norma é alterado pois há uma alteração na malha, o que torna a norma dependente do número de processos.

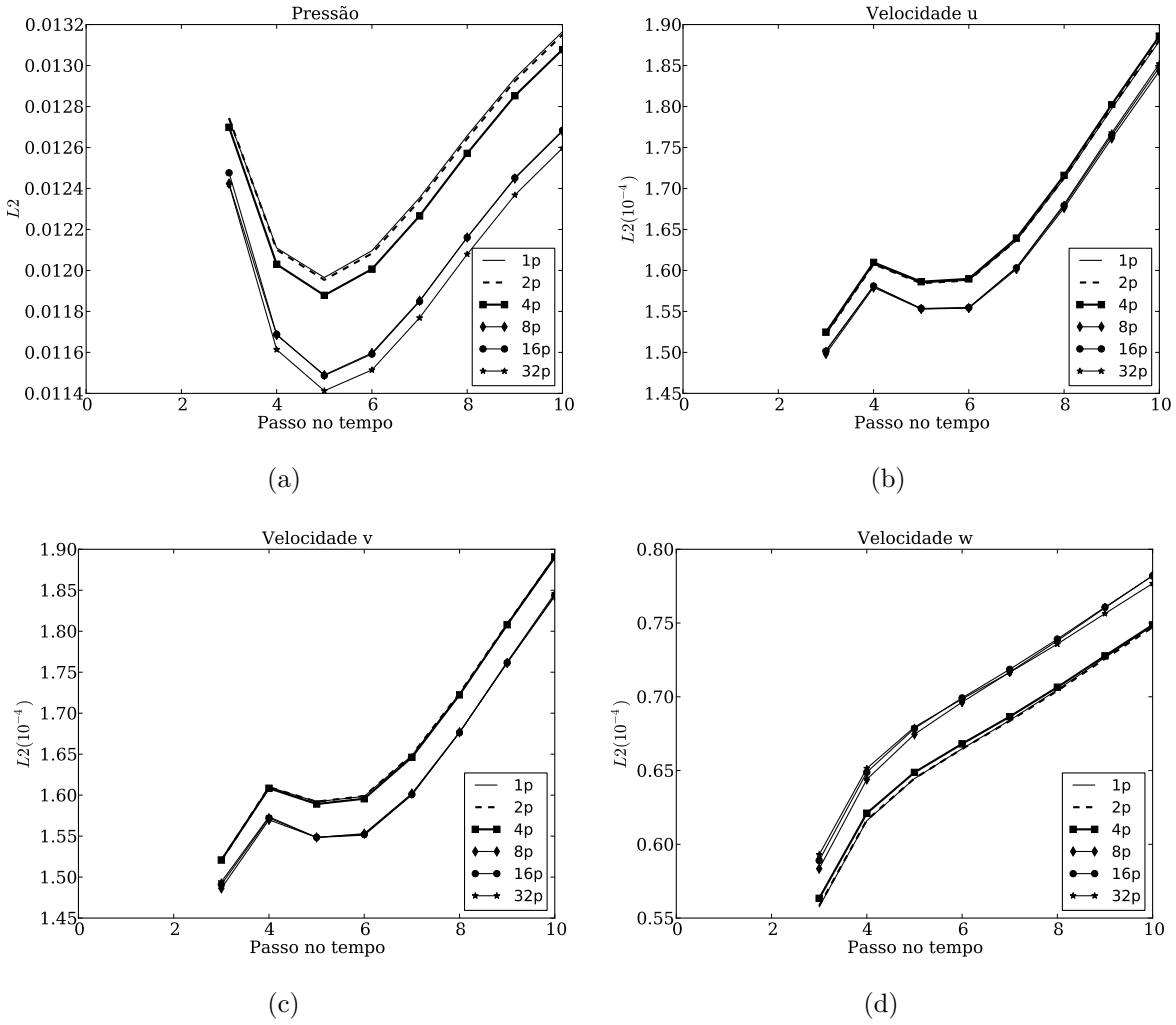
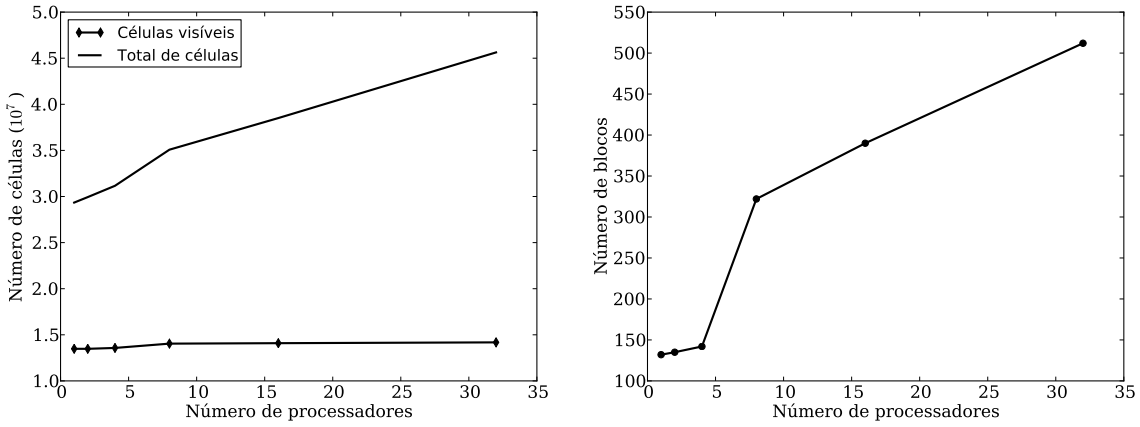


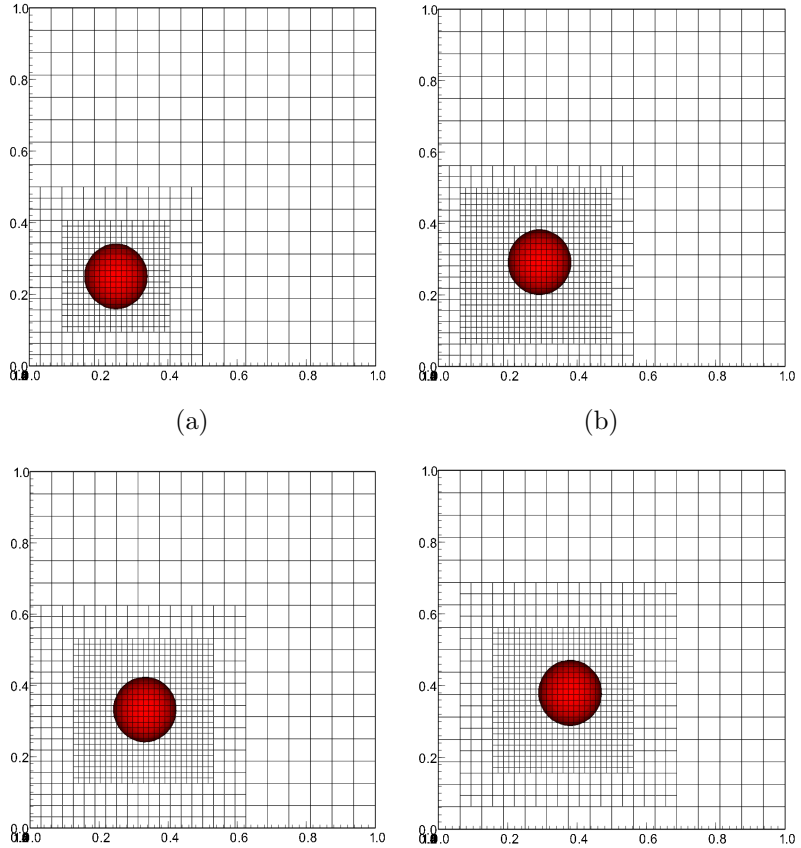
Figura 4.12: Norma L2 ao longo de 20 passos no tampo para: (a) pressão, (b) velocidade u, (c) velocidade v, (d) velocidade w

Para o teste de desempenho em paralelo em malha dinâmica, a malha composta foi refinada em função de um campo de densidade ρ variável com o tempo, como mostra a sequência de malhas apresentada na Fig.4.14. O processo de remalhamento é realizado três vezes ao longo de 20 passos no tempo (ct). Nota-se que o desempenho em paralelo apresenta uma perda em torno de 62% em relação à curva ideal. porém, observa-se que há uma diminuição em torno de 15 vezes no tempo de cálculo, utilizando 32 processos. Na Fig. 4.16(a), nota-se que as alterações bruscas nos valores das normas ocorrem devido ao processo de remalhamento.



(a) (b)

Figura 4.13: Influência da número de processos na quantidade de células e *patches*. (a) Número de células ao longo do tempo, (b) número de *patches* ao longo do tempo.



(a) (b)

(c) (d)

Figura 4.14: Evolução temporal do campo de densidade ρ sobreposta por uma malha composta $16L3$. (a) Malha inicial. (b) Passo de tempo $ct = 6$. (c) Passo de tempo $ct = 12$. (d) Passo de tempo $ct = 18$.

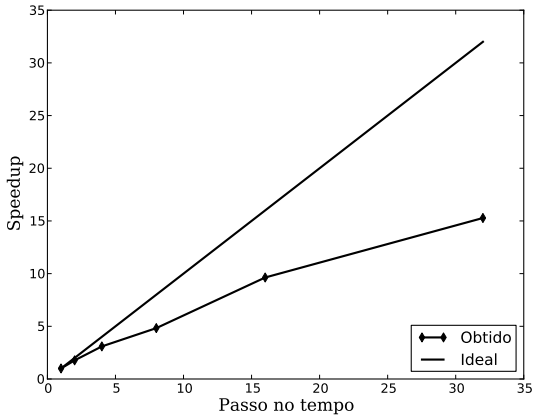


Figura 4.15: Speedup obtido para solução das Equações de Navier-Stokes durante 20 passos de tempo em uma malha composta dinâmica com nível base 16^3 mais dois níveis de refinamento.

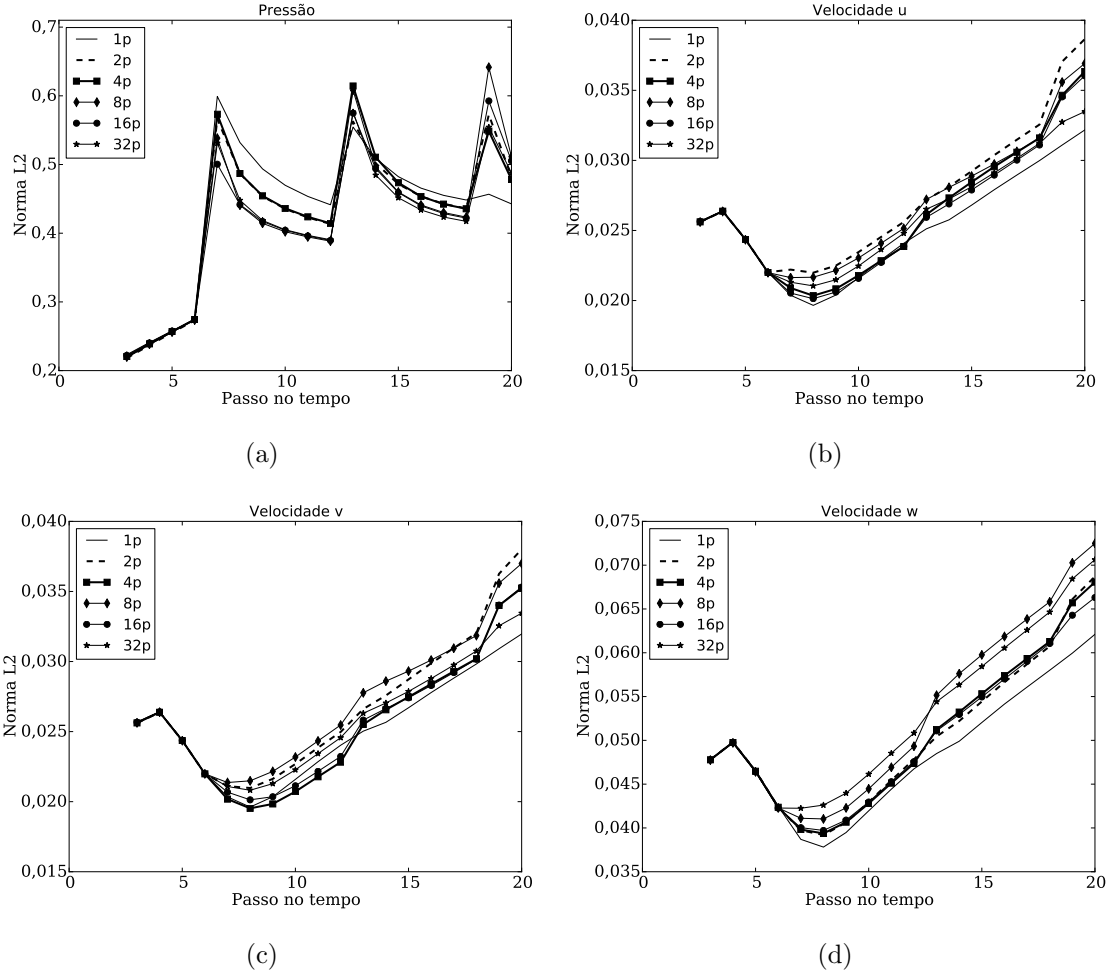


Figura 4.16: Norma L2 ao longo de 20 passos no tampo para: (a) pressão, (b) velocidade u, (c) velocidade v, (d) velocidade w

4.3 Cavidade com tampa deslizante

Para a validação da metodologia proposta, simulou-se o caso da cavidade com tampa deslizante para números de Reynolds iguais a 100, 400, 1.000 e 3.200. As dimensões do domínio computacional são $1m \times 1m \times 1m$, a velocidade de deslizamento da tampa é $u_0 = 1m/s$ e a densidade $\rho = 1kg/m^3$. Um esboço da cavidade simulada é apresentado na Fig. 4.17. A viscosidade é calculada por meio do número de Reynolds, ou seja,

$$\mu = \frac{\rho u_0 L}{Re}, \tag{4.6}$$

na qual $L = 1m$ é o comprimento da cavidade.

