UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GUILHERME GALANTE

Métodos Multigrid Paralelos em Malhas Não Estruturadas Aplicados à Simulação de Problemas de Dinâmica de Fluidos Computacional e Transferência de Calor

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Tiarajú Asmuz Diverio
Orientador

Prof. Dr. Rogério Luis Rizzi
Co-orientador

Porto Alegre, março de 2006

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Galante, Guilherme

Métodos Multigrid Paralelos em Malhas Não Estruturadas Aplicados à Simulação de Problemas de Dinâmica de Fluidos Computacional e Transferência de Calor / Guilherme Galante. – Porto Alegre: PPGC da UFRGS, 2006.

102 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2006. Orientador: Tiarajú Asmuz Diverio; Co-orientador: Rogério Luis Rizzi.

1. Malhas Não Estruturadas. 2. Decomposição de Domínios. 3. Multigrid. 4. Solução Paralela de Sistemas de Equações. 5. Volumes Finitos. I. Diverio, Tiarajú Asmuz. II. Rizzi, Rogério Luis. III. Título.

"Though I cannot change the world we're living in
I can always change myself"

AGRADECIMENTOS

Gostaria de agradecer a todos que me ajudaram de alguma forma a vencer mais esta etapa da minha vida. O meu **muito obrigado** vai para:

Minha família, em especial meus pais, que sempre me incentivaram em tudo. Mais uma missão cumprida!

Juliana, minha namorada, pelos bons momentos que passamos juntos, pelo carinho e compreensão. Foi difícil passar esses dois anos longe, mas conseguimos!

Meu orientador, o Prof. Tiarajú Diverio, pela confiança depositada em mim ao ter me aceito no mestrado, e pelo companherismo demonstrado durante esses dois anos de trabalho.

Meu co-orientador, o Prof. Rogério Rizzi, pelas muitas horas dedicadas a mim. Sem sua ajuda o trabalho teria sido muito mais difícil.

Todos colegas que compartilharam suas horas de trabalho, de risadas e de preocupações, em especial o pessoal do GMCPAD (UFRGS) e do LCAD (UNIOESTE).

Meus amigos da República Paraná que sempre estiveram comigo tanto nas difíceis e nas horas de lazer. Agora que cada um toma seu rumo, talvez nunca mais nos encontremos, mas esses dois anos não serão nunca apagados da memória.

Os meus amigos de Cascavel, pelos bons (embora poucos) momentos que passei aí na terrinha nesses dois anos.

Ao Programa de Pós Graduação em Computação do Instituto de Informática da UFRGS, por todos os recursos disponibilizados e à CAPES pelo auxílio financeiro.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

CSR	*Compressed Sparse Row*
DECK	*Distributed Execution and Communication Kernel*
EDP	Equação Diferencial Parcial
FMV	*Full Multigrid V*
FMW	*Full Multigrid W*
GB	Gigabyte
GC	Gradiente Conjugado
GHz	Gigahertz
GMCPAD	Grupo da Matemática da Computação e Processamento de Alto Desempenho
GMRES	*Generalized Minimum Residual*
LabTeC	Laboratório de Tecnologia em Clusters
MAS	Método Aditivo de Schwarz
MCS	Método do Complemento de Schur
MDD	Método de Decomposição de Domínio
MG	Multigrid
MG+MDD	Combinação de Multigrid com Métodos de
MPI	*Message Passing Interface*
NP	*Non-deterministic Polynomial-time*
ORB	*Ortogonal Recursive Bisection*
PC	*Personal Computer*
PCAM	*Partitioning, Communication, Agglomeration and Mapping*
PSLG	*Planar Straight Line Graph*
RCB	*Recursive Coordinate Bisection*
RGB	*Recursive Graph Bissection*
RSB	*Recursive Spectral Bisection*

SCSI	*Small Computer System Interface*
SEL	Sistema de Equações Lineares
SMP	*Symmetric Multiprocessing*
STRIP	*Stripwise Partitioning*
SPMD	*Single Program Multiple Data*
UCS	Universidade de Caxias do Sul
UFRGS	Universidade Federal do Rio Grande do Sul
UNIOESTE	Universidade Estadual do Oeste do Paraná
UnHIDRA	*Unstructured HIDRA*

LISTA DE SÍMBOLOS

x_i^j	Vetor x no índice i na iteração j
x'	Solução aproximada para um dado vetor x
M_i	i-ésimo nível de malha empregado nos métodos multigrid
$I_{M_n}^{M_{n-1}}$	Operador de interpolação
$I_{M_n-1}^{M_n}$	Operador de restrição
A_{M_i}	Representa a matriz relacionada à malha M_i;
x_{M_i}	Representa o vetor das incógnitas relacionada à malha M_i;
b_{M_i}	Representa o vetor dos termos independentes relacionada à malha M_i;
$S(A, x, b)$	Operador que descreve um método de solução iterativa
A^-1	Inversa da matriz A
$\|x\|_2$	Norma Euclidiana

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Fenômenos naturais, tecnológicos e industriais podem, em geral, ser modelados de modo acurado através de equações diferenciais parciais, definidas sobre domínios contínuos que necessitam ser discretizados para serem resolvidos. Dependendo do esquema de discretização utilizado, pode-se gerar sistemas de equações lineares. Esses sistemas são, de modo geral, esparsos e de grande porte, onde as incógnitas podem ser da ordem de milhares, ou até mesmo de milhões. Levando em consideração essas características, o emprego de métodos iterativos é o mais apropriado para a resolução dos sistemas gerados, devido principalmente a sua potencialidade quanto à otimização de armazenamento e eficiência computacional.

Uma forma de incrementar o desempenho dos métodos iterativos é empregar uma técnica multigrid. Multigrid são uma classe de métodos que resolvem eficientemente um grande conjunto de equações algébricas através da aceleração da convergência de métodos iterativos.

Considerando que a resolução de sistemas de equações de problemas realísticos pode requerer grande capacidade de processamento e de armazenamento, torna-se imprescindível o uso de ambientes computacionais de alto desempenho.

Uma das abordagens encontradas na literatura técnica para a resolução de sistemas de equações em paralelo é aquela que emprega métodos de decomposição de domínio (MDDs). Os MDDs são baseados 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.

Assim, neste trabalho são disponibilizados diferentes métodos de resolução paralela baseado em decomposição de domínio, utilizando técnicas multigrid para a aceleração da solução de sistemas de equações lineares. Para cada método, são apresentados dois estudos de caso visando a validação das implementações. Os estudos de caso abordados são o problema da difusão de calor e o modelo de hidrodinâmica do modelo UnHIDRA.

Os métodos implementados mostraram-se altamente paralelizáveis, apresentando bons ganhos de desempenho. Os métodos multigrid mostraram-se eficiente na aceleração dos métodos iterativos, já que métodos que utilizaram esta técnica apresentaram desempenho superior aos métodos que não utilizaram nenhum método de aceleração.

Palavras-chave: Malhas Não Estruturadas, Decomposição de Domínios, Multigrid, Solução Paralela de Sistemas de Equações, Volumes Finitos.

Parallel Multigrid Methods in Unstructured Meshes Applied to Computational Fluid Dynamics and Heat Transfer

ABSTRACT

Natural, technological and industrial phenomena, in general, can be modeled by partial differential equations, that are defined on a continuous domain and must be discretized to be solved.

The discretization process results in linear systems that must be solved at each simulation time step. Generally, these systems are sparses and have a large number of unknowns. Taking in consideration these characteristics, the use of iterative methods is more appropriate for the resolution of these systems, due to the storage optimization potential and computational efficiency.

A form to increase the performance of iterative methods is to use a multigrid method. Multigrid is a class of methods that efficiently solves a great set of algebraic equations through the acceleration of the convergence of iterative methods. Basically, the methods multigrid consider a sequence of meshes for the solution of the system of equations.

Considering that the resolution of systems of equations with numerical high-quality requires a great amount of processing and storage, becoming the use of high performance computing adequate to obtain their solution.

An approach in the literature for the resolution of linear systems in parallel is the domain decomposition methods. These methods are based on the partitioning of the computational domain in subdomains, such that the global solution for the problem is obtained by the appropriate combination of the solutions of all the subdomains.

Thus, in this work are considered parallel solvers based in domain decomposition and multigrid for the solution of linear equation systems. Two study cases are presented for the validation of the implementations. The first study case is the problem of heat diffusion. In the second, the objective is to solve the linear systems originated from UnHIDRA hydrodynamics model.

The implemented methods show good performance and scalability. The use of multigrid was very efficient in the acceleration of the iterative methods, since methods that had used this technique presented superior performance when comparated to methods that had not used any method of acceleration.

Keywords: Unstructured Meshes, Multigrid, Domain Decomposition Methods, Equations Systems Parallel solution, Finite Volume.

1 INTRODUÇÃO

Através de modelagem numérica associada à computação de alto desempenho, é possível realizar a simulação ou predição de fenômenos ou processos científicos, tecnológicos e industriais, que seriam irrealizáveis ou antieconômicos se efetivados por métodos experimentais.

Pesquisas empíricas com modelos realísticos ou com semelhança dinâmica são fundamentais, pois validam e delineiam os limites de várias aproximações para os modelos matemáticos, mas, muitas vezes, têm custo tão elevado que se tornam economicamente inviáveis (MODI, 1997). Esse fato abriu espaço para a modelagem computacional.

Muitos dos fenômenos podem ser modelados matematicamente através de equações ou sistemas de equações diferenciais parciais (EDPs). Porém, essas equações, em geral, não possuem solução analítica ou essa solução é muito custosa, sendo necessário obter uma solução aproximada através de métodos numéricos. Essa solução através de métodos numéricos necessita que o domínio seja discretizado, de modo a produzir um conjunto de pontos nos quais os algoritmos se baseiam. Esse conjunto de pontos conectados, denomina-se malha (MAVRIPLIS, 1996).

Nesses pontos, os termos das EDPs são aproximados, resultando em um sistema de equações lineares ou não lineares, que devem ser resolvidos a cada passo de tempo. Esses sistemas podem ser resolvidos por métodos diretos ou iterativos.

Os métodos diretos apresentam solução exata, exceto por erros de arredondamento devido às operações de ponto flutuante, em um número finito de passos. No entanto, o uso de métodos diretos é inadequado para a resolução de sistemas esparsos, uma vez que não aproveitam a esparsidade da matriz de coeficientes, tornando essa abordagem difícil, por problemas de armazenamento e pela dependência de operações que dificulta a sua paralelização (CISMASIU, 2002).

Os algoritmos iterativos, por sua vez, utilizam a matriz apenas como um operador para construir iterativamente uma seqüência de aproximações para a solução. E, ao contrário dos métodos diretos, são muito utilizados na resolução de sistemas de equações esparsos e de grande porte, devido a sua potencialidade quanto à otimização de armazenamento e eficiência computacional.

Existem várias formas para se aumentar o desempenho dos métodos iterativos. Em particular, emprega-se neste trabalho técnicas multigrid. Basicamente, os métodos multigrid constroem a solução utilizando uma seqüência de malhas, onde se resolve o problema na malha fina empregando as demais malhas como esquemas de correção.

Experimentos numéricos mostram que estes métodos são muito eficientes e podem ser aplicados com sucesso a uma ampla classe de problemas de computação

científica (BRIGGS, 1987). A bibliografia sobre o assunto mostra que o método é bastante geral e sua eficiência não é restrita ao tipo da malha utilizado (estruturada, não estruturada), da discretização utilizada (elementos finitos, diferenças finitas, volumes finitos) ou do tipo do sistema de equações obtido da discretização (simétrico, não simétrico) (TROTTENBERG; OOSTERLEE; SCHÜLLER, 2001), (HORNUNG; TRANGENSTEIN, 1997).

Em geral, os sistemas de equações oriundos de simulações são esparsos e de grande porte, onde as incógnitas podem ser da ordem de milhares, ou até mesmo de milhões (CANAL, 2000). Considerando tais características, uma solução com alta qualidade numérica pode requerer grande capacidade de processamento e de armazenamento, o que torna imprescindível o uso de ambientes computacionais de alto desempenho. Sob tais ambientes, simulações computacionais podem ser realizadas com um nível de detalhe que não seria viável em abordagens computacionais seqüenciais (RIZZI, 2002).

Existem, pelo menos, duas grandes abordagens para a resolução de sistemas de equações em paralelo. Em uma delas, chamada de decomposição de dados, gera-se um único sistema de equações para todo o domínio que é resolvido através de um método numérico paralelizado. A segunda abordagem consiste na utilização de métodos de decomposição de domínio. Os MDDs são baseados 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 (MARTINOTTO, 2004).

Neste trabalho, utiliza-se métodos de decomposição de domínios como abordagem de resolução de sistemas de equações em paralelo. Esta escolha é baseada nos resultados obtidos em Galante *et al.* (2004-b), onde essa abordagem mostrou melhores resultados que a paralelização dos métodos numéricos. Além disso, a literatura técnica mostra que esta é a melhor abordagem para a paralelização de problemas que envolvem a discretização de um domínio físico (SAAD, 1996; SMITH; BJORSTAD; GROPP, 1996).

Os MDDs podem ser divididos em duas grandes classes: métodos de Schwarz, onde os subdomínios apresentam uma região de sobreposição e métodos de Schur, onde os subdomínios não apresentam região de sobreposição.

Sob o escopo apresentado, foram desenvolvidas implementações de métodos multigrid paralelizados pela abordagem de decomposição de domínio, com e sem sobreposição, para resolução paralela dos sistemas de equações gerados pela discretização de equações diferenciais parciais.

1.1 Motivação e Objetivos

O GMCPAD vem trabalhando no desenvolvimento de aplicações de alto desempenho desde 1998. Um resultado deste trabalho é o modelo HIDRA, um modelo computacional paralelo com balanceamento dinâmico de carga para a simulação do escoamento e do transporte de substâncias, tridimensional e bidimensional, em corpos de água, tendo como estudo de caso o Lago Guaíba (RIZZI, 2002; DORNELES, 2003). No entanto, importantes questões matemáticas, numéricas e computacionais não puderam ser contempladas no modelo HIDRA. Essas questões são objetos de pesquisa do modelo UnHIDRA (*Unstructured* HIDRA), um aprimoramento do modelo HIDRA, desenvolvido em conjunto pela UNIOESTE, UCS e UFRGS. Uma das

modificações neste novo modelo é a utilização de malhas não estruturadas. Logo, existe a necessidade de desenvolver métodos de solução que se adaptem a esse novo tipo de malha, já que o HIDRA utilizava-se de malhas estruturadas.

Assim, neste trabalho, são apresentadas as implementações desenvolvidas para a resolução de sistemas de equações gerados a partir de malhas não estruturadas. Mais especificamente são implementados métodos de solução utilizando métodos multigrid paralelizados por decomposição de domínios, com e sem sobreposição. Essa combinação, de métodos multigrid e métodos de decomposição de domínio, é chamada MG+MDD (DOUGLAS, 1996a).

Desta forma, neste trabalho são disponibilizados diferentes métodos de resolução paralela baseado em decomposição de domínio, utilizando técnicas multigrid para a solução de sistemas de equações lineares. Para cada método, são apresentados dois estudos de caso visando a validação das implementações. Os estudos de caso abordados são o problema da difusão de calor e o modelo de hidrodinâmica do modelo UnHIDRA.

1.2 Trabalhos Relacionados e Contribuições

Alguns trabalhos e dissertações relacionados com a paralelização de métodos de resolução de sistemas de equações lineares, foram desenvolvidos no GMCPAD. Alguns dos trabalhos já concluídos são:

- Paralelização de Métodos de Solução de Sistemas Lineares Esparsos com o DECK em Clusters de PCs, dissertação de Ana Paula Canal, onde foram implementadas versões paralelas do método do GC e do Método de Thomas para matrizes do tipo banda (CANAL, 2000);

- Paralelização de Métodos de Solução de Sistemas Lineares em Clusters de PCs com as Bibliotecas DECK, MPICH e Pthreads, dissertação na qual Delcino Picinin Jr. implementou e analisou a paralelização do método do GC e do Método do GMRES, utilizando MPI, DECK e Pthreads (PICININ, 2002);

- Resolução de Sistemas de Equações Lineares através de Métodos de Decomposição de Domínio, dissertação de André Luis Martinotto onde se abordou a solução paralela de sistemas de equações lineares através de métodos de decomposição de domínio (MARTINOTTO, 2004).

Estes trabalhos abordaram a solução de sistemas gerados a partir da discretização de EDPs em malhas estruturadas. Nesse sentido, a principal contribuição desta dissertação é a *resolução paralela de sistemas de equações lineares gerados a partir de malhas não estruturadas, utilizando métodos multigrid paralelizados por métodos de decomposição de domínio.*

Diversos pacotes oferecem solução multigrid e alguns deles permitem a solução em paralelo, como por exemplo, Madpack5 (DOUGLAS, 1996b), MUDPACK (ADAMS, 1993), Diffpack (BRUASET; LANGTANGEN; ZUMBUSCH, 1998) e o ParMGridGen (MOULITSAS; KARYPIS, 2001). No entanto, a oferta de pacotes para a resolução problemas utilizando malhas não estruturadas é ainda limitada e quando existem, geralmente são restritos a uma aplicação específica.

Alguns trabalhos que exploraram o uso de multigrid em aplicações científicas, no entanto em uma abordagem seqüencial são os trabalhos apresentados em Bittencourt (1996), Rabi (1998) e Moro (2004) . Um trabalho que merece destaque por utilizar multigrid em uma abordagem paralela é a tese de doutorado de Manel Soria Guerrero, intitulada "*Parallel Multigrid Algorithms for computational fluid dynamics and Heat Transfer*" (GUERRERO, 2000). Neste trabalho o autor apresenta o uso de multigrid na solução de problemas de dinâmica de fluidos e transferência de calor, empregando malhas estruturadas. Atualmente mais de 3600 referências podem ser encontradas no MGNET `http://www.mgnet.org`, que é o repositório oficial de métodos multigrid (DOUGLAS, 2006).

A idéia de combinar métodos multigrid e decomposição de domínio não é recente (BASTIAN; HACKBUSCH; WITTUM, 1998)(CHOW et al., 2005), no entanto as implementações desenvolvidas neste trabalho diferem em diversos aspectos dos trabalhos existentes. Em particular, utilizou-se o ciclo Full Multigrid V combinado com os método aditivo de Schwarz e com o método do complemento de Schur. Além disso, utilizou-se como métodos de solução apenas métodos iterativos do subespaço de Krylov, ao invés de utilizar métodos clássicos de solução, tal como Gauss-Seidel, comumente utilizado nas abordagens multigrid.

Assim, neste trabalho, são abordadas todos as etapas necessárias para a resolução dos sistemas de equações, desde a geração das malhas não estruturadas até a resolução utilizando métodos multigrid paralelos.

1.3 Organização do Trabalho

Este texto está organizado em nove capítulos, organizado da seguinte maneira. No Capítulo 2, inicialmente, aborda-se as questões relacionadas à geração de malhas não estruturadas e as ferramentas utilizadas para a geração das malhas. Em um segundo momento serão apresentados diferentes tipos de particionamentos que podem ser adotados e a ferramentas utilizada neste trabalho. No Capítulo 3 é feito um estudo sobre alguns métodos que podem ser utilizados para a resolução de sistemas de equações oriundos da discretização de equações diferenciais parciais. Inicialmente apresenta-se uma visão geral sobre sistemas de equações e na seqüência são abordados os métodos iterativos e os métodos multigrid.

O Capítulo 4 apresenta o ambiente computacional de desenvolvimento do trabalho. Nele são abordados aspectos relativos a arquitetura e a ferramentas de programação utilizados. Ainda apresenta-se algumas métricas para a avaliação do desempenho computacional.

No Capítulo 5 apresenta-se uma visão geral dos métodos de decomposição de domínio. A ênfase foi dada aos métodos utilizados neste trabalho: o método aditivo de Schwarz e o método do complemento de Schur.

Os Capítulo 6 e 7 têm como objetivo mostrar todas as questões relacionadas às soluções implementadas neste trabalho. Aborda-se detalhadamente todos os aspectos relevantes para o desenvolvimento dos métodos propostos, desde a geração das malhas até a paralelização dos métodos de solução.

As avaliações dos resultados obtidos com as paralelizações através de testes e comparações são apresentadas no Capítulo 8. Por fim, no Capítulo 9 resume-se o que foi desenvolvido ao longo deste trabalho. São apresentadas as conclusões e as contribuições deste trabalho.

2 MALHAS: GERAÇÃO E PARTICIONAMENTO

No processo de discretização de EDPs o domínio é mapeado em uma estrutura composta por um número finito de pontos, denominado malha. Nesses pontos os termos das EDPs são aproximados, resultando em sistemas de equações que devem ser resolvidos a cada passo de tempo, quando em problemas evolutivos.

Além disso, para a resolução de um determinado problema em paralelo, a malha deve ser distribuída entre os processadores disponíveis. Com o particionamento, cada subdomínio pode ser tratado em paralelo com os demais, diminuindo o tempo total na solução do problema.

Nesse capítulo inicialmente são abordadas as questões relacionadas à geração de malhas não estruturadas, e as ferramentas utilizadas para a geração das malhas empregadas. Em um segundo momento serão apresentados diferentes tipos de particionamentos que podem ser adotados e a ferramentas utilizada neste trabalho.

2.1 Classificação dos Tipos de Malha

Basicamente, existem dois tipos de malhas, caracterizados pela conectividade dos pontos: malhas estruturadas e malhas não estruturadas (FILIPIAK, 1996).

Em uma malha estruturada, cada ponto do interior da malha é adjacente ao mesmo número de elementos. Pode-se identificar os vizinhos de cada ponto através da soma dos índices (SHEWCHUK, 1999). Pode-se identificar a vizinhança de um ponto associando um sistema de coordenadas a cada linha da malha, podendo ser facilmente armazenados em uma matriz (BERN; EPPSTEIN, 1992). Por exemplo, os vizinhos do ponto (i, j) são $(i+1, j)$, $(i, j+1)$, $(i-1, j)$ e $(i, j-1)$, como mostrado na Figura 2.1.

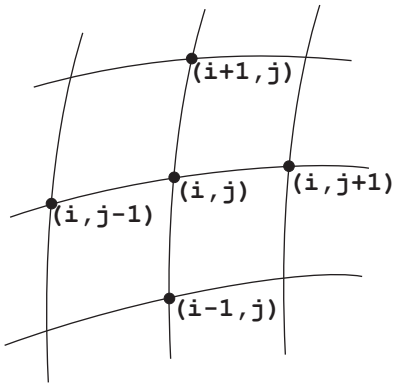


Figura 2.1: Exemplo de organização dos pontos de uma malha estruturada

Nas malhas não estruturadas, o número de vizinhos de um ponto não é necessariamente constante. Diferentemente das malhas estruturadas, nesse tipo de malha permite-se a existência de qualquer quantidade de vizinhos para um determinado ponto (BERN; PLASSMANN, 2000). Para o armazenamento desse tipo de malha, deve-se numerar todos os pontos (também conhecidos como vértices ou nodos) e os elementos (também conhecidos por polígonos ou células) formados, bem como a relação entre eles, como exemplificado na Figura 2.2.

Elemento	Vértices
1	1 2 3
2	1 2 4
3	2 3 5
4	2 5 6
5	2 4 6
6	5 6 7
7	6 7 8
8	6 8 9
9	6 9 10
10	4 6 10

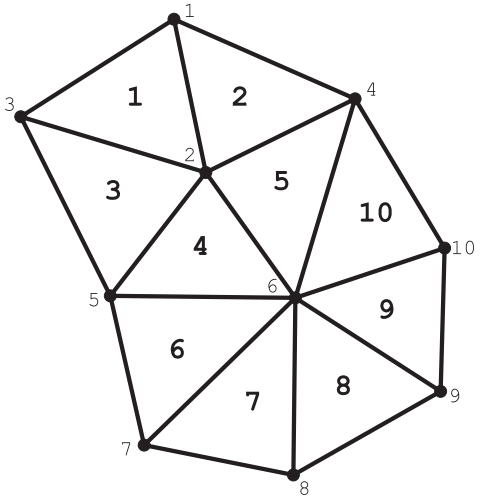


Figura 2.2: Exemplo de armazenamento dos dados de uma malha não estruturada

Nesse trabalho a discretização do domínio é feita utilizando-se malhas não estruturadas triangulares. Malhas não estruturadas conciliam uma boa representação do domínio computacional, já que diversos problemas são definidos em domínios com geometria irregular que nem sempre são apropriadamente discretizados por malhas estruturadas (SONI; THOMPSON, 2003). Para mais informações a respeito de malhas estruturadas consultar Knupp e Steinberg (1994), Mavriplis (1996) e Soni e Thompson (2003) .

2.2 Geração de Malhas Não Estruturadas Triangulares 2D

No atual estado da arte em geração de malhas não estruturadas, duas classes de métodos se destacam: *avanço de fronteira* e *triangulações de Delaunay* (FILIPIAK, 1996).

2.2.1 Avanço de Fronteira

Nos métodos de avanço de fronteira, os triângulos são gerados a partir da fronteira do domínio a ser coberto pela malha. O domínio é gradualmente preenchido por triângulos até que todo o domínio esteja completamente coberto.

O contorno do domínio é o ponto de partida para o processo de geração da malha. O primeiro passo consiste em discretizar a linha do contorno com pontos, como mostra a Figura 2.3. Esses pontos unidos por segmentos de reta (arestas) formam o contorno poligonal do domínio. Esse contorno poligonal corresponde à fronteira inicial de partida da malha (SHEWCHUK, 1997).

A partir deste polígono, adiciona-se triângulos ao domínio, com ao menos uma aresta pertencente à fronteira. A cada passo, atualiza-se a lista de arestas da fron-

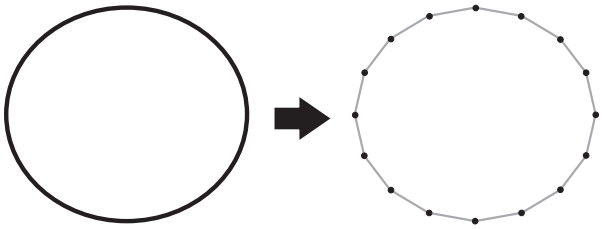


Figura 2.3: Discretização do contorno do domínio

teira, até que esta lista esteja vazia, o que significa que o processo de geração da malha está completo e todo o domínio foi coberto pela malha, como pode ser observado na Figura 2.4 (OWEN, 1998).

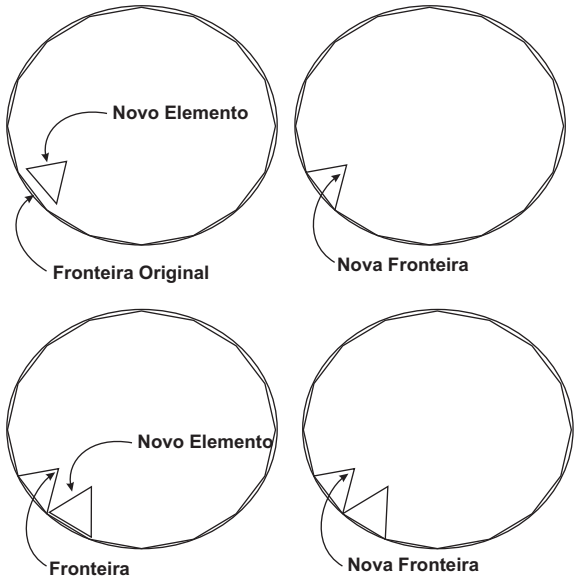


Figura 2.4: Exemplo de Avanço de Fronteira

Após a geração da malha pelo método de avanço de fronteira é comum utilizar-se um algoritmo de suavização (*smoothing*) para melhorar a qualidade da malha. O processo de amaciamento da malha consiste em ajustar os pontos dos triângulos de modo que eles permaneçam com a mesma topologia, mas possuam ângulos internos mais suaves (AUADA, 1997). Na Figura 2.5 pode-se observar a malha obtida após o processo ter sido completado.

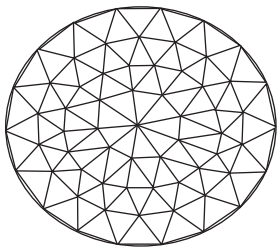


Figura 2.5: Malha gerada por Avanço de Fronteira

2.2.2 Triangulação de Delaunay

Este método é baseado em uma propriedade matemática formulada pelo matemático russo B. Delaunay em 1934 (O'ROURKE, 1998). Essa propriedade é chamada de *critério do círculo vazio*. O critério garante que o circuncírculo de um triângulo não contém em seu interior nenhum outro vértice, além dos três que definem este triângulo. Uma ilustração deste critério é mostrado na Figura 2.6.

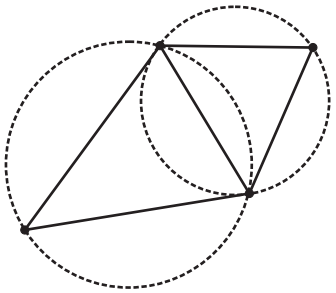


Figura 2.6: Exemplo do critério do círculo vazio

O método inicia com a discretização do contorno do domínio a ser triangularizado (Figura 2.7a). Na seqüência, triangula-se o domínio utilizando os pontos que definem o domínio (Figura 2.7b). Essa triangulação pode, ou não, representar a fronteira do domínio que está sendo coberto pela malha.

Esta triangulação é refinada gradativamente pela inserção de mais pontos (Figura 2.7c), criando triângulos adicionais, e preservando as propriedades Delaunay da malha. Esse refinamento é feito até que a malha atinja as características desejadas. Por fim a fronteira original do domínio é recuperado se o domínio for não-convexo (Figura 2.7d) (OWEN, 1998).

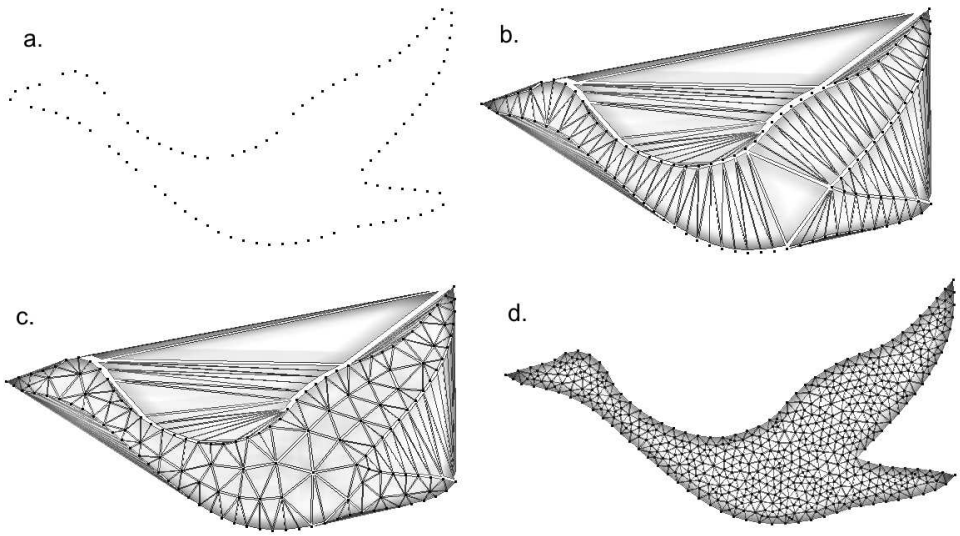


Figura 2.7: Passos de uma triangulação de Delaunay

Existem vários algoritmos diferentes para a geração das malhas através da triangulação de Delaunay. Exemplos podem ser encontrados em Fortune (1992), Modi (1997) , O'Rourke (1998) e Shewchuk (1999).

2.3 Qualidade das Malhas

Como já visto, a simulação computacional de um processo físico requer a discretização do domínio geométrico. O tempo de cálculo depende da quantidade de triângulos da malha, e a estabilidade e a convergência do método são bastante afetadas pela forma dos triângulos. Logo a qualidade de uma malha triangular deve levar em conta o número de triângulos e a forma destes.

Medidas típicas analisam o maior ou o menor dos ângulos, a razão entre a menor e a maior de suas arestas, a razão entre os raios dos círculos inscrito e circunscrito, etc, tendo por parâmetro esta relação nos triângulos eqüiláteros (SHEWCHUK, 2002).

2.3.1 Malhas Não Estruturadas Ortogonais

É importante salientar que cada aplicação pode requerer malhas com características específicas. Nos estudos de caso deste trabalho, a modelagem exige que a malha seja do tipo *não estruturada ortogonal*. Este tipo de malha está vinculada ao método de volumes finitos empregado na discretização das EDPs (CASULLI; WALTERS, 2000).

Uma malha é dita uma malha não estruturada ortogonal, se cada polígono desta malha possui um ponto, chamado de *centro*, de tal forma que o segmento que une os *centros* de dois polígonos adjacentes for ortogonal ao lado compartilhado por estes dois polígonos, conforme pode-se observar na Figura 2.8. Este *centro* nem sempre coincide com o centro geométrico do triângulo (CHENG; CASULLI, 2001).

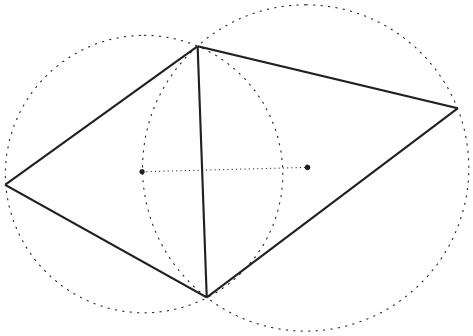


Figura 2.8: Detalhe de malha não estruturada ortogonal

Uma abordagem empregada para a obtenção destas malhas, é a utilização de uma malha de triângulos acutângulos, ou seja, as medidas dos ângulos são menores do que 90^o (CASULLI; WALTERS, 2000). Dessa forma, o centro é dado pelo circuncentro do triângulo. Caso o triângulo não se encaixe nessa exigência, o *centro* pode localizar-se fora do elemento, o que torna o elemento inválido. Assim, para a obtenção apropriada deste tipo de malhas, é desejavel que os triângulos se aproximem da forma de um triângulo eqüilátero.

Para a avaliação da qualidade da malha, pode-se utilizar duas métricas equivalentes: medição do ângulo máximo de cada triângulo e localidade do circuncírculo do triângulo. Na primeira métrica calcula-se o maior ângulo para cada triângulo. Se o triângulo contiver um ângulo maior que 90^o a malha não possui as características desejadas. Já na segunda métrica, utiliza-se uma primitiva geométrica chamada *ponto em polígono*, que determina se um ponto encontra-se no interior de um polígono (GALANTE, 2004). Dessa forma, aplica-se esta primitiva a cada triângulo da

malha para determinar se o circuncentro está, ou não, no interior do elemento. Em caso positivo para todos os triângulos, a malha é validada.

Existem softwares proprietários que geram eficientemente este tipo de malha, como por exemplo o Janet (`http://www.smileconsult.de/`) e o Argus One (`http://www.argusint.com/`), mas não foi encontrado nenhum software gratuito com estas características.

2.4 Pacotes para Geração de Malhas

Existem vários pacotes para a geração de malhas, tanto estruturadas quando não estruturadas. Uma ampla lista, mantida por Robert Schneiders, pode ver vista em: `http://www-users.informatik.rwth-aachen.de/~roberts/software.html`.

Neste trabalho são utilizados dois geradores de malha, o *Triangle* e o *Easymesh*.

2.4.1 Triangle

O Triangle é um programa desenvolvido em C, por Jonathan R. Shewchuk, para a geração e construção de malhas bidimensionais. Estas malhas podem ser geradas segundo a triangulação de Delaunay, a triangulação de Delaunay com restrição, e o diagrama de Voronoi. O Triangle é rápido, utiliza pouca memória, e calcula as triangulações através dos métodos de Delaunay e de Delaunay com restrição. As malhas são de boa qualidade e são geradas usando o algoritmo de refinamento de Delaunay de Ruppert (SHEWCHUK, 1997).

As características para a geração de malha incluem restrições de ângulo e de área do triângulo, e a inserção de buracos que podem ser definidos pelo usuário. Essas características foram importantes na escolha desse software para a geração das malhas neste trabalho. A malha pode ser visualizada através do software *Showme*, distribuído em conjunto com o Triangle.

Triangle pode ser obtido gratuitamente na Internet através do endereço `http://www-2.cs.cmu.edu/~quake/tripaper/triangle0.html`. Além do software, estão disponíveis artigos e documentação sobre o assunto.

2.4.2 Easymesh

O Easymesh, desenvolvido por Bojan Niceno, gera malhas não estruturadas bidimensionais em domínios genéricos, permitindo a geração de malhas em domínios com buracos e formados por diferentes materiais. As malhas são geradas através de triangulação de Delaunay, e a qualidade da malha é garantida por algoritmos de suavização.

As malhas podem ser visualizadas através do software *Showmesh*. Este visualizador permite visualizar a numeração dos pontos, elementos, fronteiras, além de apresentar recursos de zoom e rotação da malha.

Em `http://www-dinma.univ.trieste.it/nirftc/research/easymesh/` encontra-se o download do gerador e de visualizador, além da documentação e informações gerais sobre o software.

2.5 Particionamento da Malha

Para que um determinado problema possa ser resolvido de modo paralelo, via decomposição de domínio, usualmente, o domínio é particionado em um certo número de subdomínios, os quais são distribuídos entre os processadores. O problema principal nesse processo é como conseguir uma distribuição equilibrada de carga computacional entre os processadores e como minimizar a quantidade de comunicação entre processos, de forma a se obter um significativo ganho de desempenho ao se explorar o paralelismo.

Mais especificamente, as técnicas de particionamento de domínios devem ser desenvolvidas tendo os seguintes objetivos (AL-NASRA; NGUYEN, 1991):

- distribuir de forma balanceada a carga entre os processadores, onde cada processador deve receber, o número de elementos de forma proporcional a sua capacidade de processamento;

- minimizar o tempo de sincronização entre os processadores, através da minimização do número total de pontos da fronteira;

- o tempo gasto na partição do domínio deve ser pequeno em relação ao tempo total de solução do problema;

- o algoritmo deve ser capaz de tratar geometrias irregulares e discretizações arbitrárias.

