

IMPLEMENTAÇÃO EM PARALELO DO MÉTODO DOS ELEMENTOS FINITOS  
PARA AS EQUAÇÕES DE ÁGUAS RASAS

Ivan Slobodcicov

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA CIVIL.

Aprovada por:

---

Prof. Fernando Luiz Bastos Ribeiro, DSc.

---

Prof. Nelson Francisco Fávilla Ebecken, DSc.

---

Prof. Philippe Remy Bernard Devloo, PhD.

RIO DE JANEIRO, RJ – BRASIL  
MARÇO DE 2003

SLOBODCICOV, IVAN

Implementação em Paralelo do Método dos Elementos Finitos para as Equações de Águas Rasas [Rio de Janeiro] 2003

VII, 88p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Civil, 2003)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1 – Processamento Paralelo

2 – Método dos Elementos Finitos

3 – Equações de Águas Rasas

4 – Métodos Iterativos

I. COPPE/UFRJ      II. Título (série)

*Aos meus pais Anton e Emília*  
e  
*à Cássia, Laura, Pêtr e Sophia*

## **AGRADECIMENTOS**

Aos meus pais que embora não estejam mais presentes ao final deste trabalho, mas cujas contribuições foram significativas para que este pudesse ter começado.

Aos Profs. Fernando Luiz Bastos Ribeiro e Alvaro Luiz Gayoso de Azeredo Coutinho pela orientação, incentivo, apoio e paciência despendidos durante o desenvolvimento desta tese.

Aos colegas de trabalho que muito incentivaram e contribuíram na resolução dos problemas enfrentados.

Aos colegas Ricardo Bragança, André Bulcão e Fernando Barreto, administradores do *cluster* da PETROBRAS/CENPES, que incansavelmente e pacientemente ajudaram no desenvolvimento computacional.

Ao colega Mauro Costa pela disponibilidade e presteza no início da utilização do *cluster*.

A PETROBRAS/CENPES pela oportunidade de realização deste curso e disponibilidade dos recursos materiais.

Ao Programa de Engenharia Civil da COPPE/UFRJ e em especial à secretária acadêmica Elizabeth Cornélio pelo apoio administrativo.

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

## IMPLEMENTAÇÃO EM PARALELO DO MÉTODO DOS ELEMENTOS FINITOS PARA AS EQUAÇÕES DE ÁGUAS RASAS

Ivan Slobodcicov

Março/2003

Orientadores: Fernando Luiz Bastos Ribeiro  
Alvaro Luiz Gayoso de Azeredo Coutinho

Programa: Engenharia Civil

Este trabalho apresenta a implementação em paralelo do método dos elementos finitos aplicado na resolução de sistemas não-lineares não-simétricos obtidos a partir da discretização das equações que governam o comportamento hidrodinâmico do escoamento em águas rasas. Na determinação da solução destes sistemas não-lineares foi utilizado o método iterativo GMRES que além de ser amplamente empregado em problemas de dinâmica dos fluidos apresenta boa estabilidade e convergência.

Estruturas de dados baseadas em elemento e em aresta foram usadas para o armazenamento das matrizes, otimizando o uso da memória e o desempenho do método iterativo GMRES. O METIS, um *software* de domínio público, através de seus dois métodos (Dual ou Nodal), foi utilizado no particionamento das malhas. Resultados comparativos entre ambas as estruturas de dados e entre os dois métodos de particionamento são apresentados através dos exemplos numéricos.

A implementação paralela foi projetada para *clusters* de PCs gerenciados por um pacote de comunicação utilizando o padrão MPI.

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

PARALLEL IMPLEMENTATION OF THE FINITE ELEMENT METHOD  
FOR SHALLOW WATER EQUATIONS

Ivan Slobodcicov

March/2003

Advisors: Fernando Luiz Bastos Ribeiro

Alvaro Luiz Gayoso de Azeredo Coutinho

Department: Civil Engineering

This work presents a parallel implementation of the finite element method applied to the solution of non-linear and non-symmetric systems, arising from the discretization of the equations that govern the hydrodynamic behavior of shallow water flow. For the determination of these non-linear systems, The GMRES iterative method was used. This method is largely used in hydrodynamics problems, since it presents good stability and convergence.