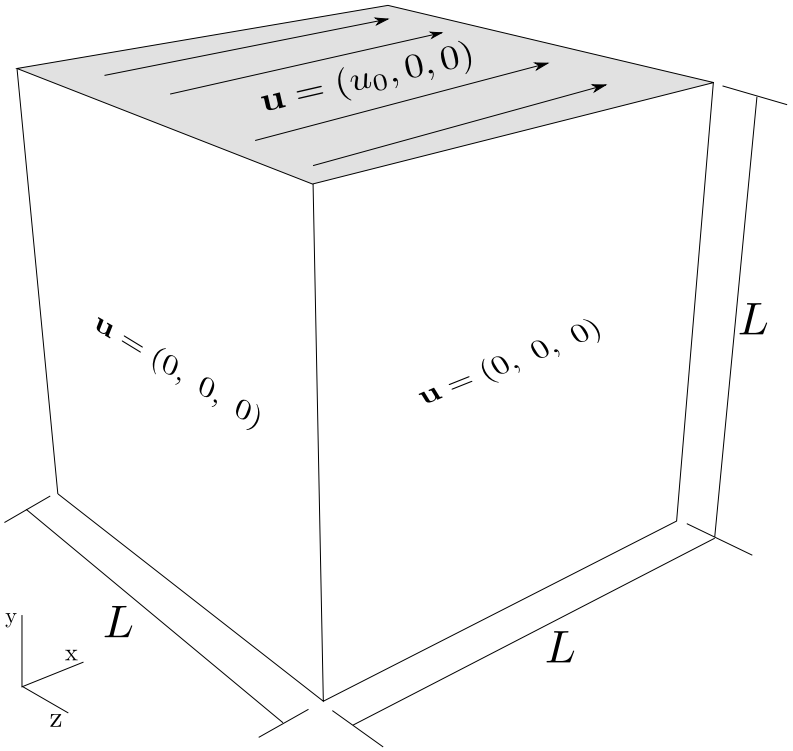


Figura 4.17: Esquema de uma cavidade com tampa delizando de comprimento L, utilizada para a validação da metodologia desenvolvida no presenta trabalho

Para os escoamentos a $Re = 100$ e $Re = 400$, os casos foram executados utilizando-se 16 processos, malha base 16^3 mais dois níveis de refinamento. Para $Re = 1000$ e $Re = 3200$, utilizou-se 32 processos, malha base 32^3 mais dois níveis de refinamento. Os resultados obtidos são comparados com os de Deshpande e Milton (1998) e Prasad e Koseff (1989) para $Re = 1000$ e $Re = 3200$ e com os de Ku, Hirsh e Taylor (1987) para $Re = 100$ e $Re = 400$. As comparações são feitas por meio do perfil da velocidade u ao longo do eixo vertical (y) da cavidade em $x/L = 0,5$ e $z/L = 0,5$ e do perfil da

velocidade v ao longo do eixo horizontal (x) da cavidade em $y/L = 0,5$ e $z/L = 0,5$. O passo de tempo, determinado pela condição CFL, é da ordem do espaçamento da malha no nível mais fino, ou seja $\Delta t = O(1/64)$ para $Re = 100$ e $Re = 400$ e $\Delta t = O(1/128)$ para $Re = 1000$ e $Re = 3200$.

As figuras 4.18 e 4.19 representam os perfis das componentes de velocidade u e v para $Re = 100$, os quais apresentam boa concordância com os resultados numéricos de Ku, Hirsh e Taylor (1987). O tempo de simulação para este caso foi de 33 minutos em paralelo e 300 minutos em serial, o que leva a um ganho de aproximadamente nove vezes no tempo total de cálculo. Observa-se nas figuras4.18 e 4.19 que há uma pequena diferença entre o resultado sequencial e o resultado em paralelo. Tal comportamento pode ser explicado pela Fig. 4.13(a), onde nota-se que há uma diferença entre o número de células visíveis do caso sequencial e paralelo.

Os perfis das componentes de velocidade u e v para $Re = 400$ estão representados nas figuras 4.20 e 4.21. Os resultados representam boa concordância com os resultados numéricos de Ku, Hirsh e Taylor (1987), exceto para os pontos de mínima velocidade v. Observa-se que estes valores são alcançados por Ku, Hirsh e Taylor (1987), os quais utilizam o método pseudo espectral de Chebyshev para a solução das equações de Navier-Stokes.

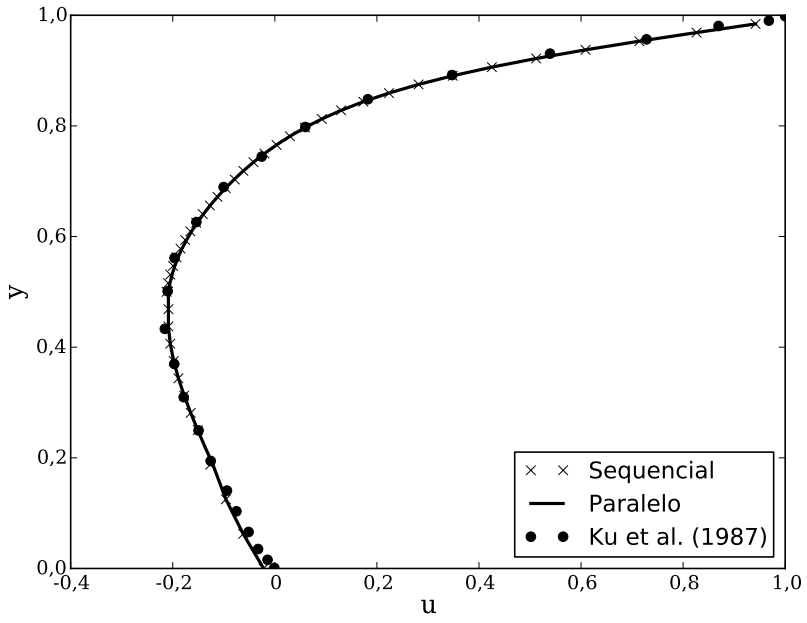


Figura 4.18: Comparação dos perfis da componente de velocidade u em $x/L = 0,5$ e $z/L = 0,5$ para $Re = 100$, obtidos com malha composta 16^3L3.

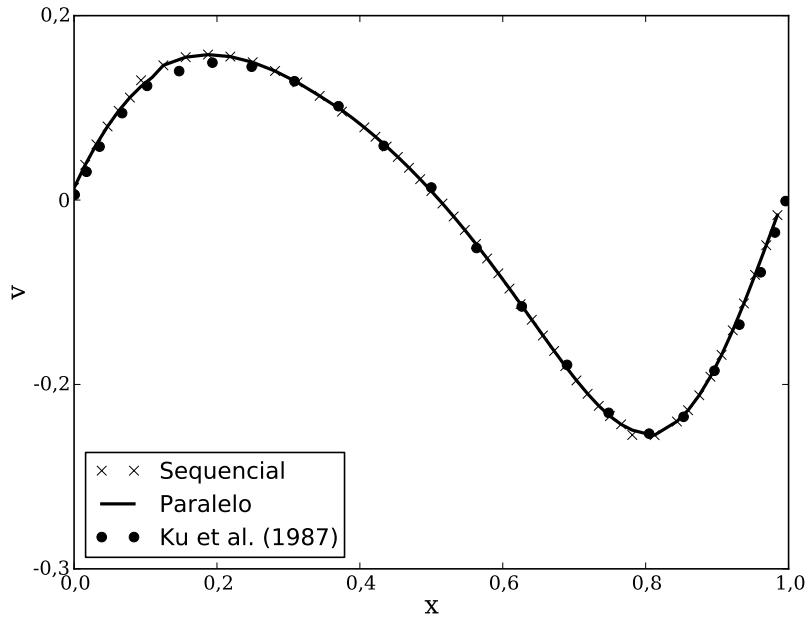


Figura 4.19: Comparação dos perfis da componente de velocidade v em $y/L = 0,5$ e $z/L = 0,5$ para $Re = 100$, obtidos com malha composta 16^3L3.

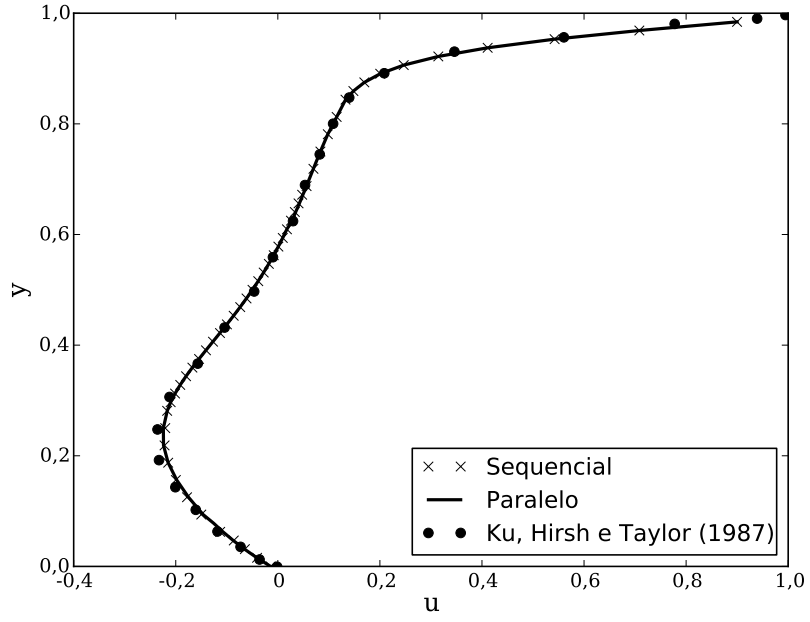


Figura 4.20: Comparação dos perfis da componente de velocidade u em $x/L = 0,5$ e $z/L = 0,5$ para $Re = 400$, obtidos com malha composta 16^3L3.

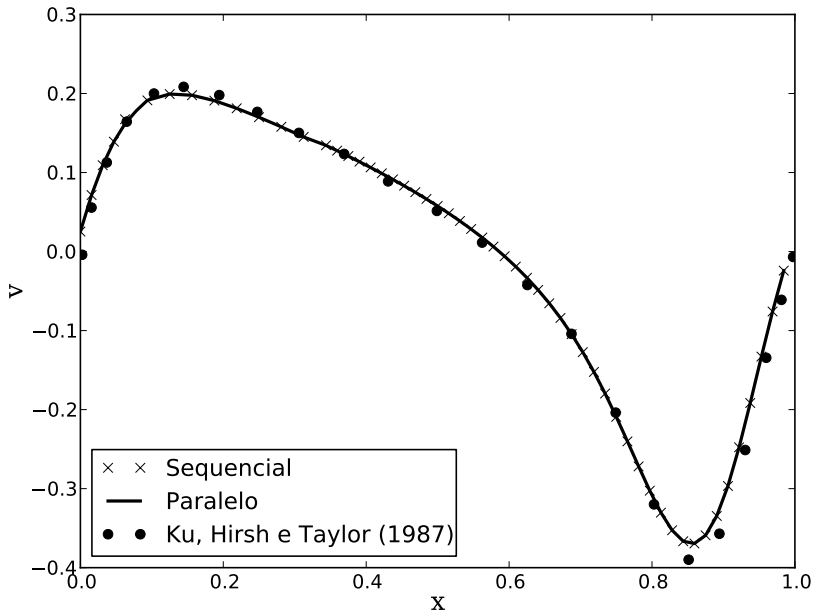


Figura 4.21: Comparação dos perfis da componente de velocidade v em $y/L = 0,5$ e $z/L = 0,5$ para $Re = 400$, obtidos com malha composta 16^3L3.

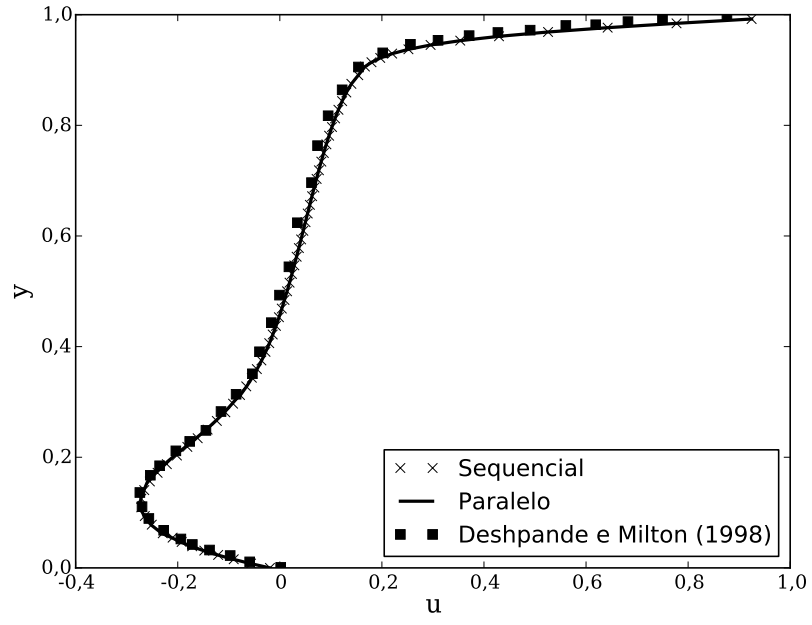


Figura 4.22: Comparação dos perfis da componente de velocidade u em $x/L = 0,5$ e $z/L = 0,5$ para $Re = 1000$, obtidos com malha composta 32^3L3.

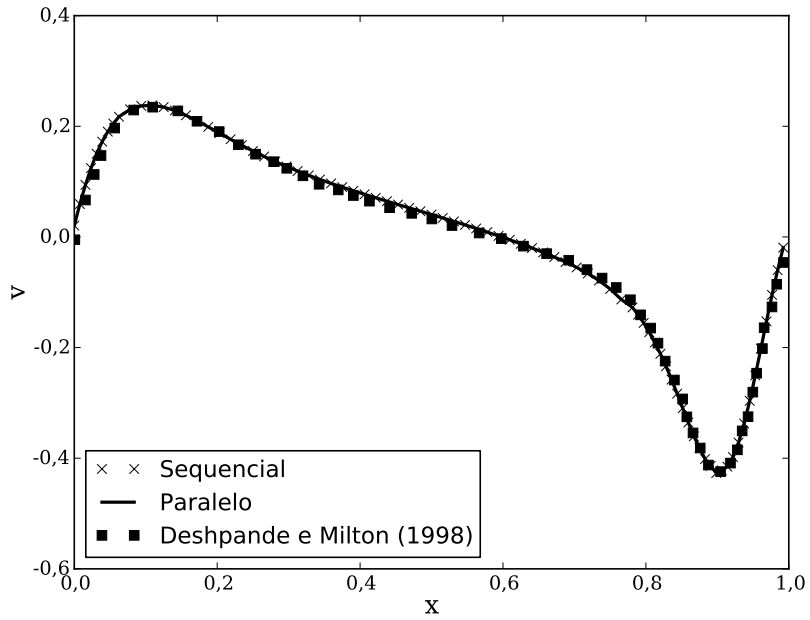


Figura 4.23: Comparação dos perfis da componente de velocidade v em $y/L = 0,5$ e $z/L = 0,5$ para $Re = 1000$, obtidos com malha composta 32^3L3.