A distribuição da carga de trabalho, considerando-se a arquitetura disponível e exigindo o balanceamento de carga e a minimização da comunicação dos processos, durante tempo de execução, é uma das etapas mais importantes da Computação Científica Paralela (RIZZI, 2002).

O problema da divisão do domínio computacional pode ser modelado como um problema de particionamento de grafos, onde os vértices representam os pontos da malha e as arestas a relação de vizinhança entre esses. Sob esta abordagem, o particionamento da malha pode ser visto como o problema de k-particionamento de grafos, que consiste em dividir um grafo em k subgrafos, de modo que cada subgrafo contenha um número semelhante de vértices, e que o número arestas entre os subgrafos seja o menor possível (DORNELES, 2003).

O particionamento de grafos é um problema NP-Difícil[1]. No entanto foram desenvolvidos diversos algoritmos que geram boas partições, e com custo computacional razoável (KARYPIS; KUMAR, 1995a).

2.5.1 Algoritmos de Particionamento

De acordo com Fjällström, os algoritmos de particionamento podem ser divididos em métodos globais, métodos locais e métodos multínivel como pode ser observado na Figura 2.9 (FJALLSTROM, 1998).

[1]Um problema é NP-Completo se é NP, ou seja, uma solução não determinística para o mesmo pode ser verificada em tempo polinomial (*Non-deterministic Polinomial*) e se todos os problemas em NP podem ser reduzidos a ele por uma transformação polinomial. Um problema NP-Difícil é um problema para o qual todos os problemas em NP podem ser reduzidos, sem ser necessariamente NP (ou seja, uma solução para o mesmo não é necessariamente verificável em tempo polinomial).

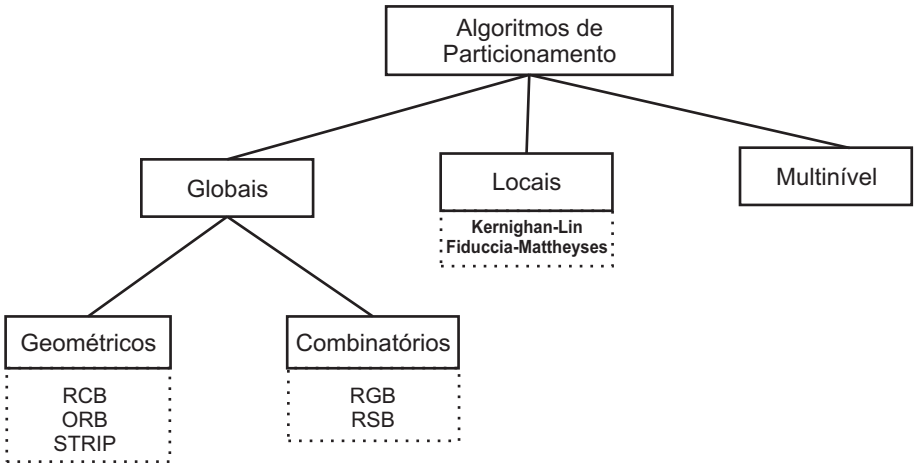


Figura 2.9: Uma classificação para os algoritmos de particionamento

Métodos globais são algumas vezes chamados de heurísticas de construção, uma vez que utilizam a descrição do grafo como entrada e geram uma partição. Os métodos globais podem ainda ser subdivididos em geométricos ou combinatórios.

Os algoritmos geométricos levam em consideração informações sobre a geometria do domínio. Alguns deles buscam dividir o domínio através de uma linhas (para domínios bidimensionais) ou planos (domínios tridimensionais). Estes algoritmos ignoram a colocação das arestas, considerando que os vértices próximos estão conectados por arestas. Exemplos dessa classe são os algoritmos STRIP (*stripwise partitioning*), RCB (*Recursive Coordinate Bisection*) e ORB (*Orthogonal Recursive Bisection*) (DORNELES, 2003),

Já os algoritmos combinatórios podem ser usados em grafos que não possuem um sistema de coordenadas associadas aos vértices, isto é, em grafos onde não há identificação do vértice como um ponto físico no espaço. Desta forma, durante o particionamento, essa classe de algoritmos considera apenas a conectividade dos vértices. Entre os métodos dessa classe cita-se: RSB (*Recursive Spectral Bisection*) (SAAD, 1996) e o RGB (*Recursive Graph Bissection*) (FJALLSTROM, 1998).

Os métodos locais, também chamados de heurísticas de melhoramento, recebem como entrada um grafo particionado e tentam melhorar a qualidade da partição, isto é, diminuir o corte de arestas, através do rearranjo dos vértices. Entre os métodos dessa classe os mais conhecidos são o Kernighan-Lin e o Fiduccia-Mattheyses (KERNIGHAN; LIN, 1970; FIDUCCIA; MATTHEYSES, 1982).

Os métodos multinível procuram combinar métodos globais e métodos locais. Esses métodos são assim chamados devido às diversas fases que compõem todo o processo de partição. Inicialmente o grafo é reduzido a algumas centenas de vértices e, então, realiza-se a partição deste grafo reduzido através de um algoritmo global, e, por fim, essa partição é projetada de volta ao grafo original. Durante o retorno ao grafo original, são feitos refinamentos na partição por um algoritmo local, com o objetivo de reduzir o número de vértices na fronteira entre domínios (KARYPIS; KUMAR, 1998).

Existem diversos pacotes e bibliotecas para particionamento de grafos. Os mais conhecidos e utilizados são: o METIS (KARYPIS; KUMAR, 1995b), JOSTLE (WALSHAW et al., 1995) e o CHACO (HENDRICKSON; LELAND, 1994). Neste trabalho utiliza-se o METIS devido a facilidade de aquisição e de uso.

2.5.2 METIS

O software METIS é um conjunto de programas especialmente desenvolvidos para realizar a partição de grafos e malhas de grande porte (KARYPIS; KUMAR, 1995b). Os algoritmos são baseados na partição multinível de grafos. Devido à alta qualidade das partições e ao reduzido tempo de processamento, o pacote METIS foi escolhido para realizar a partição das malhas.

Na Figura 2.10a pode-se observar um domínio quadrado com 394 triângulos e 224 nodos. Em 2.10b é mostrado o mesmo domínio particionado em 16 partes pelo METIS.

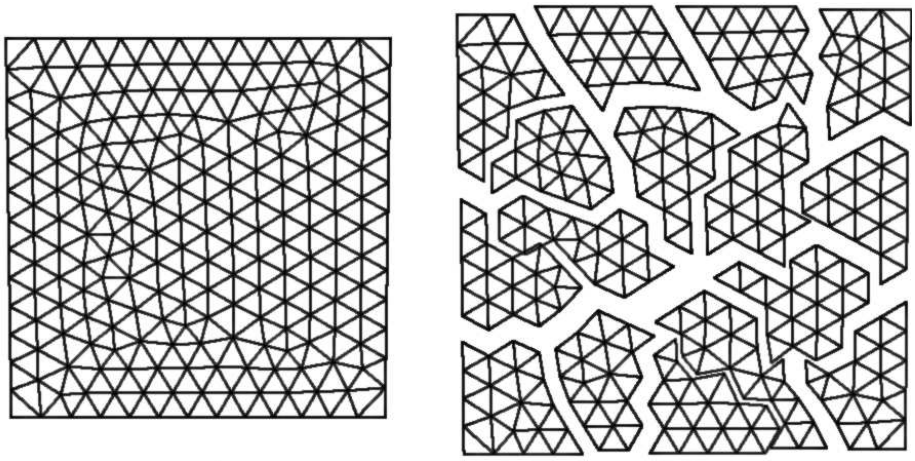


Figura 2.10: Exemplo de malha particionada em dezesseis subdomínios usando METIS

O METIS, atualmente em sua versão 4.0.1, está disponível gratuitamente através do endereço `http://www.cs.umn.edu/~karypis/metis/`, não sendo necessário licença de uso. METIS é portável para a maioria dos sistemas Unix, tais como AIX, SunOS, Solaris, IRIX, Linux, HP-UX, Free BSD e Unicos. Também está disponível uma versão pré-compilada para sistemas Windows.

Para a visualização das malhas particionadas pelo METIS, pode se utilizar o software PMVIS *(Partitioned Mesh Visualizer)*. O PVMIS permite a visualização de malhas não estruturadas 2D e 3D, com recursos de zoom, rotação, e explosão da malha particionada, como mostrado na Figura 2.10. O PMVIS está disponível para download no endereço `http://www-users.cs.umn.edu/~oztekin/pmvis/pmvis.html`.

O PMVIS está disponível para qualquer plataforma com compilador C++ e OpenGL disponíveis. O download da ferramenta e documentação pode ser obtido no endereço: `http://www-users.cs.umn.edu/~oztekin/pmvis/`.

2.6 Considerações Finais

Neste capítulo, foram discutidos os processos de geração e particionamento da malha. Estes processos irão influenciar consideravelmente nos demais passos necessários para a resolução em paralelo dos sistemas de equações tratados neste trabalho. A Figura 2.11 representa o processo da geração e particionamento das malhas. Têm-se inicialmente como entrada dados geométricos do domínio da aplicação, e como

saída do processo obtém-se a malha particionada. A geração e particionamento da malha serve de base para todo o processo de resolução de um problema, como apresentado no Capítulo 6.

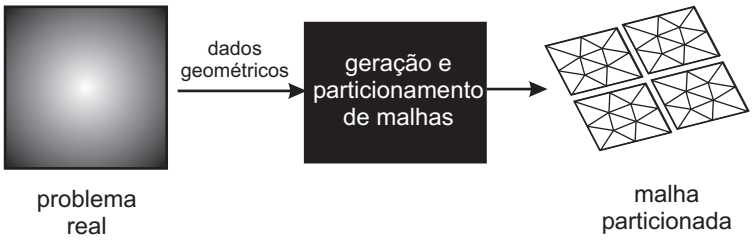


Figura 2.11: Processo de geração e particionamento da malha

Na Seção 6.1, encontram-se mais detalhes sobre a geração e particionamento das malhas utilizando as ferramentas citadas, bem como a utilização das malhas na geração dos dados de entradas para as aplicações propostas.

Para mais detalhes sobre os diversos métodos de geração de malhas não estruturadas, recomenda-se Filipiak (1996), Owen (1998), Bern e Plassmann (1996) e Soni e Thompson (2003) .

3 RESOLUÇÃO DE SISTEMAS DE EQUAÇÕES

O estudo de sistemas de equações é de grande importância na computação científica, pois estes resultam de modelos discretos provenientes de vários tipos de aplicação, tais como: programação linear, dinâmica dos fluidos, modelagem do clima e previsão meteorológica, etc. (SAAD, 1996).

O objetivo deste capítulo é apresentar alguns métodos que podem ser utilizados para a resolução de sistemas de equações oriundos da discretização de equações diferenciais parciais. Inicialmente apresenta-se uma visão geral sobre sistemas de equações e na seqüência são abordados os métodos iterativos e os métodos multigrid.

3.1 Sistemas de Equações

Um sistema de equações lineares pode ser definido como um conjunto de n equações com m variáveis independentes entre si, na forma genérica, como:

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & & & & \ddots & & \vdots & = & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & = & b_m \end{array}$$

onde a_{ij} e b_i são constantes reais e x_i são as incógnitas, para $i = 1, \cdots, m$ e $j = 1, \cdots, n$.

Usando as operações matriciais, o sistema linear acima pode ser escrito como uma equação matricial

$$Ax = b$$

para o qual se tem:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \text{e} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

A solução de um sistema de equações lineares é obtida através do cálculo de um vetor x, formado por valores que satisfaçam a igualdade $Ax = b$.

Existem, basicamente, duas classes de métodos que podem ser aplicados na resolução de sistemas de equações: os métodos diretos e os métodos iterativos (CISMASIU, 2002).

Os métodos diretos obtêm a solução baseados na fatorização da matriz de coeficientes A (FREUND; GOLUB; NACHTIGAL, 1992). Estes métodos resolvem os sistemas em um número finito e conhecido de passos e os erros que ocorrem se devem, essencialmente, aos arredondamentos na aritmética de ponto flutuante (RIZZI, 2002).

Métodos diretos são mais gerais e robustos que os métodos iterativos, pois podem ser utilizados na resolução de qualquer tipo de sistema. No entanto são inadequados para a resolução de sistemas esparsos, uma vez que não aproveitam a esparsidade da matriz de coeficientes do sistema, tornando essa abordagem impraticável, por problemas de armazenamento e pela dependência de operações e/ou de dados que dificulta a sua paralelização (PICININ, 2001; RIZZI, 2002).

Os algoritmos iterativos, por sua vez, utilizam a matriz apenas como um operador para construir iterativamente uma seqüência de vetores que converge para a solução de x (MAILLARD, 2005). E ao contrário dos métodos diretos, são freqüentemente utilizados na resolução de sistemas de equações esparsos e de grande porte, devido a sua potencialidade quanto à otimização de armazenamento e sua eficiência computacional.

Muitas aplicações científicas utilizam EDPs em sua formulação, e quando discretizadas, resultam em sistemas de equações altamente esparsos e de grande porte. Portanto, o emprego de métodos iterativos é mais apropriado nestes casos.

3.2 Métodos Iterativos

Os métodos iterativos podem ser classificados em estacionários ou não estacionários. Nos estacionários, cada iteração não envolve informações da iteração anterior e manipulam variáveis do sistema de equações lineares durante a resolução, através de operações elementares entre linhas e colunas da matriz. Os não estacionários encontram a solução através da minimização da função quadrática ou por projeção, manipulando os vetores e matrizes inteiros e incluem a hereditariedade em suas iterações, a cada iteração. Desse modo calcula-se um resíduo que é usado na iteração subseqüente (CANAL, 2000).

Neste trabalho, utiliza-se o método do resíduo mínimo generalizado (GMRES). Optou-se por este método por ser considerado um dos mais eficientes e robustos métodos iterativos para a solução de sistemas de equações lineares não-simétricos, como os gerados nos estudos de caso. Conforme se pode observar na Seção 3.3, estes métodos são empregados nos métodos multigrid para encontrar as aproximações em cada nível de malha.

3.2.1 GMRES

O GMRES é um método iterativo desenvolvido por Saad e Schultz (1986) utilizado na solução de sistemas de equações lineares de grande porte, esparsos e não simétricas (MAILLARD, 2005).

Considerando uma solução inicial $x^{(0)}$, uma solução aproximada é obtida através de $x^{(0)}+z$, onde z é um vetor no subespaço de Krylov. O GMRES procura encontrar um z tal que a norma do resíduo seja mínima, isto é, $x^{(0)}+z$ é solução do sistema

se $\left\|b - A(x^{(0)} + z)\right\|$ é minima.

O método GMRES possui como principal característica a construção de uma base ortonormal V no subespaço de Krylov. No método GMRES a base ortonomal no subespaço de Krylov é obtida através do processo de Gram-Schmidt Modificado (VALENTIM et al., 2004).

O GMRES é um método robusto e obtém a solução aproximada com norma residual mínima. Sua desvantagem é que os produtos matriz-vetor aumentam linearmente com as iterações e todos os vetores da base do subespaço de Krylov têm que ser armazenados, o que é um problema quando a dimensão m do subespaço cresce. A solução mais empregada é reinicializar o algoritmo, fixando-se a dimensão m do subespaço. Essa estratégia gera o conhecido GMRES(m) que tem a desvantagem de não ter a robustez do GMRES, uma vez que a convergência não é garantida. Mas, segundo Saad (1996) se a matriz é real e positiva, o GMRES(m) converge.

Um outro problema é encontrar o valor mais apropriado para m. Se m for muito pequeno, o GMRES(m) pode ter a convergência lenta, ou até mesmo falhar na convergência. Já para um valor muito grande de m, têm-se os inconvenientes de armazenamento em memória. Infelizmente não existem regras para a determinação de um m ideal, podendo este variar conforme o problema a ser tratado (WEISSTEIN, 2005). Neste trabalho, o valor adotado para m é 5. O Algoritmo 3.1 apresenta os principais componentes do método GMRES(m).

Algoritmo 3.1: GMRES

---

1. $r_0 = b - Ax_0$;
2. $\beta = \|r_0\|_2$;
3. $v_1 = r_0/\beta$;
4. $H_m = \{h_{ij}\}_{1 \leq i \leq m+1,\ 1 \leq j < m}$; $H_m = 0$;
5. $for(;;)$
6. $\quad h_{ij} = (w_j, v_i)$;
7. $\quad w_j = w_j - h_{ij} v_i$;
8. $\quad h_{j+1,j} = \|w_j\|_2$;
9. $\quad if(h_{j+1,j} == 0)$
10. $\quad\quad m = j$;
11. $\quad\quad goto\ 13$;
12. $\quad v_{j+1} = w_j / h_{j+1,j}$;
13. $Computar \quad y_m\ de\ \|\beta e_1 - H_m y\|_2$;
14. $x_m = x_0 + V_m y_m$

3.3 Métodos Multigrid

Ao utilizar-se um método iterativo para a resolução do sistema $Ax = b$, utilizando como aproximação inicial um vetor x_0, o método consegue rapidamente diminuir o erro se a onda for de alta freqüência, mas falha ao tentar remover as componentes suaves do erro, como pode ser visto na Figura 3.1.

Logo, conclui-se que as componentes do erro de baixa freqüência são as responsáveis pela lenta convergência demonstrada pelos processos iterativos que usam um único nível de malha (BRIGGS, 1987; BITTENCOURT, 1996). Como as componentes de altas freqüência são aquelas cujos comprimentos de onda são menores

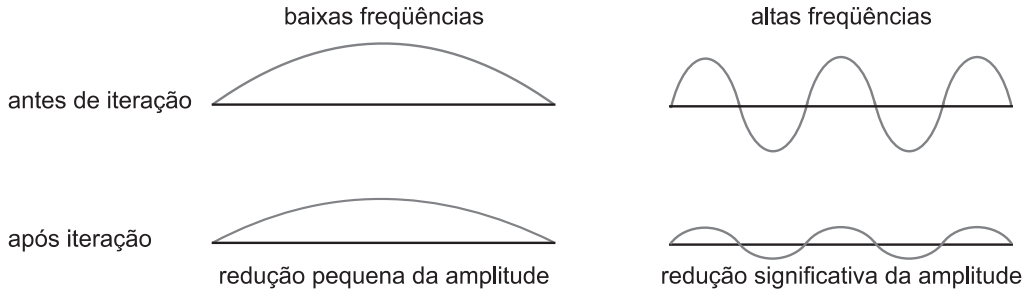


Figura 3.1: Comportamento de componentes do erro em métodos iterativos

que o espaçamento da malha computacional, a taxa de convergência cai conforme a mesma se torna mais refinada, já que com a diminuição do tamanho dos elementos da malha, os erros oscilatórios de alta freqüência acabam se tornando proporcionalmente menos oscilatórios na malha mais fina, dificultando a convergência do método. Um exemplo é mostrado na Figura 3.2

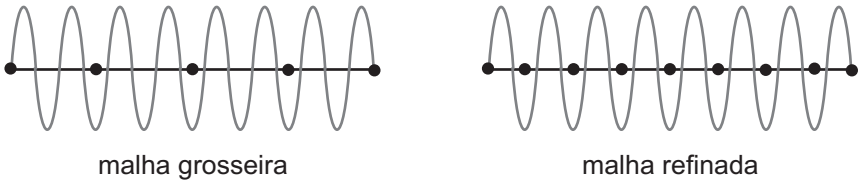


Figura 3.2: Exemplo de comportamento do erro oscilatório em malhas de diferentes refinamentos

Uma boa solução para este tipo de problema seria adaptar os métodos iterativos convencionais para que consigam eliminar tanto as componentes suaves (baixa freqüência) do erro quanto as oscilatórias (alta freqüência). Partindo desta idéia surgiram os métodos multigrid (TROTTENBERG; OOSTERLEE; SCHÜLLER, 2001).

Os métodos multigrid baseiam-se na premissa de que cada faixa de freqüência do erro deve ser suavizada no espaçamento mais adequado para tal. Para que as componentes do erro baixa freqüência possam ser eliminados com eficiência, os métodos multigrid procuram trabalhar com uma seqüência de malhas M_1, M_2, ..., M_n, cada vez mais grossas, onde então o erro pode ser rapidamente suavizado. Em cada nível de malha, as componentes do erro correspondentes são eficientemente reduzidas, acelerando o processo de convergência. Um exemplo de uma seqüência de malhas é mostrada na Figura 3.3.

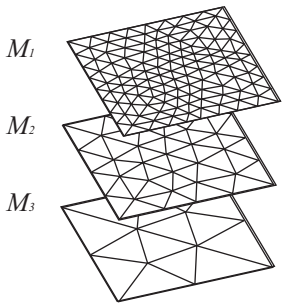


Figura 3.3: Exemplo de Seqüência de Malhas

Assim, o objetivo dos métodos multigrid é resolver o problema definido na malha mais refinada M_1, empregando as demais malhas como esquemas de correção (WESSELING, 1992).

Os métodos multigrid baseiam-se em três elementos centrais: transferência de informações entre malhas, iterações aninhadas, e correção em malha grossa (BRIGGS, 1987), descritos em detalhes nas seções 3.3.1, 3.3.2 e 3.3.3, respectivamente.

3.3.1 Transferência de Informações entre Malhas

Para que se possa utilizar os diversos níveis de malha na solução de um determinado problema, define-se dois operadores para transferência de informações ente malhas, como mostra a Figura 3.4.

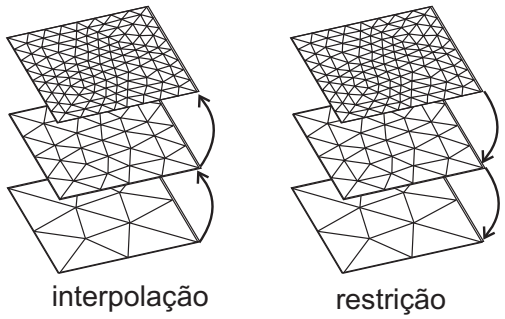


Figura 3.4: Operadores de transferência entre níveis de malha

O primeiro operador transfere informações da malha grossa M_n para a malha mais fina M_{n-1}, sendo denominado operador de *interpolação*, definido por:

$$I_{M_n}^{M_{n-1}} : M_n \rightarrow M_{n-1}$$

O segundo operador, definido por:

$$I_{M_{n-1}}^{M_n} : M_{n-1} \rightarrow M_n$$

transfere informações da malha mais fina para a mais grossa, e é conhecido como operador de *restrição*.

A forma destes operadores pode variar com o tipo de elemento ou com o tipo de problema. Diversas formas de interpolação e restrição podem ser encontradas em Briggs (1987) e Trottenberg, Oosterlee *et al.* (2001). Os operadores utilizados neste trabalho são descritos na Seção 7.1.

3.3.2 Iterações Aninhadas

A partir da definição dos operadores de transferência, pode-se definir a estratégia de iterações aninhadas (*nested iteration*). O objetivo é encontrar uma melhor aproximação inicial para a solução através de iterações em malhas mais grossas. Como o sistema gerado a partir da malha mais grossa possui um número menor de incógnitas, o custo computacional de sua resolução é menor, em comparação à resolução do sistema gerado a partir da malha mais refinada.

Logo, pode-se obter uma aproximação inicial para $Ax = b$ empregando as malha mais grossas (SARTORETTO, 2005), como mostrado no Algoritmo 3.3.

Algoritmo 3.3 : *Iteracoes_Aninhadas*

1. $x_{M_{n-1}} = I_{M_n}^{M_{n-1}}\ S(A_{M_n}, x_{M_n}, b_{M_n})$
2. $x_{M_{n-2}} = I_{M_{n-1}}^{M_{n-2}}\ S(A_{M_{n-1}}, x_{M_{n-1}}, b_{M_{n-1}})$

   $\vdots$
3. $x_{M_2} = I_{M_3}^{M_2}\ S(A_{M_3}, x_{M_3}, b_{M_3})$
4. $x_{M_1} = I_{M_2}^{M_1}\ S(A_{M_2}, x_{M_2}, b_{M_2})$

onde:

- n é a quantidade de níveis de malha utilizados;
- A_{M_i} representa a matriz relacionada à malha M_i;
- x_{M_i} representa o vetor das incógnitas relacionada à malha M_i;
- b_{M_i} representa o vetor dos termos independentes relacionada à malha M_i;
- $S(A, x, b)$ é um operador que descreve um método de solução iterativa. Esta solução pode ser completa ou apenas a execução de algumas iterações do método.

O processo é iniciado no nível mais grosseiro da malha, onde é encontrado uma solução inicial para x. Esta solução é então transferida para um nível acima através do operador de interpolação. Esse processo repete-se até que o nível mais refinado da malha seja alcançado, e por conseqüência, uma boa aproximação para x. Na Figura 3.5 apresenta-se um diagrama representando a estratégia de iterações aninhadas.

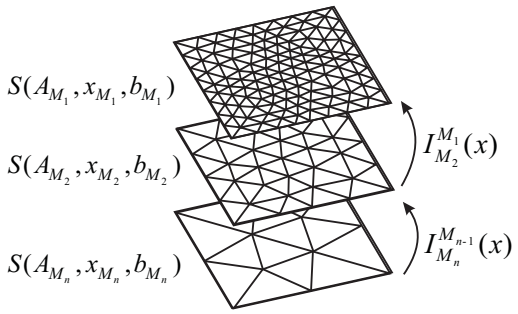


Figura 3.5: Diagrama da estratégia de iterações aninhadas

Esta estratégia não garante que ao final, a solução em M_1 não contenha componentes de erro suaves (baixa freqüência). O emprego da correção do erro proveniente das malhas grossas evita esta limitação (BITTENCOURT, 1996).

3.3.3 Correção do Erro em Malha Grossa

Para se compreender melhor o método de correção em malha grossa (*coarse grid correction*), algumas considerações iniciais devem ser feitas. Suponhamos que se queira resolver o sistema linear $Ax = b$ de modo iterativo, para tal denotamos por x a solução exata deste sistema e por x' uma solução aproximada do sistema, gerada por um método iterativo. Da relação entre x e x', pode-se definir duas medidas: o *erro* e o *resíduo*.

O erro é definido por:

$$e = x - x' \tag{3.1}$$

e infelizmente, é tão inacessível quanto a solução exata propriamente dita. Entretanto, uma medida computável de quão bem x' aproxima x é o resíduo, definido por:

$$r = b - Ax' \tag{3.2}$$

O resíduo r representa a quantidade pela qual a aproximação x' falha em satisfazer o problema original $Ax = b$.

Das definições (3.1) e (3.2) podemos escrever:

$$\begin{aligned} Ax' &= b - r \\ A(x - e) &= b - r \\ Ae &= r + Ax - b \\ Ae &= r \end{aligned} \tag{3.3}$$

Usando as equações (3.2) e (3.3), o esquema de correção em malha grossa obtém uma aproximação para o erro na malha mais grossa que é utilizada para corrigir a solução na malha mais fina. A correção utilizando a equação residual para iterar sobre o erro é dada pelo Algoritmo 3.4.

Algoritmo 3.4 : *Correcao*

1. $x_{M_{n-1}} = S(A_{M_{n-1}}, x_{M_{n-1}}, b_{M_{n-1}})$
2. $r_{M_{n-1}} = b_{M_{n-1}} - Ax_{M_{n-1}}$
3. $I_{M_{n-1}}^{M_n}(x_{M_{n-1}})$
4. $e_{M_n} \overset{(4.7)}{=} S(A_{M_n}, e_{M_n}, r_{M_n})$
5. $I_{M_n}^{M_{n-1}}(e_{M_n})$
6. $x_{M_{n-1}} = x_{M_{n-1}} + e_{M_{n-1}}$

Após iterar na malha fina M_{n-1} até que a convergência se deteriore, passa-se então a iterar na equação do resíduo em uma malha mais grossa M_n, obtendo-se uma aproximação para o erro, a qual corrige a solução x na malha fina (TROTTENBERG; OOSTERLEE; SCHÜLLER, 2001). A representação gráfica da correção em malha grossa pode ser encontrada na Figura 3.6.

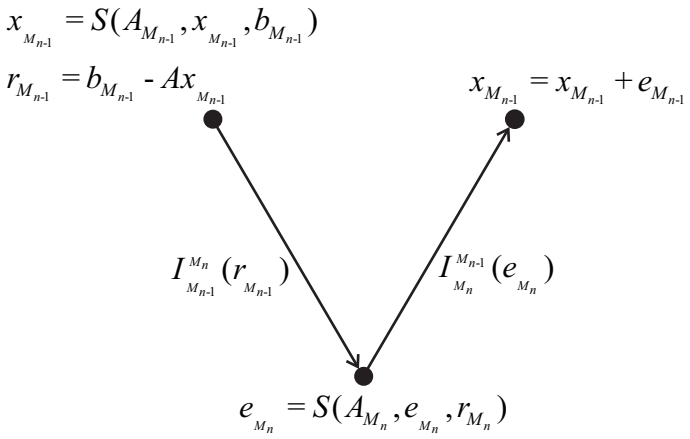


Figura 3.6: Representação gráfica da correção em malha grossa

3.3.4 Ciclos Multigrid

A combinação entre os elementos apresentados definem uma família de métodos multigrid, denominados *ciclos*. Alguns ciclos são apresentados a seguir.

3.3.4.1 Ciclos μ

A partir do uso de recursividade na estratégia de correção em malha grossa, pode-se definir uma família de métodos multigrid, denominados Ciclos μ (WESSELING, 1992; BRIGGS, 1987), dado pelo Algoritmo 3.5 (BITTENCOURT, 1996).

Algoritmo 3.5 : $M\mu(M_{n-1})$

---

1. $x_{M_{n-1}} = S(A_{M_{n-1}}, x_{M_{n-1}}, b_{M_{n-1}});$
2. $if(M_{n-1} \neq malha\ mais\ grossa)$
3. $\quad b_{M_n} = I_{n-1}^{n}(b_{M_{n-1}} - A_{M_{n-1}} x_{M_{n-1}});$
4. $\quad x_{M_n} = 0;$
5. $\quad for\ (i = 0;\ i < \mu;\ i++)$
6. $\quad\quad x_{M_n} = M\mu(M_n);$
7. $\quad x_{M_{n-1}} = x_{M_{n-1}} + I_{n}^{n-1}(x_{M_n});$
8. $x_{M_{n-1}} = S(A_{M_{n-1}}, x_{M_{n-1}}, b_{M_{n-1}});$

Na notação usada, x_{Mn} indica a solução da equação do resíduo e_{Mn}; da mesma maneira b_{Mn} é utilizado ao invés de r_{Mn}, pois os mesmos são termos independentes dos sistemas de equações residuais envolvidos.

Para $\mu = 1$ o algoritmo se reduz ao ciclo em V (Figura 3.7a), o qual partindo da malha mais fina alcança a malha mais grossa mapeando o resíduo entre as malhas, retornando para a malha mais fina aplicando as correções em cada nível. Para $\mu = 2$ o algoritmo é chamado de ciclo em W (Figura 3.7b).

3.3.4.2 Full Multigrid

Como visto na Seção 3.3.4, os ciclos V e W baseiam-se em correções em malha aplicadas recursivamente. Uma outra classe de algoritmos chamada *Full Multigrid*,

Figura 3.7: Ciclo V e Ciclo W

além de utilizar correção em malha grossa, incorpora também o conceito de iterações aninhadas. Essa classe aborda basicamente duas estratégias:

- *Full Multigrid V* (FMV);
- *Full Multigrid W* (FMW);

representados na Figura 3.8 e apresentados nos algoritmos a seguir.

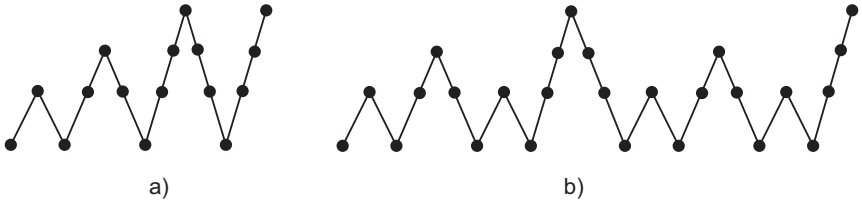


Figura 3.8: Ciclos FMV e FMW

A idéia básica do FMV é combinar o ciclo V com a técnica de iterações aninhadas. A solução inicia-se na malha mais grossa, e então usa-se esta solução como aproximação inicial para o próximo nível, e na seqüência, um ciclo V é executado. Este processo repete-se até que o nível mais alto seja alcançado. A execução do FMW é análogo à execução do FMV, mas executando ciclos W ao invés de ciclos V. O Algoritmo 3.6 apresenta uma formulação recursiva para as estratégias Full Multigrid.

Algoritmo 3.6 : $FM\mu(M_n)$

---

1. $if(M_{n-1} \neq malha\ mais\ grossa)$
2. $\quad b_{M_n} = I^n_{n-1}(b_{M_{n-1}} - A_{M_{n-1}} x_{M_{n-1}})$;
3. $\quad x_{M_n} = 0$;
4. $\quad x_{M_n} = FM\mu(x_{M_n}, b_{M_n})$;
5. $\quad x_{M_{n-1}} = x_{M_{n-1}} + I^n_{n-1}(x_{M_n})$;
6. $x_{M_{n-1}} = M\mu(M_{n-1})$;

Neste algoritmo, tomando μ igual a 1 têm-se FMV, e com μ igual a 2 obtêm-se FMW. Algumas variantes destes algoritmos podem ser utilizadas. Por exemplo, ao final de um ciclo FMV, concatenam-se vários ciclos V, sendo este esquema denominado FMVV (BITTENCOURT, 1996).

3.4 Considerações Finais

Este capítulo proporcionou uma visão geral sobre a resolução de sistemas de equações. Os sistemas podem ser resolvidos através de duas classes de métodos: métodos diretos e métodos iterativos. Dadas as características das matrizes geradas nos estudos de caso deste trabalho, apresentados no Capítulo 6, emprega-se métodos iterativos acelerados por multigrid, como apresentado na Figura 3.9.

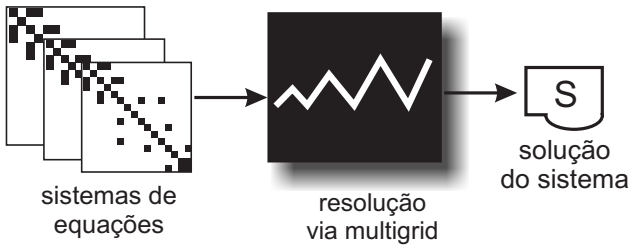


Figura 3.9: Resolução de Sistema de Equações: Visão geral

Basicamente, os métodos multigrid consideram uma seqüência de malhas para a solução do sistema de equações. O objetivo é resolver o problema na malha mais fina empregando as demais malhas como esquemas de correção. Neste trabalho, emprega-se o algoritmo full multigrid V (FMV) na aceleração do GMRES.

Uma vez que os sistemas de equações resultantes da discretização de EDPs, em aplicações realísticas, são de grande porte, é conveniente o uso de processamento paralelo. O próximo capítulo trata do ambiente de desenvolvimento de aplicações paralelas, utilizado para o desenvolvimento deste trabalho. Além disso são abordadas questões relacionadas à programação paralela.

4 PROCESSAMENTO PARALELO

Nos sistemas computacionais convencionais cada instrução do programa é executada seqüencialmente, uma após a outra pelo processador que compõem a máquina. No entanto, desde o desenvolvimento dos primeiros computadores, sempre se buscou uma forma alternativa de executar mais instruções simultaneamente. O objetivo sempre foi aumentar a velocidade de processamento para que aplicações complexas pudessem ser resolvidas cada vez mais rapidamente com o auxílio da computação. Dessa forma surgiu o processamento paralelo (CODENOTTI; LEONCINI, 1992).

O processamento paralelo pode ser definido como o processamento de informações, com ênfase na exploração de eventos concorrentes no processo computacional. O processamento paralelo implica na divisão de uma determinada aplicação em partes, de maneira que essas partes possam ser executadas concorrentemente, por vários elementos de processamento.

Nesse capítulo aborda-se o ambiente computacional utilizado, bem como alguns aspectos da programação paralela, e por fim algumas métricas para a avaliação do desempenho computacional.

4.1 Ambiente computacional

As implementações propostas nesse trabalho foram desenvolvidas para explorar o paralelismo em clusters de PCs. Nessa seção apresenta-se a arquitetura utilizada, bem como os mecanismos de software utilizados para a exploração do paralelismo nessa arquitetura.

4.1.1 Clusters

Conceitualmente, um *cluster* é uma coleção de computadores (estações de trabalho, máquinas pessoais ou SMPs), chamados de nodos, os quais são utilizados exclusivamente para obtenção de alto desempenho. Estas máquinas são fisicamente interconectadas por uma rede local ou uma rede de alto desempenho (BUYYA, 1999).

Como a principal motivação do uso de máquinas paralelas é a obtenção de desempenho, uma comparação dos modelos baseada na relação entre seu custo e o benefício resultante acaba se transformando em uma relação entre o custo e o desempenho obtido. Mesmo com as dificuldades de se comparar o desempenho entre os modelos, no caso da relação custo/desempenho, é muito clara a vantagem dos *clusters* em relação aos outros modelos de máquinas paralelas. A combinação de baixo custo de aquisição e de manutenção desses sistemas, por causa da utilização

de componentes de propósito geral, aliado às opções de redes de baixa latência, vem aumentando o interesse por essas máquinas nos últimos anos.

Por estes motivos, o uso desse tipo de arquitetura vem tendo um aumento significativo nos últimos anos. No endereço `http://www.top500.org`, de atualização semestral, que lista as 500 máquinas com maior poder de processamento do mundo, pode-se observar um número cada vez maior de *clusters*. Na edição de novembro de 2004, os *clusters* ocupavam 58,8% das posições e em junho de 2005 já totalizam 60,8% das máquinas, como pode ser visto na Figura 4.1.