Element-based as well as edge-based data structures were used for the storage of matrices in order to optimize the usage of memory and performance appearing in the GMRES iterative solver. METIS, a freeware software, which carries two different methods (Dual or Nodal) for meshes partitioning was also used. Comparative results between both data structures and both partitioning methods are presented through numerical examples.

The parallel implementation was designed for PC's *clusters* running a communication package using the MPI library.

# ÍNDICE

1. Introdução	1
2. Modelo Matemático	3
2.1 Equações de Águas Rasas em Três Dimensões	3
2.2 Modelo Baseado na Média Vertical (modelo 2DH)	7
2.2.1 Forma Divergente das Equações de Águas Rasas	14
2.2.2 Forma Advectiva das Equações de Águas Rasas	14
2.2.3 Formas Simétricas das Equações de Águas Rasas	16
2.2.3.1 Variáveis de Entropia	17
2.2.3.2 Variáveis de Velocidade	18
3. Formulação Semi-Discreta Estabilizada do Método dos Elementos Finitos	21
3.1 Aspectos Computacionais da Aproximação Semi-Discreta	25
3.2 Solução do Sistema de Equações Não-Lineares	26
4. Estrutura de Dados	28
4.1 Estrutura de Dados Baseada em Elemento e em Aresta	29
5. Implementação em Paralelo	32
5.1 Programa METIS	33
5.2 Tratamento das Fronteiras	37
5.3 Comunicação entre Processadores	40
5.4 Fluxograma	44
6. Dados e Resultados	46
6.1 Dados Geométricos	46
6.2 Dados Ambientais	49
6.3 Outros Dados	52
6.4 Resultados	53
7. Conclusões	83
8. Referências Bibliográficas	84

## 1. INTRODUÇÃO

O comportamento hidrodinâmico verificado em regiões costeiras, estuários, baías, rios, canais ou mesmo em lagoas abertas para o mar, sofre influência do vento e/ou do efeito da maré. A predição das correntes nestas regiões, devido ao ciclo de elevação da maré, é de vital importância à navegação. Outro aspecto fundamental desta predição está relacionado ao meio ambiente. Trata-se da determinação da dispersão de contaminantes, que infelizmente ainda vem ocorrendo em diversas áreas. O transporte de sedimentos associado a este tipo de escoamento também é outra área de interesse a ser investigada.

É cada vez maior o interesse pela formulação computacional representativa dos fenômenos relacionados ao escoamento com essas características. Nessa categoria, estão classificados os fenômenos onde o escoamento ocorre em uma camada de água considerada rasa. De forma simplificada podemos dizer que o escoamento em águas rasas pode ser definido como sendo aquele que ocorre quando a escala horizontal é muito maior que a sua escala vertical (profundidade). Considerando, que na maioria dos casos, o escoamento seja realizado através de finas camadas, podemos dizer que a velocidade vertical apresenta uma magnitude bastante inferior à das velocidades horizontais, e que em certos casos pode ser considerada praticamente desprezível. Sendo assim, os problemas dessa natureza podem ser razoavelmente aproximados em duas dimensões e as correspondentes equações que modelam o seu comportamento podem ser obtidas a partir das equações de *Navier-Stokes* [1] na sua forma isotérmica totalmente incompressível.

Por vários anos o método de diferenças finitas foi usado na resolução de fenômenos modelados através das equações de águas rasas. Porém, com o aumento do poder computacional nos últimos anos, uma série de problemas têm migrado para a utilização do método dos elementos finitos (MEF). Embora seja computacionalmente mais complexo, o MEF é reconhecido como tendo algumas vantagens sobre a abordagem utilizando diferenças finitas. Uma de suas principais características é a sólida base matemática empregada na sua formulação. Outra característica importante, devido à facilidade em lidar com malhas não estruturadas, é a capacidade do método em tratar os contornos, fronteiras irregulares, "ilhas" e obstruções, de uma maneira mais natural. A discretização das equações de águas rasas utilizando o MEF conduz a um sistema não-linear não-simétrico de equações diferenciais parciais hiperbólicas que caracterizam a elevação da superfície livre e a velocidade média ao longo da profundidade.