O tempo de simulação para este caso foi de 40 minutos em paralelo e 420 minutos em serial, o que leva a um ganho de aproximadamente dez vezes no tempo total de cálculo. Para $Re = 1000$ os perfis das componentes de velocidade u e v apresentam boa concordância com os resultados numéricos de Deshpande e Milton (1998), conforme mostrado nas figuras 4.22 e 4.23. Utilizando 32 processadores o tempo de simulação para este caso foi de 5 horas e 30 minutos em paralelo e 67 horas em serial, o que corresponde a um ganho de aproximadamente 12 vezes no tempo total de cálculo. Para avaliar o ganho em termos de tempo computacional, obtido com o particionamento dinâmico de domínio, simulou-se o caso da cavidade com tampa deslizante para $Re = 1000$ e 3000 passos de tempo (≈ 10 s físicos) utilizando 16 processadores. O caso foi executado em 180 minutos sem balanço dinâmico de carga e em 90 minutos com balanço de carga.

Os resultados obtidos para $Re = 3.200$ são mostrados a seguir. Os perfis das componentes u e v são comparadas com os resultados experimentais de Prasad e Koseff (1989) e numéricos de Deshpande e Milton (1998). Observa-se uma boa concordância entre os resultados, exceto pelos pontos de velocidade máxima e mínima.

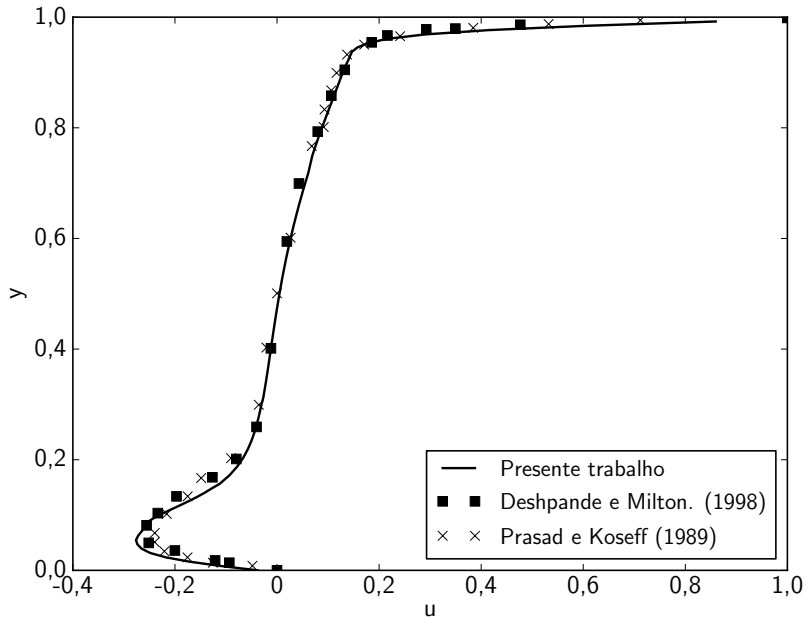


Figura 4.24: Comparação dos perfis da componente de velocidade u em $x/L = 0,5$ e $z/L = 0,5$ para $Re = 3200$, obtidos com malha composta 32^3L3.

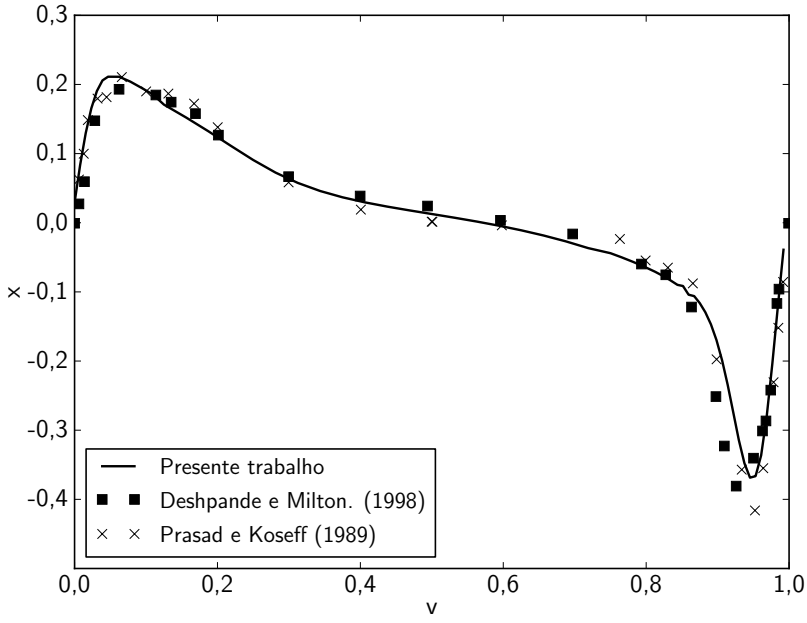
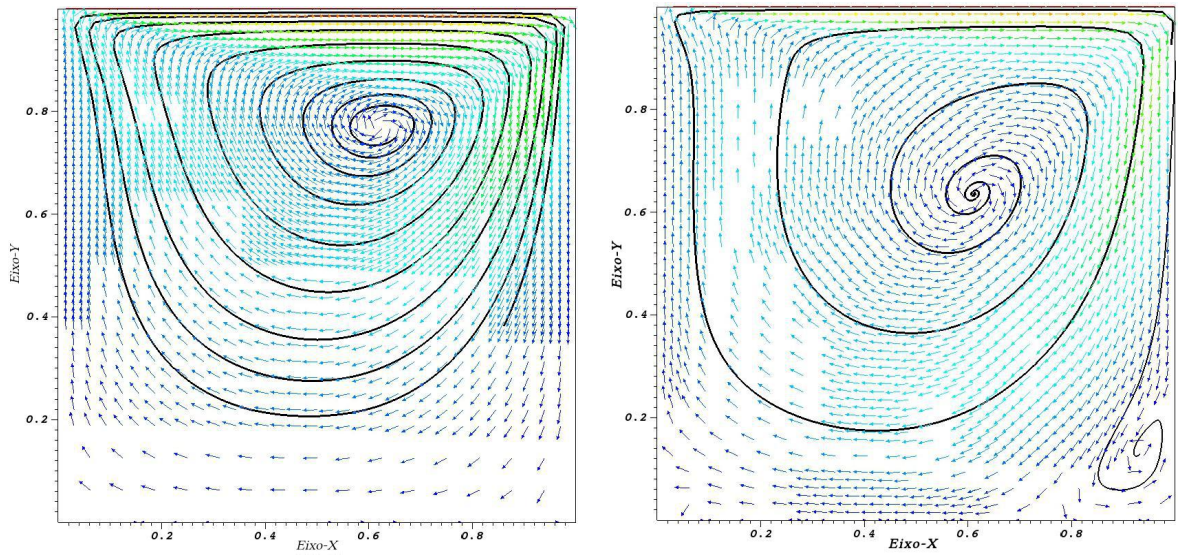


Figura 4.25: Comparação dos perfis da componente de velocidade v em $y/L = 0,5$ e $z/L = 0,5$ para $Re = 1000$, obtidos com malha composta 32^3L3.

4.3.1 Topologia dos escoamentos

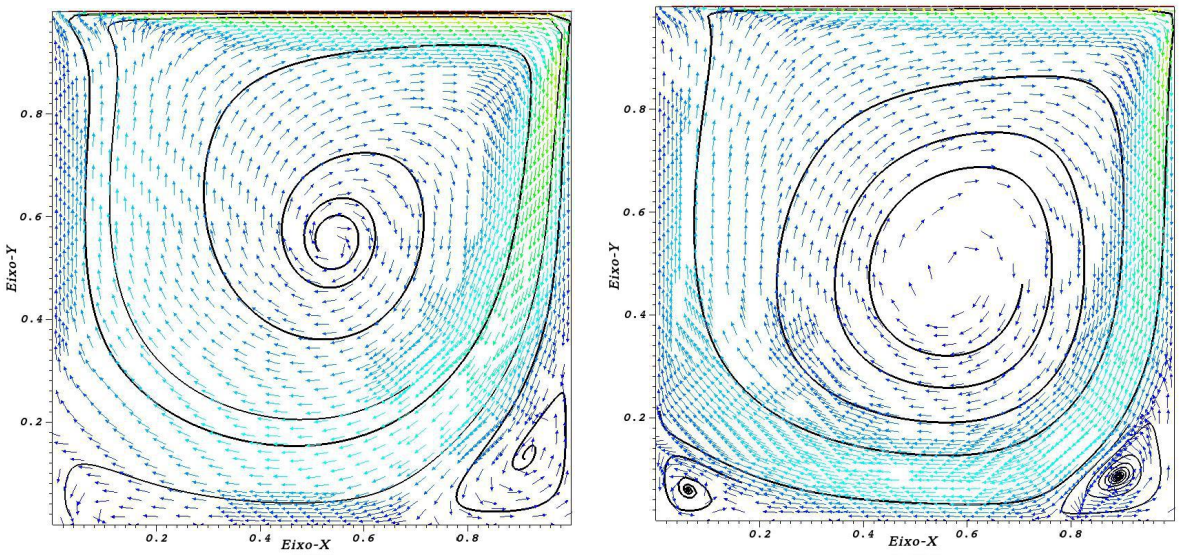
As figuras 4.26 e 4.27 mostram a representação vetorial do campo de velocidades sobrepostos por linhas de corrente. De acordo com a Fig.4.26(a), para o escoamento a $Re = 100$ observa-se somente a presença de uma recirculação central. Conforme aumenta-se o número de Reynolds, observa-se o surgimento de um vórtice secundário no canto inferior da cavidade, o qual pode ser observado nas figuras 4.26(b) e 4.27(a) e 4.27(b), para $Re = 400$, $Re = 1.000$ e $Re = 3.200$ respectivamente.



(a) (b)

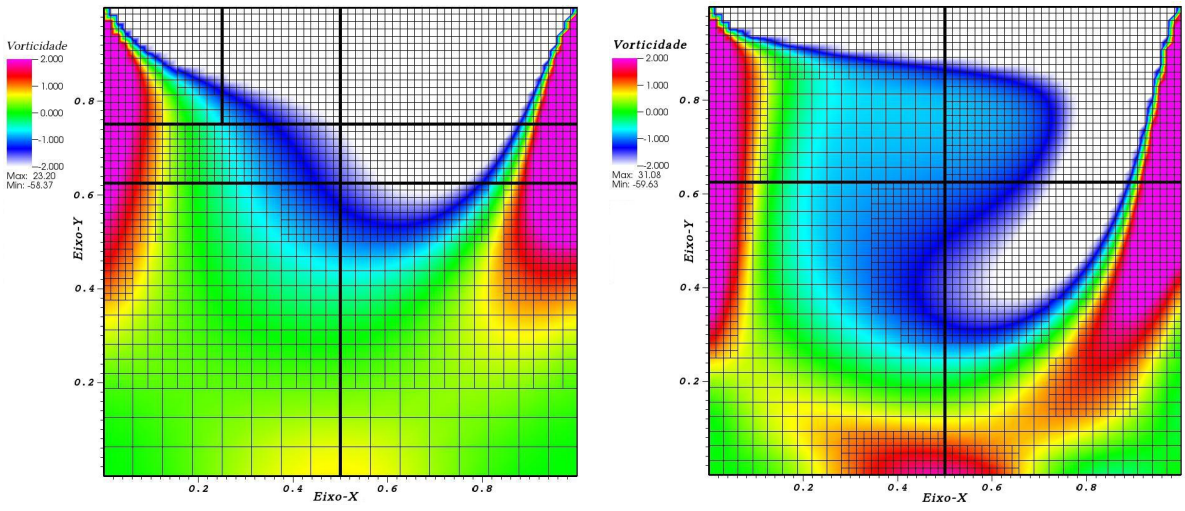
Figura 4.26: Representação vetorial do campo de velocidades com linhas de corrente sobrepostas no plano $x - y$ para $z/L = 0,7$ em malha composta $16^3 L3$.(a) $Re = 100$. (b) $Re = 400$.

As figuras 4.28 e 4.29 mostram o campo de vorticidade variando de $-2,0$ a $2,0$ sobreposto pelo refinamento adaptativo e pelas interfaces entre os processos. Para o caso da cavidade com tampa deslizante, a malha composta foi refinada em função da vorticidade, portanto, observa-se a presença de malhas mais finas nas regiões onde esta grandeza é mais elevada. Como o balanço de carga leva em consideração a quantidade de células presentes em cada processo, verifica-se que os retângulos com menor área apresentam uma quantidade maior de células refinadas.



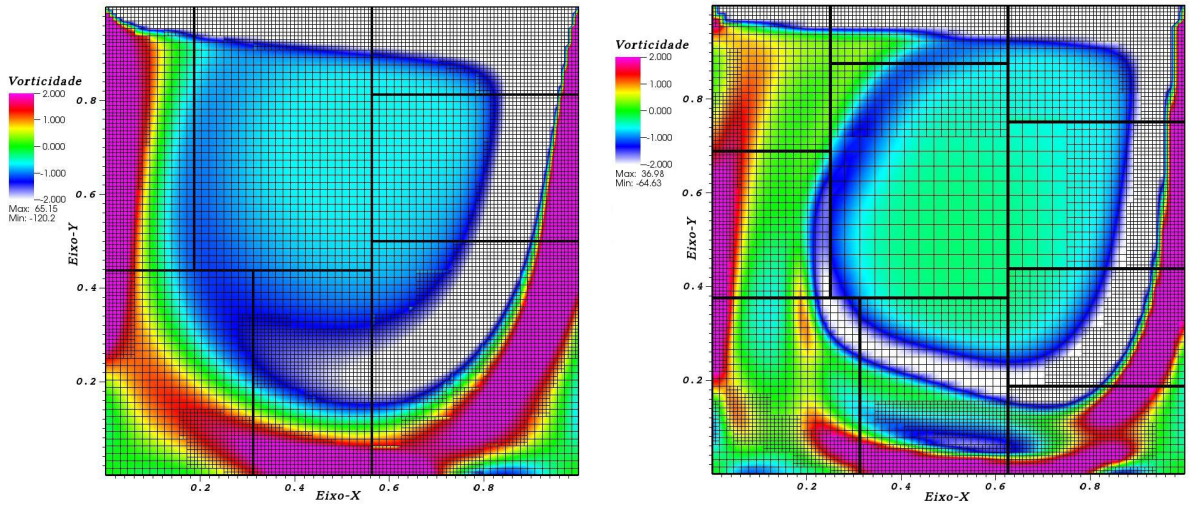
(a) (b)

Figura 4.27: Representação vetorial do campo de velocidades com linhas de corrente sobrepostas no plano $x - y$ para $z/L = 0,7$ em malha composta 32^3L3. (a) $Re = 1000$. (b) $Re = 3200$.