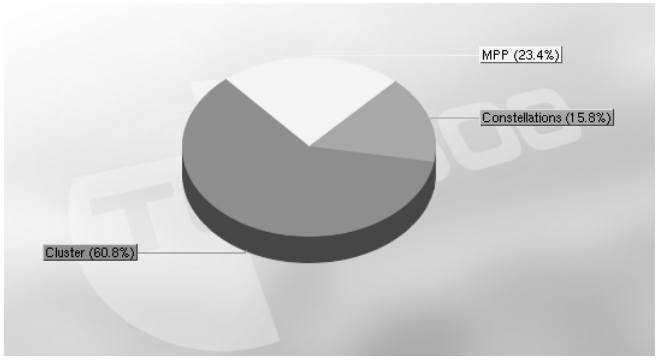


Figura 4.1: Distribuição dos tipos de arquitetura no Top 500

O nível de paralelismo a ser explorado em um *cluster* depende, em parte, do tipo de arquitetura existente, ou seja, em *clusters* formados por máquinas multiprocessadas, existe a possibilidade da exploração do paralelismo intra-nodos em conjunto com a exploração do paralelismo inter-nodos. Já em *clusters* formados por máquinas monoprocessadas, somente o paralelismo inter-nodos pode ser explorado.

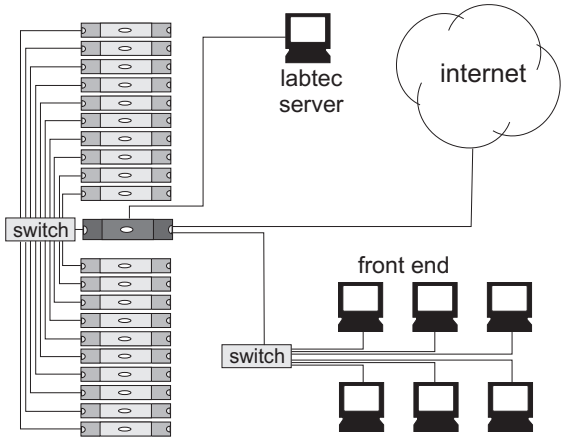


Figura 4.2: Cluster labtec

As estratégias e implementações propostas nesse trabalho foram desenvolvidas para uma exploração eficiente do paralelismo em clusters de PCs. Para tal, utilizou-se o *cluster* do Laboratório de Tecnologia em *Clusters* (LabTeC) do Instituto de Informática da UFRGS, desenvolvido em conjunto com a Dell Computadores.

O *cluster* labtec é constituído por 21 nodos, onde 20 desses são dedicados exclusivamente para processamento e 1 nodo servidor. A interconexão dos nodos de processamento é feita através de um *switch* Gigabit Ethernet. No que se refere aos nodos desse *cluster*, cada nodo de processamento do *cluster* é um Dual Pentium III

1.1 GHz, com 1 GB de memória RAM, 512 KB de cache e disco rígido SCSI com 18 GB; o nodo servidor é um Dual Pentium IV Xeon 1.8 GHz, com 1 GB de memória RAM e disco rígido SCSI com 36 GB. Uma ilustração do cluster labtec é apresentada na Figura 4.2.

4.1.2 Biblioteca de Troca de Mensagens

As bibliotecas de troca de mensagens são ferramentas que possibilitam o desenvolvimento de aplicações paralelas em máquinas com memória distribuída. A função de uma biblioteca de troca de mensagens é permitir que processos em diferentes máquinas possam trocar informações através de uma rede de interconexão.

As bibliotecas de troca de mensagens estão localizadas entre o sistema operacional e a aplicação. Essas bibliotecas são softwares que permitem o uso dos recursos do sistema operacional de maneira mais fácil. Neste trabalho utiliza-se uma biblioteca MPI para a exploração do paralelismo em *clusters*.

O MPI é um padrão para bibliotecas de troca de mensagens. Foi desenvolvido durante 1993 e 1994 por um grupo representantes de empresas, órgãos governamentais e universidades, chamado de *MPI Forum* (EL-REWINI; LEWIS, 1998). O documento que define o padrão *MPI: A Message-Passing Standard* encontra-se em `http://www.mcs.anl.gov/mpi/`.

O padrão MPI especifica a sintaxe e a semântica para 125 funções, divididas entre primitivas de gerência, primitivas de comunicação ponto a ponto e primitivas para comunicação coletiva (PACHECO, 1997).

As rotinas de comunicação compõem o núcleo principal do MPI. Existem rotinas para comunicação ponto-a-ponto, que envolvem apenas o envio e recebimento entre um par de processos. Já as coletivas, permitem o envio de mensagens de e para um grupo de processos. Essas mensagens podem ser de redução, sincronização ou distribuição de dados (SNIR et al., 1996). Na Figura 4.3 pode-se observar um exemplo simples de programa em C utilizando primitivas MPI.

```
#include "mpi.h"
main( argc, argv )
int argc;
char **argv;
{
    char message[20];
    int myrank;
    MPI_Status status;
    MPI_Init( &argc, &argv );
    MPI_Comm_rank( MPI_COMM_WORLD, &myrank );
    if (myrank == 0)
    {
        Strcpy(message,"Hello world!!!”);
        MPI_Send(message, strlen(message)+1, MPI_CHAR, 1, 99, MPI_COMM_WORLD);
        printf("rank%d sent: message\n", myrank, message);
    }
    else
    {
        MPI_Recv(message, 20, MPI_CHAR, 0, 99, MPI_COMM_WORLD, &status);
        printf("rank%d received: %s\n", myrank, message);
    }
    MPI_Finalize();
}
```

Figura 4.3: *Hello World* em MPI

Atualmente o MPI possui diversas implementações comerciais ou de domínio público. Neste trabalho utilizou-se a implementação de domínio público MPICH, do *Argonne National Laboratory*. Seu *download* e documentação completa pode ser encontrada em: `http://www-unix.mcs.anl.gov/mpi/mpich/download.html`.

4.2 Avaliação de Desempenho Computacional

Um dos fatores que justifica a necessidade de processamento paralelo é a possibilidade de aumentar a velocidade de processamento e reduzir o tempo de execução de uma tarefa. Diferentes métricas podem ser utilizadas para determinar se a utilização do processamento paralelo está sendo vantajosa e quantificar o desempenho alcançado. Dentre estas pode-se citar o tempo de execução, o *speedup* e a eficiência (EL-REWINI; LEWIS, 1998).

4.2.1 Tempo de Execução

O tempo de execução (T_{exec}) de um programa paralelo é o tempo decorrido desde o primeiro processador iniciar a execução do programa até o último terminar. A fórmula para determinar o tempo de execução é dada por:

$$T_{exec} = T_{final} - T_{inicial}$$

onde $T_{inicial}$ é o tempo do início da execução do programa e T_{final} é o tempo do término da execução do mesmo programa.

Nem sempre o tempo de execução é a métrica mais conveniente para avaliar o desempenho de um programa paralelo. Como o tempo de execução tende a variar com o tamanho do problema, o tempo de execução deve ser normalizado quando existe a comparação de desempenho em problemas de diferentes grandezas (FOSTER, 1995).

4.2.2 *Speedup*

Speedup (S_p) é uma medida utilizada para determinar o aumento de velocidade obtido durante a execução de um programa utilizando p processadores, em relação a sua execução seqüencial. O *speedup* é dada pela fórmula:

$$S_p = T_{seq}/T_{par}$$

onde T_{seq} é o tempo obtido na execução do algoritmo seqüencial e T_{par} é o tempo obtido utilizando p processadores. No entanto, em alguns casos o tempo seqüencial é substituído pelo tempo do algoritmo paralelo fazendo uso de apenas um processador, para se evitar a comparação de algoritmos diferentes.

O caso ideal é quando o $S_p = p$, isto é, a velocidade de processamento torna-se proporcional à quantidade de processadores utilizada. Mas existem três fatores principais que degradam essa situação ideal: a sobrecarga que a comunicação representa para os processadores, algoritmos que são dificilmente paralelizáveis e casos onde a granulação é inadequada para o tipo de arquitetura utilizada (ALMASI; GOTTLIEB, 1989).

4.2.3 Eficiência

Outra medida amplamente utilizada é a eficiência (E_p). Ela é definida com a relação entre o *speedup* (S_p) e o número de processadores p, ou seja, é o quanto os processadores estão sendo utilizados.

$$E_p = S_p/p$$

A eficiência varia entre 0 e 1, para eficiências variando de 0% e 100% respectivamente (ALMASI; GOTTLIEB, 1989). Por exemplo, se com a execução de uma aplicação paralela é obtido o valor de $E_p = 0.8$, esse valor indica uma eficiência de 80% na utilização dos processadores.

4.3 Considerações Finais

Neste capítulo apresentou-se o ambiente de desenvolvimento do trabalho. Inicialmente foram apresentados conceitos relativos à arquitetura utilizada (*clusters* de PCs) e as características desse tipo de arquitetura. Para maiores informações sobre arquiteturas paralelas e clusters de computadores recomenda-se De Rose (2001) e Buyya (1999).

Para a exploração do paralelismo existem diferentes bibliotecas, que proporcionam primitivas para facilitar o desenvolvimento de aplicações. Neste trabalho utilizou-se a biblioteca de troca de mensagens MPI. Para maiores informações sobre o MPI recomenda-se Snir *et al.* (1996) e Pacheco (1997).

Abordou-se também algumas métricas para a avaliação de desempenho de uma aplicação paralela. No Capítulo 8 são utilizadas estas métricas nas análises de desempenho das aplicações desenvolvidas neste trabalho.

No próximo capítulo são discutidas as técnicas utilizadas para a resolução de sistemas de equações em paralelo.

5 MÉTODOS DE DECOMPOSIÇÃO DE DOMÍNIO

A expressão *decomposição de domínio* possui diferentes interpretações em diferentes áreas do conhecimento. Não existe unanimidade na literatura técnica sobre a terminologia. Na computação, o termo geralmente é relacionado à distribuição dos dados entre os processadores em uma arquitetura de memória distribuída (SMITH; BJORSTAD; GROPP, 1996), ou ainda, relacionado ao particionamento do domínio computacional (malha, grafo) em subdomínios. Já na matemática, o termo refere-se a uma técnica para a resolução de equações diferenciais parciais (MARGETTS, 2002).

Neste trabalho, métodos de decomposição de domínio (MDD) designam um conjunto de técnicas e métodos matemáticos, numéricos e computacionais para resolver problemas em computadores paralelos (CHAN; MATHEW, 1994). Um MDD é caracterizado pela divisão do domínio computacional, que é particionado em subdomínios empregando algoritmos de particionamento, como mostrado na Seção 2.3. A solução global do problema é obtida através da combinação dos subproblemas que são resolvidos localmente. Cada processador é responsável por encontrar a solução local de um ou mais subdomínios que a ele são alocados, e então, essas soluções locais são combinadas para fornecer uma aproximação para a solução global (GALANTE, 2003).

Uma ilustração para o emprego de métodos de decomposição de domínio na solução de sistemas de equações lineares (SEL) é mostrado na Figura 5.1. A solução do sistema de equações global é obtida pela combinação das soluções dos subproblemas locais.

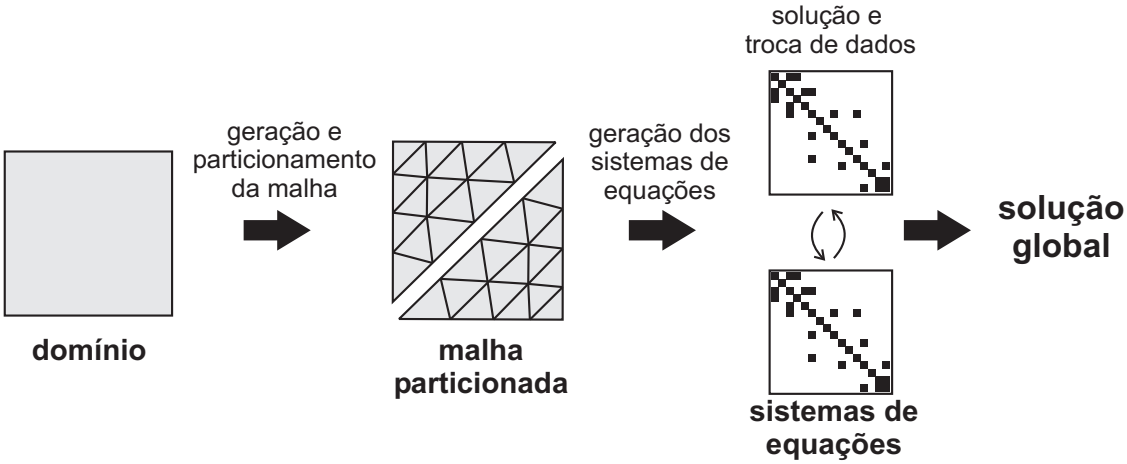


Figura 5.1: Decomposição de domínios

Abordagens paralelas via decomposição de domínio baseiam-se no fato de que cada processador pode fazer grande parte do trabalho de forma independente (SAAD, 1996). E, uma vez que os subdomínios podem ser tratados independentemente, tais métodos são atrativos para ambientes de memória distribuída.

De fato, alguns dos principais atrativos para o uso de MDDs são: a necessidade de pouca comunicação, a qual, em geral, fica restrita às fronteiras dos subdomínios; a versatilidade para trabalhar com distintos modelos matemáticos que são definidos em diferentes subregiões do domínio global; e o fato de que podem ser utilizados para a construção de pré-condicionadores para métodos iterativos (SMITH; BJORSTAD; GROPP, 1996).

Os MDDs podem ser divididos em duas classes: os métodos de Schwarz, onde os subdomínios apresentam uma região de sobreposição, que pode variar de acordo com o tipo de aproximação empregada para resolver os modelos matemáticos já discretizados, e os métodos de Schur, onde os subdomínios não apresentam região de sobreposição. Neste trabalho utiliza-se um método com sobreposição, o método aditivo de Schwarz, e um método sem sobreposição, o método do complemento de Schur. Estes métodos são abordados nas Seções 5.1 e 5.2.

5.1 Método Aditivo de Schwarz

Os MDDs de Schwarz caracterizam-se pela decomposição do domínio global Ω em n subdomínios sobrepostos Ω_i, tal que $\Omega = \bigcup_{i=1}^{n} \Omega_i$, com $\Omega_i \cup \Omega_j \neq \emptyset$, para $i \neq j$. As fronteiras artificiais são denotadas por Γ_i, e $\partial\Omega$ denota as fronteiras reais de Ω. A fronteira artificial Γ_i é parte de Ω_i que é o interior do domínio Ω, e $\partial\Omega \backslash \Gamma_i$ são os pontos de $\partial\Omega$ que não estão em Γ. Uma ilustração é mostrada na Figura 5.2.

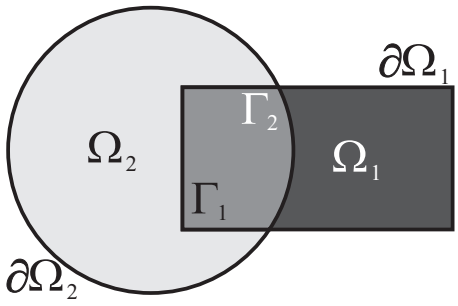


Figura 5.2: Domínio formado pela união de um disco e um retângulo com áreas sobrepostas

O primeiro MDD utilizado como método de solução foi proposto pelo matemático alemão Hermann Amandus Schwarz em 1869. No método desenvolvido por Schwarz obtém-se a solução do problema global de modo alternado em cada subdomínio, sendo que os valores calculados em um subdomínio, em uma determinada iteração, são utilizados como condição de contorno para o outro subdomínio na iteração seguinte. Este algoritmo é conhecido na literatura como MDD Alternante de Schwarz (FLEMISH, 2001).

Em 1936 Sobolev propõe uma formulação matemática abstrata para o método original de Schwarz colocando-o em rigorosas bases matemáticas. Com essa nova formulação matemática, o método original de Schwarz, passou a ser conhecido na literatura técnica como MDD multiplicativo de Schwarz. Posteriormente, Dryja e

Widlund ao analisarem as características matemáticas do MDD Multiplicativo de Schwarz, desenvolveram um novo MDD, o aditivo de Schwarz (DRYJA; WIDLUND, 1987). Neste trabalho emprega-se o MDD aditivo de Schwarz por ser a abordagem com maior potencial de paralelismo (MARTINOTTO, 2004).

De fato, o MDD aditivo de Schwarz (MAS) utiliza condições de contorno do tipo Dirichlet (RIZZI, 2002). Assim, um subdomínio obtém as condições de contorno através do conhecimento dos valores das células adjacentes aos subdomínios vizinhos na iteração anterior. Assim, os subdomínios, durante uma iteração, podem ser resolvidos independentemente. Uma versão para operador diferencial parcial, para o MAS, pode ser escrita como:

$$\begin{cases} L_i u_i^k = f_i, u \in \Omega_i \\ u_i^k = g, u \in \partial\Omega_i \backslash \Gamma \\ u_i^k = g^{k-1}, u \in \Gamma \end{cases} \tag{5.1}$$

onde $L_i u_i^k = f_i, u \in \Omega_i$ representa a solução no interior de Ω; $u_i^k = g, u \in \partial\Omega_i \backslash \Gamma$ representa a solução na fronteira real de Ω; e $u_i^k = g^{k-1}$ a solução na fronteira artificial Γ de Ω.

Note-se que para resolver (5.1) no tempo k é necessário o conhecimento dos valores das células de contorno no nível de tempo anterior $(k-1)$, como pode ser visto em $u_i^k = g^{k-1}$. Assim, para resolver o problema em paralelo em arquiteturas de memória distribuída deve-se trocar informações, chamadas de condições de contorno (CC), entre os subdomínios, até que a convergência seja alcançada.

No MAS, todos os subdomínios usam a solução da última iteração em cada subdomínio como CC para os subdomínios adjacentes, de modo que cada um deles pode ser resolvido independentemente, ficando as comunicações restritas às fronteiras e às sobreposições. Além disso, supondo que $\Omega_i \cap \Omega_j \cap \Omega_k \neq \oslash, \forall i \neq j \neq k$, pode-se mostrar que o algoritmo converge, e a presença de regiões sobrepostas assegura a continuidade da solução e de suas derivadas (DEBREU; BLAYO, 1998).

O uso de células de sobreposição pelo MAS, requer a especificação de CC de Dirichlet. Neste caso o vetor dos termos independentes do sistema $Au = b$ é escrito como $b^k = b^0 + \beta * u^{k-1}$, onde β é o coeficiente associado ao valor nodal da célula do domínio adjacente, calculado na iteração anterior, e u^{k-1} é a incógnita calculado neste passo de tempo.

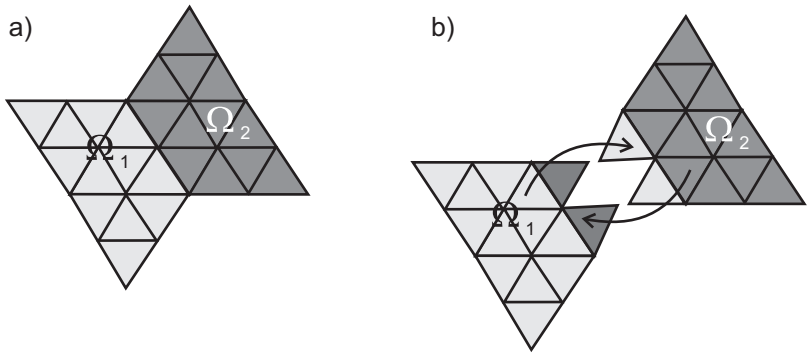


Figura 5.3: Dominio sem sobreposição (a) domínio com sobreposição (b). Detalhe de troca de dados entre dois subdomínios em (b)

Na Figura 5.3 apresenta-se um exemplo de um domínio partcicionados sem sobreposição (a) e um em um segundo momento, apresentando as células de sobreposição (b). Pode-se observar também nesta figura a troca de dados entre subdomínios com uma célula de sobreposição. A troca de dados é feita enviando informações das células do domínio Ω_1 para as células da área de sobreposição de Ω_2 correspondente e vice-versa.

E importante ressaltar que a taxa de convergência dos métodos de Schwarz é sensível ao número de células na região de sobreposição. Com o aumento da área de sobreposição, maior será a velocidade de convergência. Em compensação maior será o tamanho dos subdomínios Ω_i e, conseqüentemente, o custo computacional para o cálculo das soluções locais.

5.1.1 Convergência do Método Aditivo de Schwarz

Nesta seção apresenta-se um exemplo de resolução de um sistema de equações através do método aditivo de Schwarz, utilizando dois processos. O domínio utilizado é apresentado na Figura 5.4. Para a construção da matriz dos coeficientes utilizou-se a equação da difusão do calor, apresentada no Capítulo 6.

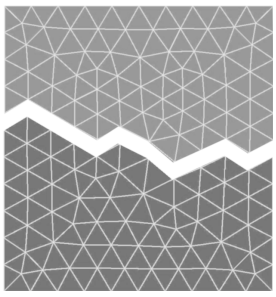


Figura 5.4: Domínio computacional, formado por 236 elementos triangulares

Na Figura 5.5 pode-se observar a convergência da solução no MAS. Na primeira iteração a solução é feita sem nenhuma troca de dados anterior. Logo, as soluções encontradas são as soluções locais sem a contribuição dos vizinhos. Já na segunda iteração a solução é obtida após a troca das contribuições dos subdomínios vizinhos.

Note que a solução, ao fim da segunda iteração, vai se tornando mais homogênea. Na terceira iteração, após a segunda troca de dados, a solução converge para uma solução contínua entre os subdomínios.

5.2 Método do Complemento de Schur

O método do complemento de Schur (MCS) foi desenvolvido na década de 70. Neste MDD o domínio Ω é particionado em subdomínios sem sobreposição, da forma que:

$$\Omega = \bigcup_{i=1,s} \Omega_i \text{ tal que } \Omega_i \cap \Omega_j = \emptyset.$$

A continuidade da solução entre os subdomínios é garantida através da solução de um sistema de interface (sistema correspondente às células pertencentes às fronteiras artificiais criadas pelo particionamento). O sistema de interface é conhecido na literatura como complemento de Schur (SMITH; BJORSTAD; GROPP, 1996).

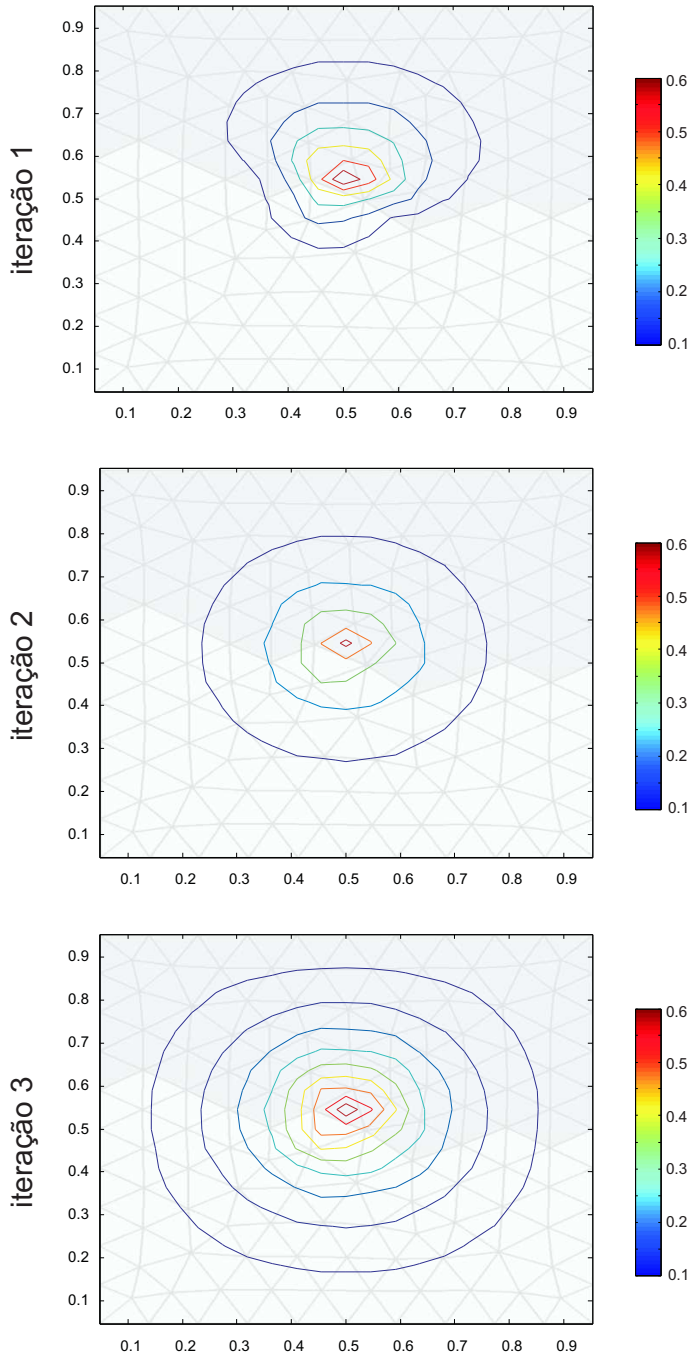


Figura 5.5: Exemplo de convergência do método aditivo de Schwarz

Existem diversas variantes do MCS, diferenciadas pela forma de particionamento e de numeração das células utilizada. Neste trabalho utiliza-se a abordagem descrita em Saad (1996). Nesta abordagem, em cada subdomínio Ω_i, as células locais são ordenadas de forma que as células pertencentes à interface (fronteira com subdomínio vizinho) são listadas depois das células internas do subdomínio, como mostra a Figura 5.6. Tal ordenação apresenta algumas vantagens, incluindo uma comunicação entre processos mais eficiente (SAAD; SOSONKINA, 2000).

Adotando essa ordenação, o vetor de incógnitas x_i pode ser particionado em duas partes:

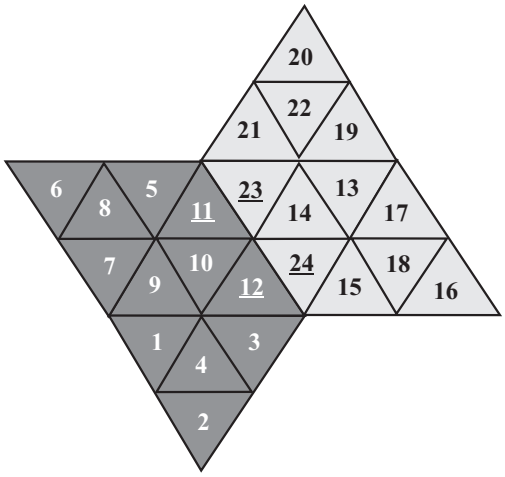


Figura 5.6: Esquema de numeração das células no método do complemento de Schur

$$x_i = \begin{bmatrix} u_i \\ y_i \end{bmatrix}$$

onde o subvetor u_i representa as células internas do subdomínio e o subvetor y_i representa as células da interface do subdomínio Ω_i. Da mesma forma, o vetor dos termos independentes b_i pode ser particionado em:

$$b_i = \begin{bmatrix} f_i \\ g_i \end{bmatrix}$$

onde f_i representa as células internas e g_i representa as células da interface.

Cada matriz é chamada de matriz local. Esta matriz local é formada por quatro submatrizes:

$$A_i = \begin{bmatrix} B_i & E_i \\ F_i & C_i \end{bmatrix}$$

onde B_i é uma matriz associada às células internas do subdomínio Ω_i, as matrizes E_i e F_i representam as interações entre as células internas e as células de interface do subdomínio Ω_i, e a matriz C_i representa as células pertencentes à interface do subdomínio (SAAD; SOSONKINA, 2000). Tem-se também uma estrutura adicional E_{ij} que representa a interação entre a interface local e a interface dos subdomínios vizinhos Ω_j para o sistema local do subdomínio Ω_i. A Figura 5.7 mostra um exemplo de matriz.

Com isso, as equações locais podem ser escritas, para $1, ..., n$ processos, como:

$$\begin{array}{rcrcrcl} B_i u_i & + & E_i y_i & & & = & f_i \\ F_i u_i & + & C_i y_i & + & \sum_{j \in N_i} E_{ij} y_j & = & g_i \end{array}$$

O termo $E_{ij} y_i$ é a contribuição dos subdomínios vizinhos Ω_j para o sistema local do subdomínio Ω_i e N_i é o conjunto de subdomínios vizinhos ao subdomínio Ω_i. Destaca-se que é a soma dessas contribuições que garante a continuidade da solução entre os subdomínios (MARTINOTTO, 2004).

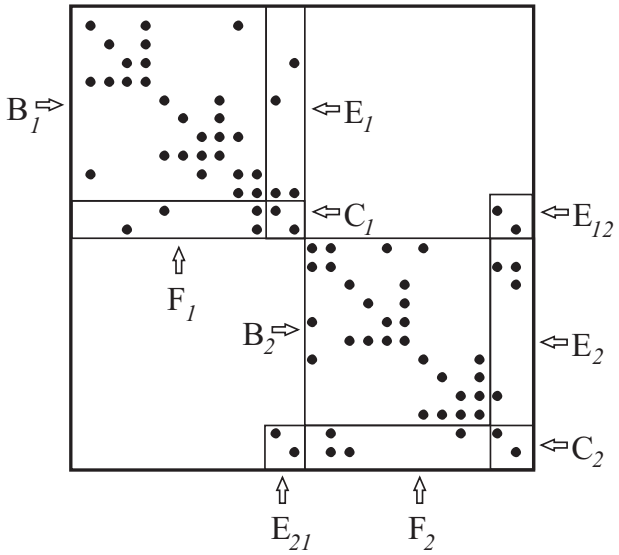


Figura 5.7: Matriz formada a partir da Figura 5.6

Isolando u_i na primeira equação temos:

$$u_i = B_i^{-1}(f_i - E_i y_i) \tag{5.2}$$

Substituindo u_i na segunda equação, obtém-se:

$$S_i y_i + \sum_{j \in N_i} E_{ij} y_j = g_i - F_i B_i^{-1} f_i \tag{5.3}$$

onde $S_i = C_i - F_i B_i^{-1} E_i$. Esta matriz é conhecida na literatura como complemento de Schur.

Resolvendo (5.3) encontra-se a solução para o vetor y_i, que está relacionada às incógnitas das regiões de interface. No entanto, os valores de y_j são desconhecidos. Assim, deve-se resolver

$$S_i y_i = g_i - F_i B_i^{-1} f_i, \tag{5.4}$$

na qual uma aproximação grosseira para y_i é obtida, já que não considera a colaboração das contribuições dos subdomínios vizinhos. Na seqüência, utiliza-se esta aproximação em (5.3), de modo a obter uma solução mais acurada para y_i. Então cada processo utiliza os valores encontrados para y_i para encontrar, de modo independente, os valores de u_i em (5.2), que são os valores das incógnitas referentes à parte interna dos subdomínios.

5.2.1 Matrizes Inversas

Na formulação do método do complemento de Schur, existe a necessidade do uso de matrizes inversas. Este é o principal obstáculo do método, pois os métodos convencionais de inversão destróem a esparsidade das matrizes e além disso, possuem um alto custo computacional. Em Charão (2001), Galante (2003) e Martinotto (2004), os autores optaram por construir aproximações para as matrizes inversas através de métodos polinomiais. Essas aproximações baseiam-se na série de Neumann para obter uma aproximação M^{-1} para B^{-1} (MARTINOTTO, 2004). Se a série é truncada com $k = 0$, obtém-se que:

$$M^{-1} = D^{-1}.$$

Truncando-se a série com $k = 1$, tem-se:

$$M^{-1} = D^{-1} - D^{-1}(L + L^T)D^{-1}$$

onde, D^{-1} é a inversa da diagonal principal, L é a parte inferior à diagonal principal e L^T é denota a transposta de L.

Estas aproximações têm a vantagem de terem pouco custo computacional, além de manterem a estrutura da matriz, evitando preenchimento (*fill-in*), no entanto, a aproximação M^{-1} por elas geradas apresentam baixa qualidade numérica.

Neste trabalho optou-se em utilizar uma abordagem em que são feitas manipulações algébricas para resolver o problema das inversas. Dado o sistema:

$$\underbrace{(C_i - F_i B_i^{-1} E_i)}_{A} \underbrace{y_i}_{x} = \underbrace{g_i - F_i B_i^{-1} f_i}_{b} \tag{5.5}$$

pode-se resolve-lo através de um método iterativo, como o GC ou GMRES, fazendo-se algumas modificações nas operações de multiplicação de matriz-vetor existentes nestes métodos. Para exemplificar, é mostrado o cálculo do resíduo (r), operação comum nos métodos iterativos, que é dada por:

$$r = b - Ax.$$

Inicialmente, encontra-se o valor para b, representado em (5.5) por $g_i - F_i B_i^{-1} f_i$:

$$g_i - F_i B_i^{-1} f_i = b$$

$$F_i B_i^{-1} f_i = \underbrace{b + g_i}_{\alpha}$$

$$B_i^{-1} f_i = \underbrace{F_i^{-1}\alpha}_{\gamma}$$

Resolvendo o sistema de equações resultante:

$$B_i \gamma = f$$

encontra-se o valor para γ, e por retrosubstituição, determina-se o valor para α, e por conseqüência determina-se b.

O segundo passo é calcular o produto Ax. Dado que $A = C_i - F_i B_i^{-1} E_i$, temos:

$$Ax = (C_i - F_i B_i^{-1} E_i)x$$

$$Ax = \underbrace{(C_i - F_i B_i^{-1} E_i)x}_{\beta}$$

$$\beta = \underbrace{C_i x}_{\rho} - \underbrace{F_i B_i^{-1} E_i x}_{\phi}$$

$$\beta = \rho - \phi \tag{5.6}$$

O valor de ρ é facilmente encontrado através de um produto matriz-vetor. Já o valor de ϕ é obtido por:

$$F_i B_i^{-1} E_i x = \phi$$

$$B_i^{-1} \underbrace{E_i x}_{w} = \underbrace{F_i^{-1} \phi}_{\varphi}$$

Resolvendo o sistema de equações resultante:

$$B\varphi = w$$

encontra-se o valor para φ, e por retrosubstituição, determina-se o valor para ϕ. Assim, obtém-se um vetor β equivalente ao produto matriz-vetor Ax. Desse modo o resíduo $r = b - Ax$ é obtido por uma operação de subtração de vetores $r = b - \beta$.

Com a utilização desta abordagem algébrica, a qualidade numérica da solução depende apenas da precisão do método escolhido para a resolução dos sistemas de equações.

5.3 Considerações Finais

Este capítulo apresentou uma visão geral dos métodos de decomposição de domínio. A ênfase foi dada aos métodos utilizados neste trabalho: o método aditivo de Schwarz e o método do complemento de Schur. Estes métodos são utilizados em conjunto com multigrid para a obtenção de um método de resolução paralela, descrita no Capítulo 6. Essa combinação, de métodos multigrid e métodos de decomposição de domínio é chamada MG+MDD (DOUGLAS, 1996a).

Uma visão completa das áreas de utilização de métodos de decomposição de domínio, bem como os últimos avanços obtidos nestes métodos, podem ser obtidos no fórum *International Conferences on Domain Decomposition Methods* (`http://www.ddm.org`), que realiza encontros anuais desde 1987 sobre métodos de decomposição de domínio.

6 GERAÇÃO DE HIERARQUIA DE MALHAS E DE SISTEMAS DE EQUAÇÕES

Este capítulo aborda as questões relacionadas aos métodos de solução implementados neste trabalho. Como se pode observar no diagrama da Figura 6.1, este trabalho está dividido em quatro atividades, que incluem:

1. geração e particionamento de malhas;
2. criação da hierarquia de malhas;
3. montagem dos sistemas de equações lineares;
4. resolução dos sistemas em paralelo.

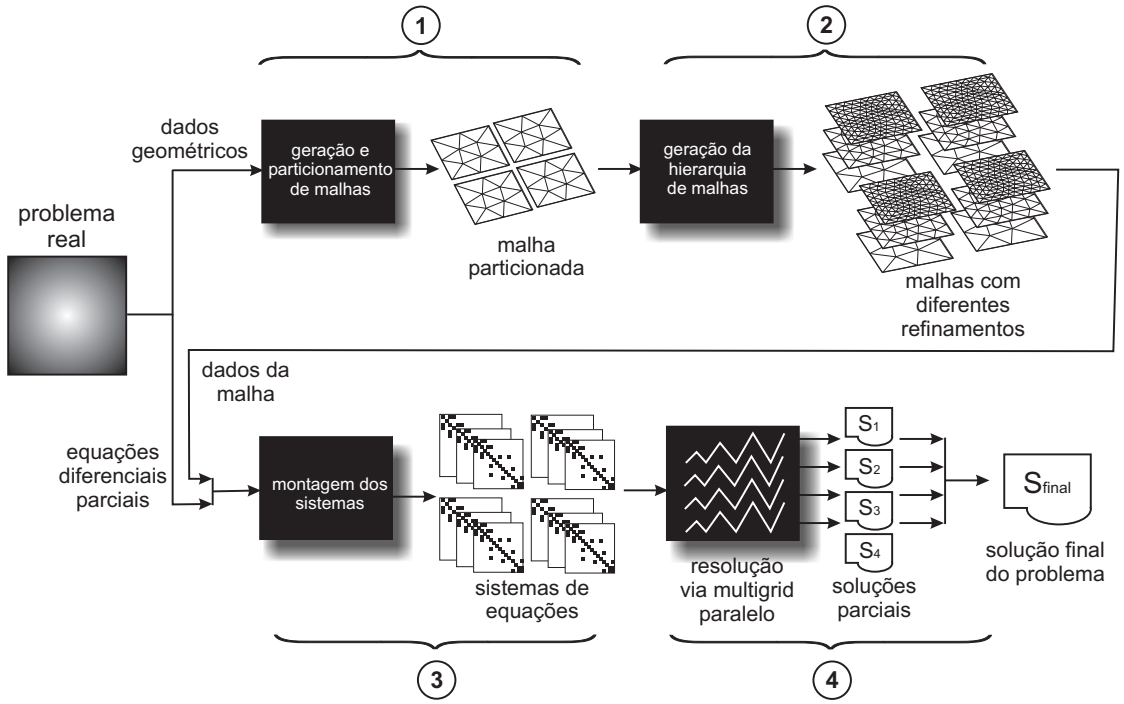


Figura 6.1: Passos para a solução do problema

A primeira atividade compreende todo o processo de geração e particionamento das malhas. Uma vez geradas, essas malhas servem de entrada de dados para o módulo que gera a hierarquia de malhas necessárias para os métodos multigrid, conforme mostrado na Seção 4.2. Na seqüência, discretiza-se a EDP do problema a

ser tratado utilizando as malhas previamente geradas, obtendo como resultado os sistemas de equações, que posteriormente são resolvidos através de métodos paralelizados. As três primeiras atividades são descritas detalhadamente ao longo deste capítulo. Já a paralelização dos métodos multigrid é apresentada no Capítulo 7.