Uma das grandes dificuldades na resolução numérica deste tipo de problema está associada ao caráter convectivo dominante [1] destas equações, que não são capazes de evitar oscilações espúrias. Uma maneira encontrada para resolver este problema é através da utilização de métodos estabilizados [2, 3, 4, 5, 6] acoplados ao MEF. Neste trabalho foi adotada esta metodologia utilizando a formulação variacional semi-discreta estabilizada do MEF. Esta formulação, semi-discreta, considera que apenas as variáveis espaciais são aproximadas por elementos finitos, enquanto que as variáveis temporais são aproximadas por operadores de diferenças finitas. Para a estabilização, foram introduzidos dois termos: *SUPG (Steamline Upwind Petrov-Galerkin)* e *CAU (Consistent Approximate Upwind)*. O primeiro deles, *SUPG* [6], introduz a quantidade de difusão necessária para eliminar as oscilações espúrias e o segundo, *CAU* [2], atua nas regiões de mais alto gradiente minimizando eventuais flutuações.

Soluções transientes podem ser obtidas a partir de um algoritmo preditor multi-corretor marchante no tempo, o que significa dizer que uma seqüência de sistemas lineares deve ser resolvida. Para a resolução destes sistemas foi adotado o método iterativo *GMRES (Generalized Minimum Residual)* [7], desenvolvido para sistemas não-simétricos, acoplado a um preconditionador diagonal. Estruturas de dados baseadas em elemento e em aresta [8] foram utilizadas no produto matriz-vetor implementado no método iterativo.

Este trabalho possui como um dos principais objetivos a implementação de um algoritmo que permita a redução no tempo de simulação através da paralelização das rotinas de cálculo. Para tanto, toda a implementação matricial, tanto na sua montagem como em suas atualizações, além do produto matriz-vetor utilizado no método iterativo foi realizada de forma paralela. O produto escalar necessário na determinação do resíduo, e que é utilizado como condição de parada do *loop* não-linear, também foi paralelizado.

Para a representação dos resultados foram feitas simulações do comportamento da maré na entrada da Lagoa de Araruama, Brasil, a partir de três malhas não estruturadas [9] obtidas através de sucessivos refinamentos. Estas malhas foram particionadas utilizando-se o *software* METIS [10], de domínio público, o qual dispõe de dois métodos distintos para o seu particionamento (Dual ou Nodal). Resultados comparativos foram feitos entre ambas as estruturas de dados e entre os dois métodos de particionamento, evidenciando o desempenho de cada uma destas abordagens através da medida do seu ganho. Toda a simulação foi executada em *clusters* de PC's onde a biblioteca de comunicação instalada utilizou o padrão MPI [11].

## 2. MODELO MATEMÁTICO

A modelagem utilizada na descrição das equações de águas rasas em duas dimensões, também denominada de modelagem baseada na média vertical (modelo 2DH), é obtida a partir da integração vertical das equações tridimensionais de *Navier-Stokes* para escoamentos incompressíveis com condições de contorno, de fundo e de superfície, incluídas. A principal limitação da modelagem 2DH é que ela não considera os efeitos da variação da velocidade e densidade na direção vertical. Contudo, o modelo 2DH pode ser adequado, uma vez que o escoamento da camada compreendida entre o fundo e a superfície livre comporta-se de forma homogênea, com suas velocidades horizontais sendo predominantes. Assim sendo, o escoamento pode ser razoavelmente aproximado em duas dimensões.

### 2.1 EQUAÇÕES DE ÁGUAS RASAS EM TRÊS DIMENSÕES

O escoamento isotérmico incompressível em três dimensões é governado pelas seguintes equações de *Navier-Stokes* [1]

$$\frac{\partial u_1}{\partial t} + \frac{\partial}{\partial x_i} (u_i u_1) + \frac{1}{\rho} \frac{\partial p}{\partial x_1} - \frac{1}{\rho} \left( \frac{\partial \tau_{i1}}{\partial x_i} \right) + f u_2 = 0 \quad (1)$$

$$\frac{\partial u_2}{\partial t} + \frac{\partial}{\partial x_i} (u_i u_2) + \frac{1}{\rho} \frac{\partial p}{\partial x_2} - \frac{1}{\rho} \left( \frac{\partial \tau_{i2}}{\partial x_i} \right) - f u_1 = 0 \quad (2)$$