(a) (b)

Figura 4.28: Campo de vorticidade no plano $x - y$ em $z/L = 0,5$ sobreposto pelo refinamento adaptativo e pelas interfaces entre os processos em malha composta 16^3L3. (a) $Re = 100$; (b) $Re = 400$.



(a) (b)

Figura 4.29: Campo de vorticidade no plano $x - y$ em $z/L = 0,5$ sobreposto pelo refinamento adaptativo e pelas interfaces entre os processos em malha composta 32^3L3. (a) $Re = 1000$. (b) $Re = 3200$.

Uma ferramenta bastante utilizada na análise da topologia de escoamentos consiste no chamado critério Q. De acordo com Pinho (2006), o critério Q é definido por Jeong e Hussain (1995) como a norma euclidiana nos pontos em que a norma do tensor vorticidade (V) é maior que a norma do tensor deformação (S). Seguindo a definição, a equação que representa o critério Q é dada por

$$Q = \frac{1}{2}\left(|V|^2 - |S|^2\right). \tag{4.7}$$

sendo o tensor vorticidade e o tensor deformação respectivamente iguais a

$$V = \frac{1}{2}\left(\nabla\mathbf{u} - \nabla\mathbf{u}^T\right), \tag{4.8}$$

$$S = \frac{1}{2}\left(\nabla\mathbf{u} + \nabla\mathbf{u}^T\right). \tag{4.9}$$

Desenvolvendo a Eq. 4.7 chega-se a

$$Q = -2\left(\left(\frac{\partial u}{\partial y}\frac{\partial v}{\partial x}\right) - \left(\frac{\partial u}{\partial z}\frac{\partial w}{\partial x}\right) - \left(\frac{\partial v}{\partial z}\frac{\partial w}{\partial y}\right)\right) - \left(\frac{\partial u}{\partial x}\right)^2 - \left(\frac{\partial v}{\partial y}\right)^2 - \left(\frac{\partial w}{\partial z}\right)^2. \tag{4.10}$$

As figuras 4.30 e 4.31 mostram a iso-superfície do critério Q para $Re = 1000$ e $Re = 3200$ sobreposta respectivamente pelo particionamento do domínio e pelos *patches* que compõem a malha composta. Nota-se a presença de recirculação central, que ocupa toda a região central da cavidade, para ambos números de Reynolds. Para $Re = 1000$, observa-se a presença de pequenas estruturas nas regiões laterais. Tais estruturas podem ser notadas mais claramente no escoamento a $Re = 3200$.

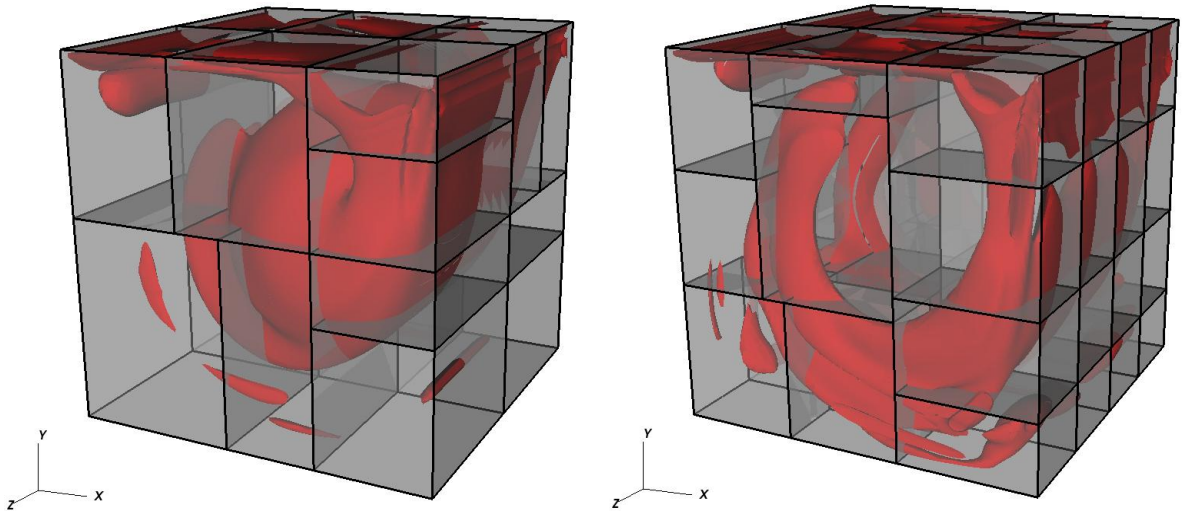


Figura 4.30: Iso-superfície do critério $Q = 0,25$ sobreposto pelo particionamento de domínio para uma malha composta $32L3$.(a) $Re = 1.000$; (b) $Re = 3.200$.

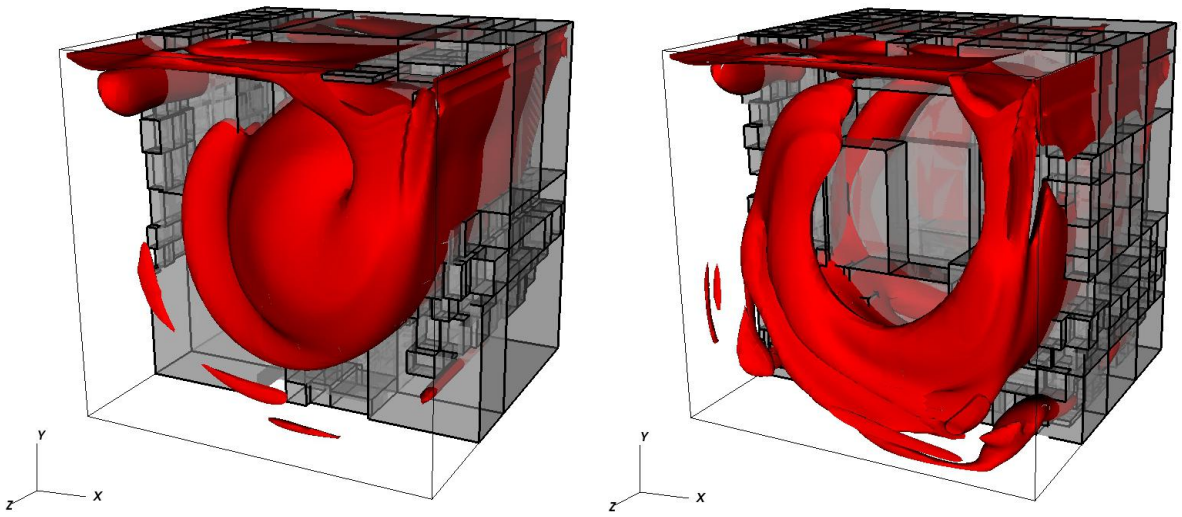


Figura 4.31: Iso-superfície do critério $Q = 0,25$ sobreposto pelo refinamento adaptativo para uma malha composta $32L3$.(a) $Re = 1000$; (b) $Re = 3200$.

4.3.2 Balanço de carga

O caso da cavidade com tampa deslizante também é utilizado para a validação da metodologia proposta para particionamento de domínio, no qual verifica-se distribuição da carga computacional nos processos ao longo da simulação. Esta carga é representada pelo número de células computacionais em todos os níveis de refinamento para cada processo. Na utilização do algoritmo RCB para o particionamento de domínio foi utilizada uma tolerância de 25% para desbalanceamento da carga, ou seja, a diferença entre o número de células de cada processo deve ser menor ou igual a 25%. Entretanto, esta tolerância pode ser extrapolada, principalmente em casos onde a região de refino é muito pequena, ou quando a partição de domínio possui razão de aspecto muito grande.

A Figura 4.32 mostra uma sequência temporal contendo o campo de vorticidade na direção do perfil de entrada sobreposto pelo particionamento de domínio e pela malha composta. Observa-se que o refinamento concentra-se na região superior da cavidade para os instantes iniciais e distribui-se para a região inferior para os instantes finais. Tal comportamento comprometeria de sobremaneira o desempenho em paralelo caso não fosse levado em conta o particionamento dinâmico de domínio com balanço de carga.

Nas figuras 4.33 a 4.35, são mostradas as evoluções temporais do balanço de carga para seis instantes de tempo. Observa-se um aumento do número de células em todas as figuras, pois nos instantes iniciais da simulação as regiões de alta vorticidade concentram-se na região superior da cavidade, forçando a malha adaptativa a se concentrar nesse local. De forma geral, observa-se uma distribuição razoável da carga computacional ao longo da simulação, porém, em alguns instantes de tempo há processos que possuem carga até 50% maior que os demais processos.

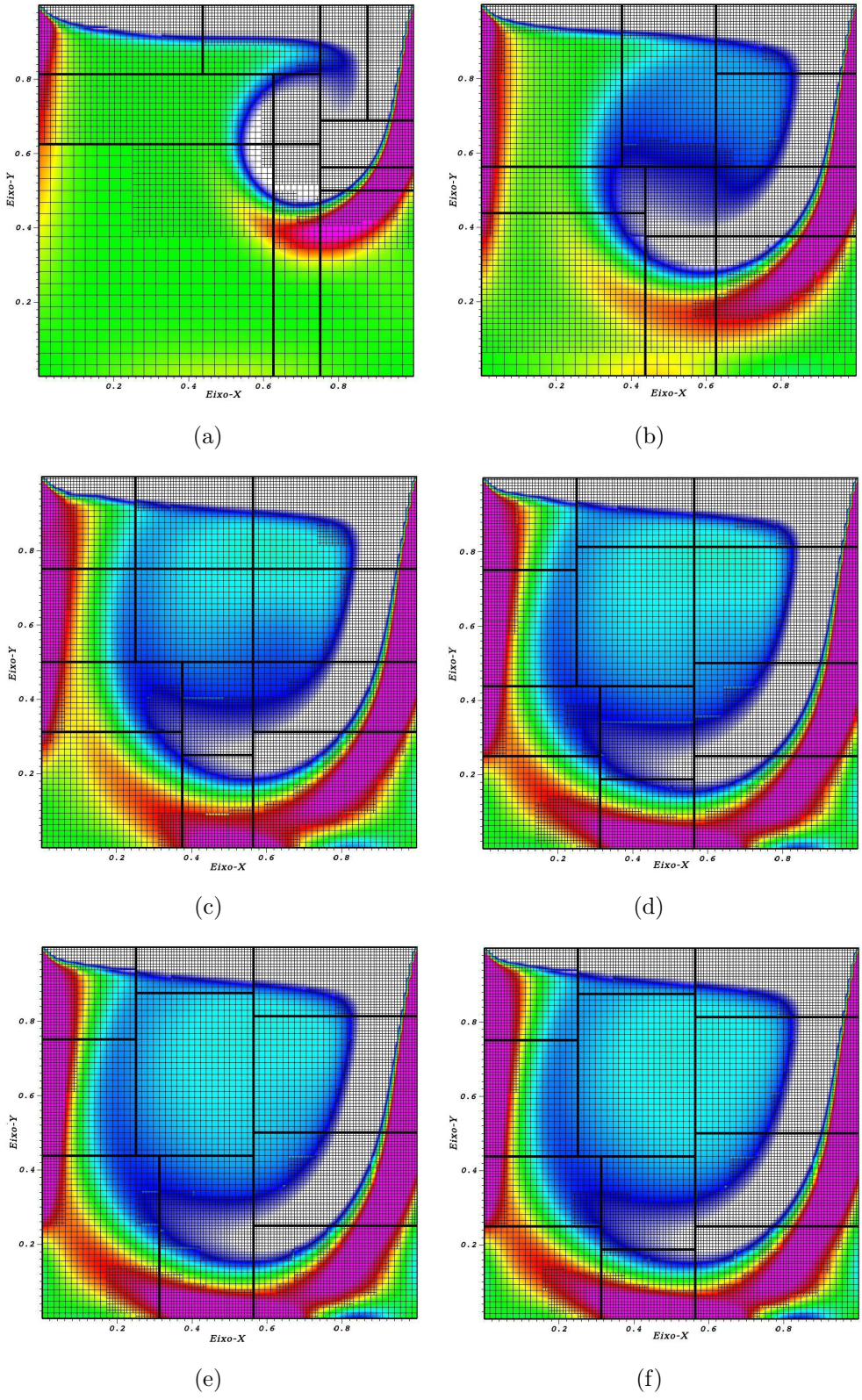


Figura 4.32: Evolução temporal da malha composta e do particionamento dinâmico em função da vorticidade no plano $x - y$ para $z/L = 0,5$ na malha composta $32L3$ para $Re = 1000$.(a)9s; (b)19s; (c) 28s; (d) 38s; (e) 44s; (f) 50s.