6.1 Geração e Particionamento de Malhas

A geração e particionamento das malhas é a atividade que serve como base ao restante do trabalho. É nesta fase que as malhas são geradas e posteriormente particionadas por um software específico.

Como já apresentado na Seção 2.2, neste trabalho são utilizados dois pacotes para a geração de malhas não estruturadas 2D, o *Triangle* e o *Easymesh*.

A entrada de dados para estes dois pacotes são semelhantes. Como dado inicial para a geração das malhas têm-se um arquivo de texto contendo um gráfico de linhas retas planares (PSLG[1],*Planar Straight Line Graph*), que descreve o contorno do domínio a ser coberto pela malha, através de segmentos de reta.

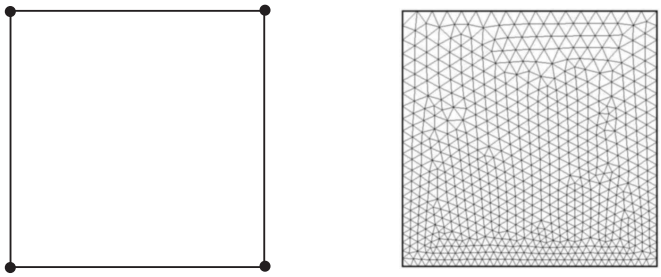


Figura 6.2: Exemplo de PSLG e respectiva malha

Como saída dos programas obtém-se diversos arquivos contendo dados sobre a malha gerada. Estes dados incluem:

- coordenadas dos vértices da malha;
- conectividade dos triângulos (quais nodos formam um determinado triângulo);
- vizinhança de cada triângulo.

No Anexo C apresenta-se o formato dos arquivos de entrada e saída utilizados na geração de malhas.

Como citado na Seção 2.3, para o particionamento das malhas emprega-se o pacote METIS. Como entrada de dados utiliza-se as informações sobre a conectividade dos triângulos geradas pelos softwares de geração de malhas, como pode-se observar na Figura 6.3.

O pacote METIS fornece dois diferentes programas para o particionamento de malhas, o *partdmesh* e o *partdmesh*.

A diferença entre estes dois programas é que o *partnmesh* converte a malha em um gráfico nodal (isto é, cada vértice da malha se transforma em um vértice do grafo), já o *partdmesh* converte a malha em um grafo dual (isto é, cada elemento transforma-se um vértice do gráfico).

[1]Grafo formado a partir de um conjunto de vértices e segmentos que não se interceptam, exceto nas suas extremidades. Este grafo define uma região planar, e esta pode ou não conter buracos.

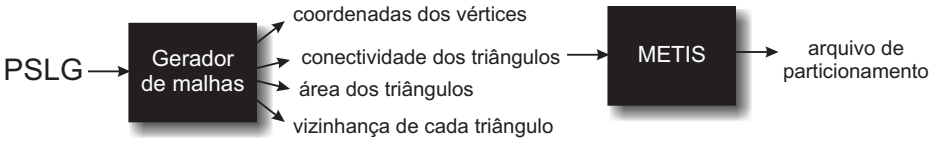


Figura 6.3: Relacionamento entre gerador de malha e METIS

Para ambos os programas, os dados de saída resumem-se a uma lista das respectivas partições de cada triângulo. Neste trabalho, utiliza-se o *partdmesh*, já que na solução dos estudos de caso as incógnitas estão centralizadas no interior do triângulo, portanto o particionamento deve ser feito baseado nos elementos da malha, e não nos vértices.

É importante ressaltar que nesta fase apenas a malha mais grosseira é gerada. Assim, hierarquia de malhas é gerada tendo como ponto de partida esta malha grosseira.

6.2 Criação da Hierarquia de Malhas

A criação da hierarquia de malhas é um passo muito importante na solução de problemas utilizando métodos multigrid. Existem algum pacotes disponíveis que geram de modo automático os diversos níveis de malha, como é o caso do MGridGen/ParMGridGen. Estes pacotes partem de uma malha mais refinada e geram as demais malhas grosseiras através de algoritmos multível (MOULITSAS; KARYPIS, 2001).

Embora esses pacotes gerem de forma eficiente a hierarquia de malhas necessárias optou-se por implementar um gerador próprio para as malhas. Dessa maneira implementou-se um módulo, chamado *MGTool*, capaz de gerar malhas com diferentes refinamentos. Essa escolha de projeto deve-se a dois principais motivos. O primeiro refere-se ao tipo especial de malha utilizado neste trabalho, as malhas não estruturadas ortogonais, não geradas pelos pacotes MGridGen/ParMGridGen. O segundo motivo deve-se as otimizações feitas nas malhas geradas pelo MGTool, que faz com que os mecanismos de transferência entre malhas não necessitem de comunicação entre processos, já que o particionamento da malha é mantido para todos os níveis de malha.

O *MGTool* tem como entrada de dados, os arquivos de saída dos geradores de malha e o arquivo de saída do METIS; e como saída produz os diversos níveis de malha desejado, como mostrado na Figura 6.4.

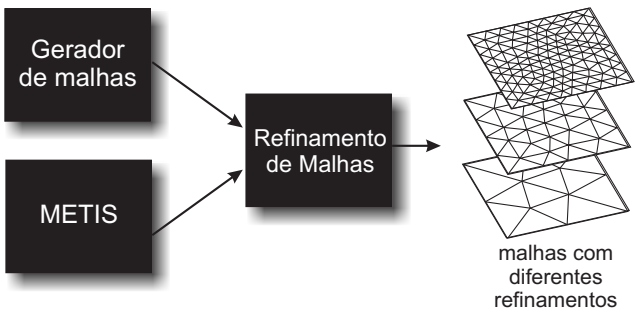


Figura 6.4: Entradas e saídas para o módulo de refinamento de malhas

Para o refinamento das malhas adotou-se o uma estratégia conhecida na literatura como *h-refinement*, caracterizada pela subdivisão dos elementos do domínio. No módulo implementado, dada uma malha inicial, gera-se as demais malhas através da subdivisão sucessiva dos elementos em quatro subelementos, como observa-se na Figura 6.5.

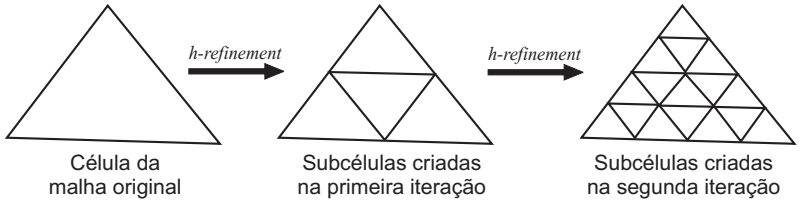
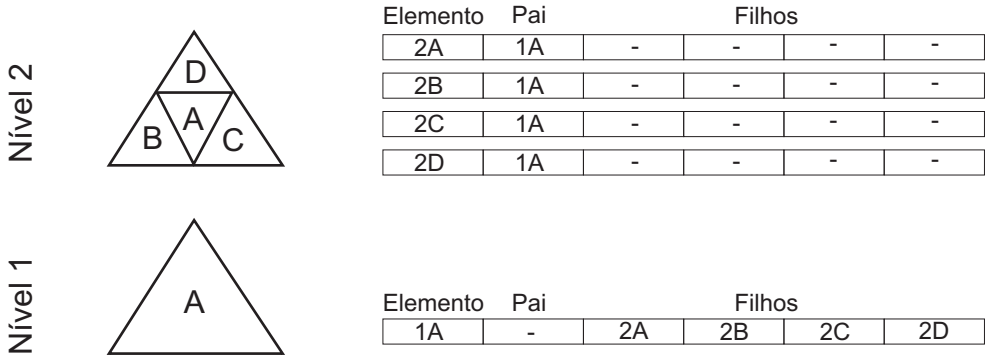


Figura 6.5: Esquema de refinamento de malha

Após a geração da hierarquia das malhas, deve-se relacionar cada nível com os níveis adjacentes. Nessa fase cria-se para cada elemento da malha uma relação dos triângulos para os quais deve receber informações, e para os quais deve enviar. Um exemplo é mostrado na Figura 6.6, onde o triângulo 1A é dividido em outros 4 triângulos, 2A, 2B, 2C e 2D. Na notação utilizada neste trabalho, o triângulo 1A é o "pai" dos triângulos 2A, 2B, 2C e 2D, e por sua vez estes triângulos são chamados de "filhos" do triângulo 1A.



Elemento	Pai	Filhos			
2A	1A	-	-	-	-
2B	1A	-	-	-	-
2C	1A	-	-	-	-
2D	1A	-	-	-	-

Elemento	Pai	Filhos			
1A	-	2A	2B	2C	2D

Figura 6.6: Exemplo de hierarquia de dois níveis de malha. Os números representam o nível e as letras identificam o triângulo na malha. No lado direito da figura, a tabela descreve o relacionamento entre os níveis adjacentes de malha.

Esta relação "pai-filho" é utilizada pelos procedimentos de interpolação e restrição presentes nos métodos multigrid, conforme descrito na Seção 3.3.1.

Dependendo do tipo de MDD a ser empregado, existe um tratamento diferente para a malha. Se o MDD empregado for o método aditivo de Schwarz, o módulo ainda é responsável pela criação das áreas de sobreposição requeridas por este método. Caso o MDD escolhido seja o método do complemento de Schur, o módulo deve renumerar os elementos de acordo com o especificado para o método, conforme já abordado na Seção 5.2.

6.3 Montagem dos Sistemas de Equações Lineares

O *MGTool*, além de montar a hierarquia de malhas e as relações entre níveis adjacentes, é também responsável pela montagem das matrizes para cada nível de

malha.

A geração das matrizes é feita de forma distribuída, onde cada processo gera as matrizes de acordo com o subdomínio que lhe foi atribuído. O processo de geração difere dependendo do MDD a ser utilizado. No caso do método aditivo de Schwarz ser utilizado, gera-se a matriz estendida, ou seja, gera-se a matriz também considerando as células de sobreposição. Já no caso do método do complemento de Schur, gera-se as submatrizes B_i, C_i, E_i, F_i e E_{ij}, conforme descrito na Seção 5.2.

Em particular, os sistemas são gerados a partir da discretização de duas EDPs: a EDP da difusão de calor e a EDP do cálculo do nível da superfície livre da hidrodinâmica do modelo UnHIDRA.

A difusão de calor é a movimentação de energia que ocorre devido à diferença entre temperaturas. O calor sempre flui das regiões de maior temperatura para regiões de temperatura inferior (PAAR; ATHANAS; EDWARDS, 1996). Assim, este estudo de caso consiste em determinar como o calor se propaga em um corpo, dado uma condição inicial e as condições de contorno.

No estudo de caso do modelo de hidrodinâmica, utiliza-se uma simplificação do modelo de hidrodinâmica do UnHIDRA, onde objetiva-se calcular a variação do nível da água. O modelo matemático completo empregado é chamado de equações *shallow water* (ESW) (RIZZI, 2002).

De acordo com a discretização feita para a EDP, gera-se um estêncil computacional. Este estêncil indica a posição dos pontos presentes em uma EDP. Nas discretizações utilizadas neste trabalho, gerou-se um estêncil de quatro pontos (ou 4-pontos). Uma representação é mostrada na Figura 6.7, onde pode-se observar relação de dependência existente entre um triângulo i e seus vizinhos. Note que o triângulo i depende dos triângulos i_1, i_2 e i_3.

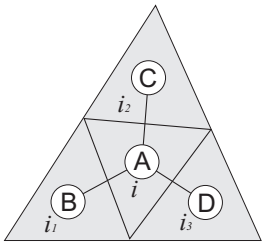


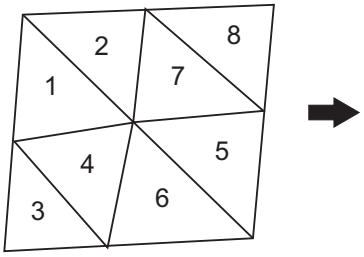
Figura 6.7: Molécula computacional

Na Figura 6.8, é mostrado um exemplo de montagem de uma matriz. O estêncil é aplicado a cada elemento da malha, resultando em uma linha da matriz. Com este estêncil, as matrizes geradas têm no máximo quatro elementos por linha, e apesar de possuir uma disposição simétrica das posições da matriz, o mesmo não ocorre para os valores.

Mais detalhes sobre a discretização da difusão de calor bidimensional são apresentados no Anexo A.

6.3.1 Esquemas de Armazenamento de Matrizes

Os sistemas gerados a partir da discretização de EDPs geralmente apresentam um número suficientemente elevado de elementos nulos. Esse fato torna compensador a procura por esquemas de armazenagem que permitam tirar partido da existência dos seus poucos elementos não nulos (JUDICE; PATRICIO, 1996).



	1	2	3	4	5	6	7	8
1	A	B		C				
2	B	A					C	
3			A	B				
4	B		C	A		D		
5					A	B	C	
6				B	C	A		
7		B			C		A	D
8							B	A

Figura 6.8: Exemplo de matriz formada a partir da molécula computacional. A geometria do domínio e a vizinhança de cada triângulo determina a localidade dos elementos não-nulos da matriz

Existem diversos esquemas para armazenamento de matrizes. Para um estudo completo veja Saad (1996), Eijkhout (1992) e Silva (2005). Para o desenvolvimento deste trabalho utilizou-se o formato *compressed sparse row* (CSR).

O formato CSR armazena apenas os elementos não nulos de uma matriz esparsa e sua estrutura é baseada em três vetores, como pode ser visto no exemplo ilustrado pela Figura 6.9, sendo um vetor são do tipo de dados da matriz, e dois vetores são de inteiros (SAAD, 1996), de modo que:

1. *elems*: armazena os valores não nulos da matriz. O tamanho do vetor *val* é dado pelo número de elementos não nulos da matriz;

2. *cols*: armazena a coluna da qual os valores contidos em *elems* foram obtidos na matriz. O tamanho de *cols* é dado pelo número de elementos não nulos da matriz;

3. *ptrs*: armazena os ponteiros que indicam quantos valores não nulos cada linha possui. A primeira posição desse vetor recebe "0" (zero), a segunda posição recebe o valor da posição anterior somado com o número de elementos não nulos da primeira linha do vetor e assim sucessivamente. O tamanho do vetor *ptrs* é dado por $N + 1$, onde N é o número de linhas da matriz.

$$\begin{pmatrix} e_1 & e_2 & 0 & 0 & 0 \\ 0 & e_3 & e_4 & 0 & 0 \\ 0 & e_5 & e_6 & 0 & 0 \\ 0 & 0 & e_7 & e_8 & e_9 \\ 0 & 0 & 0 & e_{10} & e_{11} \end{pmatrix}$$

$$elems = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}\}$$

$$cols = \{0, 1, 1, 2, 1, 2, 2, 3, 4, 3, 4\}$$

$$ptrs = \{0, 2, 4, 6, 9, 11\}$$

Figura 6.9: Exemplo de matriz armazenada em formato CSR

O formato CSR não considera qualquer informação sobre a estrutura da matriz, armazenando os elementos não nulos em posições contíguas na memória. Com a

utilização deste formato de armazenamento ao invés de se armazenar N^2 elementos, são necessários apenas $n + N + 1$ posições (onde N é a dimensão da matriz e n é o número de elementos não nulos).

Pode-se alterar o CSR colocando a diagonal principal em um vetor separado. Essa alteração diminui o total de armazenamentos no vetor de índices e também o total de acesso a este, o que pode resultar em um ganho de desempenho (PICININ, 2001).

6.4 Considerações Finais

Neste capítulo, mostrou-se o desenvolvimento de algumas etapas necessárias para o emprego de métodos multigrid na resolução de sistemas de equações lineares.

Inicialmente abordou-se as questões de geração e particionamento das malhas empregadas neste trabalho. Os pacotes utilizados na geração da malhas foram o *EasyMesh* e o *Triangle*, já para o particionamento utilizou-se o METIS. No Anexo C são mostrados exemplos de arquivos de entrada e saída empregados na geração das malhas.

Em um segundo momento apresentou-se a criação da hierarquia das malhas necessárias para os métodos multigrid. A geração dos múltiplos níveis de malha foi feita utilizando o algoritmo h-refinement, que se baseia na subdivisão dos elementos da malha para a criação de novos elementos.

Na seqüência, apresentou-se a montagem dos sistemas de equações baseada nos estudos de caso tratados neste trabalho. A montagem dos sistemas se dá através da aplicação de um estêncil computacional (relacionado à discretização de uma particular EDP) aos elementos da malha. Dois estudos de caso são utilizados: transferência de calor e uma simplificação do modelo de hidrodinâmica do UnHIDRA. Um maior detalhamento destes modelos são descritos no Anexo A.

7 MULTIGRID PARALELO

Os sistemas de equações lineares estão entre os mais freqüentes problemas que devem ser tratados pela computação científica. Por se tratarem geralmente de sistemas de grande porte, uma alternativa viável de se obter a solução em tempo útil é empregar métodos de resolução paralelizados, empregando *clusters* como ambiente computacional.

Nesta seção apresenta-se o método proposto para a resolução de sistemas de equações em paralelo. Uma visão geral do processo de resolução é mostrado na Figura 7.1.

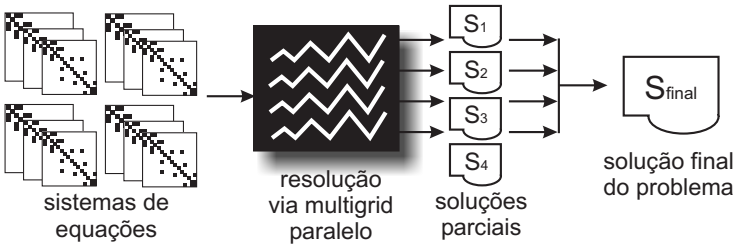


Figura 7.1: Visão geral da resolução dos sistemas de equações através do multigrid paralelo

O primeiro passo da resolução consiste na leitura dos arquivos de entrada contendo os sistemas de equações. Cada arquivo contém informações que pertencem à um subdomínio específico, e por conseqüência, a um processo.

Após a leitura de seus respectivos arquivos de entrada, cada processo fica responsável por calcular independentemente a solução do sistema de equações referente ao seu conjunto de dados. Nesse trabalho, utilizou-se métodos multigrid paralelizados através da abordagem de decomposição de domínio.

Ao término da resolução, cada processo cria um arquivo de saída contendo a sua parte da solução. Estas diversas soluções obtidas podem então ser reunidas para formar a solução global do sistema.

Conforme apresentado na Seção 3.3, os métodos multigrid são baseados em três etapas centrais: transferência de informações entre malhas, iterações aninhadas e correção em malha grossa. Para a construção destas etapas, considera-se basicamente três elementos básicos:

1. transferência de informações (restrição e interpolação);

2. cálculo do resíduo da solução;

3. resolução de sistemas de equações.

Dessa forma, a obtenção do multigrid paralelo se dá através da execução de cada uma destas rotinas paralelizadas. As implementaçõs foram feitas em linguagem C, utilizando a biblioteca de trocas de mensagens MPICH 1.2.7. A paralelização dos elementos apresentada ao longo do restante do capítulo.

7.1 Restrição e Interpolação Paralelas

Os elementos principais discutidos anteriormente revelam a necessidade de operadores para transferir informações entre as malhas. Inicialmente assume-se que a malha grossa possui um quarto do número de incógnitas da malha fina ou ainda que o tamanho dos elementos grossos é quatro vezes maior daqueles da malha fina.

Utilizando a tabela de relacionamentos entre níveis, como a apresentada na Seção 6.2, pode-se descrever os operadores de restrição e interpolação.

O operador de restrição $I_{M_{fina}}^{M_{grossa}}$ utilizado neste trabalho é baseado no operador *full-weighted* descrito em Trottenberg *et al.* (2001) e Wesseling (1992). O operador de restrição é dado por:

$$x_i = \frac{1}{4}x_{filho_1(i)} + \frac{1}{4}x_{filho_2(i)} + \frac{1}{4}x_{filho_3(i)} + \frac{1}{4}x_{filho_4(i)}$$

e ilustrado na Figura 7.2.

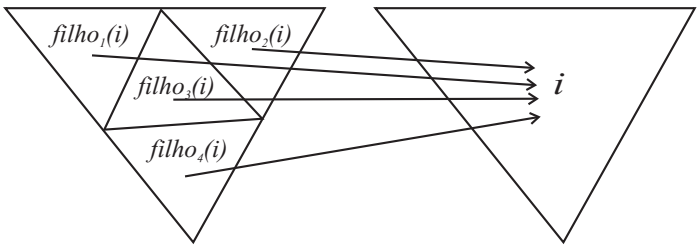


Figura 7.2: Operador de restrição

Antes de definir o operador de interpolação, é necessário considerar mais uma vez os elementos i_1, i_2 e i_3 como sendo os vizinhos do elemento i na malha fina. Dessa forma o operador de interpolação $I_{M_{grossa}}^{M_{fina}}$ é definido por:

$$x_i = \frac{1}{4}(x_{pai(i)} + \sum_{j=1}^{3} x_{pai(i_j)})$$

e é ilustrado na Figura 7.3

Note que da forma como o operador foi definido, o elemento x_i da malha fina recebe-se duas contribuições do elemento pai e outras duas dos elementos vizinhos do pai.

É importante salientar que em casos onde o triângulo i esteja na fronteira artificial do domínio, não se considera a contribuição recebida de outros subdomínios. Nesses casos, considera-se apenas as contribuições locais, e os pesos são alterados. Esta decisão de desconsiderar os elementos de outros subdomínios evita a necessidade de comunicação entre processos.

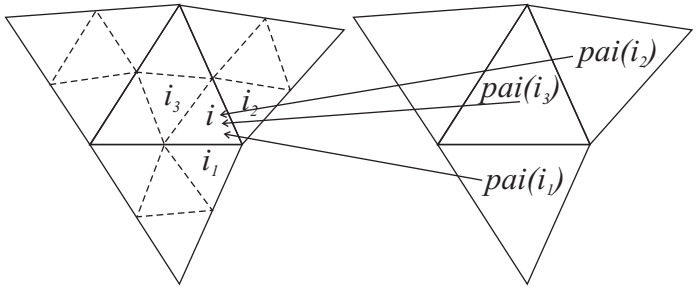


Figura 7.3: Operador de interpolação

7.2 Cálculo do Resíduo em Paralelo

O cálculo do resíduo é utilizado no multigrid no procedimento de correção em malha grossa. O cálculo do resíduo é dado por:

$$r = b - Ax'$$

onde r é o vetor de resíduo, b é o vetor dos termos independentes, A é a matriz de coeficientes e x' é uma aproximação da solução. Logo se tem duas operações a serem efetuadas, um produto matriz por vetor e uma subtração de vetores:

$$r = \underbrace{b - \underbrace{Ax'}_{matriz-vetor}}_{subvetor}$$

Portanto, a paralelização do cálculo do resíduo consiste em executar estas operações de tal maneira que cada processo trabalhe sob seu conjunto de dados.

Na operação de subtração de vetores paralela, cada processo executa a operação sobre suas partes dos vetores, como pode ser visto na Figura 7.4

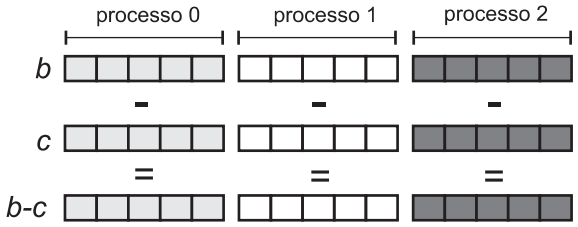


Figura 7.4: Subtração de vetores em paralelo

Já na operação de multiplicação de matriz por vetor, cada processo multiplica sua parte da matriz A pelo vetor x. No entanto, durante a multiplicação, alguns elementos podem não estar disponíveis. Isso ocorre devido ao fato destes elementos estarem armazenados nos vetores locais de outros processos. Um exemplo é mostrado na Figura 7.5, onde o processo 0 necessita dos elementos x_5 e x_8, alocados no processo 1 e o processo 1 necessita dos elementos x_1 e x_3, alocados no processo 0.

Para resolver este problema de disposição de dados, utilizou-se uma técnica semelhante à descrita em Picinin (2002) , onde a multiplicação é dividida em dois passos. O primeiro passo consiste em efetuar a operação de multiplicação matriz por vetor utilizando apenas os dados locais do processo.

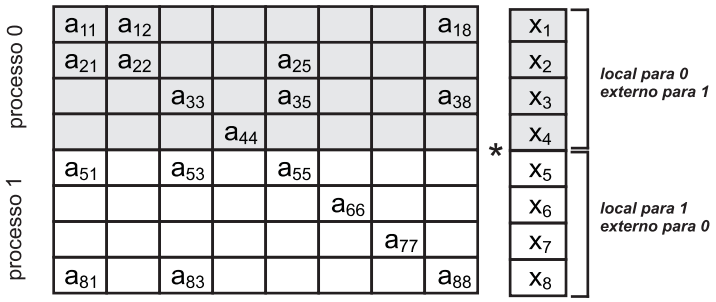


Figura 7.5: Multiplicação matriz por vetor em paralelo

O segundo passo consiste em efetuar a operação de multiplicação matriz por vetor dos elementos que possuem dependências de dados e complementar os resultados obtidos no passo anterior. Para que isso ocorra, existe a necessidade de comunicação entre os processos, para que estes troquem os dados necessários. De modo a gerenciar essas trocas, cada processo possui uma lista contendo quais elementos deve enviar e uma lista das posições em que recebe os valores enviados pelos outros processos. Essas listas são geradas durante o processo de geração das matrizes, onde é possível verificar a dependência de dados entre os processos.

Assim, o resultado local c do produto matriz por vetor pode ser escrito como:

$$c = A * x^k_{local} + A * x^k_{externo}$$

onde x^k_{local} é a parte local do vetor e $x^k_{externo}$ é a porção do vetor recebida dos outros processos, na iteração k do método iterativo.

7.3 Resolução dos Sistemas de Equações em Paralelo

Durante a resolução de sistemas através de multigrid, é necessário resolver diversos subsistemas nos diversos níveis de malha, como por exemplo nos procedimentos de iterações aninhadas e correção em malha grossa.

No procedimento de iterações aninhadas resolve-se o sistema na malha mais grossa de modo a encontrar uma melhor aproximação inicial para as malhas mais finas. Já na correção em malha grossa, deve-se resolver um sistema $Ae = r$ de modo a encontrar uma aproximação para o erro e, que é utilizado para a correção da solução, conforme apresentado na Seção 3.3.3.

Geralmente, na solução destes subsistemas em um multigrid seqüencial, utiliza-se métodos iterativos, tal como Gauss-Seidel, gradiente conjugado ou GMRES (BITTENCOURT, 1996; WESSELING, 1992).

No entanto, como deseja-se resolver os sistemas de equações de modo paralelo, emprega-se os dois métodos de decomposição de domínios apresentados no Capítulo 5. Dessa forma, pode-se optar pela resolução dos sistemas pelo método aditivo de Schwarz ou pelo método do complemento de Schur.

7.3.1 Resolução pelo Método Aditivo de Schwarz

Após o particionamento e a expansão dos subdomínios para criação da região de sobreposição necessárias para o método aditivo, cada processador é responsável pela geração dos sistemas de equações locais.

Na geração dos sistemas de equações locais utilizou-se uma numeração local, sendo que as células pertencentes à região de sobreposição são numeradas depois das células internas. Os sistemas de equações locais são armazenados utilizando o formato CSR, apresentado na Seção 6.3.1.

Além dos sistemas de equações, determina-se estruturas necessárias para a coordenação da comunicação entre os processos. Estas estruturas consistem em uma lista que armazena a identificação dos subdomínios vizinhos, e mais duas listas para cada vizinho. A primeira delas, contendo as posições do vetor solução a serem enviadas no fim de cada iteração. E a segunda, contendo as posições do vetor de termos independentes que receberão os valores calculados no subdomínio vizinho. A Figura 7.6 ilustra o domínio particionado e expandido com uma célula de sobreposição. As posições dos elementos com sublinhados duplos são aquelas que serão enviadas para o subdomínio vizinho, e as com sublinhados simples correspondem às posições que receberão os dados.

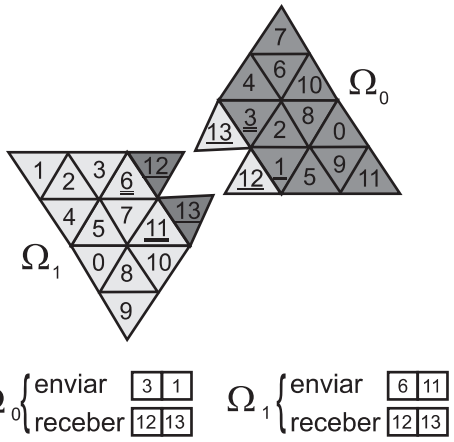


Figura 7.6: Estrutura de dados para a comunicação no aditivo de Schwarz

Com estas informações cada processo calcula a solução do sistema de equações referente ao seu subdomínio, conforme o algoritmo da Figura 7.7.

Considerando as características dos sistemas gerados nos estudos de caso, utilizou-se o GMRES, descrito na seção 3.2.1, para a solução dos subsistemas locais a cada ciclo de Schwarz.

7.3.2 Resolução pelo Método do Complemento de Schur

Para resolver um determinado problema utilizando o método do complemento de Schur, a numeração das células deve ser feita de modo que as células pertencentes às fronteiras artificiais são numeradas após as células internas. Com essa numeração, os sistemas locais gerados podem ser escritos como:

$$\begin{aligned} B_i u_i &+ E_i y_i &&= f_i \\ F_i u_i &+ C_i y_i + \sum_{j \in N_i} E_{ij} y_j &&= g_i \end{aligned}$$

No desenvolvimento deste trabalho optou-se por armazenar as submatrizes B_i, E_i, F_i, C_i e E_{ij} em estruturas CSR distintas. Da mesma forma, optou-se pelo armazenamento dos subvetores u_i, y_i, f_i, g_i separadamente. Optou-se por essa forma de armazenamento, porque ela facilita a implementação das operações de álgebra linear que compõem o método do complemento de Schur.

Inicialização: escolha uma solução inicial u_i^0 *para cada subdomínio* Ω_i;
defina o erro máximo para a solução ε;

1. *resolva o sistema com a solução inicial* u_i^0
2. $k = 1$;

do

3. *enviar dados das fronteiras da iteração* u_i^{k-1} *para os vizinhos;*
4. *receber dados das fronteiras da iteração* u_i^{k-1} *dos vizinhos;*
5. *resolver o sistema utilizando os novos valores de fronteira recebidos;*
6. *obter* u_i^k ;
7. k ++;

until $max\left(\left\| u_i^k - u_i^{k-1} \right\|_2 / \left\| u_i^k \right\|_2\right) \leq \varepsilon$

Figura 7.7: Algoritmo do método aditivo de Schwarz

O algoritmo do MCS é apresentado na Figura 7.8.

Inicialização: defina o erro máximo para a solução ε;

1. *Calcular uma aproximação inicial para* $S_i y_i = g_i - F_i B_i^{-1} f_i$

do

2. *enviar* y_i *para os vizinhos;*
3. $y_i' = y_i$;
4. *receber* y_j *dos vizinhos;*
5. *calcular* $\sum_{j \in N_i} E_{ij} y_j$
6. *calcular* $S_i y_i = g_i - F_i B_i^{-1} f_i - \sum_{j \in N_i} E_{ij} y_j$

until $\max(y_i - y_i') < \varepsilon$

7. *calcular* $B_i u_i = f_i - E_i y_i$

Figura 7.8: Algoritmo do método do complemento de Schur

Conforme mostrado na Seção 5.2, durante o cálculo dos sistemas de interface (linhas 1 a 6) é necessário o cálculo das contribuições dos subdomínios vizinhos no sisema de equações local. No cálculo das contribuições é necessária a troca de informações entre os subdomínios.

Basicamente cada subdomínio necessita apenas de uma lista de subdomínios vizinhos. Nenhuma outra informação é necessária, já que os dados a serem enviados é sempre o subvetor y_i, que está relacionada às células de fronteira.

Uma vez que a solução do sistema de interface é conhecida o sistema de equações correspondente às células internas de cada subdomínio pode ser resolvido de maneira

totalmente independentemente (linha 7). Para a resolução dos subsistemas utilizou-se o GMRES, dado que os sistemas resolvidos de são do tipo não simétricos.

7.4 Considerações Finais

Este capítulo abordou a paralelização dos métodos multigrid. Os métodos multigrid foram paralelizados através da decomposição de domínio.

Mais especificamente, os métodos multigrid paralelos são obtidos pela paralelização dos elementos que o compõe. Dessa forma, cada processador fica responsável pelos dados relacionados a um subdomínio e a solução global é dada pela combinação apropriada destas partes.

Para mais informações sobre a implementação de métodos de decomposição de domínio, recomenda-se Martinotto (2004) e Charão (2001). Para informações sobre a paralelização de métodos multigrid, recomenda-se Guerrero (2000) e Douglas (1996-b).

No próximo capítulo são mostrados os resultados obtidos com as paralelizações desenvolvidas nesse trabalho. São apresentados testes de desempenho e testes de qualidade numérica.

8 ESTUDOS DE CASO: ANÁLISE DE RESULTADOS

Neste capítulo são apresentados os resultados obtidos com as paralelizações desenvolvidas neste trabalho. Essas foram implementadas em linguagem C, utilizando o compilador gcc 2.95.4 sobre o sistema operacional Mandriva Linux e como biblioteca de troca de mensagens foi utilizado o MPICH 1.2.7.

A apresentação dos resultados está organizada da seguinte forma: inicialmente são apresentados os resultados obtidos com a resolução dos sistemas originados no problema de transferência de calor. Em seguida são apresentados os resultados obtidos com a resolução dos sistemas da hidrodinâmica do UnHIDRA. Por fim, apresenta-se uma análise da qualidade numérica dos métodos utilizados.

8.1 Transferência de Calor

Um problema clássico de aplicação de métodos numéricos é a transferência de calor em uma placa plana. O processo de transferência de calor em uma placa retangular, cujos lados estão submetidos a diferentes temperaturas T_1, T_2, T_3 e T_4, como ilustra a Figura 8.1, ocorre pela troca de calor entre partículas do material de um ponto com mais energia para outro com menos. Esse processo é conhecido como condução do calor.

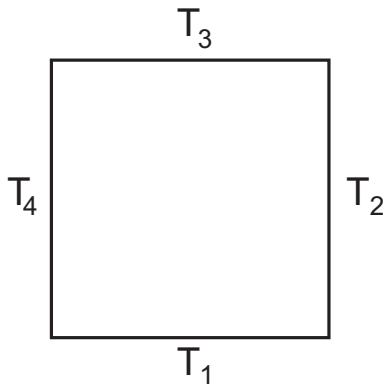


Figura 8.1: Placa plana homogênea

Considerando que todos os pontos da placa estejam a uma temperatura inicial T_0 e sendo esta temperatura diferente das temperaturas das bordas, o problema que se coloca é determinar a temperatura em qualquer ponto interno da placa em um dado instante de tempo. Neste experimento utilizou-se um quadrado unitário com temperaturas $T_1 = 1^oC$ e $T_2 = T_3 = T_4 = 0^oC$.

Para a realização dos testes empregando multigrid, utilizou-se 4 níveis de malha, com 1337, 5348, 21392 e 85568 triângulos, respectivamente. Nos demais testes, sem o emprego do multigrid, utilizou-se apenas a malha mais refinada, que é a malha onde procura-se a solução do problema. Na Figura 8.2 encontra-se a malha com 1337 e a mesma malha particionada em 20 subdomínios.

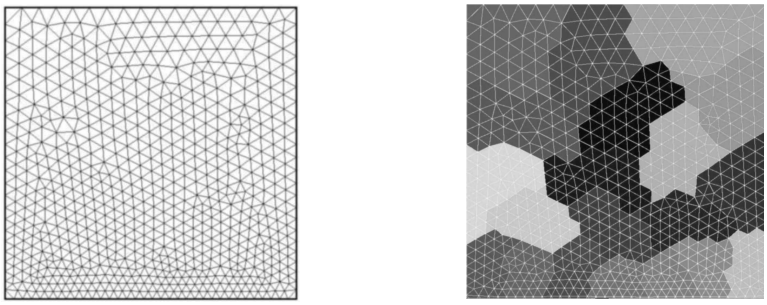


Figura 8.2: Malha com 1337 triângulos original e particionada em 20 subdomínios

Na Figura 8.3, são mostrados alguns passos da resolução do problema da transferência de calor utilizando 5 processos. Em (a) apresenta-se a solução no primeiro passo de tempo, em (b) após 5 passos, em (c) após 10 passos e em (d) após 20 passos de tempo.