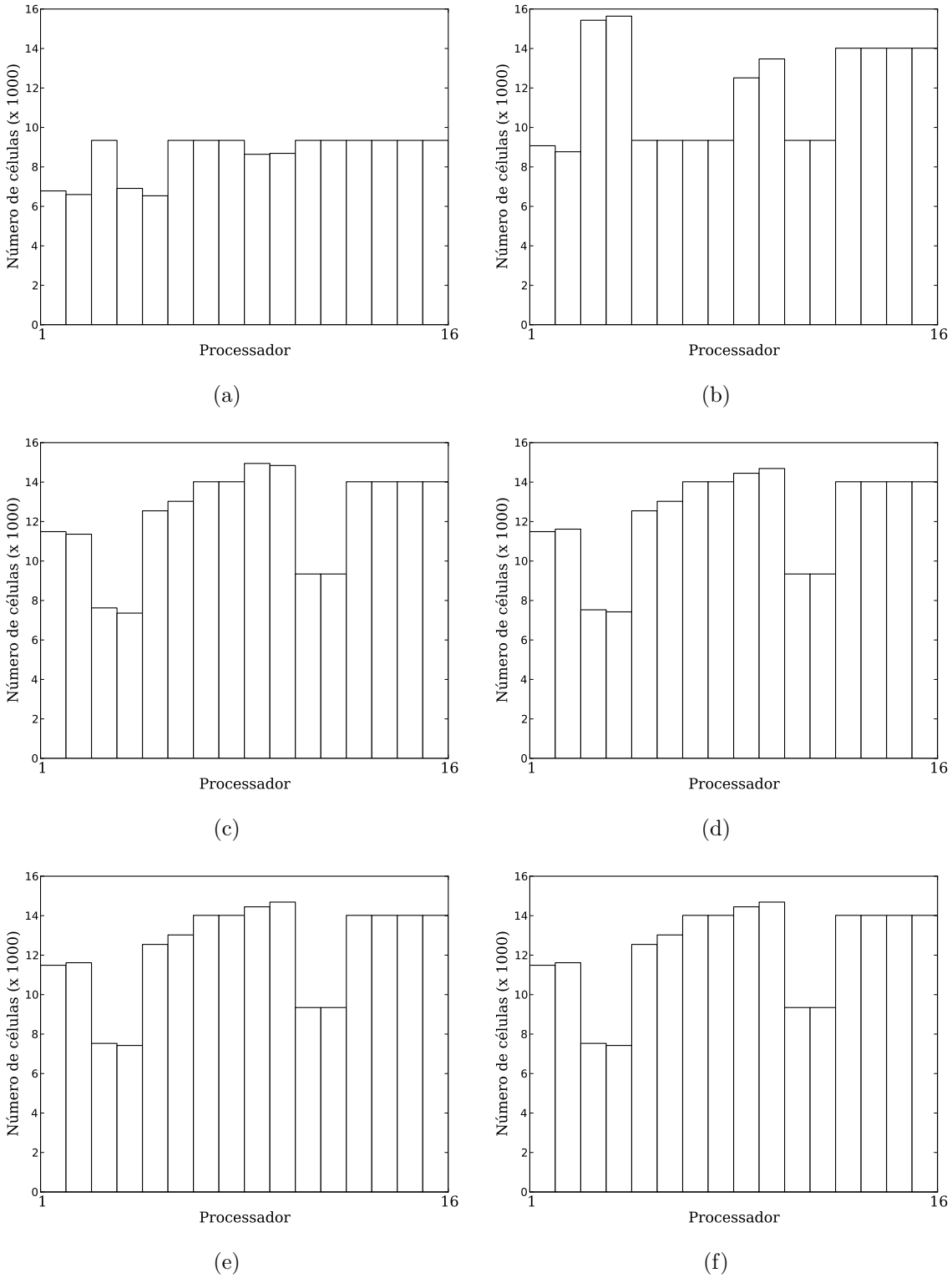


Figura 4.33: Evolução temporal da distribuição das células computacionais em 16 processos para $Re = 100$.(a)3s; (b)6s; (c) 9s; (d) 13s; (e) 16s; (f) 19s.

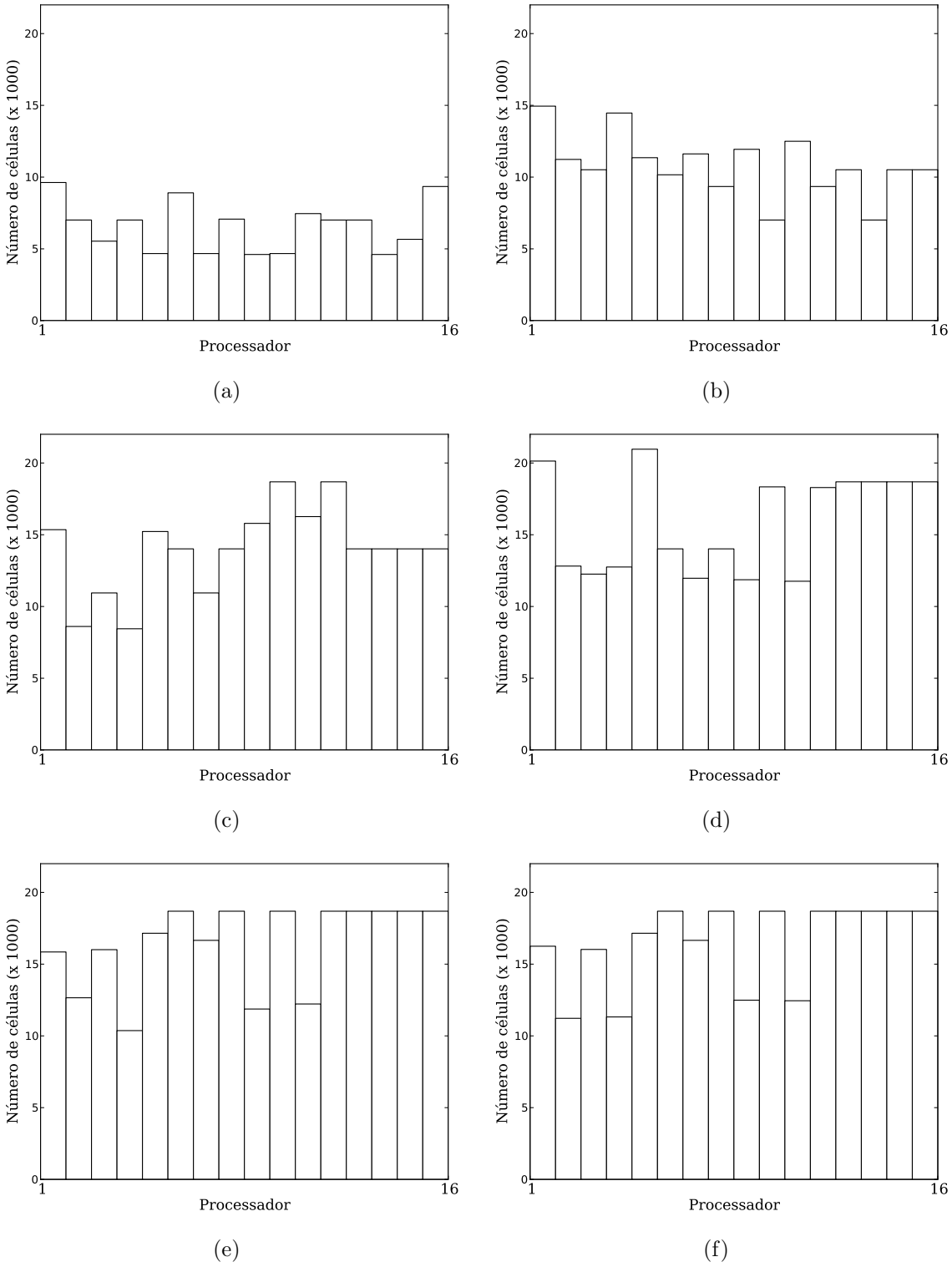


Figura 4.34: Evolução temporal da distribuição das células computacionais em 16 processos para $Re = 400$.(a)3s; (b)6s; (c) 9s; (d) 13s; (e) 16s; (f) 19s.

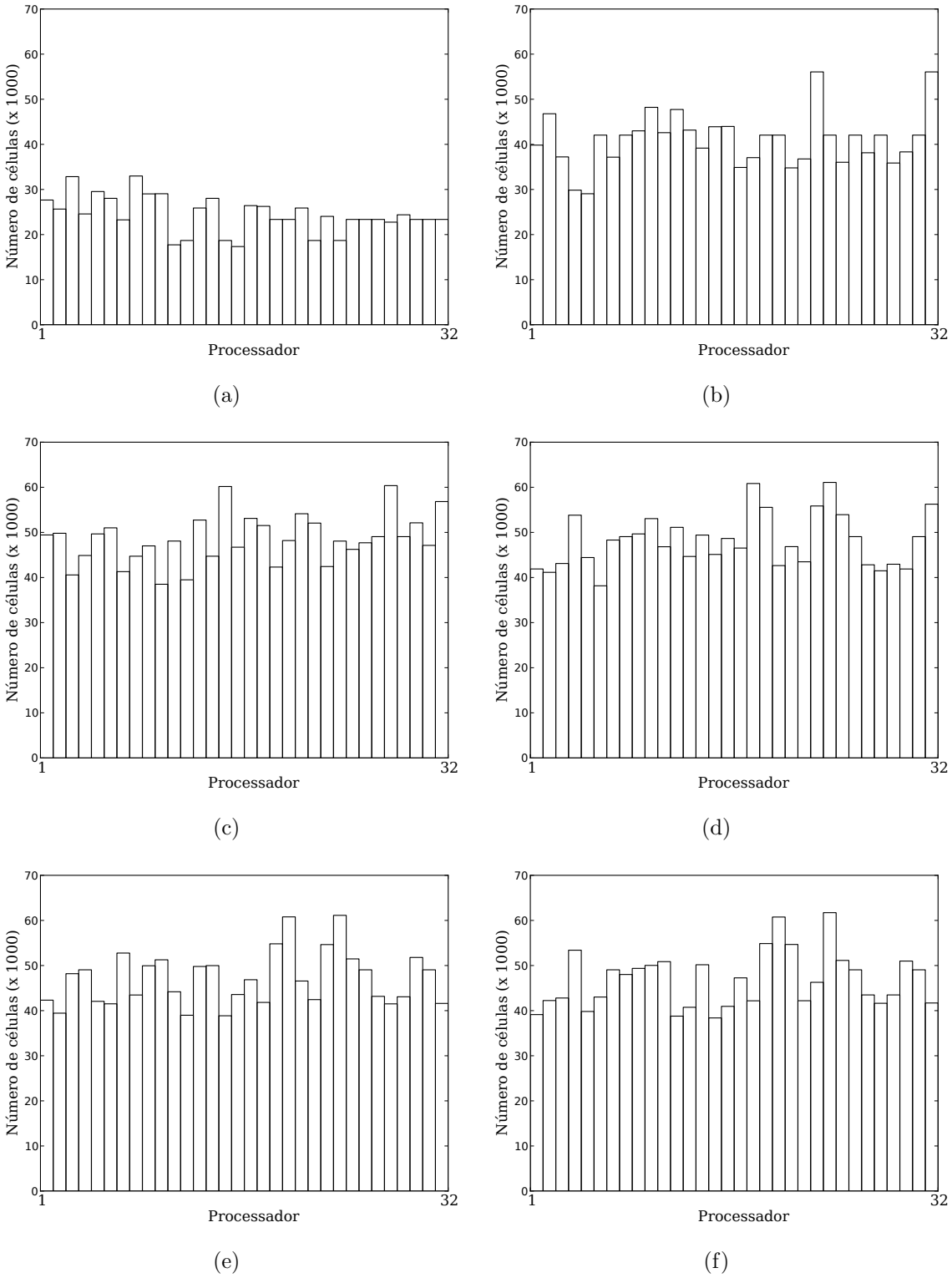


Figura 4.35: Evolução temporal da distribuição das células computacionais em 32 processos para $Re = 1000$.(a)9s; (b)19s; (c) 28s; (d) 38s; (e) 44s; (f) 50s.

4.4 Jato incompressível

A simulação de um jato incompressível, baseada na simulação encontrada no trabalho de Calegari (2012), foi utilizada na validação da metodologia proposta. A Fig. 4.36 mostra um esboço do bocal de entrada esférico de diâmetro d_0. Na região do bocal de entrada são usadas condições de contorno de Dirichlet para as componentes da velocidade e condição Neumman nula para a correção de pressão. Nas demais faces do domínio computacional, são usadas condições Neumann nulas para as componentes da velocidade e condição de contorno de Dirichlet para a correção de pressão. O mesmo caso foi executado em processamento paralelo e sequencial.

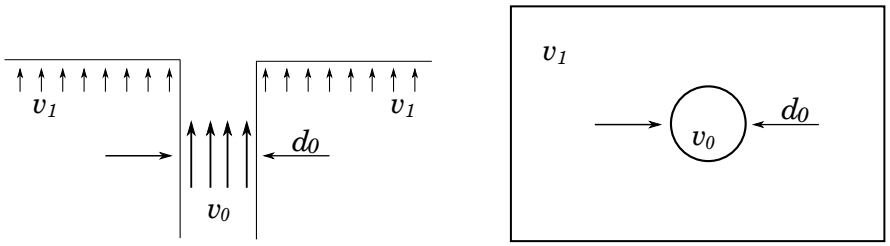


Figura 4.36: Esboço do bocal de entrada e da componente de velocidade na direção do escoamento.

Segundo os dados da simulação do jato laminar do trabalho de Calegari (2012), o diâmetro do bocal de entrada é $d_0 = 0,004m$ e as dimensões do domínio computacional são $0,16m \times 0,16m \times 0,16m$. A velocidade de da região de entrada do domínio $v_0 = 0,25m/s$ na saída do bocal e $v_1 = 0,002m/s$ no restante da região de entrada. Para a simulação deste escoamento a densidade e a viscosidade dinâmica são constantes, $\rho = 1,184kg/m^3$ e $\mu = 1,85 \times 10^-5kg/ms$. O número de Reynolds deste escoamento baseado no diâmetro do bocal é $Re \approx 65$.

Para a simulação computacional foi utilizada uma malha composta por 32^3 células no nível base mais três níveis de refinamento, divididos em 16 processadores. O nível de refinamento mais fino possui espaçamento $6 \times 10^{-4}m$, o que garante ao menos seis células na região do bocal de entrada. O critério utilizado para refinamento adaptativo foi baseado valor da componente da velocidade na direção do escoamento. O passo de tempo é da ordem do espaçamento da malha do nível mais fino, ou seja, $\Delta t = O(1/256)$. Foram realizadas 30000 integrações no tempo, o que resulta em $7,5s$ físicos. O critério de refinamento utilizado nesta simulação foi o mesmo de Calegari (2012), baseado na norma do tensor de tensões viscosas e no valor da componente de velocidade na direção do escoamento.

A Figura 4.37 mostra a comparação entre o decaimento da componente de velocidade

v obtida no presente trabalho com os resultados de Calegari (2012) e Boersma, Brethouwer e Nieuwstadt (1998). Segundo Calegari (2012) é importante ressaltar que, conforme Turns (1996), a comparação do decaimento é válida para $y/r_0 > 0,375Re_{r_0}$, que neste caso é igual a $y/d_0 \approx 7$. Segundo Boersma, Brethouwer e Nieuwstadt (1998), o decaimento da componente de velocidades v na linha de centro de um jato laminar é dada pela expressão

$$\frac{v^*}{v_0} = 0,375Re_{r_0}\frac{r_0}{y}, \tag{4.11}$$

na qual v^* representa o valor da velocidade v na linha de centro do jato em $x = 0,08m$ e $z = 0,08m$, v_{d_0} é a velocidade do escoamento na saída do bocal, r_0 é o raio da saída do bocal e Re_{r_0} é o número de Reynolds baseado no raio da saída do bocal.

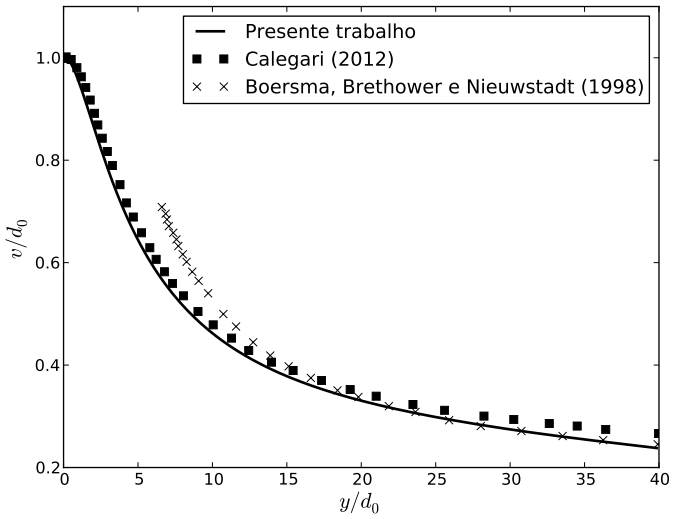
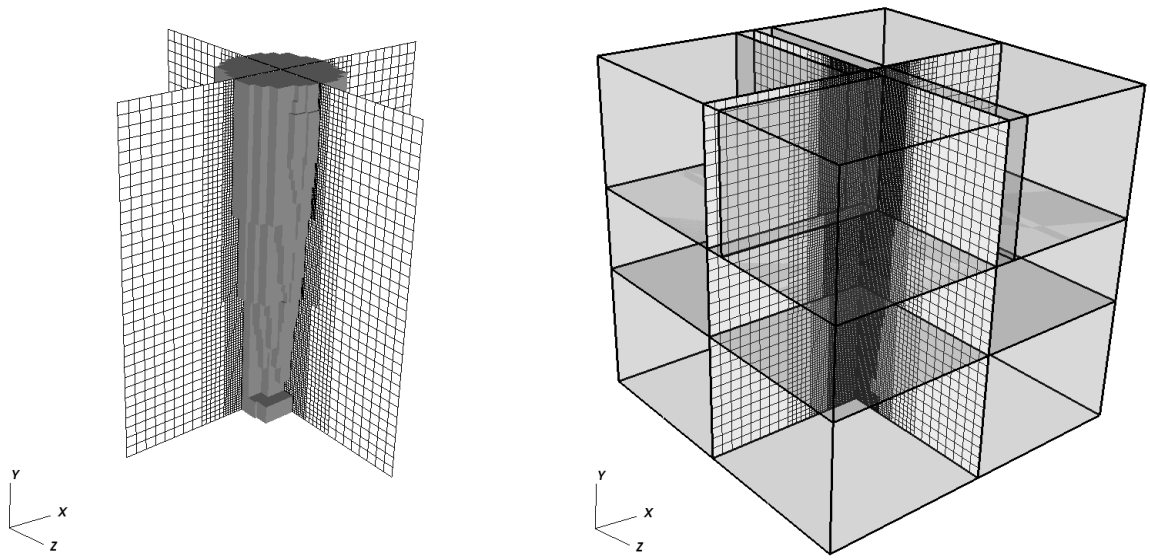


Figura 4.37: Decaimento da componente de velocidades v na linha de centro do domínio computacional.

Para comparar os ganhos em termos de tempo computacional obtidos com a utilização de malhas adaptativas em relação a malhas uniformes, Calegari (2012) simulou o mesmo problema utilizando malha uniforme com 256^3 células. Segundo seus resultados, a simulação com malha adaptativa chegou a ser 146 vezes mais rápida nos instantes iniciais e 16 vezes mais rápida nos instantes finais. No presente trabalho, o caso executado em processamento sequencial levou aproximadamente 150 horas para ser finalizado. O caso em processamento paralelo com 16 processadores foi executado em aproximadamente 17 horas, o que resulta em um ganho de aproximadamente nove vezes em termos de tempo computacional. Apesar da metodologia proposta não apresentar um *speedup* satisfatório, foram obtidos ganhos razoáveis em termos de tempo computacional. Para avaliar o ganho obtido com a utilização do particionamento dinâmico de domínio, simulou-se o caso do

jato incompressível para 10000 passos de tempo (≈ 3 s físicos) utilizando 16 processadores. O caso foi executado em 150 minutos sem balanço dinâmico de carga e em 60 minutos com balanço de carga.

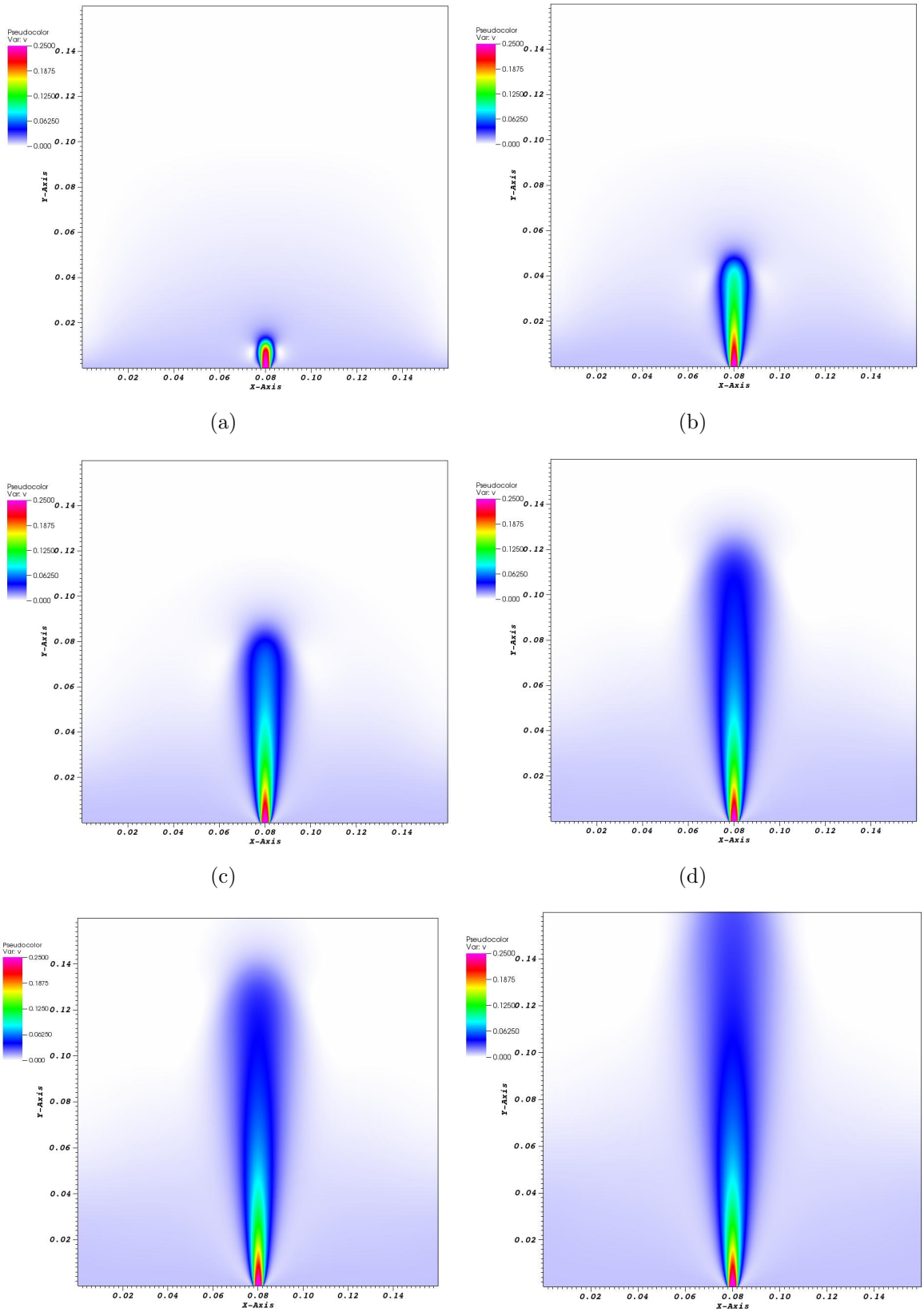
A Figura 4.38 mostra uma malha adaptativa tridimensional sobreposta pelos *patches* do nível mais fino e pelo particionamento de domínio. A evolução temporal do campo de velocidade v para um plano de corte em $z = 0,08m$ é mostrado na Fig. 4.39. Para o mesmo plano de corte, o campo de velocidades sobreposto pelos *patches* do nível mais fino l_{top} é mostrado na Fig. 4.40, onde observa-se que o refinamento distribui-se ao redor do núcleo potencial do jato, pois um dos critérios para refinamento foi baseado na intensidade da velocidade v. A Fig. 4.41 mostra uma sequência temporal contendo o campo de velocidades v sobreposto pelo particionamento de domínio e pela malha composta.



(a) (b)

Figura 4.38: Malha tridimensional com refinamento adaptativo dividida em 16 processos. (a) Sobreposta pelos *patches* do nível mais fino l_{top}, (b) sobreposta pelo particionamento de domínio.

Na Figura 4.42, mostra-se a evolução temporal do balanço de carga para vários instantes de tempo, na qual observa-se um número inferior de células nos instantes iniciais da simulação. Isto ocorre pois, nos instantes iniciais, o valor da componente de velocidade v é maior na região próxima à saída do bocal, forçando a malha adaptativa a se concentrar nesse local. De forma geral, observa-se uma distribuição razoável da carga computacional ao longo da simulação, porém, em alguns instantes de tempo há processos que possuem carga até 50% maior que os demais processos.



(a) (b)

(c) (d)

(e) (f)

Figura 4.39: Evolução temporal da componente de velocidade v para um plano de corte em $z = 0,08m$. (a) 0,1s; (b) 0,75s; (c) 2s; (d) 4s; (e) 5s; (e) 5s; (f) 7,5s.

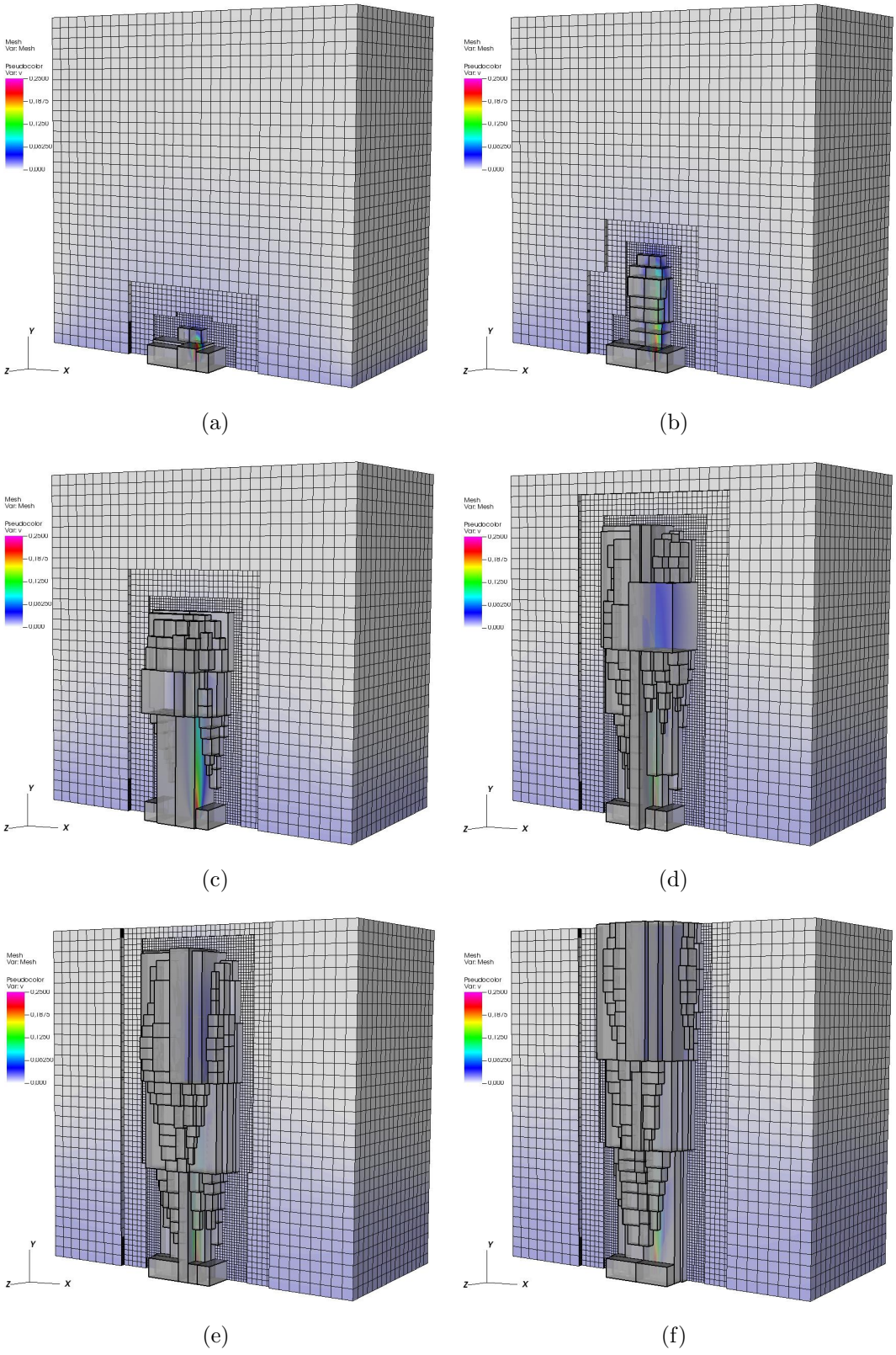


Figura 4.40: Evolução temporal da componente de velocidade v sobreposta pelos *patches* do nível mais fino para um plano de corte em $z = 0,08m$. (a) 0,1s; (b) 0,75s; (c) 2s; (d) 4s; (e) 5s; (f) 7,5s.