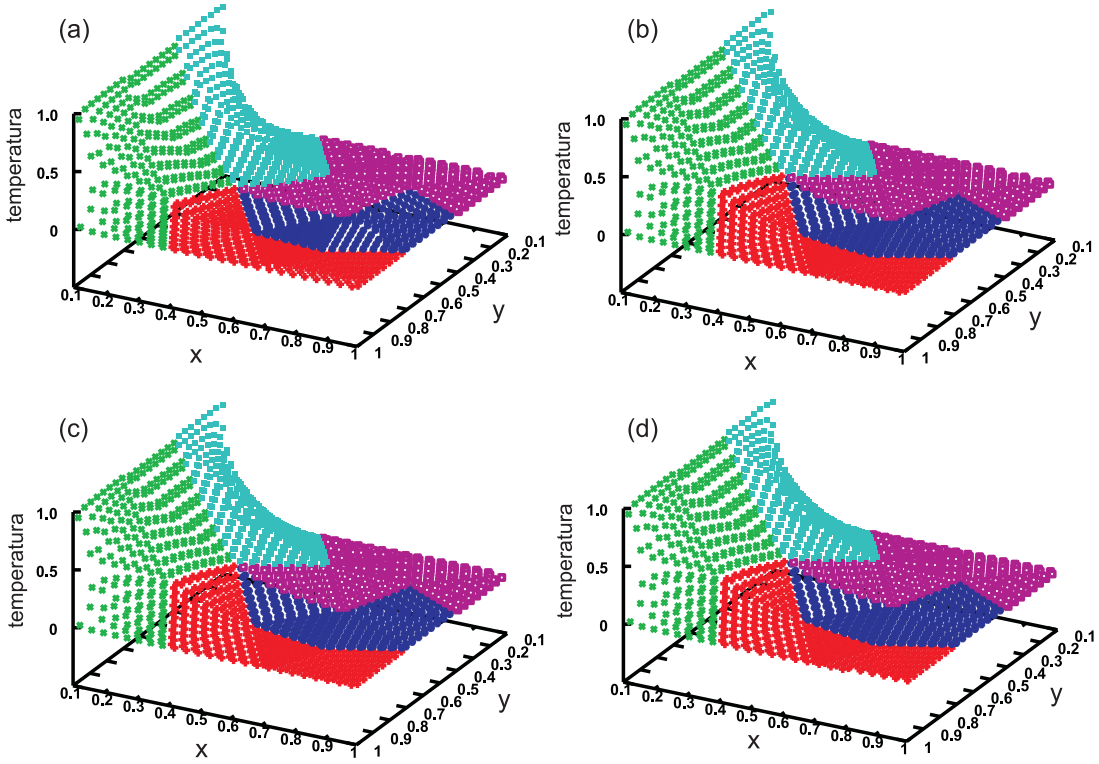


Figura 8.3: Passos da resolução do problema de transferência de calor

Pode-se notar que a diferença na curva de temperatura é bastante acentuada entre a primeira e a quinta iteração. Após a décima iteração a temperatura tende a entrar em equilibrio, tornando as mudanças na curva menos perceptíveis.

8.1.1 Análise de Desempenho

Os testes do problema de trasferência de calor foram efetuados utilizando 19 nodos do *cluster* labtec do Instituto de Informática da UFRGS. Utilizou-se dois processos por nodo, de modo a aproveitar a característica dos nodos de serem bi-processados.

Na análise de desempenho avaliou-se o tempo de execução e a eficiência das implementações. Na tomada de tempo foram feitas 20 execuções de cada implementação e o tempo considerado foi a média aritmética dessas. Dos tempos coletados, o maior e o menor valores foram descartados. Resultados anômalos não foram considerados. Foram simulados 20 passos de tempo. Os tempos de execução são dados em segundos.

8.1.1.1 Multigrid-Aditivo de Schwarz

As Figuras 8.4 e 8.5 mostram, respectivamente, o tempo de execução e a eficiência da resolução do problema da transferência de calor, utilizando o método de solução que combina multigrid ao método aditivo de Schwarz. Ainda nesses gráficos, é feito uma comparação deste método com o método aditivo de Schwarz sem o emprego de multigrid.

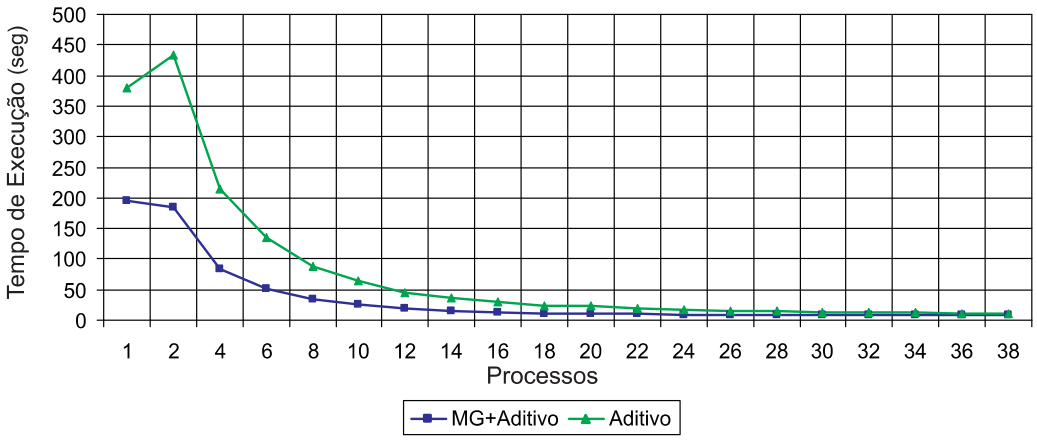


Figura 8.4: Tempo de Execução: MG+Aditivo versus Aditivo

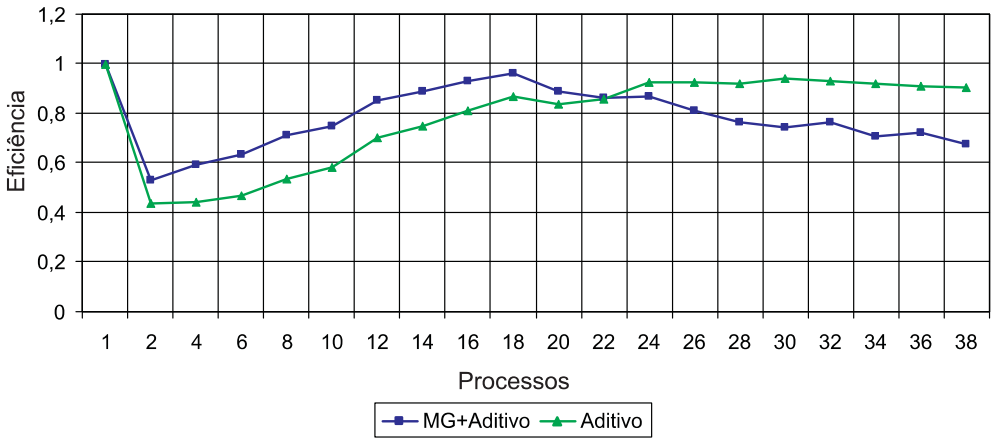


Figura 8.5: Eficiência: MG+Aditivo versus Aditivo

Baseado no comportamento do gráficos de eficiência, pode-se notar que os ganhos de desempenho são representativos com o uso de até 22 processos, onde obteve-se 18,94 de ganho de desempenho. A partir disso há ganhos de desempenho, porém menos significativos, tendo como ganho máximo de desempenho de 26,08 com 36 processos. Esse comportamento deve-se ao aumento da quantidade de comunicação e pela redução do processamento necessário para cada subdomínio, já que os domínios possuem tamanho reduzido quando o número de partições é alto. Além disso, com o aumento da quantidade de processos, cada subdomínio passa a ter mais vizinhos, o que acarreta o aumento das áreas de sobreposição, e por conseqüência o aumento dos sistemas locais.

A eficácia dos métodos multigrid na aceleração dos métodos iterativos também pode ser observada nos gráficos. Em média, a implementação utilizando multigrid foi 2,01 vezes mais rápida que a implementação que não emprega multigrid.

8.1.1.2 Multigrid-Complemento de Schur

As Figuras 8.4 e 8.5 mostram, respectivamente, o tempo de execução e a eficiência da resolução do problema da transferência de calor, utilizando o método de solução que combina multigrid ao método do complemento de Schur. Ainda nesses gráficos, é feito uma comparação deste método com o método do complemento de Schur sem o emprego de multigrid.

Estes resultados foram obtidos utilizando a abordagem de solução explícita para o cálculo do complemento de Schur, conforme abordado na Seção 5.2.1. Nesta implementação também obteve-se bons resultados com a paralelização. Uma excessão ocorre quando utilizado 2 processos. Esse comportamento ocorre devido ao tamanho elevado dos sistemas de fronteira (complemento de Schur) a serem calculados explicitamente.

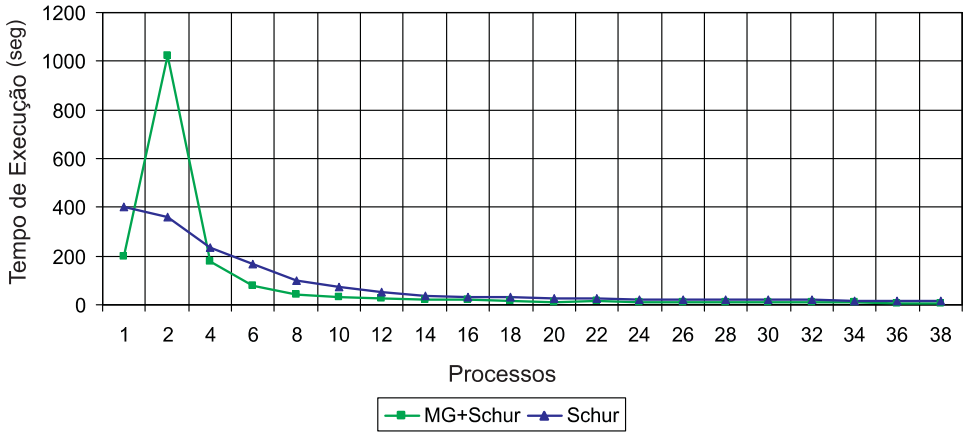


Figura 8.6: Tempo de Execução: MG+Schur versus Schur

O ponto de máxima eficiência alcançado pelo algoritmo foi alcançado quando executado utilizando 20 processos, onde obteve-se ganhos de desempenho na ordem de 16,11 vezes, com 79% de eficiência. O ganho máximo de desempenho do algoritmo foi de 27,07 vezes, utilizando 36 processos, com 75% de eficiência.

No gráfico de eficiência, pode-se observar a presença de picos. Estes picos são causados pela resolução dos sistemas de interface, que em alguns casos, necessitam de mais iterações para alcançar a precisão desejada, conforme pode-se observar na Tabela 8.1.

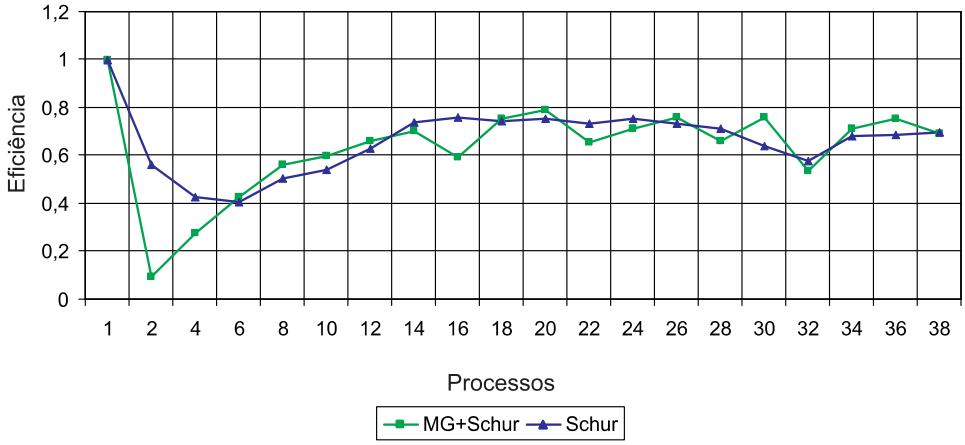


Figura 8.7: Eficiência: MG+Schur versus Schur

Tabela 8.1: Iterações necessárias para a convergência do complemento de Schur. São consideradas as iterações em todos os níveis de malha

Processos	2	4	6	8	10	12	14	16	18	20
Iterações	52	28	28	27	40	32	31	45	30	27
Processos	22	24	26	28	30	32	34	36	38	
Iterações	46	37	27	34	24	44	41	34	35	

Os resultados da combinação de multigrid com o método do complemento de Schur foram semelhantes ao do método anteriomente apresentado, apresentando desempenho de 2,02 superior ao da implementação que não emprega multigrid.

Avaliou-se também a abordagem do método do complemento de Schur utilizando as inversas das diagonais (vide Seção 5.2.1), onde utiliza-se como aproximação da matriz apenas a inversa da diagonal. Este método é mais simples de ser implementado, no entanto a qualidade numérica é insatisfatória, como será analisado na Seção 8.3. Essa aborgagem é denominada, neste trabalho, de método do complemento de Schur polinomial.

Os resultados de tempo de execução e eficiência são mostrados nas Figuras 8.8 e 8.9, respectivamente. Da mesma forma, analisou-se as implementações com e sem o emprego de multigrid.

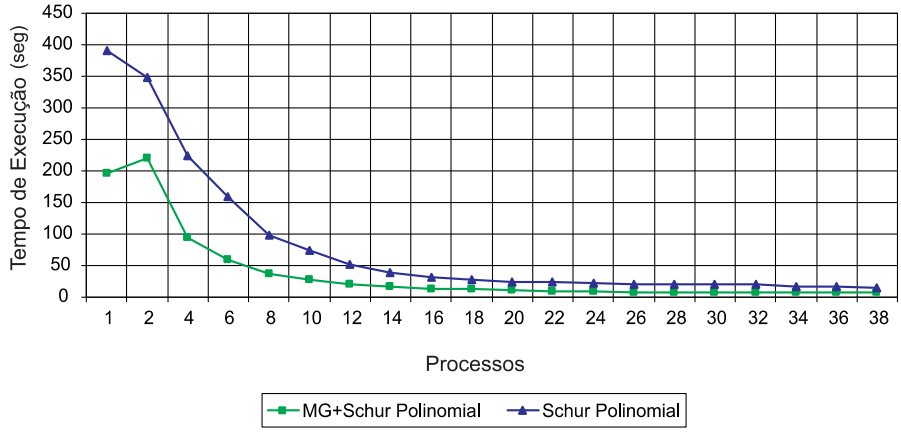


Figura 8.8: Tempo de Execução: MG+Schur Polinomial versus Schur Polinomial

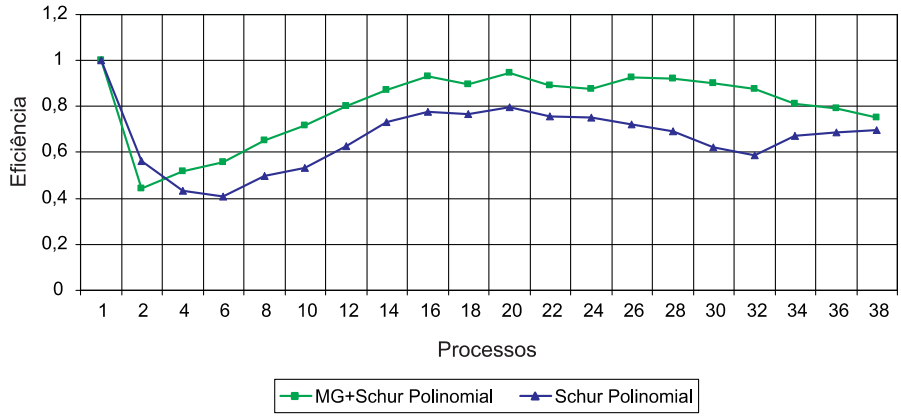


Figura 8.9: Eficiência: MG+Schur Polinomial versus Schur Polinomial

Pode-se notar uma maior homogeneidade nos resultados contidos no gráficos de e eficiência, já que o cálculo dos sistemas de interface são aproximados de modo mais simples e o número de iterações é mais uniforme, como pode ser visto na Tabela 8.2. Com essa simplificação na solução, obteve-se bons ganhos de desempenho, chegando a 28,53 vezes com 38 processos. Nesta implementação, os ganhos com o uso do multigrid foram ainda mais significativos, apresentando 2,42 vezes mais desempenho do que o mesmo algoritmo sem o uso do multigrid.

Tabela 8.2: Iterações necessárias para a convergência do complemento de Schur com aproximação polinomial. São consideradas as iterações em todos os níveis de malha

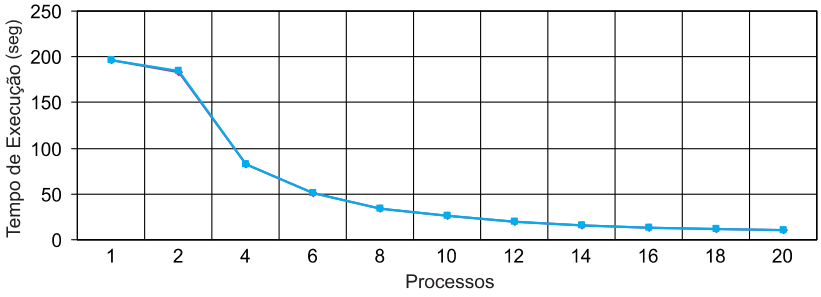
Processos	2	4	6	8	10	12	14	16	18	20
Iterações	43	23	24	26	27	28	26	27	25	28
Processos	22	24	26	28	30	32	34	36	38	
Iterações	32	34	27	23	24	24	24	27	31	

8.1.1.3 Contenção de Memória

Nesta seção faz-se uma análise comparativa sobre a execução dos métodos propostos utilizando um ou dois processos por nodo do cluster. Essa análise é baseada nos estudos de Picinin (2002) e Martinotto (200), onde os autores analisam a ocorrência de contenção de memória no caso do uso de dois processos em um nodo bi-processado.

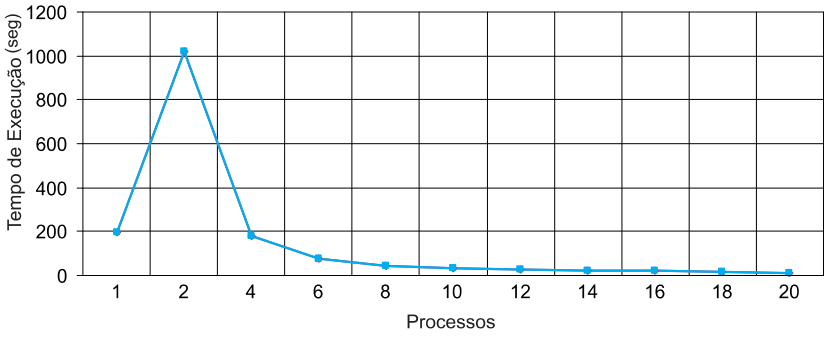
Os testes foram feitos utilizando 20 processos MPI. Primeiramente, as aplicações foram testadas utilizando 1 processo para cada um dos 20 nodos e posteriomente 2 processos, utilizando 10 nodos. Os resultados obtidos são mostrados nas Figuras 8.10 e 8.11. Pela semelhança nos tempos de execução, é quase impossível visualizar a diferença nos gráficos. Dessa maneira, pode-se observar nas tabelas na parte inferior de cada gráfico os respectivos tempos e a diferença entre eles.

Pelos resultados mostrados, pode-se dizer que para as aplicações testadas, a contenção de memória não é significativa, e por conseqüência o uso de dois processadores por nodo não afeta o desempenho da aplicação. A diferença máxima obtida nos testes foi de aproximadamente 1,12%. Diferenças no tempo de execução com essa grandeza ocorrem normalmente em diferentes execuções do mesmo algoritmo sob mesmas configurações de teste.



20 nodos	196,13	183,06	82,214	51,354	34,524	26,273	19,345	15,757	12,98	11,271	10,236
10 nodos	196,13	185,12	83,043	51,659	34,474	26,13	19,181	15,751	13,152	11,361	11,03
Diferença	–	2,06	0,829	0,305	-0,05	-0,143	-0,164	-0,006	0,172	0,09	0,794

Figura 8.10: MG+Aditivo: Execução utilizando 10 e 20 nodos



20 nodos	196,13	1019,9	177,35	76,257	43,586	32,751	24,893	20,046	20,78	14,43	12,34
10 nodos	196,13	1021,1	178,23	76,708	43,585	32,79	24,832	19,987	20,758	14,436	12,389
Diferença	--	1,02	0,88	0,451	-0,001	0,039	-0,061	-0,006	-0,022	0,006	0,049

Figura 8.11: MG+Schur: Execução utilizando 10 e 20 nodos

Uma abordagem diferente, empregando duas *threads* internamente a um processo MPI pode apresentar melhores resultados na exploração do paralelismo intra-nodos, já que dessa forma reduz-se a quantidade de processos à metade e dessa forma otimiza-se a quantidade de comunicação necessária.

8.1.1.4 Comparação dos Métodos

Na Figuras 8.12 é mostrado um comparativo dos tempos de execução dos método multigrid-aditivo, multigrid-schur e multigrid-schur polinomial na resolução do problema de transferência de calor. O gráfico foi dividido em duas partes para melhor visualização. Na primeira parte (esquerda) pode-se visualisar os tempos de execução para 1, 2 e 4 processos. Já na segunda parte (direita), estão os tempos de execução empregando de 6 até 38 processos.

Como pode ser observado, em termos de tempo, o método que se sobressai na resolução deste problema é o método multigrid-schur polinomial, no entanto, como já dito anteriormente, o método apresenta baixa qualidade numérica. Assim, método que combina melhor desempenho e qualidade é o *multigrid combinado ao método aditivo de Schwarz*.

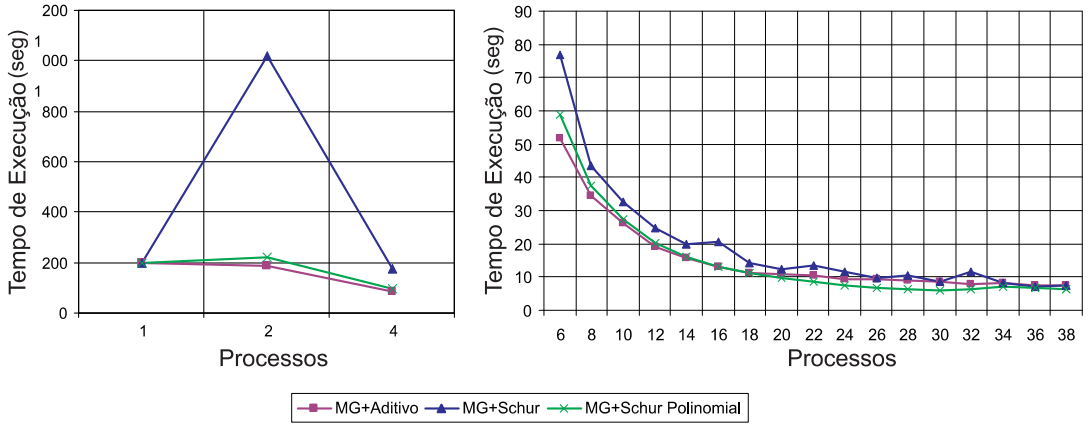


Figura 8.12: Comparação de tempo de execução dos métodos na solução do problema de transferência de calor

8.2 Hidrodinâmica

Nos testes da hidrodinâmica utilizou-se como domínio computacional o Lago Guaíba. O Guaíba é um corpo de água doce que se situa entre o chamado delta do Jacuí e a Lagoa dos Patos. Banha toda a região metropolitana de Porto Alegre e tem uma área total de aproximadamente $468km^2$ e profundidade média $4{,}0m$. Com uma extensão de aproximadamente $50km$, o Guaíba deságua na Lagoa dos Patos e apresenta seções de até $15km$ de largura. A opção por utilizar o Lago Guaíba como estudo de caso deve-se ao fato de que existem alguns dados da geometria e de parâmetros físicos para definir as condições iniciais e de contorno do problema. Além disso, outros trabalhos do grupo, como por exemplo Rizzi (2002), Picinin (2002) e Martinotto (2004), já utilizaram este domíno como estudo de caso, podendo assim ser possível a comparação dos resultados obtidos.

Neste estudo de caso utilizou-se 4 níveis de malha, com 2818, 11272 , 45088 e 180352 triângulos, respectivamente. A malha mais grosseira e alguns detalhes das malhas refinadas são mostrados na Figura 8.13.

8.2.1 Análise de Desempenho

Os testes do problema de hidrodinâmica foram efetuados utilizando os 18 nodos do *cluster* Krusty, do projeto UnHIDRA, na Universidade Estadual do Oeste do Paraná. O cluster Krusty é formado por 18 nodos Pentium 4 3.0 GHz, memória RAM de 1 GB, 1MB de memória cache e HD de 80 GB. Os nodos do *cluster* são interconectados por rede Gigabit-Ethernet.

Na análise de desempenho avaliou-se o tempo de execução e a eficiência das implementações. Na tomada de tempo foram feitas 20 execuções de cada implementação e o tempo considerado foi a média aritmética dessas. Dos tempos coletados, o maior e o menor valores foram descartados. Resultados anômalos não foram considerados. Foram simulados 100 passos de tempo. Os tempos de execução são dados em segundos.

8.2.1.1 Multigrid-Aditivo de Schwarz

As Figuras 8.14 e 8.15 mostram, respectivamente, o tempo de execução e a eficiência da resolução do problema da hidrodinâmica, utilizando o método de solução

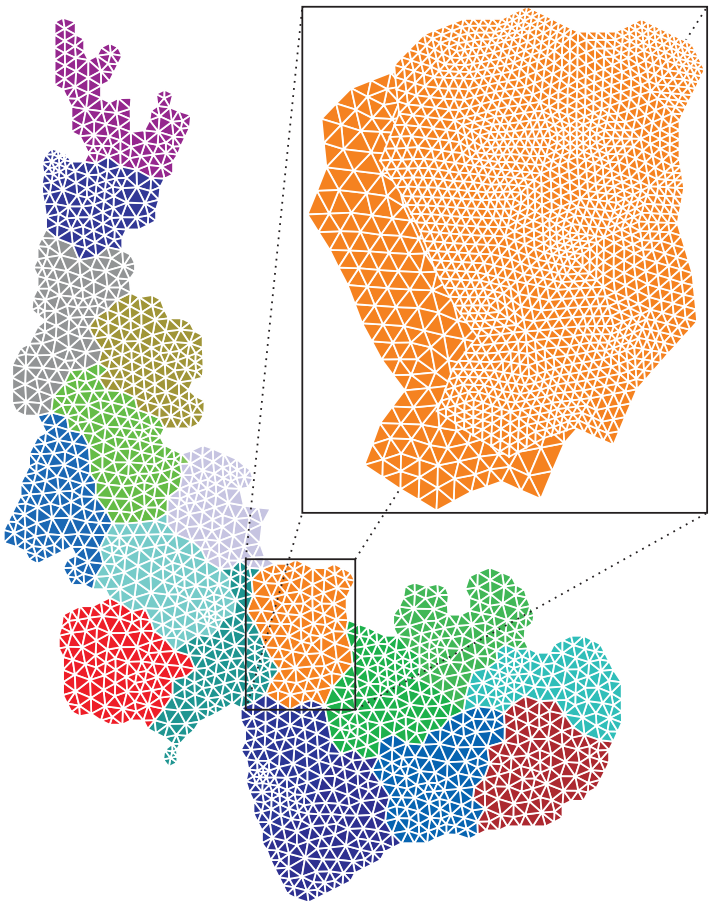


Figura 8.13: Guaíba: malha particionada em desesseis subdomínios. No detalhe pode-se observar dois níveis de refinamentos da hierarquia de malhas

que combina multigrid ao método aditivo de Schwarz. Da mesma forma como foi feito no problema da transferência de calor, compara-se os métodos com e sem o uso do multigrid.

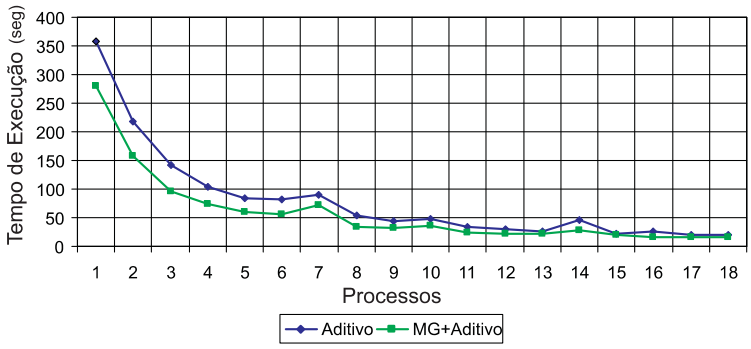


Figura 8.14: Tempo de Execução: MG+Aditivo versus Aditivo

Baseado no comportamento do gráfico de eficiência, pode-se notar que o método mostrou-se bastante escalável para esta aplicação, com a presença de picos na execução com 7 e 14 processos. Esses picos ocorrem devido ao particular particionamento do domínio, que exige que mais iterações sejam necessárias para a convergência do método.

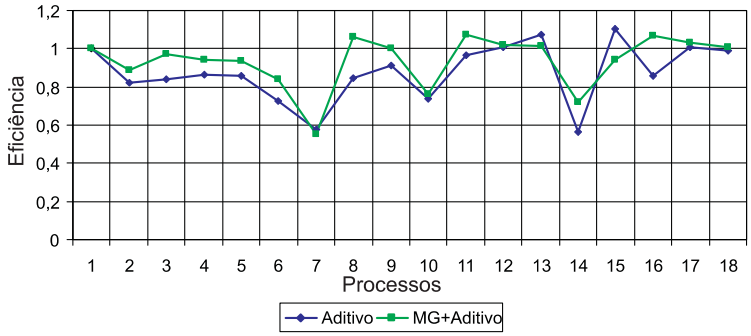


Figura 8.15: Eficiência: MG+Aditivo versus Aditivo

Na comparação entre os métodos com e sem o emprego de multigrid, pode-se notar, que para esta aplicação o ganho foi menos significativo que o obtido no problema da transferência de calor. Em média, a implementação utilizando multigrid foi 40% mais rápida que a implementação que não emprega multigrid.

8.2.1.2 Multigrid-Complemento de Schur

As Figuras 8.16 e 8.17 mostram, respectivamente, o tempo de execução e a eficiência da resolução obtidos na resolução do problema da hidrodinâmica, utilizando o método de solução que combina multigrid ao método do complemento de Schur. Observa-se ainda nesses gráficos, a comparação deste método com o método do complemento de Schur sem o emprego de multigrid.

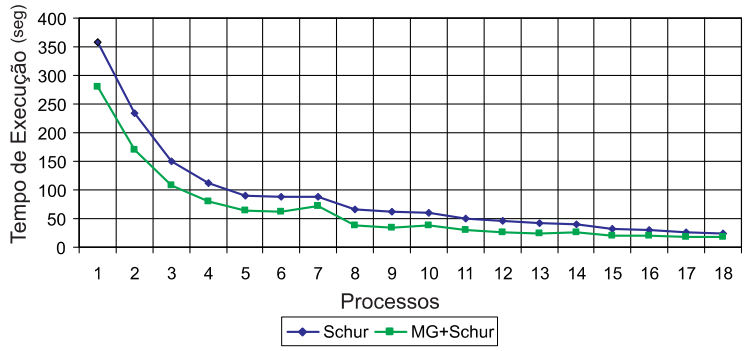


Figura 8.16: Tempo de Execução: MG+Schur versus Schur

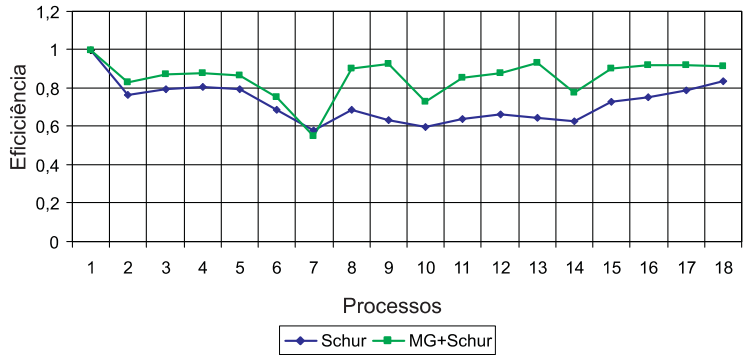


Figura 8.17: Eficiência: MG+Schur versus Schur

O método do complemento de Schur combinado ao multigrid também mostrou-se escalável para o problema da hidrodinâmica, apresentando bom ganho de desempenho e eficiência média de 84%. Mais uma vez, pode-se notar picos no tempo de execução quando utiliza-se 7, 10 e 14 processos, reafirmando a necessidade de mais iterações para a convergência nesses casos (ver Tabela 8.3). Os ganhos com o uso do multigrid para este método foram, em média, 53% em relação ao método sem multigrid.

Tabela 8.3: Iterações necessárias para a convergência do complemento de Schur. São consideradas as iterações em todos os níveis de malha

Processos	2	3	4	5	6	7	8	9	10
Iterações	20	19	18	22	24	44	21	23	42
Processos	11	12	13	14	15	16	17	18	
Iterações	26	28	21	42	23	20	22	20	

Avaliou-se também o método do complemento de Schur polinomial na solução da hidrodinâmica. Os resultados de tempo de execução e eficiência são mostrados nas Figuras 8.18 e 8.19, respectivamente. Analisou-se também as implementações com e sem o emprego de multigrid.

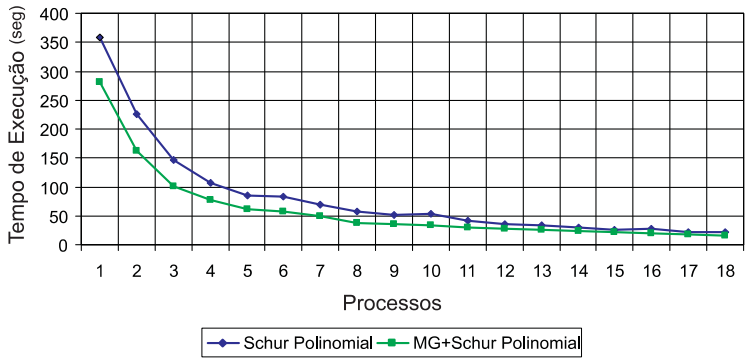


Figura 8.18: Tempo de Execução: MG+Schur Polinomial versus Schur Polinomial

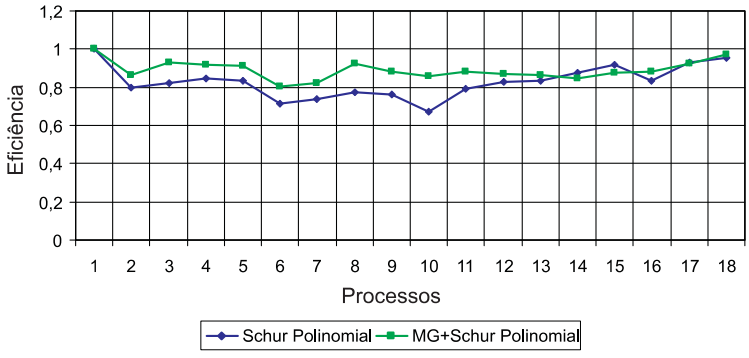


Figura 8.19: Eficiência: MG+Schur Polinomial versus Schur Polinomial

O método apresentou bom desempenho e eficiência, 88% em média. Ao contrário dos resultados obtidos com o problema de transferência de calor, os ganhos com a abordagem simplificada do método de Schur não foram tão significativos, sendo 38% mais rápida que o mesmo algoritmo sem o uso do multigrid.

8.2.1.3 Comparação dos Métodos

Na Figuras 8.20 é mostrado um comparativo dos tempos de execução dos método multigrid-aditivo, multigrid-schur e multigrid-schur polinomial.

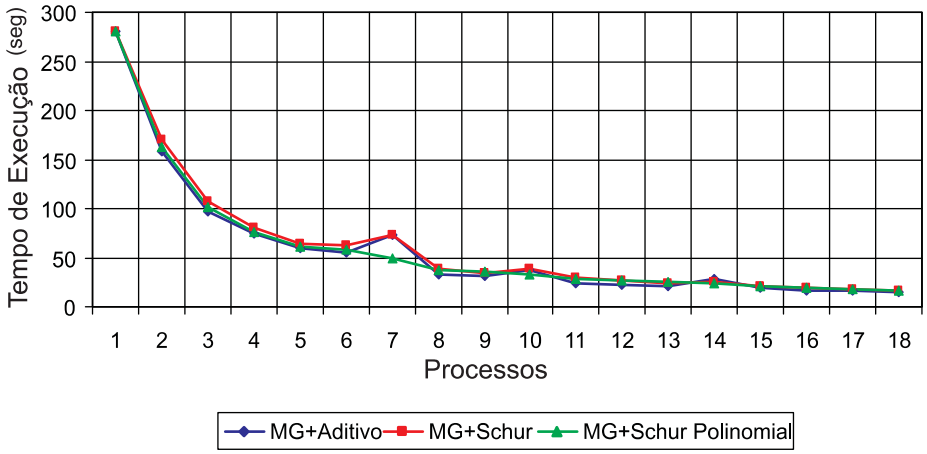


Figura 8.20: Comparação de tempo de execução dos métodos na solução do problema de hidrodinâmica

Pode-se notar que tempos obtidos foram muito equivalentes, sendo qualquer um deles uma boa opção para a resolução do problema de hidrodinâmica. A escolha por um determinado método fica a cargo da qualidade numérica desejada, e das demais questões relacionadas à ele, como por exemplo geração do sistema de equações, que é mais complexa para os métodos de Schur do que para os métodos de Schwarz.

8.3 Análise da Qualidade Numérica

Para uma verificação da qualidade númerica das versões paralelas desenvolvidas utilizou-se o procedimento descrito em Rizzi (2002). Inicialmente, a solução numérica monoprocessada é comparada a um *benchmark*. Neste trabalho, o *benchmark* empregado foi a solução obtida no software Matlab. Então, considerando que a solução numérica monoprocessada é, numericamente, a correta, um modo de avaliar a solução paralela é empregar a métrica do erro relativo para todas as células do domínio computacional. O erro relativo é dado por

$$E_R = |\varphi_i - \varphi_i'|/\varphi_i$$

onde φ_i é a solução tida como exata e φ_i' é a solução aproximada.

Para os testes de qualidade numérica utilizou-se um domínio quadrado com as mesmas condições iniciais consideradas para a análise de desempenho. A malha utilizada contém 54, 216, 864 e 3456 triângulos para o primeiro, segundo, terceiro e quarto nível, respectivamente. Avaliou-se o erro no interior do domínio e nas fronteiras. Os erros médios obtidos nos problemas da transferência de calor e da hidrodinâmica encontram-se nas Tabelas 8.4 e 8.5.

É importante considerar que os erros obtidos podem variar de acordo com a malha utilizada, as condições de contorno empregadas, a precisão dos métodos entre outros fatores relevantes.