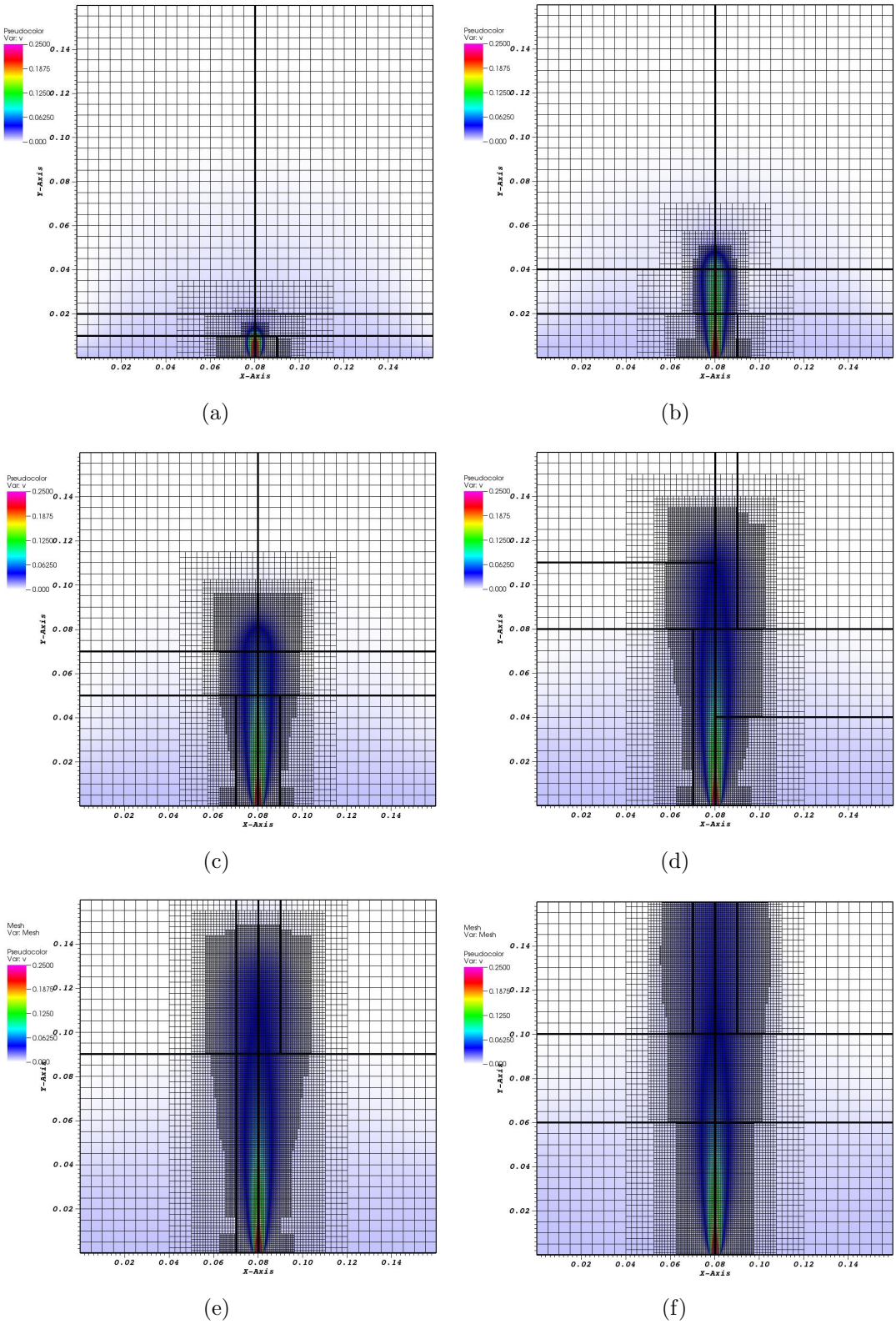


Figura 4.41: Evolução temporal da componente de velocidade v sobreposta pela malha composta $32L4$ e pelo particionamento de domínio para um plano de corte em $z = 0,08m$. (a) 0,1s; (b) 0,75s; (c) 2s; (d) 4s; (e) 5s; (f) 7,5s.

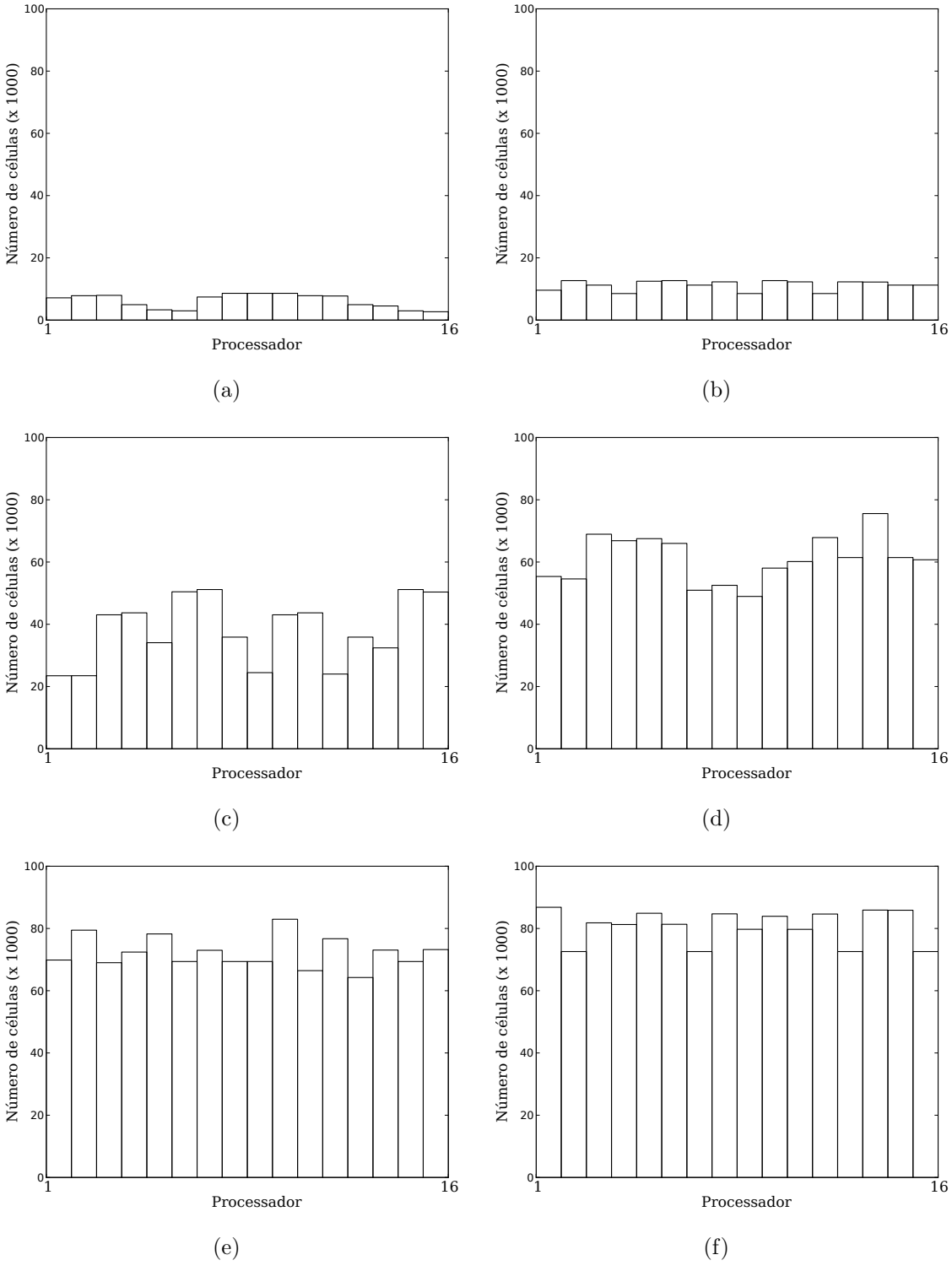


Figura 4.42: Evolução temporal da distribuição de células por processador. (a) 0,1s; (b) 0,75s; (c) 2s; (d) 4s; (e) 5s; (f) 7,5s.

Capítulo 5

Conclusões e Trabalhos Futuros

O presente trabalho teve como motivação principal a necessidade de se desenvolver uma ferramenta computacional capaz de simular fenômenos físicos com elevado grau de complexidade no menor tempo possível. Neste cenário o conceito de malhas adaptativas já é amplamente difundido, entretanto o custo computacional presente na simulação de alguns escoamentos, como ecoamentos envolvendo combustão ou multifásicos, é ainda proibitivo no contexto nacional. Na tentativa de superar tal dificuldade, foi proposta uma metodologia de paralelização para solução da equações de Navier-Stokes incompressível em malhas adaptativas envolvendo particionamento de domínio com balanceamento dinâmico da carga computacional.

A metodologia proposta baseia-se no particionamento e distribuição do domínio de cálculo em vários processadores, nos quais as EDPs são devidamente solucionadas separadamente. A comunicação de dados entre os processadores foi feita por meio da interface MPI (*Message Passing Interface*). A principal dificuldade apresentada no desenvolvimento da presente tese está no gerenciamento da estrutura de dados envolvida no processo de paralelização, a qual deve ser alterada frequentemente, devido ao comportamento dinâmico da malha composta.

O código computacional resultante, empregado nas simulações computacionais de escoamentos incompressíveis em paralelo, é novo no cenário nacional e representa uma das contribuições do presente trabalho. Tal código, denominado AMR3D, teve por base o trabalho de Nós (2007), o qual apresenta simulações tridimensionais utilizando malhas adaptativas e um modelo de campo de fase para escoamentos incompressíveis. Escrito em Fortran90, o AMR3D tem sido desenvolvido em um esforço conjunto e contínuo do IME-USP e do MFLab/FEMEC-UFU por vários anos.

Uma alteração muito importante no código AMR3D foi o armazenamento dos arquivos

de visualização utilizando a biblioteca HDF5 (*Hierarchical Data Format 5*). A biblioteca HDF5, muito utilizada para para armazenar e gerenciar dados, possibilita ao usuário a criação de arquivos em formato hierárquico. Dessa forma é possível organizar os dados de saída em um conjunto de níveis, sendo que cada nível possui um conjunto de *patches*. A estrutura HDF5 em paralelo desenvolvida no presente trabalho foi baseada na estrutura HDF5 utilizada pela biblioteca CHOMBO (`https://commons.lbl.gov/display/chombo/`). Esta medida reduziu o tamanho dos arquivos de saída em mais de três vezes e o tempo de escrita de cada arquivo em mais de 50 vezes.

Para avaliar a consistência do presente trabalho, a metodologia proposta foi devidamente verificada por meio de equações manufaturadas e verificada por meio de dois problemas clássico em mecânica dos Fluidos: cavidade com tampa deslizante e jato incompressível laminar. O balanço de carga também foi avaliado, apresentando uma boa uniformidade na distribuição da carga computacional. A variação da carga varia entre 10% e 40% conforme os casos apresentados. Seguramente a metodologia não apresenta uma distribuição de carga ótima, mas é muito superior ao que poderia ser sem balanceamento de carga. Para a fase atual do desenvolvimento do presente trabalho, os ganhos em tempo de processamento na paralelização do código computacional são muito satisfatórios. Para os casos da cavidade com tampa deslizante, obteve-se um ganho em torno de nove vezes no tempo computacional para 16 processadores.

5.1 Trabalhos futuros

1. Implementação de uma estrutura de dados mais eficiente para o cálculo das células fantasmas em processamento paralelo;
2. Implementação de uma estrutura de dados em processamento paralelo que permita a utilização de mais de uma célula fantasma;
3. Implementação de um módulo para reinicialização da simulação numérica no código AMR3D em processamento paralelo utilizando a biblioteca *HDF5*;
4. Aplicação da metodologia proposta junto com a metodologia de Fronteira Imersa para simular escoamentos em geometrias complexas;
5. Aplicação da metodologia proposta junto com a metodologia de *Front tracking* para simular escoamentos envolvendo transporte e deformação de geometrias complexas e escoamentos bifásicos;
6. Aplicação da metodologia proposta junto com a metodologia de Volume de Fluido para a simulação de escoamentos bifásicos.

Referências Bibliográficas

ALMGREN, A. S.; BELL, J. B.; COLELLA, P.; HOWELL, L. H. An adaptive projection method for the incompressible euler equations. In: *11th AIAA Computational Fluid Dynamics Conference*. FL: [s.n.], 1993.

ALMGREN, A. S.; BELL, J. B.; COLELLA, P.; HOWELL, L. H. An adaptive projection method for the incompressible navier-stokes equations. In: *World Congress on Computational and Applied Mathematics*. Atlanta, GA: [s.n.], 1994.

ALMGREN, A. S.; BELL, J. B.; COLELLA, P.; HOWELL, L. H.; L., W. M. A conservative adaptive projection method for the variable density incompressible navier-stokes equations. *Journal of Computational Physics*, v. 142, p. 1–46, May 1998.

BADALASSI, V. E.; CENICEROS, H. D.; BANERJEE, S. Computation of multiphase systems with phase field models. *J. Comput. Phys.*, Academic Press Professional, Inc., San Diego, CA, USA, v. 190, n. 2, p. 371–397, set. 2003. ISSN 0021-9991. Disponível em: `http://dx.doi.org/10.1016/S0021-9991(03)00280-8`.

BAKHVALOV, N. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Computational Mathematics and Mathematical Physics*, v. 6, n. 5, p. 101 – 135, 1966. ISSN 0041-5553. Disponível em: `http://www.sciencedirect.com/science/article/pii/0041555366901182`.

BALSARA, D.; NORTON, C. Highly parallel structured adaptive mesh refinement using parallel language-based approaches. *Journal of Parallel Computing*, v. 27, p. 37–70, 2001.

BERGER, M.; BOKHARI, S. A partitioning strategy for nonuniform problems on multiprocessors. *IEEE Transactions on Computers*, IEEE Computer Society, Los Alamitos, CA, USA, v. 36, p. 570–580, 1987. ISSN 0018-9340.

BERGER, M.; RIGOUTSOS, I. An algorithm for point clustering and grid generation. *Systems, Man and Cybernetics, IEEE Transactions on*, v. 21, n. 5, p. 1278 –1286, sep/oct 1991. ISSN 0018-9472.

BERGER, M. J.; COLELLA, P. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, v. 82, p. 64–84, 1989.

BERGER, M. J.; OLIGER, J. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, v. 53, n. 3, p. 484 – 512, 1984.

BHAVSAR, S.; SHEE, M.; PARASHAR, M. A characterization of distribution techniques for dynamic adaptive grid hierarchies. In: *5th U.S. National Congress on Computational Mechanics (USACM)*. Boulder, CO, USA: [s.n.], 1999. p. 198.

BOERSMA, B. J.; BRETHOUWER, G.; NIEUWSTADT, F. T. M. A numerical investigation on the effect of the inflow conditions on the self-similar region of a round jet. *Physics of Fluids*, v. 10, p. 11, 1998.

BRAMBLE, J. H.; PASCIAK, J. E.; XU, J. Parallel Multilevel Preconditioners. *Mathematics of Computation*, v. 55, n. 191, p. 1–22, 1990. Disponível em: `http://www.jstor.org/stable/2008789`.

BRANDT, A. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, v. 31, n. 138, p. 333 –390, april 1977. ISSN 1088-6842(e) 0025-5718(p).

BRIGGS, W. L.; EMDEN, H. V.; MCCORMICK, S. F. *A multigrid tutorial (2. ed.)*. [S.l.]: SIAM, 2000. I-XII, 1-193 p. ISBN 978-0-89871-462-3.

CALEGARI, P. C. *Simulação Computacional de Escoamentos Reativos com Baixo Número de Mach Aplicando Técnicas de Refinamento Adaptativo de Malhas.* Tese (Doutorado) — Universidade de São Paulo, 2012.

CENICEROS, H. D.; NóS, R. L.; ROMA, A. M. Three-dimensional, fully adaptive simulations of phase-field fluid models. *Journal of Computational Physics*, v. 229, n. 17, p. 6135 – 6155, 2010.

CENICEROS, H. D.; ROMA, A. M.; SILVEIRA-NETO, A.; VILLAR, M. M. A robust, fully adaptive hybrid level-set/front-tracking method for two-phase flows with an accurate surface tension computation. *Communications in Computational Physics*, v. 8, p. 51–94, 2010.

CHANDRA, S.; STEENSLAND, J.; PARASHAR, M.; CUMMINGS, J. An experimental study of adaptive application sensitive partitioning strategies for samr applications. In: *2nd Los Alamos Computer Science Institute Symposium*. [S.l.: s.n.], 2001.

CHORIN, A. J. Numerical solution of the navier-stokes equations. *Mathematics of Computation*, American Mathematical Society, v. 22, n. 104, p. 745–762, 1968.

CHORIN, A. J.; MARSDEN, J. E. *A Mathematical Introduction to Fluid Mechanics*. New York: Springer, 1993. 192 p.

CORRSIN, S.; UBEROI, M. S. Further experiments on the flow and heat transfer in a heated turbulent air jet. *National Advisory Committe for Aeronautics*, Report 988, 1950.

CORRSIN, S.; UBEROI, M. S. Spectrums and diffusion in a round turbulent jet. *National Advisory Committe for Aeronautics*, 1950.

DESHPANDE, M. D.; MILTON, S. G. Kolmogorov scales in a driven cavity flow. *Fluid Dynamics Research*, v. 22, n. 6, p. 359, 1998. Disponível em: `http://stacks.iop.org/1873-7005/22/i=6/a=A04`.

DEVINE, K.; BOMAN, E.; HEAPBY, R.; HENDRICKSON, B.; VAUGHAN, C. Zoltan data management service for parallel dynamic applications. *Computing in Science and Engineering*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 4, p. 90–97, March 2002.

FEDORENKO, R. P. The speed of convergence of one iterative process. *Ussr Computational Mathematics and Mathematical Physics*, v. 4, p. 227–235, 1964.

FERZIGER, J. H.; PERIC, M. *Computational Methods for Fluid Dynamics*. [S.l.]: Springer, 2001. 437 p.

FOERSTER, H.; STÜBEN, K.; TROTTENBERG, U. *Non-standard multigrid techniques using checkered relaxation and intermediate grids*. [s.n.], 1980. (Preprint. Sonderforschungsbereich 72. Approximation und Optimierung. Universität Bonn). Disponível em: `http://books.google.com.br/books?id=KS0iGwAACAAJ`.

GREENBAUM, A. A multigrid method for multiprocessors. *Applied Mathematics and Computation*, v. 19, n. 1–4, p. 75 – 88, 1986. ISSN 0096-3003. Disponível em: `http://www.sciencedirect.com/science/article/pii/0096300386900974`.

GRIFFITH, B. E.; HORNUNG, R. D.; MCQUEEN, D. M.; PESKIN, C. S. An adaptive, formally second order accurate version of the immersed boundary method. *Journal of Computational Physics*, v. 223, n. 1, p. 10 – 49, 2007. ISSN 0021-9991. Disponível em: `http://www.sciencedirect.com/science/article/pii/S0021999106004207`.

HACKBUSCH, W. *A multi-grid method applied to a boundary problem with variable coefficients in a rectangle*. [S.l.], 1977.

HIRSCH, C. *Numerical Computation of Internal and External Flows: Fundamentals of numerical discretization*. [S.l.]: Wiley, 1988. 515 p.

HORTMANN, M.; PERIć, M.; SCHEUERER, G. Finite volume multigrid prediction of laminar natural convection: Bench-mark solutions. *International Journal for Numerical Methods in Fluids*, John Wiley and Sons, Ltd, v. 11, n. 2, p. 189–207, 1990. ISSN 1097-0363. Disponível em: http://dx.doi.org/10.1002/fld.1650110206.

HUANG, W. C.; TAFTI, D. K. A parallel adaptive mesh refinement algorithm for solving nonlinear dynamical systems. *The International Journal of High Performance Computing Applications*, v. 18, p. 171–181, 2004.

HÜLSEMANN, F.; KOWARSCHIK, M.; MOHR, M.; RÜDE, U. Parallel geometric multigrid. Book Chapter. Orifinal file 10.1.1.83.4637.pdf.

JEONG, J.; HUSSAIN, F. On the identification of a vortex. *Journal of Fluid Mechanics*, v. 285, p. 69–94, 1995. Disponível em: http://dx.doi.org/10.1017/S0022112095000462.

JOHANSSON, H. A characterization of a hybrid and dynamic partitioner for samr applications. In: *In Proceedings of The 16th IASTED International Conference on Parallel and Distributed Computing and Systems*. [S.l.: s.n.], 2004.

JOHANSSON, H.; VAKILI, A. A patch-based partitioner for parallel samr applications. In: *Proceedings of Parallel and Distributed Computing and Systems*. Orlando, Florida, USA: [s.n.], 2008.

KU, H. C.; HIRSH, R. S.; TAYLOR, T. D. A pseudospectral method for solution of the three-dimensional incompressible navier-stokes equations. *Journal of Computational Physics*, v. 70, n. 2, p. 439 – 462, 1987. ISSN 0021-9991. Disponível em: http://www.sciencedirect.com/science/article/pii/0021999187901902.

LAN, Z.; TAYLOR, V. E.; BRYAN, G. A novel dynamic load balancing scheme for parallel systems. *Journal of Parallel and Distributed Computing*, Academic Press, Inc., Orlando, FL, USA, v. 62, p. 1763–1781, December 2002. ISSN 0743-7315.

LIMA-E-SILVA, A.; SILVEIRA-NETO, A.; DAMASCENO, J. Numerical simulation of two dimensional flows over a circular cylinder using the immersed boundary method. *Journal of Computational Physics*, v. 189, p. 351–370, 2003.

MALISKA, C. R. *Transferência de Calor e Mecânica dos Fluidos Computacional*. Rio de Janeiro: LTC-Livros Técnicos e Científicos Editora S.A., 2004.

MARIANO, F. P. *Solução Numérica das Equações de Navier-Stokes Usando uma Hibridação das Metodologias Fronteira Imersa e Pseudo-Espectral de Fourier*. 151 p. Tese (Doutorado) — Universidade Federal de Uberlândia, 2011.

MCCORMICK, S. *Multigrid Methods*. Society for Industrial and Applied Mathematics, 1987. (Frontiers in Applied Mathematics). ISBN 9780898712148. Disponível em: http://books.google.com.br/books?id=594AN97YHtwC.

MOREIRA, L. Q. *Modelagem Matematica de Jatos em Jatos em Desenvolvimento Espacial Usando a Metodologia Pseudo Espectral de Fourier*. 177 p. Tese (Doutorado) — Universidade Federal de Uberlândia, 2011.

MORRIS, P. The spatial viscous instability of axisymmetric jets. *Journal of Fluid Mechanics*, v. 77, n. 3, p. 511–529, 1976.

NETO, A. S.; ROMA, A. M.; VILLAR, M. M.; PIVELLO, M. R.; LIMA, R. S. *Desenvolvimento de Modelagem Matemática e Simulação Numérica para Análise de Escoamentos Bifásicos Anulares*. Relatório N. 3, Uberlândia: Universidade Federal de Uberlândia; Petrobras. Março, 2010.

NóS, R. L. *Simulações de Escoamentos tridimensionais Bifásicos Empregando Métodos Adaptativos e Modelos de Campo de Fase*. Tese (Doutorado) — Universidade de São Paulo, 2007.

PARASHAR, M.; BROWNE, J. C. On partitioning dynamic adaptive grid hierarchies. In: *Proceedings of the 29th Annual Hawaii International Conference on System Sciences*. [S.l.: s.n.], 1996. p. 604–613.

PARASHAR, M.; BROWNE, J. C.; EDWARDS, C.; KLIMKOWSKI, K. A common data management infrastructure for adaptive algorithms for pde solutions. In: *Proceedings of the 1997 ACM/IEEE conference on Supercomputing (CDROM)*. New York, NY, USA: ACM, 1997. (Supercomputing '97), p. 1–22.

PINHO, F. A. A. *Simulação Numérica de Grandes Escalas em Cavidades Tridimensionais com Tampa Deslizante Utilizando Modelagem Dinâmica.* Tese (Doutorado) — Universidade Federal de Uberlândia, 2006.

POPINET, S. Gerris: A tree-based adaptive solver for the incompressible euler equations in complex geometries. *Journal of Computational Physics*, v. 190, p. 572–600, 2003.

Prasad, A. K.; Koseff, J. R. Reynolds number and end-wall effects on a lid-driven cavity flow. *Physics of Fluids*, v. 1, p. 208–218, fev. 1989.

R., C. Investigation of flow in an axially symmetric heated jet of air. *National Advisory Committe for Aeronautics*, 1943.

RABI, J. A. *Aplicação do Método Multigrid na Solução Numérica de Problemas 2-D Simples de Mecânica dos Fluidos e Transferência de Calor.* Dissertação (Mestrado) — Institudo Tecnológico de Aeronáutica., 1998.

RANTAKOKKO, J. A framework for partitioning structured grids with inhomogeneous workload. *Journal of Parallel Algorithms and Applications*, v. 13, p. 135 – 151, 1998.

RANTAKOKKO, J. Partitioning strategies for structured multiblock grids. *Parallel Computing*, v. 26, n. 12, p. 1661–1680, 2000. Graph Partitioning and Parallel Computing.

ROMA, A. M. *Multilevel Self Adaptive Version of the Immersed Boundary Method.* Tese (Doutorado) — New York University, 1996.

ROMA, A. M.; PESKIN, C. S.; BERGER, M. J. An adaptive version of the immersed boundary method. *Journal of Computational Physics*, v. 153, p. 509–534, 1999.

SAGAN, H. *Space-Filling Curves.* New York: Springer-Verlag, 1994.

SILVEIRA-NETO, A. A turbulência nos fluidos aplicada. *Apostila da Disciplina Mecânica dos Fluidos do Programa de Pós-Graduação da Universidade Federal de Uberlândia*, v. 1, 2002.

STANLEY S. A.; SARKAR, S. Influence of nozzle conditions and discrete forcing on the developing region of turbulent planar jets. *AIAA Journal*, v. 38, n. LBNL-43150, p. 1–40, 1999.

STEENSLAND, J. *Efficient Partitioning of Dynamic Structured Grid Hierarchies.* 139 p. Tese (Doutorado) — Uppsala UniversityUppsala University, Division of Scientific Computing, Numerical Analysis, 2002.

STEENSLAND, J.; RAY, J. A partitioner-centric model for structured adaptive mesh refinement partitioning trade-off optimization: Part i. *International Journal of High Performance Computing Applications*, Sage Publications, v. 19, p. 409–422, 2005.

SUSSMAN, M.; ALMGREN, A. S.; BELL, J. B.; COLELLA, P.; HOWELL, L. H.; WELCOME, M. L. An adaptive level set approach for incompressible two-phase flows. *Journal of Computational Physics*, v. 148, n. 1, p. 81 – 124, 1999.

TANNEHILL, J.; ANDERSON, D.; PLETCHER, R. *Computational Fluid Mechanics and Heat Transfer.* [S.l.]: Taylor and Francis, London, 1997. 816 p.

TEMAM, R. Une méthode d'approximation de la solution des équations de Navier-Stokes. *Bull. Soc. Math. France*, v. 96, p. 115–152, 1968. ISSN 0037-9484.

THUNé, M. Partitioning strategies for composite grids. *Parallel Algorithms and Applications*, Taylor and Francis, Uppsala, Sweden, v. 11, p. 325 – 348, 1997.

TROTTENBERG, C. W. O. U.; SCHULLER, A. *Multigrid*. San Diego, CA, USA: Academic Press, 2001.

TURNS, S. R. *An introduction to combustion*. [S.l.]: McGraw-Hill, 1996. 565 p.

VEDOVOTO, J. M. *Mathematical and numerical modeling of turbulent reactive flows using a hybrid LES/PDF methodology*. 194 p. Tese (Doutorado) — Universidade Federal de Uberlândia, 2011.

VILLAR, M. M. *Análise Numérica Detalhada de Escoamentos Multifsicos Bidimensionais*. Tese (Doutorado) — Universidade Federal de Uberlândia, 2007.

WALSH, P. C.; ZINGG, D. W. Solution adaptation of unstructured grids for two-dimensional aerodynamic computations. *AIAA Journal*, v. 39, p. 831–837, 2001.

WISSINK, A.; HORNUNG, R.; KOHN, S.; SMITH, S.; ELLIOTT, N. Large scale parallel structured amr calculations using the samrai framework. In: *Supercomputing, ACM/IEEE 2001 Conference*. [S.l.: s.n.], 2001. p. 22 – 22.

WISSINK, A. M.; HYSOM, D.; HORNUNG, R. D. Enhancing scalability of parallel structured amr calculations. In: *Proceedings of the 17th annual international conference on Supercomputing*. New York, NY, USA: ACM, 2003. (ICS '03), p. 336–347. ISBN 1-58113-733-8. Disponível em: `http://doi.acm.org/10.1145/782814.782861`.

WYGNANSKI, I.; FIEDLER, H. Some measurements in the self-preserving jet. *Journal of Fluid Mechanics*, v. 38, n. 03, p. 577–612, 1969.

XU, C.; TANG, T. Stability analysis of large time - stepping methods for epitaxial growth models. *SIAM Journal on Numerical Analysis*, v. 44, n. 4, p. 1759–1779, 2006. Disponível em: `http://epubs.siam.org/doi/abs/10.1137/050628143`.

Apêndice A

Configuração do *cluster* utilizado

O *cluster* utilizado nas simulações em processamento paralelo foi um SGI/Altix XE 1300 composto por:

- 344 núcleos computacionais;
- 1,66TB de memória RAM;
- 19TB de espaço em disco (*storage raid* 5);
- *Headnode* SGI/Altix XE 270:
  - 2 Intel Xeon Processor E5520, 2.27GHz, 8MB cache, 24GB DDR3 1333 MHz, 16 cores;
- 4 nós SGI/Altix XE 340:
  - 2 quad core Intel Xeon Processor E5540 2.53GHz 8MB cache, 24GB DDR3 1333 MHz, 16 cores;
- 20 nós SGI/Altix XE 340:
  - 2 six core Intel Xeon Processor E5650 2.67GHz 12MB cache, 48GB DDR3 1333 MHz, 24 cores.

Total de 1 *rack* com 30 nós computacionais, 1 nó servidor, 1 *storage* (IS220 Storage, capacidade máxima de 24TB), 2 *switches* gigabit SMC8824M TigerStack II e 1 *switch* infiniband Voltarie 4063.