Tabela 8.4: Erros na solução do problema de transferência de calor

–	MG+Aditivo	MG+Schur	MG+Schur Polinomial
Interior	1,321%	2,725%	8,983%
Fronteira	1,942%	2,862%	10,332%

Tabela 8.5: Erros na solução do problema de hidrodinâmica

–	MG+Aditivo	MG+Schur	MG+Schur Polinomial
Interior	1,865%	2,102%	7,392%
Fronteira	2,028%	2,167%	7,455%

8.4 Considerações Finais

Apresentou-se neste capítulo a avaliação dos resultados obtidos com as implementações desenvolvidas neste trabalho.

Em um primeiro momento analisou-se os resultados obtidos na solução do problema de transferência de calor. Os métodos implementados mostraram-se escaláveis, apresentando bons desempenhos. A aceleração por métodos multigrid mostrou-se eficiente. Os métodos que utilizaram esta técnica apresentaram tempo de execução, em média, duas vezes menor do que os métodos que não utilizaram nenhum método de aceleração. Apresentou-se também um breve estudo sobre a contenção de memória com o uso de dois processos por nodo no *cluster* labtec. De acordo com os resultados observados, a contenção não foi significativa para a aplicação testada.

Analisou-se também os resultados obtidos com a solução da hidrodinâmica. Mais uma vez, os métodos implementados mostraram-se eficientes na solução do problema. Nestes testes, pode-se observar ganhos de 40% no desempenho com o uso do multigrid.

Por fim, analisou-se a qualidade numérica dos métodos. O método Multigrid-Aditivo apresentou uma boa qualidade numérica, mostrando-se uma boa opção para a resolução paralela de sistemas de equações. Um outro bom resultado foi obtido pela abordagem Multigrid-Schur utilizando a abordagem que calcula explicitamente o complemento de Schur. Esta abordagem apresentou uma qualidade numérica muito superior àquela que utiliza a inversa da diagonal como aproximação para a matriz inversa.

9 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho teve como objetivo apresentar a solução de problemas científicos através de métodos multigrid paralelizados através de decomposição de domínios.

9.1 Revisão do Trabalho Desenvolvido

Inicialmente, no Capítulo 1 introduziu-se o trabalho, citando alguns trabalhos relacionados, a motivação e os principais objetivos deste trabalho. Nos quatro capítulos que seguem, apresenta-se a revisão bibliográfica dos assuntos pertinentes a essa dissertação.

O Capítulo 2 discutiu os processos de geração e particionamento da malha. No Capítulo 3 abordou-se as questões relacionados à resolução de sistemas de equações. O Capítulo 4 focou o ambiente de desenvolvimento do trabalho. O Capítulo 5 apresentou uma visão geral dos métodos de decomposição de domínio.

Os próximos dois capítulos, 6 e 7, contemplaram características da aplicação que foi desenvolvida. No Capítulo 6, mostrou-se o desenvolvimento de algumas etapas necessárias para o emprego de métodos multigrid na resolução de sistemas de equações lineares. No Capítulo 7 abordou-se a paralelização dos métodos multigrid. Os métodos multigrid foram paralelizados através da abordagem de decomposição de domínio.

Por fim, no Capítulo 8 são apresentados os resultados obtidos com os métodos de solução implementados.

9.2 Conclusões

O principal objetivo deste trabalho foi o desenvolvimento e a avaliação de desempenho das soluções paralelas obtidas através da combinação de métodos multigrid e métodos de decomposição de domínio (MG+MDD). Mais especificamente combinou-se o método Full Multigrid V com dois métodos de decomposição de domínio: o método aditivo de Schwarz e o método do complemento de Schur.

Além dos métodos de solução, desenvolveu-se também toda uma estrutura de suporte para a resolução utilizando os métodos MG+MDD. Para as tarefas de geração da hierarquia de malhas e montagem dos sistemas de equações foi implementado um módulo chamado de *MGTool* responsável pela geração da hierarquia de malhas e pela montagem dos sistemas de equações. Através *do MGTool* foram geradas as malhas para os estudos de caso, bem como os sistemas de equações tanto para o método aditivo de Schwarz como para o método do complemento de Schur.

De acordo com os resultados observados no Capítulo 8, os métodos implementados mostraram-se altamente paralelizáveis, apresentando bons ganhos de desempenho. Pode-se observar que os métodos multigrid mostraram-se eficiente na aceleração dos métodos iterativos, já que métodos que utilizaram esta técnica apresentaram desempenho superior aos métodos que não utilizaram nenhum método de aceleração. Para o problema de transferência de calor, os métodos empregando multigrid convergiram, em média, duas vezes mais rapido. Já no problema de hidrodinâmica, os ganhos foram menos significativos, mostranto convergência 40% mais rápida.

A avaliação do método de Multigrid-Schwarz mostrou bons resultados na solução dos dois problemas tratados, apresentando boa escalabilidade e boa qualidade numérica nos dois estudos de caso apresentados, apresentando a melhor relação desempenho e qualidade de solução.

Avaliou-se também duas diferentes abordagens do método Multigrid-Schur. Na primeira abordagem, o cálculo do complemento de Schur é feito explicitamente, através da resolução de subsistemas de equações. Este método apresentou desempenho satisfatório e também uma boa qualidade numérica, mostrando-se também, assim como o método que emprega o método aditivo de Schwarz, uma boa alternativa de solução para problemas científicos. Na segunda abordagem, os bons resultados limitam-se ao desempenho, já que a qualidade numérica mostrou-se insuficiente para a resolução de problemas realísticos, devido principalmente a erros numéricos introduzidos pelo uso de aproximações polinomiais no cálculo das inversas locais.

No estudo sobre a contenção de memória, avaliou-se o impacto no desempenho na utilização de dois processos por nodo no *cluster* labtec. Comparou-se a execução de 20 processos MPI utilizando um processador por nodo, utilizando 20 nodos e dois processadores por nodo, utilizando 10 nodos. De acordo com os resultados obtidos, pode-se observar que o uso de dois processadores por nodo não causou uma contenção de memória/barramento significativa. Ganhos de desempenho significativos podem ser alcançados utilizando os dois processadores do nodo de modo mais eficiente, de modo que se dispare um processo MPI por nodo, e este utilizando-se de duas *threads*. Dessa forma pode-se explorar eficientemente tanto o paralelismo inter-nodo como o paralelismo intra-nodo. Neste trabalho essa abordagem não foi empregada devido às características síncronas dos algoritmos.

Como principal conclusão tem-se que a combinação de métodos multigrid e decomposição de domínios mostraram-se uma boa opção na solução de sistemas de equações provenientes da discretização de equações diferenciais parciais. Embora a solução por estes métodos possua algumas fases bastante complexas, como a geração da hierarquia de malhas e a geração dos sistemas de equações para os diversos níveis da malha, os ganhos de desempenho justificam estas questões.

9.3 Contribuições

Desde 2000 o GMCPAD vem trabalhando com diferentes abordagens para a resolução de sistemas de equações em paralelo. Em Canal (2000) e Picinin (2002) explorou-se o paralelismo de dados; em Martinotto (2004) foi a vez dos métodos de decomposição de domínios serem utilizados. Assim a contribuição deste trabalho ao grupo, é fornecer mais uma alternativa de métodos para a solução de sistemas em paralelo.

O objetivo do trabalho foi o desenvolvimento de métodos métodos multigrid paralelizados pela abordagem de decomposição de domínio, utilizados na resolução paralela dos sistemas de equações gerados pela discretização de equações diferenciais parciais em malhas não estruturadas. Algumas contribuições deste trabalho foram:

1. Metodologia para a geração de malhas não estruturadas ortogonais;
2. Técnica para a geração da hierarquia de malhas necessária para os métodos multigrid;
3. Modelos de transferência de calor e de hidrodinâmica que serviram de estudo de caso nos testes dos métodos propostos. Estes modelos podem ser utilizados posteriormente em outros trabalhos;
4. Disponibilização dos métodos multigrid combinados a métodos de decomposição de domínio, com e sem sobreposição;
5. Nova técnica para o cálculo das inversas no método do complemento de Schur;
6. Análise de desempenho e qualidade numérica dos métodos implementados.

Os estudos realizados durante o desenvolvimento desse trabalho resultaram na publicação de seis (6) trabalhos que foram publicados em eventos internacionais, nacionais e regionais. Todos os artigos desenvolvidos encontram-se no Anexo C.

A principal publicação aceita até o momento, é o artigo que fará parte dos anais do VECPAR 06 (International Meeting on High Performance Computing for Computational Science), que será realizado em julho na cidade do Rio de Janeiro-RJ.

Além disso, foi publicado um artigo no evento WSCAD 2004 (Quinto Workshop em Sistemas Computacionais de Alto Desempenho) realizado em Foz do Iguaçu-PR. O artigo apresentado foi escolhido como o melhor do evento.

Também foram publicados 4 artigos em eventos regionais, os quais foram: WSGPPD 2004, ERAD 2005, WSGPPD 2005 e ERAD 2006.

Por fim, foram submetidos trabalhos para o evento PARA'06 (Umea, Suécia) e para a revista Parallel Computing.

9.4 Trabalhos Futuros

Neste trabalho foram desenvolvidas atividades que visaram paralelização de métodos multigrid através de métodos de decomposição de domínio. Porém, alguns pontos importantes não puderam ser contemplados nesta dissertação, o que é inevitável quando se determina os objetivos a serem atingidos dentro de um determinado tempo. A seguir pontua-se algumas atividades que podem ser desenvolvidas para dar continuidade ao trabalho:

1. Empregar os métodos desenvolvidos na resolução de sistemas originados em outras aplicações.
2. Pesquisar novas abordagens para os operadores de interpolação e restrição, de modo que se obtenha métodos de transferência entre malhas mais acurados;
3. Utilizar multigrid como método de resolução local nos métodos de decomposição de domínios, obtendo uma abordagem MDD-Multigrid;

4. Desenvolver novos mecanismos para a geração da hierarquia de malhas;

5. Comparar os mecanismos de geração de hierarquia de malhas com os mecanismos disponíveis em outros pacotes;

6. Aplicar os métodos multigrid em malhas adaptativas. O uso de malhas adaptativas em aplicações paralelas requer também a implementação de algoritmos para balanceamento de carga;

7. O desenvolvimento de estruturas de dados visando o uso efetivo das bibliotecas BLAS 1, 2 e 3 para a solução das operações matriciais que compõem os métodos iterativos.

Os métodos desenvolvidos nesse trabalhos podem ser agregados a outras aplicações, de modo a oferecer flexibilidade e eficiência na resolução de sistemas de equações.

REFERÊNCIAS

ADAMS, J. C. MUDPACK-2: multigrid software for approximating elliptic partial differential equations on uniform grids with any resolution. **Appl. Math. Comput.**, New York, NY, USA, v.53, n.2-3, p.235–249, 1993.

AL-NASRA, M.; NGUYEN, D. An Algorithm for Domain Decomposition in Finite Element Analysis. **Computer and Structures**, [S.l.], v.39, p.277–289, 1991.

ALMASI, G. S.; GOTTLIEB, A. **Highly parallel computing**. [S.l.]: Benjamin-Cummings Publishing Co., Inc., 1989.

AUADA, R. B. **Utilização de Malhas Não-Estruturadas em Dinâmica dos Fluídos Computacional**. 1997. 182pp. Dissertação (Mestrado em Engenharia Mecânica) — Universidade de São Paulo, São Paulo.

BASTIAN, P.; HACKBUSCH, W.; WITTUM, G. Additive and Multiplicative Multi-Grid — A Comparison. **Computing**, [S.l.], v.60, n.4, p.345–364, 1998.

BERN, M.; PLASSMANN, P. Mesh Generation. In: SACK, J.; URRUTIA, J. (Ed.). **Handbook of Computational Geometry**. [S.l.]: Elsevier Science, 2000.

BERN, M. W.; EPPSTEIN, D. Mesh generation and optimal triangulation. In: DU, D.-Z.; HWANG, F. K.-M. (Ed.). **Computing in Euclidean Geometry**. [S.l.]: World Scientific, 1992. p.23–90. (Lecture Notes Series on Computing).

BITTENCOURT, M. L. **Métodos Iterativos e Multigrid Adaptáveis em Malhas Não Estruturadas**. 1996. Tese (Doutorado em Engenharia Mecânica) — Faculdade de Engenharia Mecânica, UNICAMP, Campinas-SP.

BLUMBERG, A. F.; KANTHA, L. H. Open boundary condition for circulation models. **Journal of Hydraulic Engineering**, [S.l.], v.111, p.237–255, 1985.

BRIGGS, W. **A Multigrid Tutorial**. Philadelphia: SIAM, 1987.

BRUASET, A. M.; LANGTANGEN, H. P.; ZUMBUSCH, G. W. Domain decomposition and multilevel methods in Diffpack. In: CONFERENCE ON DOMAIN DECOMPOSITION, 9., 1998. **Proceedings...** Cambridge: Domain Decomposition Press, 1998. p.77–662.

BUYYA, R. **High Performance Cluster Computing**: Architecture and Systems. [S.l.]: Prentice Hall, 1999. v.1.

CANAL, A. P. **Paralelização de Métodos de Resolução de Sistemas Lineares Esparsos com o DECK em Clusters de PCs**. 2000. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

CASULLI, V.; WALTERS, R. A. An Unstructured Grid, Three-Dimensional Model based on the Shallow Water Equations. **International Journal for Numerical Methods in Fluids**, [S.l.], v.3, 2000.

CHAN, T. F.; MATHEW, T. P. Domain Decomposition Algorithms. In: **Acta Numerica 1994**. [S.l.]: Cambridge University Press, 1994. p.61–143.

CHARÃO, A. S. **Multiprogrammation Parallèle Générique des Méthodes de Décomposition de Domaine**. 2001. Tese (Doutorado em Informática) — Institut National Polytechnique de Grenoble.

CHENG, R. T.; CASULLI, V. Evaluation of the UnTRIM Model for 3-D Tidal Circulation. In: INTERNATIONAL CONFERENCE ON ESTUARINE AND COASTAL MODELING, 7., 2001, St. Petersburg, FL - USA. **Proceedings...** [S.l.: s.n.], 2001. v.1.

CHOW, E.; FALGOUT, R.; HU, J.; TUMINARO, R.; YANG, U. M. A Survey of Parallelization Techniques for Multigrid Solvers. **SIAM Frontiers of Parallel Processing For Scientific Computing**, [S.l.], 2005. Disponível em: <http://www.llnl.gov/CASC/people/chow/pubs/parmg-survey.pdf>. Acesso em jan. 2006.

CISMASIU, I. **Parallel Algorithms for Non-Conventional Finite Element Computations on Distributed Architectures**. 2002. Tese (Doutorado em Engenharia Civil) — Universidade Técnica de Lisboa, Lisboa.

CODENOTTI, B.; LEONCINI, M. **Introduction to Parallel Processing (International Computer Science Series)**. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 1992.

DE ROSE, C. A. F. Arquiteturas Paralelas. In: ERAD, 1., 2001, Gramado-RS. **Anais...** Porto Alegre: SBC/CRAD, 2001. p.3–33.

DEBREU, L.; BLAYO, E. On the Schwarz Alternating Method for Solving Oceanic Models on Parallel Computers. **Journal of Computational Physics**, [S.l.], v.141, p.93–111, 1998.

DORNELES, R. V. **Particionamento de Domínio e Balanceamento de Carga no Modelo HIDRA**. 2003. Tese (Doutorado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre - RS.

DOUGLAS, C. C. A review of numerous parallel multigrid methods. In: ASTFALK, G. (Ed.). **Applications on Advanced Architecture Computers**. Philadelphia: SIAM, 1996. p.187–202.

DOUGLAS, C. C. A Sparse Matrix Approach to Abstract Multilevel Solvers on Serial and Parallel Computers. **ZAMM**, [S.l.], v.76, p.139–142, 1996.

DOUGLAS, C. C. MGNet: a multigrid and domain decomposition network. **ACM SIGNUM Newsletter**, [S.l.], v.27, p.2–8, 2006. Disponível em: <http://www.mgnet.org/>. Acesso em: set. 2005.

DRYJA, M.; WIDLUND, O. B. **An Additive Variant of the Schwarz Alternating Method for the Case of Many Subregions**. New York: Department of Computer Science, Courant Institute, 1987.

EIJKHOUT, M. **LAPACK working note 50**: distributed sparse data structures for linear algebra operations. Knoxville: Computer Science Department, University of Tennessee, 1992. Disponível em: <http://www.cs.utk.edu/∼library/1992.html>. Acesso em: out. 2005.

EL-REWINI, H.; LEWIS, T. G. **Distributed and parallel computing**. [S.l.]: Manning Publications Co., 1998.

FIDUCCIA, C. M.; MATTHEYSES, R. M. A linear-time heuristic for improving network partitions. In: CONFERENCE ON DESIGN AUTOMATION, 19., 1982, Las Vegas. **Proceedings...** New York: IEEE Press, 1982. p.175–181.

FILIPIAK, M. **Mesh Generation**. Edinburg: EPCC, 1996. Watch Report.

FJALLSTROM, P.-O. Algorithms for graph partitioning: a survey. In: LINKöPING ELECTRONIC ARTICLES IN COMPUTER AND INFORMATION SCIENCE, 1998, Linköping. **Proceedings...** [S.l.: s.n.], 1998. Disponível em: <http://www.ep.liu.se/ea/cis/1998/010/>. Acesso em mar. 2006.

FLEMISH, B. **The Alternating Schwarz Method**: Mathematical Foudantion and Parallel Implementation. 2001. Dissertação (Mestrado em Matemática) — Departament of Mathematics, Iowa State University, Ames.

FORTUNE, S. Voronoi Diagrams and Delaunay Triangulations. In: DU, D.-Z.; HWANG, F. (Ed.). **Computing in Euclidean Geometry**. [S.l.]: World Scientific, 1992. (Lecture Notes Series on Computing).

FOSTER, I. **Designing and Building Parallel Programs**. [S.l.]: Addison-Wesley, 1995. Disponível em: <http://www.mcs.anl.gov/dbpp>. Acesso em mar. 2006.

FREUND, R.; GOLUB, G.; NACHTIGAL, N. Iterative solution of linear systems. In: ISERLES, A. (Ed.). **Acta Numerica 1992**. [S.l.]: Cambridge University Press, 1992. p.57–100.

GALANTE, G. **Métodos de Decomposição de Domínios para a Solução Paralela de Sistemas de Equações Lineares**. 2003. Trabalho de Conclusão (Bacharelado em Informática) — Universidade Estadual do Oeste do Paraná, Cascavel, PR.

GALANTE, G. **Geração de Malhas Não Convexas e Adaptativas**. 2004. Trabalho Individual (Mestrado em Ciência da Computação) — Instituto de Informática - UFRGS, Porto Alegre.

GALANTE, G.; DIVéRIO, T.; RIZZI, R.; MARTINOTTO, T.; DORNELES, R.; PICININ, D. Comparação entre Métodos de Decomposição de Domínio e Decomposição de Dados na Solução de Sistemas de Equações. In: WORKSHOP EM SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO, 5., 2004, Foz do Iguaçu. **Anais...** Foz do Iguaçu: SBC, 2004. p.98–104.

GUERRERO, M. S. **Parallel multigrid algorithms for computational fluid dynamics and heat transfer**. 2000. Tese (Doutorado em Engenharia Industrial) — Department de Màquines i Motors Tèrmics, Universitat Politècnica de Catalunya, Terassa.

HENDRICKSON, B.; LELAND, R. **The Chaco user's guide — Version**. Albuquerque: Sandia National Laboratories, 1994. (Technical Report SAND94-2692). Disponível em: <http://www.cs.sandia.gov/pub/papers/bahendr/guide.ps.gz>. Acesso em mar. 2006.

HORNUNG, R.; TRANGENSTEIN, J. Adaptive Mesh Refinement and Multilevel Iteration for Flow in Porous Media. **Journal of Computational Physics**, [S.l.], v.136, p.522–545, 1997. Disponível em: <http://www.math.duke.edu/∼johnt/amr.html>. Acesso em: mar. 2006.

JUDICE, J.; PATRICIO, J. M. **Sistemas de Equações Lineares**. Coimbra: Universidade de Coimbra, 1996.

KARYPIS, G.; KUMAR, V. Analysis of Multilevel Graph Partitioning. In: ACM/IEEE CONFERENCE ON SUPERCOMPUTING, 1995. **Proceedings...** New York: ACM Press, 1995.

KARYPIS, G.; KUMAR, V. **METIS, Unstructured Graph Partitioning and Sparse Matrix Ordering System. Version 2.0**. Minneapolis: University of Minnesota, Department of Computer Science, 1995.

KARYPIS, G.; KUMAR, V. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. **SIAM Journal on Scientific Computing**, [S.l.], v.20, n.1, p.359–392, 1998. Disponível em: <http://www.cs.umn.edu/∼karypis>. Acesso em: mar. 2006.

KERNIGHAN, B.; LIN, S. An effective heuristic procedure for partitioning graphs. **The Bell System Technical Journal**, [S.l.], p.291 – 308, 1970.

KNUPP, P.; STEINBERG, S. **Fundamentals of Grid Generation**. Boca Raton, FL: CRC Press, 1994.

MAILLARD, N. Algoritmos Matriciais em Processamento de Alto Desempenho. In: ERAD, 5., 2005, Canoas - RS. **Anais...** Canoas: SBC/UFPel/UCPel/UNILASALLE/UCS, 2005.

MARGETTS, L. **Parallel Finite Element Analysis**. 2002. Tese (Doutorado em Engenharia Civil) — University of Manchester, Manchester.

MARTINOTTO, A. L. **Resolução de Sistemas de Equações Lineares através de Métodos de Decomposição de Domínio**. 2004. Dissertação de Mestrado

(Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

MAVRIPLIS, D. J. Mesh Generation and Adaptivity for Complex Geometries and Flows. In: PEYRET, R. (Ed.). **Handbook of Computational Fluid Mechanics**. London: Academic, 1996.

MODI, A. **Unstructured Mesh Generation on Planes and Surfaces using Graded Triangulation**. Bombay: Department of Aerospace Engineering - Indian Institute of Technology, 1997.

MORO, R. C. **Aplicação da Técnica Multigrid em Transferência de Calor Computacional**. 2004. Dissertação (Mestrado em Engenharia Mecânica) — Pós-Graduação em Engenharia Mecânica, Curitiba.

MOULITSAS, I.; KARYPIS, G. **Mgridgen/Parmgridgen Serial/Parallel Library for Generating Coarse Grids for Multigrid Methods**. Minneapolis: University of Minnesota, Department of Computer Science, 2001.

O'ROURKE, J. **Computational Geometry in C**. [S.l.]: Cambridge University Press, 1998.

OWEN, S. A Survey of Unstructured Mesh Generation Technology. In: INTERNATIONAL MESHING ROUNDTABLE, 7., 1998. **Proceedings...** [S.l.: s.n.], 1998. p.239–267.

PAAR, K.; ATHANAS, P. M.; EDWARDS, C. M. Implementation of a finite difference method on a custom computing platform. In: HIGH-SPEED COMPUTING, DIGITAL SIGNAL PROCESSING, AND FILTERING USING RECONFIGURABLE LOGIC, 1996. **Proceedings...** [S.l.: s.n.], 1996. p.44–53.

PACHECO, P. S. **Parallel Programming with MPI**. San Francisco: Morgan Kaufmann, 1997. 418p.

PICININ, D. **Paralelização do Algoritmo do Gradiente Conjugado através da Biblioteca MPI e de Threads**. 2001. Trabalho Individual (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

PICININ, D. **Paralelização de Métodos de Solução de Sistemas Lineares em Clusters de PCs com as Bibliotecas DECK, MPICH e Pthreads**. 2002. Dissertação de Mestrado (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

RABI, J. A. **Aplicação do Método Multigrid na Solução numérica de Problemas 2-D Simples de Mecânica dos Fluidos e Transferência de Calor**. 1999. Dissertaçao (Mestrado em Ciências) — Centro Técnico Aerospacial - ITA, São José dos Campos - SP.

RIZZI, R. L. **Modelo Computacional Paralelo para a Hidrodinâmica e para o Transporte de Massa Bidimensional e Tridimensional**. 2002. Tese (Doutorado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre-RS.

SAAD, Y. **Iterative Methods for Sparse Linear Systems**. [S.l.]: PWS Publishing Company, 1996.

SAAD, Y.; SOSONKINA, M. Distributed Schur Complement Techniques for General Sparse Linear Systems. **SIAM J. Sci. Comput.**, Philadelphia, PA, USA, v.21, n.4, p.1337–1356, 2000.

SARTORETTO, F. **Appunti per le Lezioni di Calcolo Numerico**. Veneza: Università degli Studi di Venezia, 2005.

SHEWCHUK, J. R. **Delaunay Refinement Mesh Generation**. 1997. Tese (Doutorado em Ciência da Computação) — School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania. (Technical Report CMU-CS-97-137).

SHEWCHUK, J. R. **Lecture Notes on Delaunay Mesh Generation**. Berkeley: University of California at Berkeley, 1999. (Notas de Aula). Disponível em: <http://www.cs.berkeley.edu/∼jrs/mesh/>. Acesso em: mar. 2006.

SHEWCHUK, J. R. **What is a Good Linear Element? - Interpolation, Conditioning, and Quality Measures**. Disponível em: <http://www.cs.berkeley.edu/∼jrs/papers/elem.pdf>. Acesso em: mar. 2006.

SILVA, M. Sparse matrix storage revisited. In: CONFERENCE ON COMPUTING FRONTIERS, 2., 2005, Ischia, Italy. **Proceedings...** New York: ACM, 2005. p.230–235.

SMITH, B.; BJORSTAD, P.; GROPP, W. **Domain Decomposition**: Parallel Multilevel Methods for Elliptic Partial Differential Equations. Cambridge: Cambridge University Pres, 1996.

SNIR, M.; OTTO, S.; HUSS-LEDERMANN, S.; WALKER, D.; DONGARRA, J. **MPI**: The Complete Reference. [S.l.]: MIT Press, 1996.

SONI, B. K.; THOMPSON, J. F. Mesh Generation. In: DONGARRA, J.; FOSTER, I.; FOX, G.; GROPP, W.; KENNEDY, K.; TORCZON, L.; WHITE, A. (Ed.). **Sourcebook of parallel computing**. San Francisco: Morgan Kaufmann, 2003. p.543–573.

TROTTENBERG, U.; OOSTERLEE, C. W.; SCHÜLLER, A. **Multigrid**. Oxford, UK: Academic Press, 2001. 631p. With contributions by A. Brandt, P. Oswald and K. Stüben.

VALENTIM, E.; PESSOA, L.; MELOTTI, B.; VALLI, A.; CATABRIGA, L. Comparações entre os Métodos GMRES e LCD na Implementação do Método de Newton Inexato em Problemas de Difereças Finitas. In: BRAZILIAN CONGRESS OF THERMAL ENGINEERING AND SCIENCES, ENCIT, 10., 2004, Rio de Janeiro. **Proceedings...** Rio de Janeiro: ABCM, 2004. p.2–8.

WALSHAW, C.; CROSS, M.; JOHNSON, S.; EVERETT, M. JOSTLE: Partitioning of Unstructured Meshes for Massively Parallel Machines. In: PARALLEL COMPUTATIONAL FLUID DYNAMICS: NEW ALGORITHMS AND APPLICATIONS, 1995. **Proceedings...** Amsterdam: Elsevier, 1995. p.273–280.

WEISSTEIN, E. W. **Generalized Minimal Residual Method**. Disponível em: <http://mathworld.wolfram.com/GeneralizedMinimalResidualMethod.html>. Acesso em jun. 2005.

WESSELING, P. **Introduction to Multigrid Methods**. Chichester: John Wiley & Sons, 1992.

ANEXO A FORMULAÇÃO MATEMÁTICA DOS ESTUDOS DE CASO

Neste Anexo apresenta-se a formulação dos problemas da transferência de calor e da hidrodinâmica, utilizados como estudo de caso neste trabalho.

As discretizações foram feitas através da abordagem de volumes finitos. O método dos volumes finitos foi introduzido no campo da dinâmica dos fluidos computacional na década de 70. Baseia-se na forma integral das equações que devem ser resolvidas e que são discretizadas diretamente no espaço físico. O domínio solução é subdividido em um número finito de volume de controle (VC) contíguos onde a integração é executada, e as equações de conservação são aplicadas em cada VC, sendo o centróide de cada um dos VC o ponto computacional em que as variáveis são calculadas. Interpolação é empregada para representar os valores das variáveis na superfície do VC em termos do valor do ponto computacional

A.1 Difusão de Calor Bidimensional

A equação para a difusão de calor bidimensional é dada por:

$$\frac{\partial T}{\partial t} = \mu \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) \tag{A.1}$$

onde T é a temperatura, t é o tempo, e μ é o coeficiente de difusividade térmica (m^2/s).

Integrando a equação (A.1) no espaço e no tempo, temos;

$$\int_\Omega \int_t \frac{\partial T}{\partial t} d\Omega dt = \int_\Omega \int_t \mu \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) d\Omega dt \tag{A.2}$$

ou equivalentemente;

$$\int_\Omega \int_t \frac{\partial T}{\partial t} d\Omega dt = \mu \int_\Omega \int_t \bigtriangledown \left(\bigtriangledown T \right) d\Omega dt \tag{A.3}$$

Usando o teorema da divergência de Gauss, obtém-se:

$$\int_\Omega \int_t \frac{\partial T}{\partial t} d\Omega dt = \mu \int_{\partial\Omega} \int_t \left(\bigtriangledown T \right) d\partial\Omega dt \tag{A.4}$$

Considerando que T não tem variação no espaço em um intervalo de tempo, então uma discretização para o lado esquerdo da equação (A.4) pode ser como:

$$\int_{\Omega_i} \int_t \frac{\partial T}{\partial t} d\Omega_i dt \simeq P_i \frac{(T_i^{n+1} - T_i^n)}{\Delta t} \tag{A.5}$$

onde P_i é a área do elemento i.

Uma possível aproximação para o lado direito da equação (A.5) sobre os lados do elemento i pode ser expressada como:

$$\mu \int_{\partial\Omega} \int_t (\bigtriangledown T)\, d\partial\Omega dt \simeq \mu \left((T_{i1}^n - T_i^n)\frac{\lambda_{j1}}{\delta_{j1}} + (T_{i2}^n - T_i^n)\frac{\lambda_{j2}}{\delta_{j2}} + (T_{i3}^n - T_i^n)\frac{\lambda_{j3}}{\delta_{j3}} \right) \quad \text{(A.6)}$$

onde λ_j é o tamanho do lado j, δ_j é a distância entre os centros dos elementos que compartilham o lado j e tal que os elementos ip $(p = 1, 2, 3)$ compartilham o lado jp $(p = 1, 2, 3)$ com o elemento i, como na Figura A.1.

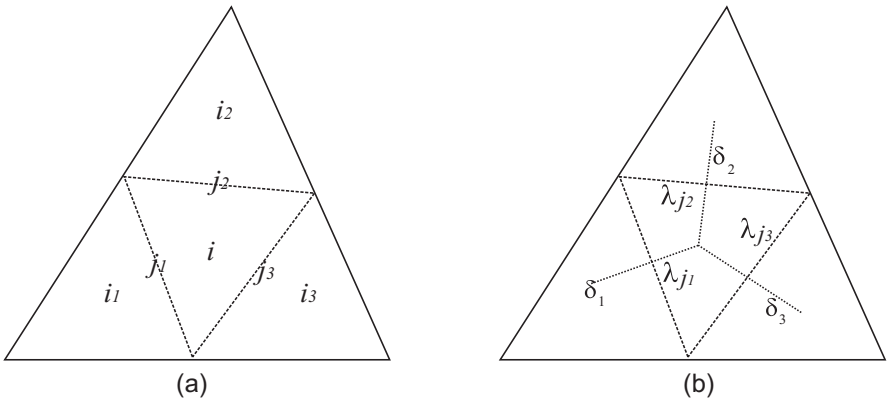


Figura A.1: Detalhe de malha. Em (a) são mostradas as notações para os triângulos vizinhos e para os lados do triângulo. Em (b) são mostradas as notações para a distância entre centros e tamanho dos lados

Usando (A.5) e (A.6), uma aproximação implícita para (A.4) é dada por:

$$P_i(T_i^{n+1} - T_i^n) = \mu\Delta t \left((T_{i1}^{n+1} - T_i^{n+1})\frac{\lambda_{j1}}{\delta_{j1}} + (T_{i2}^{n+1} - T_i^{n+1})\frac{\lambda_{j2}}{\delta_{j2}} + (T_{i3}^{n+1} - T_i^{n+1})\frac{\lambda_{j3}}{\delta_{j3}} \right)$$

Assim,

$$(T_i^{n+1} - T_i^n) = \frac{\mu\Delta t}{P_i} \left((T_{i1}^{n+1} - T_i^{n+1})\frac{\lambda_{j1}}{\delta_{j1}} + (T_{i2}^{n+1} - T_i^{n+1})\frac{\lambda_{j2}}{\delta_{j2}} + (T_{i3}^{n+1} - T_i^{n+1})\frac{\lambda_{j3}}{\delta_{j3}} \right)$$

fazendo $\frac{\mu\Delta t}{P_i} = \alpha_i$, tem-se:

$$T_i^{n+1} - \alpha_i(T_{i1}^{n+1} - T_i^{n+1})\frac{\lambda_{j1}}{\delta_{j1}} - \alpha_i(T_{i2}^{n+1} - T_i^{n+1})\frac{\lambda_{j2}}{\delta_{j2}} - \alpha_i(T_{i3}^{n+1} - T_i^{n+1})\frac{\lambda_{j3}}{\delta_{j3}} = T_i^n$$

Colocando os termos T_i^{n+1} em evidência:

$$\left[1 + \alpha_i \left(\frac{\lambda_{j1}}{\delta_{j1}} + \frac{\lambda_{j2}}{\delta_{j2}} + \frac{\lambda_{j3}}{\delta_{j3}} \right) \right] T_i^{n+1} - \alpha_i \frac{\lambda_{j1}}{\delta_{j1}} T_{i1}^{n+1} - \alpha_i \frac{\lambda_{j2}}{\delta_{j2}} T_{i2}^{n+1} - \alpha_i \frac{\lambda_{j3}}{\delta_{j3}} T_{i3}^{n+1} = T_i^n$$

Assim, pode-se definir o estêncil computacional, formado por quatro pontos A, B, C e D:

$$A = \left[1 + \alpha_i \left(\frac{\lambda_{j1}}{\delta_{j1}} + \frac{\lambda_{j2}}{\delta_{j2}} + \frac{\lambda_{j3}}{\delta_{j3}} \right) \right]$$

$$B = -\alpha_i \frac{\lambda_{j1}}{\delta_{j1}}$$

$$C = -\alpha_i \frac{\lambda_{j2}}{\delta_{j2}}$$

$$D = -\alpha_i \frac{\lambda_{j3}}{\delta_{j3}}$$

Com este estêncil, as matrizes geradas têm no máximo quatro elementos por linha, e apesar de possuir uma disposição simétrica das posições da matriz, o mesmo não ocorre para os valores.

A.2 Hidrodinâmica

O modelo matemático de escoamentos de superfície livre com densidade constante, escrito em variáveis primitivas, e que pode ser obtido a partir das equações de Navier-Stokes assumindo a decomposição de Reynolds e o escoamento como sendo hidrostático, é chamado de equações *shallow water* (ESW) (RIZZI, 2002).

As equações de superfície livre são dadas por:

$$\frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x}\left(\int_{-h}^{\eta} u dz\right) + \frac{\partial}{\partial y}\left(\int_{-h}^{\eta} v dz\right) = 0 \tag{A.7}$$

e

$$\frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x}\left(\int_{-h}^{\eta} u dz\right) + \frac{\partial}{\partial y}\left(\int_{-h}^{\eta} v dz\right) = \frac{\eta^* - \eta}{\tau_d} \tag{A.8}$$

A equação (A.7) é usada no interior do dominio, e a equação (A.8) é para o caso de fronteiras abertas *inflow* e *outflow*, como mostra a Figura A.2.

Figura A.2: Fronteiras abertas *inflow* e *outflow*

Antes de prosseguir com a discretização da EDP, é necessário introduzir algumas notações utilizadas:

- η_i^n: nível no elemento i e passo de tempo n;

- τ_d: coeficiente de sobre-relaxação obtido empiricamente;
- η^*: o nível especificado na fronteira externa adjacente as fronteiras aberta;
- Δt: passo de tempo
- $U^{*^n}(j)$: velocidade U do fluxo referente ao lado j e passo de tempo n;
- U_{ar}: velocidade do vento;
- Δz: altura da camada;
- δ_j: distância entre os centros dos triângulos que compartilham o lado j;
- λ_j: tamanho do lado j;
- P_i: área do elemento i;
- $S_{i,l}$: função sinal, ligada a direção do fluxo, onde i é o índice do triângulo e l é o índice do lado;
- r_T: coeficiente de atrito na superfície;
- $i(j,1)$ e $i(j,2)$: índice de fluxo. O fluxo sempre vai do elemento $i(j,1)$ para $i(j,2)$;
- g: gravidade;
- μ: viscosidade do meio;
- γ_{T_j}: atrito do vento na superfície;
- θ: coeficiente de implicidade do método;

Ainda descreve-se dois operadores discretos G e M, dados por:

$$G = \Delta z_j^n \left[U_j - g\frac{\Delta t}{\delta_j}(1-\theta)(\eta^n_{i(j,2)} - \eta^n_{i(j,1)}) \right] + \Delta t \gamma_{T,j} U_{ar,j}$$

$$M = \Delta z_j^n \frac{\mu \Delta t}{\Delta z_j^n} + \Delta t \gamma_{T,j} \Delta t$$

Assim, uma discretização para a equação (A.7) é como:

$$\eta_i^{n+1} = \eta_i^n + \frac{g\theta^2\Delta t^2}{P_i} \sum_{l=1}^{3} S_{i,l} \frac{\lambda_{j(i,l)}}{\delta_{j(i,l)}} \left[(\Delta z)^T M^{-1} \Delta z \right]^n_{j(i,l)} (\eta^{n+1}_{i(j,2)} - \eta^{n+1}_{i(j,1)})$$

$$-(1-\theta)\frac{\Delta t}{P_i} \sum_{l=1}^{3} S_{i,l}\lambda_{j(i,l)} \left[(\Delta z)^T U \right]^n_{j(i,l)} - \theta \frac{\Delta t}{P_i} \sum_{l=1}^{3} S_{i,l}\lambda_{j(i,l)} \left[(\Delta z)^T M^{-1} G \right]^n_{j(i,l)}$$

Assim tem-se,

$$\eta_i^{n+1} - \frac{g\theta^2\Delta t^2}{P_i} \sum_{l=1}^{3} S_{i,l} \frac{\lambda_{j(i,l)}}{\delta_{j(i,l)}} \left[(\Delta z)^T M^{-1} \Delta z \right]^n_{j(i,l)} \eta^{n+1}_{i(j,2)}$$

$$+\frac{g\theta^2\Delta t^2}{P_i}\sum_{l=1}^{3}S_{i,l}\frac{\lambda_{j(i,l)}}{\delta_{j(i,l)}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j(i,l)}\eta^{n+1}_{i(j,1)}=\eta^n_i \tag{A.9}$$

$$-(1-\theta)\frac{\Delta t}{P_i}\sum_{l=1}^{3}S_{i,l}\lambda_{j(i,l)}\left[(\Delta z)^T U\right]^n_{j(i,l)}-\theta\frac{\Delta t}{P_i}\sum_{l=1}^{3}S_{i,l}\lambda_{j(i,l)}\left[(\Delta z)^T M^{-1}G\right]^n_{j(i,l)}$$

Uma possível discretização para a equação (A.8) pode ser como:

$$(1+\frac{1}{\tau_d})\eta^{n+1}_i-\frac{g\theta^2\Delta t^2}{P_i}\sum_{l=1}^{3}S_{i,l}\frac{\lambda_{j(i,l)}}{\delta_{j(i,l)}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j(i,l)}\eta^{n+1}_{i(j,2)}+$$

$$+\frac{g\theta^2\Delta t^2}{P_i}\sum_{l=1}^{3}S_{i,l}\frac{\lambda_{j(i,l)}}{\delta_{j(i,l)}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j(i,l)}\eta^{n+1}_{i(j,1)}=\frac{\Delta t\eta^*}{\tau_d} \tag{A.10}$$

$$+(1+\frac{1}{\tau_d})\eta^n_i-(1-\theta)\frac{\Delta t}{P_i}\sum_{l=1}^{3}S_{i,l}\lambda_{j(i,l)}\left[(\Delta z)^T U\right]^n_{j(i,l)}-\theta\frac{\Delta t}{P_i}\sum_{l=1}^{3}S_{i,l}\lambda_{j(i,l)}\left[(\Delta z)^T M^{-1}G\right]^n_{j(i,l)}$$

onde, diferentemente de A.9, a expressão A.10 agrega as condições de contorno aberto do tipo Blumberg-Khanta (BLUMBERG; KANTHA, 1985).

Dessa forma, podemos definir os estênceis de 4-pontos para a hidrodinâmica, da mesma forma que fora definido para o problema da transferência de calor. Para os casos dos elementos internos tem-se:

$$A=1+\frac{g\theta^2\Delta t^2}{P_i}\frac{\lambda_{j1}}{\delta_{j1}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j1}+\frac{g\theta^2\Delta t^2}{P_i}\frac{\lambda_{j2}}{\delta_{j2}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j2}$$

$$+\frac{g\theta^2\Delta t^2}{P_i}\frac{\lambda_{j3}}{\delta_{j3}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j3}$$

$$B=-\frac{g\theta^2\Delta t^2}{P_i}\frac{\lambda_{j1}}{\delta_{j1}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j1}$$

$$C=-\frac{g\theta^2\Delta t^2}{P_i}\frac{\lambda_{j2}}{\delta_{j2}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j2}$$

$$D=-\frac{g\theta^2\Delta t^2}{P_i}\frac{\lambda_{j3}}{\delta_{j3}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j3}$$

Também é necessário a definição da contribuição para o vetor dos termos independentes b:

$$b_i=\eta^n_i-(1-\theta)\frac{\Delta t}{P_i}\sum_{l=1}^{3}S_{i,l}\lambda_{j(i,l)}\left[(\Delta z)^T U\right]^n_{j(i,l)}-\theta\frac{\Delta t}{P_i}\sum_{l=1}^{3}S_{i,l}\lambda_{j(i,l)}\left[(\Delta z)^T M^{-1}G\right]^n_{j(i,l)}$$

Da mesma forma, define-se o estêncil para os elementos da fronteira. Na fronteira tem-se dois casos distintos, fronteira aberta (*inflow* e *outflow*) e fronteira fechada.

Para o primeiro caso, fronteira aberta, supõe-se que o lado j_3 do triângulo i seja adjacente à fronteira, então obtém-se:

$$A=1+\frac{1}{\tau_d}+\frac{g\theta^2\Delta t^2}{P_i}\frac{\lambda_{j1}}{\delta_{j1}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j1}+\frac{g\theta^2\Delta t^2}{P_i}\frac{\lambda_{j2}}{\delta_{j2}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j2}$$

$$+\frac{g\theta^2\Delta t^2}{P_i}\frac{\lambda_{j3}}{\delta_{j3}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j3}$$

$$B = -\frac{g\theta^2\Delta t^2}{P_i}\frac{\lambda_{j1}}{\delta_{j1}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j1}$$

$$C = -\frac{g\theta^2\Delta t^2}{P_i}\frac{\lambda_{j2}}{\delta_{j2}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j2}$$

Note que não existe a contribuição da fronteira para a matriz, logo o estêncil possui apenas 3 pontos. Esta contribuição da fronteira é passada para o vetor dos termos independentes b, como segue:

$$b_i = \frac{1}{\tau_d}\eta_i^n + \eta_i^n - (1-\theta)\frac{\Delta t}{P_i}\sum_{l=1}^{3} S_{i,l}\lambda_{j(i,l)}\left[(\Delta z)^T U\right]^n_{j(i,l)}$$

$$-\theta\frac{\Delta t}{P_i}\sum_{l=1}^{3} S_{i,l}\lambda_{j(i,l)}\left[(\Delta z)^T M^{-1} G\right]^n_{j(i,l)}$$

$$-\frac{g\theta^2\Delta t^2}{P_i}S_{i,3}\frac{\lambda_{j(i,3)}}{\delta_{j(i,3)}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j(i,3)}\eta^* + \frac{\Delta t\eta^*}{\tau_d}$$

sendo que η^* recebe o mesmo valor do nível no centro da célula adjacente à fronteira.

Para o caso em que a fronteira é fechada, o estêncil é dado por:

$$A := 1 + \frac{1}{\tau_d} + \frac{g\theta^2\Delta t^2}{P_i}\frac{\lambda_{j1}}{\delta_{j1}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j1} + \frac{g\theta^2\Delta t^2}{P_i}\frac{\lambda_{j2}}{\delta_{j2}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j2}$$

$$B = -\frac{g\theta^2\Delta t^2}{P_i}\frac{\lambda_{j1}}{\delta_{j1}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j1}$$

$$C = -\frac{g\theta^2\Delta t^2}{P_i}\frac{\lambda_{j2}}{\delta_{j2}}\left[(\Delta z)^T M^{-1}\Delta z\right]^n_{j2}$$

Note que para o ponto A do estêncil, não há a contribuição da fronteira, já que a mesma é fechada. Da mesma forma, o vetor b não recebe nenhuma contribuição:

$$b_i = \eta_i^n - (1-\theta)\frac{\Delta t}{P_i}\sum_{l=1}^{3} S_{i,l}\lambda_{j(i,l)}\left[(\Delta z)^T U\right]^n_{j(i,l)} - \theta\frac{\Delta t}{P_i}\sum_{l=1}^{3} S_{i,l}\lambda_{j(i,l)}\left[(\Delta z)^T M^{-1} G\right]^n_{j(i,l)}$$

Assim, dado os diferentes tipos de estênceis utilizados na hidrodinâmica, aplica-se o estêncil apropriado a cada elemento da malha. Os sistemas gerados com esta discretização, assim como no caso da difusão de calor, são não simétricos.

ANEXO B FORMATO DE ARQUIVOS DE ENTRADA E SAÍDA NA GERAÇÃO DE MALHAS

Este anexo tem como objetivo mostrar exemplos de arquivos de entrada e saída utilizados no processo de geração de malhas.

Como dado inicial para a geração das malhas têm-se um arquivo de texto contendo um PSLG. Um exemplo para um PSLG de um domínio quadrado é dado na Figura B.1.

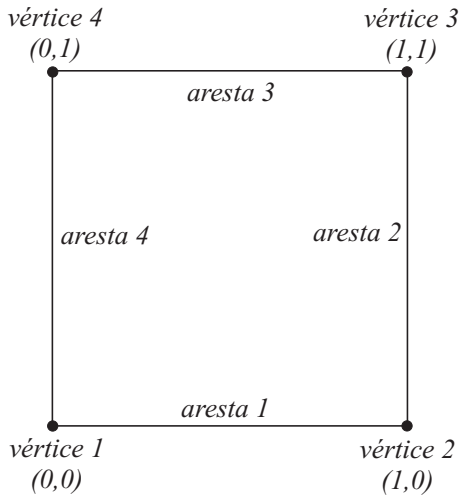


Figura B.1: PSLG

O arquivo de entrada para o gerador de malha, relacionado ao exemplo da figura acima, seria como:

```
4 #quantidade de vertices

# X    Y  -> coordenada dos vertices
  0    0  # -> vertice 1
  1    0  # -> vertice 2
  1    1  # -> vertice 3
  0    1  # -> vertice 4

4 #quantidade de segmentos

# V1   V2
```

```
1    2  # -> vertices que formam o segmento 1
2    3  # -> vertices que formam o segmento 2
3    4  # -> vertices que formam o segmento 3
4    1  # -> vertices que formam o segmento 4
```

Submetendo o arquivo ao gerador de malha, obtém-se a malha mostrada na Figura B.2. Dessa forma define-se os arquivos de saída, como apresentados a seguir.

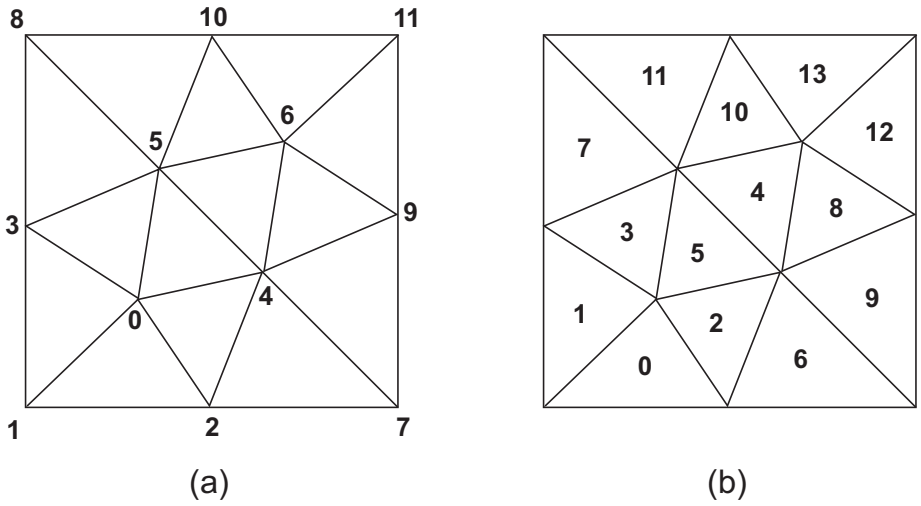


Figura B.2: Malha exemplo. Em (a) a numeração dos vértices, em (b) a numeração dos triângulos

O arquivo *exemplo.n* define as coordenadas dos vértices da malha. Neste arquivo, a primeira linha contém a quantidade de vértices n, seguidos por n linhas contendo as coordenadas dos mesmos.

```
#exemplo.n -> arquivo de vertices
12 # -> quantidade de vertices
0.35714 0.35714
0.00000 0.00000
0.50000 0.00000
0.00000 0.50000
0.70000 0.30000
0.30000 0.70000
0.64286 0.64286
1.00000 0.00000
0.00000 1.00000
1.00000 0.50000
0.50000 1.00000
1.00000 1.00000
```

O arquivo *exemplo.e* define a conectividade dos triângulos, ou seja, por quais vértices um triângulo é formado. Neste arquivo, a primeira linha contém a quantidade de triangulos t, seguidos por t linhas contendo os três vértices que formam o t-ésimo elemento. Note que a numeração inicia-se em 0 (zero).

```
#exemplo.e -> arquivo de conectividade dos elementos
14 # -> quantidade de triangulos
   0    1    2
   0    3    1
   0    2    4
   0    5    3
   6    0    4
   6    5    0
   4    2    7
   5    8    3
   4    9    6
   4    7    9
   6   10    5
   5   10    8
   6    9   11
   6   11   10
```

O arquivo *exemplo.v* define a vizinhança de um elemento da malha. Este arquivo é formado por t linhas, contendo os três vizinhos do t-ésimo elemento. A presença de "-1" significa que o elemento está na fronteira do domínio.

```
#exemplo.v -> arquivo de vizinhança dos triângulos
  -1    2    1
  -1    0    3
   6    4    0
   7    1    5
   2    8    5
   3    4   10
  -1    9    2
  -1    3   11
  12    4    9
  -1    8    6
  11    5   13
  -1    7   10
  -1   13    8
  -1   10   12
```

A partir destes arquivos pode-se encontrar todas as informações da malha, tais como a área, tamanho das arestas e centro. Tais informações são utilizadas na geração da hierarquia de malhas e na geração dos sistemas de equações.

ANEXO C PUBLICAÇÕES

Neste anexo encontram-se as publicações desenvolvidas ao longo deste trabalho.

C.1 Publicações Aceitas

- "UnHIDRA: Um Modelo Computacional Paralelo Multifísica". WSGPPD 2004, Porto Alegre-RS.
- "Comparação entre Métodos de Decomposição de Domínio e Decomposição de Dados na Solução de Sistemas de Equações". WSCAD 2004, Foz do Iguaçú-PR.
- "Geração de Malhas e Métodos de Solução no Modelo UnHIDRA". ERAD 2005, Canoas-RS.
- "Paralelização de Métodos Multgrid para Solução de Sistemas de Equações em Clusters de PCs", WSGPPD 2005, Porto Alegre-RS.
- "Solução Paralela de Sistemas de Equações através de Métodos Multigrid", ERAD 2006, Ijuí-RS.
- "A Multigrid-DDM Schur Elliptic Equation Solver in Unstructured Meshes". VECPAR 2006, Rio de Janeiro-RJ.
- "A Multigrid-Schwarz for the Solution of Elliptic Equations in Unstructured Meshes". PARA'06, Umea, Suécia.

C.2 Publicações Submetidas

- "Parallel Multigrid Solver in Unstructured Meshes: A Domain Decomposition Approach". SBAC-PAD 2005, Rio de Janeiro-RJ.
- "Hydrodynamics of the Guaiba River simulated for the HIDRA: data decomposition versus domain decomposition". Parallel Computing, Elsevier Science Publishers (em avaliação).

A Multigrid-DDM Schur Elliptic Equation Solver in Unstructured Meshes

Guilherme Galante[1], Rogerio L. Rizzi[2] and Tiaraju A. Diverio[1]

[1] PPGC, Instituto de Informática
Universidade Federal do Rio Grande do Sul
CP 15064, 91501-970, Porto Alegre, RS, Brazil
[2] Centro de Ciências Exatas e Tecnológicas
Universidade Estadual do Oeste do Paraná
Rua Universitária, 2069, 85801-110, Cascavel, PR, Brazil
ggalante@inf.ufrgs.br

Abstract. This work shows the parallel Multigrid-MDD Schur method for the solution of elliptic equations. In the proposed method the solution is obtained by a multigrid method parallelized by domain decomposition techniques, more specifically by the Schur complement method. It is also shown some issues related to the generation and partitioning of the mesh hierarchy. In the case study, we used the 2D heat diffusion equation in unstructured meshes. The implementations was developed to explore parallelism in clusters, using message passing.

1 Introduction

The large linear equations systems, that raises of the discretization of partial differential equations (PDE) in technological and scientific problems, require efficient solution. The use of direct methods is inadequate to solve these systems, once a time that do not use the advantage of the coefficients sparsity, making this approach difficult, by storage problems and for the dependence of operations that difficult its parallelization.

The iterative algorithms, however, use only the matrix as operator to iteratively construct a convergent sequence of solutions. And, in contrast of the direct methods, are very used in the resolution of sparses and large equations systems, due its storage optimization potential and computational efficiency. Currently parallel solutions by iterative methods, combine preconditioned Krylov sub-spaces methods as local solver, with domain decomposition methods [1][16].

Other approach used in the equation systems resolution are the multigrid methods. Multigrid methods originated in the 1960s with the work of Fedorenko and Bakhvalov. They were further developed in the 1970s by Brandt, and are now the preferred methods for solving elliptic partial differential equations [19]. The advantage of multigrid is the speed - multigrid algorithms only require order N operations to solve elliptic equations, where N is the number of mesh points.

[1] Candidate to the best student paper award

Since 1980, the number of publications related to the multigrid methods had increased substantially, and currently there are more than 3600 references that can be found in the MGNET (*www.mgnet.org*), who is the official repository of multigrid methods [9].

In this paper, we describe our solver, MG-Schur, that solves elliptic equations using multigrid methods parallelized by domain decomposition methods. The idea of combining multigrid and domain decomposition methods is not new [3][8], although there are a number of features of our code that distinguish it from the discussions we have seen. In particular, we use the Full Multigrid V cycle combinated with the Schur complement domain decomposition method to explore the parallelism. Moreover, our method uses only Krylov space iterative methods, instead classical iterative solvers, like Gauss-Seidel, normally used in the multigrid approaches.

This paper is structured as follows: in section 2 generation and the partitioning of meshes are discussed, and a tool *MGTool* is presented; in section 3 a overview of multigrid is presented; in section 4 the proposed parallel multigrid method is presented; in section 5 the study case is described; the section 6 summarizes the results obtained and in section 7 some conclusions and future works are presented.

2 Mesh Generation and Partitioning

In general, problems that use simulation are based on mathematical models that are expressed through Partial Differential Equations (PDE). This PDE, generally, does not have known analytical solution, being necessary the use of discretization and approximation methods, as finite volume or finite element, so that they can be numerically solved.

With the use of numerical techniques of solution of PDE, the region of the domain Ω is not treated as continuous, but like a discrete and finite set of points or subdomains in which the variables are calculated. This discrete set of points or subdomains constitutes the mesh.

In this work the domain discretization is made using unstructured triangular meshes. Unstructured meshes conciliate good representation of the computational domain, since diverse problems are defined in domains with irregular geometry that not always are appropriately discretized by structured meshes. More, specifically we used a special type of meshes, called unstructured orthogonal mesh. It is assumed that within each triangle, there exists a point (hereafter called center) such that the segment joining the centers of two adjacent triangles and the side shared by the two triangles have a nonempty intersection and are orthogonal to each other [6]. The use of this type of mesh simplifies some issues in the PDE discretization, when using finite volumes method.

In the multigrid methods, is necessary create a hierarchy of meshes. In contrast of the work of Chan and Smith [7], we consider the multilevel mesh hierarchy generation starting from a coarse mesh.

The initial coarse mesh is generated by the software *Easymesh* [13]. The *Easymesh* is a program that generates two dimensional, unstructured, Delaunay and constrained Delaunay triangulations in general domains. The mesh quality is achieved by the use of smoothing algorithms.

Once the coarser mesh was generated, we can create the mesh hierarchy. For the generation of the mesh levels we have implemented a tool called *MGTool*. The tool takes the data generated by *Easymesh* and generates the mesh hierarchy, as shown in Fig. 1.

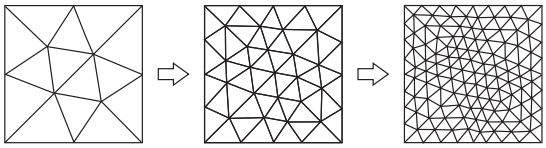
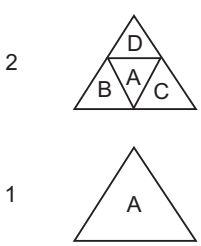


Fig. 1. Example of 3-level mesh hierarchy.

For the refinement of the meshes we adopted a strategy known in literature as h-refinement, characterized for the subdivision of the cells of the domain [10], and is similar to creation of the mesh hierarchy in structured meshes. Refined meshes are created through successive subdivisions of the triangles of the coarse mesh in four subtriangles. In the example shown in Fig. 2, the triangle 1A is refined into four triangles, 2A, 2B, 2C and 2D. Then, the triangle 1A is the "parent" of the triangles 2A, 2B, 2C and 2D, and the triangles 2A, 2B, 2C and 2D are the "children" of triangle 1A.



Cell	Parent	Children			
2A	1A	-	-	-	-
2B	1A	-	-	-	-
2C	1A	-	-	-	-
2D	1A	-	-	-	-

Cell	Parent	Children			
1A	-	2A	2B	2C	2D

Fig. 2. Example of a 2-level mesh hierarchy. The numbers on the left denote the resolution level, and the letters label triangles of the mesh. On the right side, the table that describes the relationship between adjacent levels.

Besides generating the mesh hierarchy, the *MGTool* generate the matrices to each mesh level and also creates information necessary to the interpolation and restriction operators, explained later in Section 3. This informations describes the relationship between adjacent levels, indicating how the data will be transfered from the "parents" to the "children", and vice versa.

To solve the problems in parallel, is necessary that the mesh be partitioned and distributed among the available processors. The partitioning must distribute the workload to each processor in a proportional way. Moreover, due the necessity of information exchange, the partitioning must be made in order to reduce the boundaries among subdomains, and consequently the communications among the processors, since the information exchange is restricted to the boundaries [14]. For the mesh partitioning *MGTool* uses the *METIS* package. The *METIS* is a software package especially developed to large graphs and meshes partitioning [12]. In the Fig. 3 is shown a example of domain partitioned by *METIS*.

Fig. 3. Rectangular domain partitioned in 20 subdomains with METIS.

The *MGTool* performs the partitioning in the coarser mesh and replicates to the other levels. This approach prevents the load unbalancing, and is advantageous for simulations that run on a large number of processors [8].

3 Multigrid

Multigrid methods is a group of algorithms and techniques that efficiently solve equations systems through the acceleration of the convergence of iterative methods. Basically, the methods multigrid consider a sequence of meshes for the solution of equations system.

To solve a equation system with a multigrid methods in a refined mesh of size N, we introduce meshes of size $N/4$, $N/16$, etc., covering the computational domain. On each grid an associated equation is solved. The solution on the coarse meshes quickly captures the long wavelength features of the solution, and solving on the fine mesh captures the short wavelength features.

The multigrid algorithms are based on three central ideas [18]:

1. communication among meshes;
2. nested iterations;
3. and coarse mesh correction.

A meshes in the multigrid hierarchy communicates with the adjacent ones through restriction and interpolation operators. Restriction takes data on a mesh and restricts it to the next coarsest mesh, defined by:

$$I_N^{N/4} : N \rightarrow N/4$$

Interpolation takes data on a mesh and interpolates it onto the next finest mesh. The operator is defined by:

$$I_{N/4}^{N} : N/4 \rightarrow N$$

In the technique of nested iterations, the objective is find a better initial guess for the solution using iterations in coarse meshes. The coarse meshes have a lower number of variables, and consequently, the computational cost of the iteration is reduced, in relation to an iteration in the most refined mesh. Then, one better initial guess for $Ax = b$ can be given using the coarsest mesh. An example is show in Fig. 4

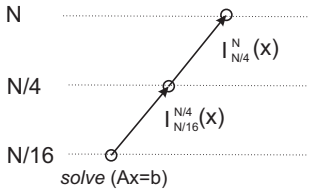


Fig. 4. Nested iterations. In this example, the process starts in the $N/16$ mesh that obtains a initial guess to $N/4$, and $N/4$ compute a initial guess to N.

This strategy does not guarantee that the solution solution in N does not contains soft error components (low frequency). The usage of the correction of error in the coarse meshes prevents this limitation. Iterating in the fine mesh until the errors have been removed, the residual equation is iterated equation in the coarse mesh to get an approximation of the error. Then, is interpolated to the fine mesh, where the first guess is corrected. The correction using the residual equation to iterate the error is shown in Fig. 5.

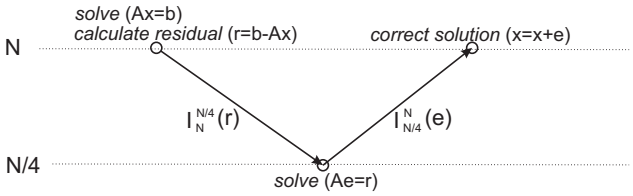


Fig. 5. Coarse grid correction.

Using combinations between the strategy of coarse mesh correction and the use of nested iterations, a family of multigrid methods can be defined [19].

3.1 Full Multigrid V

By the distinct combinations is possible to generate different algorithms that are known in technique literature. A summary of these different approaches can be found in [18][19]. In this work the *full multigrid V*, or simply FMV, is used.

The FMV strategy initiates in the coarsest mesh, for the acquisition of an initial solution with low computational cost for the superior levels. Then, the levels number is incremented by one, and a coarse mesh correction is performed. This process is repeated until all the levels are involved. In the Figure 6, the FMV scheme is shown, where the black points represents the operations (linear system solution, residual and error calculations), and the arrows represents the transference of information between the mesh levels.

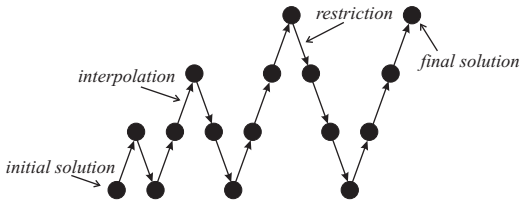


Fig. 6. Full multigrid V.

The theoretical results and numerical experiments show that these methods are efficient and can be applied to an huge types of problems on the scope of the scientific computing. The literature shows the generality of the method, being possible to use it in distinct types of meshes, as well the different discretization methods [4].

4 Parallel MG-Schur

The proposed method uses the domain decomposition approach to explore the parallelism em clusters. Domain decomposition methods (DDM) denotes a set of mathematical, numerical and computational methods and techniques to solve problems in parallel computers.

A DDM is characterized by the division of the computational domain, that is partitioned in subdomains using partitioning algorithms. The global solution of the problem is obtained by the combination of subproblems that are locally solved. Each processor is responsible for finding the local solution of one or more subdomains [17]. In parallel solutions by domain decomposition, the subdomains can be almost treated independently. Therefore, such methods are attractive for distributed memory environments.

4.1 DDM Parallel Solver

In this work we used the Schur complement method. In the Schur complement method, the domain is partitioned and the the cells are reordered such that the interface points are listed last after the interior points.

With this local ordering, each local vector of unknows x_i is split into two parts: the subvector u_i of internal vector components followed by the subvector y_i of local interface vector components. The right-hand side b_i is conformally split into the subvectors f_i and g_i. When partitioned according to this splitting, the local matrix A_i, residing in the processor i has the form

$$A_i = \begin{bmatrix} B_i & E_i \\ F_i & C_i \end{bmatrix},$$

so the local equations can be written as:

$$\begin{aligned} B_i u_i + E_i y_i &= f_i \\ F_i u_i + C_i y_i + \sum_{j \in N_i} E_{ij} y_j &= g_i \end{aligned}$$

where N_i is the set of indices for subdomains that are the neighbors to the subdomain i, and the term $E_{ij} y_i$ the contribution to the local equation from the neighboring subdomain j.

Eliminating the variable u in the first equation:

$$u_i = B_i^{-1}(f_i - E_i y_i) \tag{1}$$

and substituting u in the second equation:

$$S_i y_i + \sum_{j \in N_i} E_{ij} y_j = g_i - F_i B_i^{-1} f_i \tag{2}$$

where $S_i = C_i - F_i B_i^{-1} E_i$, and is the local Schur complement.

With this formulation, each processor needs to know the processors with which it must communicate and the list of interface points. Thus, the solution y_i in the interface is obtained, and are used to find the internal variables u_i. For more details about the Schur complement method, see [15].

The Schur complement method is used as solver in all the linear systems in the FMV. Each local matrix is allocated to a processor, and in particular, each processor needs to know the processors with which must communicate and the list of interface points.

4.2 FMV operations

The other operations required in the FMV method is now discussed. As well the solver, the restriction, interpolation and residual operations are also calculated in parallel.

For the data transfer operators, we choose $I_N^{N/4}$ to be the full-weighted restriction operator [18], and the interpolation $I_{N/4}^N$ is given by $x_N = \frac{1}{4}\sum_{i=1}^{4} x_{N/4}$, where $x_{N/4}$ is the corresponding values in the coarse mesh. In this operations, each processor is responsible only for its domain. None communication is necessary because no external data are needed.

The residual calculation is necessary in the coarse mesh correction. The residual equation is given by:

$$r_i = b_i - A_i x_i$$

however, as the matrix and the vectors was splitted to the Schur complement method, the residual vector must be splitted in two parts t_i and v_i, where t_i is related to internal components and v_i is related to interface components. Thus, the residual equation must be written as follow:

$$\begin{aligned} t_i &= f_i - B_i u_i + E_i y_i \\ v_i &= g_i - F_i u_i + C_i y_i + \sum_{j \in N_i} E_{ij} y_j \end{aligned}$$

As seen before, the term $E_{ij}y_j$ is the contribution of the neighboring subdomains, therefore, the domain must communicates with its neighbors to receive the required data.

5 Study Case: heat diffusion equation

For numerical and computational experiments, it was used the discretized heat diffusion PDE through the finite volume method under triangular unstructured meshes. The finite volume method subdivides the computational domain in not overlapped subdomains the called finite volumes, that here are the triangles of the mesh.

The finite volume method assures the method quality solution basing the approaches in the conservation principles. For example, the principle of the mass conservation affirms that the mass cannot be created nor be destroyed; if the flow for inside of a region exceeding that one that leaves, the mass will be accumulated inside of this. This approach used here can be implemented by integration of the differential equations, assuring local conservation and, consequently, global [2].

Considering the equation:

$$\frac{\partial T}{\partial t} = \mu \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) \tag{3}$$

that model the temperature diffusion, where μ is the diffusion constant.

Integrating the equation (3) in time and space:

$$\int_\Omega \int_t \frac{\partial T}{\partial t} d\Omega dt = \int_\Omega \int_t \mu \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) d\Omega dt \tag{4}$$

or equivalently

$$\int_{\Omega}\int_{t}\frac{\partial T}{\partial t}d\Omega dt = \mu\int_{\Omega}\int_{t}\bigtriangledown\left(\bigtriangledown T\right)d\Omega dt \tag{5}$$

Using the Gauss divergence theorem:

$$\int_{\Omega}\int_{t}\frac{\partial T}{\partial t}d\Omega dt = \mu\int_{\partial\Omega}\int_{t}\left(\bigtriangledown T\right)d\partial\Omega dt \tag{6}$$

Considering T does not vary in the space, so a discretization to the left hand side of equation(6) is:

$$\int_{\Omega_i}\int_{t}\frac{\partial T}{\partial t}d\Omega_i dt \simeq P_i\frac{(T_i^{n+1}-T_i^n)}{\Delta t} \tag{7}$$

where P_i is the area of $i-th$ triangle.

Approximating the right hand size of (6) using the sides of the $i-th$ triangle:

$$\mu\int_{\partial\Omega}\int_{t}\left(\bigtriangledown T\right)d\partial\Omega dt \simeq \mu\left((T_{i1}^n-T_i^n)\frac{\lambda_{j1}}{\delta_{j1}}+(T_{i2}^n-T_i^n)\frac{\lambda_{j2}}{\delta_{j2}}+(T_{i3}^n-T_i^n)\frac{\lambda_{j3}}{\delta_{j3}}\right) \tag{8}$$

where λ_j is the size of $j-th$ side, δ_j is the distance between the centers (as defined in Section 2) of the triangle that share the $j-th$ side and the triangles i_p $(p=1,2,3)$ share the side j_p $(p=1,2,3)$ with the element i, how we can observe in Fig. 7.

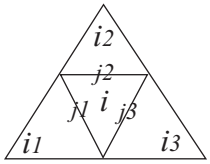


Fig. 7. Triangle and triangle side notations

Using (7) and (8), a implicit approximation to (6) can be written by:

$$P_i(T_i^{n+1}-T_i^n) = \mu\Delta t\left((T_{i1}^{n+1}-T_i^{n+1})\frac{\lambda_{j1}}{\delta_{j1}}+(T_{i2}^{n+1}-T_i^{n+1})\frac{\lambda_{j2}}{\delta_{j2}}+(T_{i3}^{n+1}-T_i^{n+1})\frac{\lambda_{j3}}{\delta_{j3}}\right) \tag{9}$$

Thus,

$$(T_i^{n+1}-T_i^n) = \frac{\mu\Delta t}{P_i}\left((T_{i1}^{n+1}-T_i^{n+1})\frac{\lambda_{j1}}{\delta_{j1}}+(T_{i2}^{n+1}-T_i^{n+1})\frac{\lambda_{j2}}{\delta_{j2}}+(T_{i3}^{n+1}-T_i^{n+1})\frac{\lambda_{j3}}{\delta_{j3}}\right) \tag{10}$$

considering $\frac{\mu\Delta t}{P_i} = w_i$:

$$T_i^{n+1}-w_i(T_{i1}^{n+1}-T_i^{n+1})\frac{\lambda_{j1}}{\delta_{j1}}-w_i(T_{i2}^{n+1}-T_i^{n+1})\frac{\lambda_{j2}}{\delta_{j2}}-w_i(T_{i3}^{n+1}-T_i^{n+1})\frac{\lambda_{j3}}{\delta_{j3}} = T_i^n \tag{11}$$

Isolating the T_i^{n+1} therm, the results above can be written in the matrix form as:

$$\left[1 + w_i \left(\frac{\lambda_{j1}}{\delta_{j1}} + \frac{\lambda_{j2}}{\delta_{j2}} + \frac{\lambda_{j3}}{\delta_{j3}}\right)\right] \cdot T_i^{n+1} - w_i \frac{\lambda_{j1}}{\delta_{j1}} T_{i1}^{n+1} - w_i \frac{\lambda_{j2}}{\delta_{j2}} T_{i2}^{n+1} - w_i \frac{\lambda_{j3}}{\delta_{j3}} T_{i3}^{n+1} = T_i^n \tag{12}$$

The assembly of the matrix, is done using the mesh information of the entire domain. Each internal triangle of the domain generates one row of the coefficient matrix, where the number of non null terms depends on the number of neighbor cells of the triangle corresponding to that row of the matrix. The matrix generated is sparse, with a maximum of 4 elements per row, and stored in CSR format. As the local matrices are sparse and non-symmetric, we used the MG-Schur with GMRES(m), with $m = 5$.

The solution of the heat transfer equation was obtained for a square domain, with $1m^2$ and $\mu = 0.1$. This domain was discretized in a 4 level mesh hierarchy generated by *MGTool*. The mesh levels contain 1706, 6824, 27296 and 109184 triangles, respectively. In this test we used constant boundaries conditions on the edges of the square (0^oC), and two triangles, in opposite corners, with 100^oC as initial value ($t = 0$).

6 Results

The implementations of this paper are being developed to explore parallelism in clusters. Conceptually, cluster is a collection of computers (workstations, personal computers or SMPs), called nodes, which are used exclusively for achieve high performance [5]. These machines are physically interconnected by a local network or a high performance network. The use of this architecture has a significant increase in the last years due, mainly, the low cost and the system scalability.

The implemented algorithms were run in the cluster of the Laboratory of Technology in Clusters (LabTeC) of UFRGS Computer Science Institute, developed in association with the Dell Computers. The cluster labtec is formed by 21 nodes, where 20 are dedicated exclusively for processing and one server node. The interconnection of the processing nodes is made through one Gigabit Ethernet switch. Each processing node is a Dual Pentium III 1,1 GHz, with 1 GB of RAM memory, 512 KB of cache and 18 GB SCSI hard disk; the server node is a Dual Pentium IV Xeon 1,8 GHz, with 1 GB of memory RAM and 36 GB SCSI hard disk.

In this type of architecture parallel programming is usually explicit, requiring complete control over implementation strategies and over the implementation itself, and, in this context, parallelism is obtained through the division-conquer paradigm. From the programming point of view, the SPMD (Single Program Multiple Data) paradigm was used. For the development of parallel applications in distributed memory machines, like clusters, is necessary the use of an messages exchange library. All the algorithms were implemented using C programming

language and the MPICH 1.2.5 message passing library [11] over Linux operating system.

The MG-Schur method was tested with 1, 2, 4, 8, 12, 16 and 20 processes. We also compare the MG-Schur to a Schur complement method without multigrid. The figures 8, 9 and 10 show execution time, speedup and computational efficiency obtained, respectively. These results were obtained without considering the opening and reading of input data and considering 15 cycles, where each cycle is composed by the maximum number of iterations necessary to reach the desired accuracy. In the tests the error is 10^{-5}.

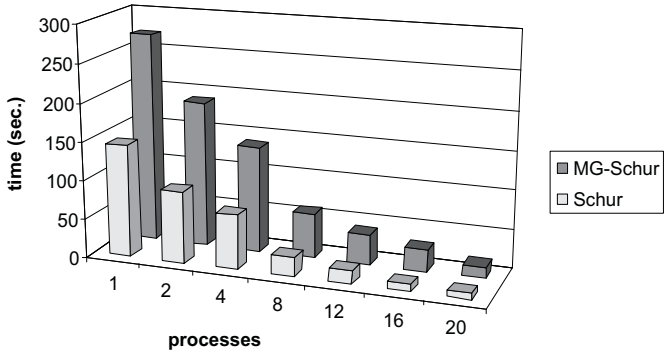


Fig. 8. Execution time x number of processes.

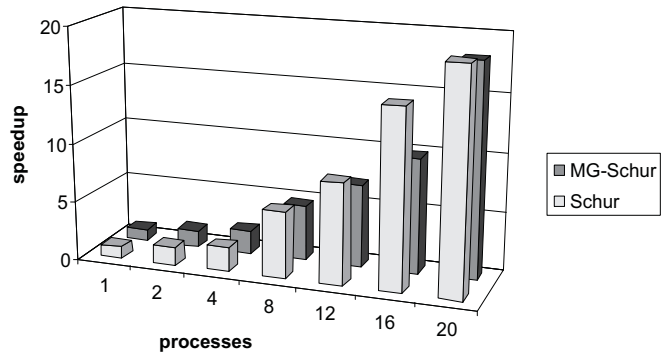


Fig. 9. Speedup x number of processes.

Accordly to the figures, we can observe that both tested methods are scalable but the MG-Schur is almost twice faster than Schur method.

It should be noted, that the efficiency decreases when using 4, 8 and 12 processes. This fact occurs probably due to the solution of the large Schur com-

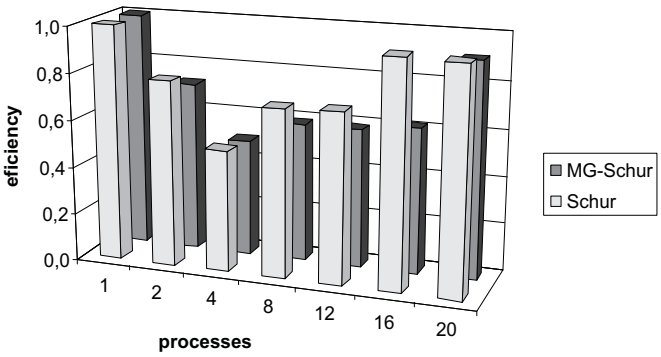


Fig. 10. Efficiency x number of processes.

plement interface systems, that are generated when the domain partitioning presents large artificial boundaries among the subdomains.

In the Fig. 11 are presented 3 steps of the solution obtained with MG-Schur with 20 processes.

As the simulation runs, heat is transmitted from warmer parts of the domain to cooler parts of the domain, and gradually converging to a stable state. The presented solution is continuous among the subdomains, showing a effective parallel solution.

7 Conclusion and Future Work

In this work we presented a parallel implementation of a multigrid method, using Schur complement domain decomposition method. The numerical results obtained when running the solver in several processors are consistent with the results obtained running it in one processor, and the difference between both results are due to the accuracy desired.

The experiments performed have shown that the proposed implementation has shown to be computationally efficient, good scalability, and good numerical quality.

As future work, three important issues will subject of more research: implementation of a flexible multigrid algorithm to allow the solution by diverse types of multigrid cycles, parallelization of the multigrid by overlapping domain decomposition, and adapt *MG-Tool* to the solution of the hydrodynamics of UnHidra model. The UnHidra is a multi-physics parallel computational model for the simulation of substance transport and for the 2D and 3D hydrodynamic flow in water bodies, using unstructured meshes.

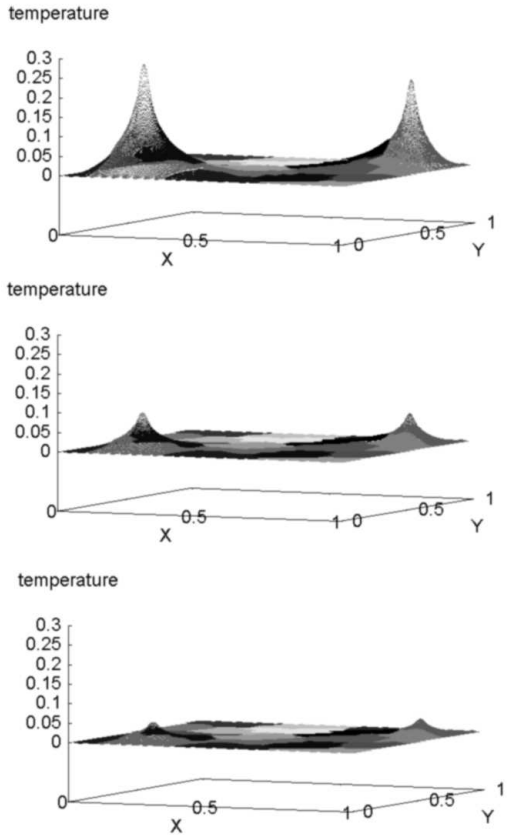


Fig. 11. 3-steps solution of the heat diffusion equation. From top to bottom, $t = 3$, $t = 6$ and $t = 12$

8 Acknowledgments

Thanks to CAPES for financed the master thesis related to this work. Some issues of this research are included to the UnHIDRA project (CT-HIDRO, Processo 502858/2003-6, Edital MCT/CNPq/CT-HIDRO 01/2003).

References

1. S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.
2. T. Barth. Numerical methods and error estimation for conservation laws on structured and unstructured meshes. Von Karman Institute, LS04-2003, 2003, pp 1-65, 2003.

3. P. Bastian, W. Hackbusch, and G. Wittum. Additive and multiplicative multi-grid: a comparison. Computing, 60:345364, 1998.
4. W. Briggs. *A Multigrid Tutorial*. SIAM, Philadelphia, 1987.
5. R. Buyya. *High Performance Cluster Computing: Architecture and Systems*, volume 1. Prentice Hall, 1999.
6. V. Casulli and R. Walters A. An Unstructured Grid, Three-Dimensional Model based on the Shallow Water Equations. International journal for numerical methods in fluids, v. 3, p. 331-348. 2000.
7. T. F. Chan and B. F. Smith. Domain decomposition and multigrid algorithms for elliptic problems on unstructured meshes. Electronic Transactions on Numerical Analysis. Volume 2, pp. 171-182, December 1994.
8. E. Chow, R. Falgout, J. Hu, R. Tuminaro and U. Meier Yang. A Survey of Parallelization Techniques for Multigrid Solvers to appear in Frontiers of Parallel Processing For Scientific Computing, SIAM book series , 2005.
9. C. C. Douglas. MGNet: a multigrid and domain decomposition network. *http://www.mgnet.org*.
10. M. Filipiak. Mesh Generation. [S.l.]: EPCC, Edinburgh, 1996. Watch Report.
11. W. Groop, et al. A High Performance, Portable Implementation of the MPI Message Passing Interface Standard. Parallel Computer, v.22, n.6, p.789-828, Sep. 1996.
12. G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.
13. B. Niceno. EasyMesh(Version 1.4), freely available mesh generator on the web site: http://www-dinma. univ.trieste.it/∼nirftc/research/easymesh/.
14. P.-O. Fjällström. Algorithms for Graph Partitioning: a survey. In *Linköping Electronic Articles in Computer and Information Science*, volume 3, Linköping, 1998. Department of Computer and Information Science, Linköping University.
15. Y. Saad and M. Sosonkina Distributed Schur Complement Techniques for General Sparse Linear Systems. SIAM Journal on Scientific Computing. October, 1997.
16. Y. Saad and M. Sosonkina. pARMS: a package for solving general sparse linear systems of equations. In R. Wyrzykowski, J. Dongarra, M. Paprzycki, and J. Wasniewski, editors, *Parallel Processing and Applied Mathematics*, volume 2328 of *Lecture Notes in Computer Science*, pages 446–457, Berlin, 2002. Springer-Verlag.
17. B. Smith, P. Bjorstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Pres, Cambridge, 1996.
18. U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. 2001. With contributions by A. Brandt, P. Oswald and K. Stüben.
19. P. Wesseling. *Introduction to Multigrid Methods*. John Wiley & Sons, Chichester, 1992.

A Multigrid-Schwarz for the Solution of Elliptic Equations in Unstructured Meshes

Guilherme Galante[1], Rogerio L. Rizzi[2] and Tiaraju A. Diverio[1]

[1] PPGC, Universidade Federal do Rio Grande do Sul
CP 15064, 91501-970, Porto Alegre, RS, Brazil
[2] CCET, Universidade Estadual do Oeste do Paran
ggalante@inf.ufrgs.br

Abstract. This work shows a parallel method for the solution of equations systems originated from the discretization of elliptic partial differential equations. The proposed solution method is obtained by a multigrid method parallelized through domain decomposition techniques, more specifically using the additive Schwarz method. In the case study, we used the 2D heat diffusion equation in unstructured meshes. The implementations was developed employing message passing to explore parallelism in clusters.

1 Introduction

The large linear equations systems, that raises of the discretization of partial differential equations (PDE) in technological and scientific problems, require efficient solution. One of the approaches used in the equation systems resolution are the multigrid methods. Multigrid methods originated in the 1960s with the work of Fedorenko and Bakhvalov. They were further developed in the 1970s by Brandt, and are now the preferred methods for solving elliptic partial differential equations [7]. The main advantage of multigrid is the speed - multigrid algorithms only require order N operations to solve elliptic equations, where N is the number of mesh points.

In this paper, we describe our solver, Multigrid-Schwarz, that solves elliptic equations using multigrid methods parallelized by domain decomposition methods in unstructured meshes. The idea of combining multigrid and domain decomposition methods is not new [1][3], although there are a number of features of our code that distinguish it from the discussions we have seen. In particular, we use Full Multigrid V cycle [7] combinated with the additive Schwarz domain decomposition method to explore the parallelism [6]. Moreover, our method uses only Krylov space iterative methods, instead classical iterative solvers, like Gauss-Seidel, generally used in the multigrid approaches.

This paper is structured as follows: in section 2 we present a overview of multigrid methods; in section 3 the proposed parallel multigrid method is detailed; the section 4 summarizes the results obtained and in section 5 some conclusions and future works are presented.

2 Multigrid Overview

The multigrid methods are iterative methods of equation systems resolution. This class of methods are strongly dependent of the initial guess attributed to the variables of the problem.

The great difference of the multigrid methods in relation to other iterative methods, is that it solves equation systems in different mesh sizes, to improve the initial guess attributed to the variables of the problem. This set of operations can be interpreted as a pre-processing of the linear system, making the solver execute a lower number of iterations to reach the criterion of convergence specified for the original linear system.

Each time the system moves from the finest mesh to some coarser mesh and back again is known as a cycle. There are a number of different kinds of cycle behaviours [7]. In this work the *full multigrid V*, or simply FMV, is used. The implementation is discussed in section 3.

3 Multigrid-Schwarz Method

Basically, the parallel FMV was obtained by the parallelization of the three main routines:

1. equation system solver;
2. communication among meshes;
3. residual calculation.

The parallelization uses the domain decomposition approach to explore the parallelism in clusters. Domain Decomposition Methods (DDM) denotes a set of mathematical, numerical and computational methods and techniques to solve problems in parallel computers.

A DDM is characterized by the division of the computational domain, that is partitioned in subdomains using partitioning algorithms. Each processor is responsible for finding the local solution of one or more subdomains [6]. The global solution of the problem is obtained by the combination of subproblems, that are locally solved.

The parallel solution was obtained using additive Schwarz domain decomposition method, which is characterized by the decomposition of the domain in K subdomains overlapped. In this method, all the subdomains use the solution of the last iteration as boundary condition, so that each subdomain can be solved independently, the communications being restricted to the boundary. As the local matrices are sparse non-symmetric, the GMRES algorithm was employed to obtain a solution of the equation sub-systems.

As well the solver, the communication among meshes and the residual calculations are also calculated in parallel. A mesh in the multigrid hierarchy communicates with the adjacent ones through restriction and interpolation operators. Restriction takes data on a mesh and restricts it to the next coarsest mesh and interpolation takes data on a mesh and interpolates it onto the next finest mesh.

For the data transfer operators, we choose I_{coarse}^{fine} to be the full-weighted restriction operator [7], and the interpolation I_{fine}^{coarse} is given by $x_N = \frac{1}{4}\sum_{i=1}^{4} x_{fine}$, where x_{fine} is the corresponding values in the coarse mesh. In this operations, each processor is responsible only for its domain. None communication is necessary because no external data are needed.

The residual equation is given by $r = b - Ax$. This calculation involves two operations: a matrix-vector multiplication and a vector subtraction. The parallelization of the vector subtraction is straightforward and do not need any communication. As the matrix and the vectors was splitted among the subdomains, the matrix-vector computation must be splitted in two parts. In the first part, we calculate the internal portion, without any communication among subdomains. In the second part, we calculate the product contribution of the neighboring subdomains, therefore, the domain must communicates with its neighbors to receive the required data.

4 Results

For numerical and computational experiments, we used the heat diffusion PDE, discretized through finite volume method on triangular unstructured meshes. The equation is given by:

$$\frac{\partial T}{\partial t} = \mu \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) \tag{1}$$

where μ is the diffusion constant, t is time and T is the temperature. The solution of the heat transfer equation was obtained for a square domain, with 1 m^2 and $\mu = 1$. This domain was discretized in a 4 level mesh hierarchy, containing 1337, 5348, 21392 and 85568 triangles, respectively. This mesh hierarchy was generated by the *MGTool*, a tool created for the multigrid implementation support [5]. In this test we used constant temperature 0^oC on three edges of the square, and 1^oC on the fourth edge.

The implemented algorithms were run in the cluster of the Laboratory of Technology in Clusters (LabTeC) of UFRGS Computer Science Institute. The cluster labtec is formed by 20 nodes Dual Pentium III 1,1 GHz, with 1 GB of RAM memory.

The Multigrid-Schwarz method was tested up to 38 processes. We also compare the Multigrid-Schwarz to a additive Schwarz method without multigrid. The Figure 1 show the execution time and the speedup obtained. These results were obtained without considering the opening and reading of input data and considering 5 time steps, where each step is composed by the maximum number of iterations necessary to reach the desired accuracy. In the tests, the considered solution error is 10^{-5}.

Accordly to the figure, we can observe that both tested methods are scalable but the Multigrid-Schwarz is almost twice faster than additive Schwarz method, showing the efficiency of the multigrid methods.

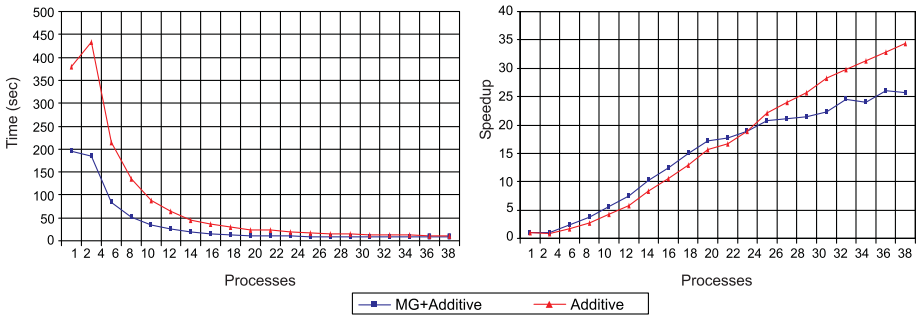


Fig. 1. Execution Time: Multigrid-Schwarz vs Additive Schwarz

5 Conclusion

In this work we presented a parallel implementation of a multigrid method, using additive Schwarz domain decomposition method. The numerical results obtained running the solver in several processors are consistent with the results obtained running it in one processor. The performed experiments shown that the proposed implementation is computationally efficient, scalable, and haves good numerical quality.

As future work, we will adapt the Multigrid-Schwarz method to the solution of UnHidra hydrodynamics model. The UnHidra is a multi-physics parallel computational model for the simulation of substance transport and for 2D and 3D hydrodynamic flow in water bodies, using unstructured meshes.

References

1. P. Bastian, W. Hackbusch, and G. Wittum. Additive and multiplicative multi-grid: a comparison. Computing, 60:345364, 1998.
2. T. F. Chan and B. F. Smith. Domain decomposition and multigrid algorithms for elliptic problems on unstructured meshes. Electronic Transactions on Numerical Analysis. Volume 2, pp. 171-182, December 1994.
3. E. Chow, R. Falgout, J. Hu, R. Tuminaro and U. Meier Yang. A Survey of Parallelization Techniques for Multigrid Solvers. Frontiers of Parallel Processing For Scientific Computing, SIAM book series , 2005.
4. M. Filipiak. Mesh Generation. [S.l.]: EPCC, Edinburgh, 1996. Watch Report.
5. G. Galante A Multigrid-DDM Schur Elliptic Equation Solver in Unstructured Meshes. In VECPAR 2006, 7th Int. Meeting on High Performance Computing for Computational Science, Rio de Janeiro, Brazil, 2006. Note: To appear.
6. B. Smith, P. Bjorstad, and W. Gropp. Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations. Cambridge University Pres, Cambridge, 1996.
7. P. Wesseling. Introduction to Multigrid Methods. John Wiley & Sons, Chichester, 1992.

Hydrodynamics of the Guaiba River simulated by the HIDRA model: data decomposition versus domain decomposition

Guilherme Galante, Rogerio L. Rizzi, Andre L. Martinotto, Tiaraju A. Diverio, Philippe O. A. Navaux

PPGC, Instituto de Informatica, UFRGS
CP 15064, 91501-970, Porto Alegre, RS, Brasil
{ggalante, diverio}@inf.ufrgs.br

Abstract

In this paper we present a parallel computational model for the simulation of the hydrodynamic flow in water bodies. For the parallel solution of the systems of equations in PC clusters, we focused the parallelization of the numerical methods and the use of domain decomposition methods. Both approaches was efficient in the solution of systems of equations, obtaining good performance. In the tests, the data decomposition approach revealed more adequate for a little amount of processes, while that the domain decomposition approach revealed more scalable, showing a better behavior with a large amount of processes.

Key words: HIDRA model, data decomposition, domain decomposition

1 Introduction

Computational models are essential in the simulation of scientific and technological processes as Ambiental Studies and Global Oceanic Modeling, etc. Thus, the goal of this work was to develop appropriate strategies of solutions in parallel to the HIDRA model, whose objective is to simulate complex hydrodynamic processes that occur in water bodies, particularly lakes, coastal regions and estuaries.

A specific application for the computational model presented here, is the simulation of the hydrodynamics and the transport of pollutants in the Guaiba River. The Guaiba River has an area of approximately $470km^2$. It has a length of approximately $50km$ and presents sections of up to $15km$ wide.

The Guaiba River is very important for the fluvial transport, irrigation and the water supply in the cities of the region, but it suffers constant emission of industrial and domestic pollutants. The Fig. 1 illustrates a satellite image of the Guaiba River.

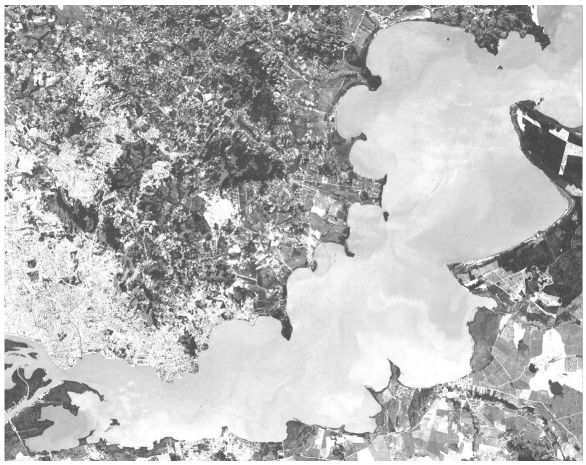


Fig. 1. Satellite image of the Guaiba River

Although the model was applied to the Guaiba River, it is independent of the particular application. The model can be used for the computational simulation of the flow and the substance transport in any shallow water body.

This work is motivated by the fact of that the disponibilization of computational models with high performance allows the simulation in details of the circulation and the substance transport in water bodies. These questions are essential for the understanding of the related ecological activities to these environments. Such importance is related to the fact of that the potable water supply in the planet for the human activities can become scarse. Therefore the increasing pollution of the environment for the man is preoccupying, once the availability of water is relatively small.

Some characteristics of the parallel computational model for 2D and 3D hydrodynamics and substance transport developed in this work are: discretization by finite differences in conservative form; use of an horizontal space-staggered mesh; use of cartesian coordinates for the vertical mesh; use of Sweby Flux-limited method and Gross Beta method to calculate transport; use of Casulli strategy to discretize and solve de hydrodynamics; solution via Cholesky algorithm to the horizontal velocities in hydrodynamics and via Thomas algorithm for the substance transport; use of solution strategies based in the parallelization of the numerical methods and domain decomposition.

This characteristics follow those originally specified for Casulli [7]. Examples of models of this nature, found in the technical literature, are those of Paglieri [8] and Ding [9], where the solution is obtained in parallel, using Domain Decomposition Methods and Fast Fourier Transform, respectively. Vollebregt [10]

uses the ADI (Alternating Direction Implict) combined with strategies of data transposition and methods of subespao of Krylov to explore the parallelism. Zhu [11] developed a parallel solution throught data decomposition.

In HIDRA model, some aspects are considered, for example, definition of mathematical model, the partitioning of the computational domain, the parallel implementation of numerical methods and load balancing. In this paper we describe the questions related to the partitioning of the computational domain and the parallel solution of the systems of equations generated for the model of hydrodynamics of the HIDRA.

This work is organized as follow. In section 2 we show the model of hydrodynamics used in the HIDRA. Section 3 shows to the methods and strategies of parallel solution that had been used in the resolution of the problem. In section 4 the performance results of the implemented methods are shown. Finally, the last section presents the conclusions and and future activities.

2 The HIDRA Hydrodynamic Model

In this section, we presents our hydrodynamics model, the HIDRA. Some characteristics of the parallel computational model, for 2D and 3D hydrodynamics and substance transport, developed in this work are explained in the sections 2.1 and 2.2.

2.1 Coordinate Systems and Discrete Models

The meshes where the variables are calculated are Arakawa C in horizontal and z-coordinate in vertical. To conciliate the computational simplicity with a good representation of topografy we have used the partial cell approach [gri00], where the vertical thickness of the cells can vary with the topography, and the cells, which are kept rectangular, can remain not completely filled.

The 3D mesh consists, therefore, of cells defined by $(\Delta x, \Delta y, \Delta z_k^n)$, where Δz_k^n denotes the vertical spacing, which depends on the position of the k-th layer in relation to the free surface, and can vary in time. If the heights of the cell faces are not compretely formed by water, as in the case of cells adjacent to the bottom and/or free surface, we use as vertical spacing the height of the water columns in these faces.

The mathematical model of free surface flow with constant density, described with primitive variables and obtained from Reynolds-averaged Navier-Stokes equations assuming that the flow is hydrostatic, is called Shallow Water Equation (SWE) [12]. Under a mathematical point of view, SWE form a hyperbolic, non-linear system of partial differential equations (PDEs) to an uncompressible fluid with free surface. The system is composed by the momentum equation for horizontal velocities and by the continuity equation. Integrating the continuity equation over the water column and using the free-surface cinematic boundary condition the equation for the integration between layers is obtained.

The boundaty conditions (BC) in the free surface are obtained considering that there is no flow through this interface. The BC in the bottom of the water body are obtained imposing that there is no flow through this interface. The lateral BC are closed and opened with inflow or outflow. In the open lateral boundaries a Dirichlet BC is specified, defining the water levels or flow. In the closed lateral boundaries there is no flow, and the velocities in these boundaries are given by the component of the velocity. The initial conditions (IC) are obtained imposing that the velocity is null in the entire domain and there is an initial water level.

In his work Casulli [7] has shown, through a stability analysis of the characteristic equations of the 2D SWE vertically integrated, that celerity is given by $c = (gH)^{\frac{1}{2}}$. The terms g and $H = h + \eta$ are coefficients of the elevation gradients of the momentum PDE and of the derivatives of the velocities in the continuity equation, respectively. Thus, discretizing these terms implicitly the stability of the numerical solution does not depend on the celerity, even when the non-linear terms are approximated, conveniently, through an explicit approach. Thus, we discretize implicitly the terms of the free surface elevation gradient and vertical diffusion int he momentum equation in horizontal direction. The advective terms, the horizontal diffusion and Coriolis force are discretized explicitly. The continuity and vertically integrated continuity PDE are discretized implicitly.

With these choices, and considering the θ-method to improve the degree of implicity and temporal accuracy in the vertical component of velocity and of the gradients of surface elevation, we have a system of linear equations, symmetric defined-positive (SDP), whose unknowns are the barotropical terms (gradients of water levels). The solution of this systems is obtained parallelization of the numerical methods or domain decomposition. This issues are discussed in the next section.

3 Parallel Solution: methods and strategies

The practical domains of problems under the scope of environment studies are huge and require a great capacity of processing and storage. Presently, PC clusters have shown to be an accessible and efficient option, due to their excellent cost/benefit ratio. In this section we presents the methods and strategies used in the solution of the equation system of the hydrodynamics of the HIDRA model.

3.1 Domain Partitioning

To solve a huge problem in parallel, it is necessary to divide the computational domain among the available processors. With domain partitioning we want obtain a good computational load balance and the minimization of the communication necessary to solve the dependencies generated by the frontiers resulting from the partitioning.

Partitioning the graph in as many parts as there are processes, aiming at the minimization of the number of the edges between them, problem known as k-partitioning, is a NP-Hard problem. Heuristic approaches are the only viable ones and, for this reason, we used METIS package and RCB algorithm to generate the subdomains. Fig. 2 shows an example of partitioning of the domain of Guaiba Lake, in 16 subdomains, using METIS.

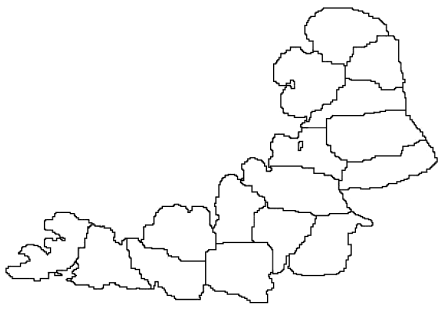


Fig. 2. Guaiba Lake domain partitioned in 16 subdomains

3.2 Data Decomposition vs Domain Decomposition

To the parallel solution we have focused in two approaches: the parallelization of the numerical methods and the use of domain decomposition methods. In this first approach just one equation system is generated for the entire domain, which is solved through a parallelized numerical method, showed in Fig. 3 (a). In the second approach, we use a domain decomposition method, so that the

global solution of the problem is obtained by the appropriate combination of the solutions obtained in each subdomain, as showed in Fig. 3 (b).

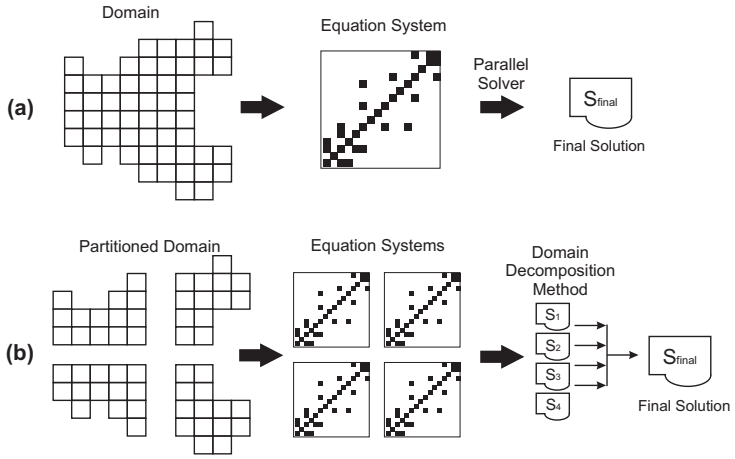


Fig. 3. (a) Data decomposition versus (b) Domain decomposition

In this paper, the parallelization of the numerical methods is called data decomposition, where the operations and the data are distributed among the available processors and processed in parallel.

Local communication and group communication, are supplied by the libraries of message passing. This libraries are the responsible for the control of the information among the involved nodes in the parallel solution of iterative methods used to solve the equation systems.

This approach was used in the parallelization of the linear algebra operations of the Krylov subspace iterative methods [4]. In this paper we used the Conjugate Gradient method (CG). The CG is composed by linear algebra operations, and the paralelism is achieved by the parallel execution of the subtraction of vector, addition of vector, inner product and matrix-vector multiplication.

Domain Decomposition Methos (DDM) are a set of mathematical and computational tecniques where the global solution for the problem is obtained by the appropriate combination of the solutions of all the subdomains. As different subdomains can be treated independently, such methods are attractive for use inparallel environments [5]. In this paper, the parallel solution was obtained using Schwarz additive domain decomposition method. The method is characterized by the decomposition of the open and limited domain Ω in N subdomains Ω_k overlapped. All the subdomains use the solution of the last iteration as boundary condition, so that each subdomain can be solved independently, the communications being restricted to the boundary. Furthermore, supposing that $\Omega = \bigcup_{i=1}^{n} \Omega_i$, with $\Omega_i \cup \Omega_j \neq \emptyset$, for $i \neq j$, it is possible to show that the algorithm converges, and the presence of overlapped regions ensures the continuity of the solution and its derivatives between the different subdomains [6]. The algorithm is showed in Fig. 4.

***Start**: choose a inicial guess* $u_i^{(0)}$ *to each subdomain* Ω_i
determine the tolerance error ε

1. solve the equation system using the approximation $u_i^{(0)}$

2. $k = 1$*;*

do

3. send the boundary data $u_i^{(k-1)}$ *to the neightbours;*

4. receive from neightbour the boundary data $u_i^{(k-1)}$*;*

5. Solve the system using the data received;

6. Calculate $u_i^{(k)}$*;*

7. k *++;*

until $max\left(\left\|u_i^{(k)} - u_i^{(k-1)}\right\|_2 / \left\|u_i^{(k)}\right\|_2\right) \leq \varepsilon$

Fig. 4. Schwarz additive domain decomposition method

Since the 5-points stencil was used in the PDE approximations, the two cells overlap are exchanged at each cycle of the solver until the global convergence criterion is fulfilled. As the local matrices are sparse and symmetric positive definite (SPD), the conjugated gradient algorithm was employed, to obtain a solution of the equation sub-systems [4].

4 Tests and Results

In this section, we presentes the results obtained with the implementations developed in this work. The results were obtained using a cluster of PCs in II-UFRGS, constituted by twenty nodes Pentium III dual 1.1 GHz with 1 GB of RAM memory, connected by a Gigabit-Ethernet network. We used the message passing library MPICH 1.2.7 and the creation and management of threads was made by OpenMP, using the Omni precompiler 1.4.

We have used 20 MPI processes, one for each node, to explore the internode parallelism. Considering the architecture of the cluster nodes, each process had two threads to explore the intranode parallelism. Thus, a process is responsible by a amount of data that are shared by the two internal threads. In this way, we used efficiently the 40 processors available in the cluster.

The solved systems was generated in the horizontal mesh of the domain. For the tests, we used equations systems generated by the discretization of the Guaiba Lake with 3 different cells sizes: $\triangle x = \triangle y = 200m$, that results in linear equation systems with 11.506 equations, $\triangle x = \triangle y = 100m$, that results in linear equation systems with 46.024 equations, and $\triangle x = \triangle y = 50m$, that results in linear equation systems with 184.096 equations. This systems are respectively called Guaiba200, Guaiba100 and Guaiba50.

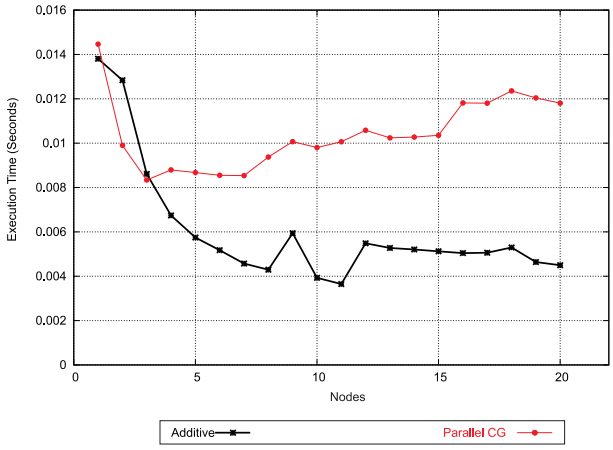


Fig. 5. Guaiba200 - Additive Schwarz vs parallel CG

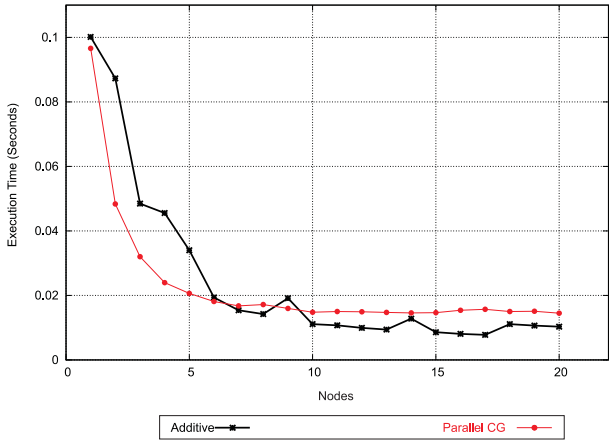


Fig. 6. Guaiba100 - Additive Schwarz vs parallel CG

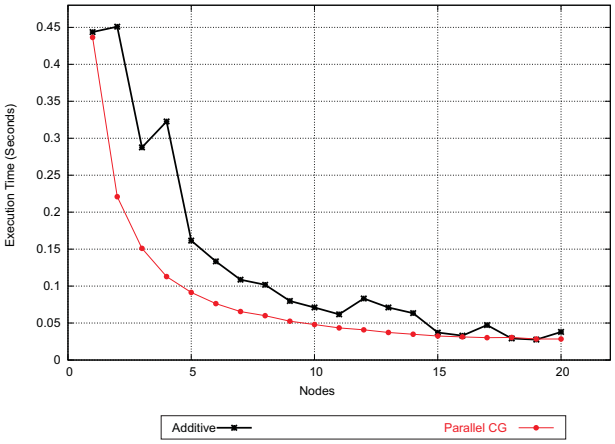


Fig. 7. Guaiba50 - Additive Schwarz vs parallel CG

The picture 5, 6 and 7 show a time comparation obtained from the implementations. It is possible to verify in these figures, that the parallel CG method have better performance when using few subdomains. However, when the number of subdomains increase, and the granularity of the problem became small,

the additive Schwarz methos shows better performance. This occur because the additive Schwarz method uses a little amount of external data, needing few communications. We can still observe, the presence of peaks in the execution time of the additive Schwartz, caused for the increase of the number of iterations, necessary for the convergence of the method.

The parallel CG needs of a big number of inner product operations in each iteration and this operation causes the sincronization of all processes. Thus, due to the low necessity of comunication, we conclude that the additive Schwarz method is more scalable than the parallel CG.

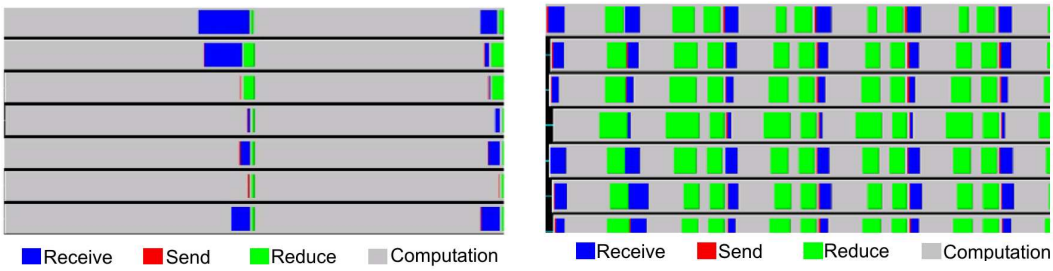


Fig. 8. Additive Schwarz method and parallel CG execution profile

The pictures 8 illustrates the profile ot the parallel executions of the additive Schwarz and the parallel CG, respectively. This executions was made with the Guaiba100 with 7 nodes. Analising the picture 8 we can identify why the additive Schwarz have a better performance than the parallel CG. The reduce operations indicates, in both methods, a inner product and it causes a sincronization of all processes, and consequentely a fall in the performance of the application.

5 Conclusion

In this work we have presented a parallel computational model where 2D and 3D hydrodynamics are simulated. More specifically, we focused the solution of the systems of equations originated from hydrodynamics model. The solution was obtained through two distinct approches, data decomposition and domain decomposition.

Both approaches was efficient in the solution of equations systems, obtaining good performance. In the presented tests, the data decomposition approach revealed more adequate for a little amount of processes, while that the domain decomposition approach revealed more scalable, showing a better behavior with a large amount of processes.

References

[1] RIZZI, R. L.; DORNELES, R. V.; et al. Parallel Computational Model with Dynamic Load Balancing in PC Clusters. In: VECPAR2004 6TH INTERNATIONAL MEETING ON HIGH PERFOMANCE COMPUTING FOR COMPUTATIONAL SCIENCE, 2004, Valencia, 2004.

[2] GRIFFIES, S. M. et al. Developments in Ocean Climate Modelling. Ocean Modelling, v. 2, p. 123-192, 2000.

[3] KARYPIS, G.; KUMAR, V. METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. 1998.

[4] SAAD, Y. Iterative Methods for Sparse Linear Systems. Boston: PWS Publishing Company, 1996.

[5] SMITH, B.; BJORSTAD, P.; GROPP, W. Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations. Cambridge: Cambridge University, 1996. 224 p.

[6] DEBREU, L.; BLAYO, E. On the Schwarz Alternating Method for Oceanic Models on Parallel Computers. Journal of Computational Physics, v. 141, p. 93-111, 1998.

[7] CASULLI, V.; CATTANI, E. Stability, Accuracy and Efficiency of a Semi-Implicit Method for Three-Dimensional Shallow Water Flow. Computers Math. Applic., [S.l.], v. 27, n. 4, p. 99-112, 1994.

[8] PAGLIERI, L.; AMBROSI, D.; FORMAGGIA, L.; QUARTERONI, A.; SCHEININE, A. L. Parallel Computation for Shallow Water Flow: A Domain Decomposition Approach. Parallel Computing. v. 23, p. 1261-1277, 1997.

[9] DING, X. Numerical Solution of the Shallow-Water Equations on Distributed Memory Systems. 1998. 73 p. Master of Science (Computer Science). University of Toronto.

[10] VOLLEBREGT, E. A. H. Parallel Software Development Techniques for Shallow Water Models. 1997. 133p. Ph.D. Dissertation (Hydraulic and Environmental Engineering) - Delft University of Technology, Delft.

[11] ZHU, J.; JOHNSON, B.; BANGALORE, P.; HUDDLESTON, D.; SKJELLUM. A. On the Parallelization of CH3D. 1998.

[12] WEIYAN, T. Shallow Water Hydrodynamics: Mathematical Theory and Numerical Solution for a Two-dimensional System of Shallow Water Equations, Elsevier, Amsterdan, 1992.