$$\frac{\partial u_3}{\partial t} + \frac{\partial}{\partial x_i} (u_i u_3) + \frac{1}{\rho} \frac{\partial p}{\partial x_3} - \frac{1}{\rho} \left( \frac{\partial \tau_{i3}}{\partial x_i} \right) + g = 0 \quad (3)$$

e pela equação da continuidade

$$\boxed{\frac{\partial u_i}{\partial x_i} = 0} \quad (4)$$

onde  $\rho(x_i)$  é a massa específica do fluido,  $u_i(x_i, t)$  são as componentes da velocidade,  $p(x_i, t)$  é a pressão,  $g$  é a aceleração da gravidade,  $f$  é o fator de Coriolis,  $\tau_{ij}$  são as tensões cisalhantes,  $x = (x_1, x_2, x_3)$  é o vetor posição e  $t$  é o tempo. As

equações (1-3) representam a conservação da quantidade de movimento ou equilíbrio dinâmico. A equação (4) representa a conservação da massa do sistema e é também a condição de incompressibilidade.

As tensões podem ser eliminadas das equações de equilíbrio utilizando a relação constitutiva

$$\tau_{ij} = \mu \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \quad (5)$$

onde  $\mu$  é a viscosidade dinâmica do fluido. Desta forma, temos um sistema de quatro equações e quatro incógnitas:  $u_1, u_2, u_3, p$ .

Em escoamentos de águas rasas (quase horizontal), o parâmetro horizontal  $L$  deve ser maior que o parâmetro vertical  $H$ . Conforme Ribeiro *et al* [12], para que um escoamento seja considerado quase horizontal, a seguinte relação deve ser satisfeita:

$$\frac{H}{L} < \frac{1}{20} \quad (6)$$

Esta hipótese mostra que somente ondas longas, isto é, ondas onde o comprimento é maior que a altura, são levadas em consideração. Nesta situação, as acelerações e tensões verticais podem ser desprezadas, reduzindo a equação da quantidade de movimento na direção de  $x_3$  a

$$\frac{\partial p}{\partial x_3} = -\rho g \quad (7)$$

Integrando esta equação na direção vertical temos

$$\int_{x_3}^{\eta} \frac{\partial p}{\partial x_3} dx_3 = - \int_{x_3}^{\eta} \rho g dx_3 \quad (8)$$

Seja  $(x_1, x_2) \in \Omega \subset \mathfrak{R}^2$  definido com sendo os pontos no plano horizontal  $x_3 = 0$  representando a superfície não perturbada e seja  $x_3 \in [-h, \eta]$  o ponto na direção vertical onde  $h(x_1, x_2)$  representa a profundidade e  $\eta(x_1, x_2, t)$  a elevação da

superfície, ambas medidas a partir da superfície não perturbada (Figura 1). O domínio de interesse é a camada de largura  $H = h + \eta$  confinada entre a superfície livre, descrito pela função  $x_3 = \eta(x_1, x_2, t)$  e o fundo, dado pela função  $x_3 = -h(x_1, x_2)$ .

Assumindo que a massa específica não varia na direção de  $x_3$  podemos escrever que

$$p(x_i, t) = p(x_1, x_2, \eta, t) + \rho g(\eta - x_3) \quad (9)$$

onde  $p(x_1, x_2, \eta, t)$  é a pressão atmosférica. Esta última equação é a expressão para a pressão hidrostática, somente válida para os escoamentos quase horizontais. Se a pressão atmosférica e a massa específica são constantes, então temos que

$$\frac{\partial p}{\partial x_i} = \rho g \frac{\partial \eta}{\partial x_i} \quad (i = 1, 2) \quad (10)$$

Aplicando estes resultados, as equações de equilíbrio nas direções de  $x_1$  e  $x_2$  são reduzidas a

$$\frac{\partial u_1}{\partial t} + \frac{\partial}{\partial x_i} (u_i u_1) + g \frac{\partial \eta}{\partial x_1} - \frac{1}{\rho} \left( \frac{\partial \tau_{i1}}{\partial x_i} \right) + f u_1 = 0 \quad (11)$$

$$\frac{\partial u_2}{\partial t} + \frac{\partial}{\partial x_i} (u_i u_2) + g \frac{\partial \eta}{\partial x_2} - \frac{1}{\rho} \left( \frac{\partial \tau_{i2}}{\partial x_i} \right) - f u_2 = 0 \quad (12)$$

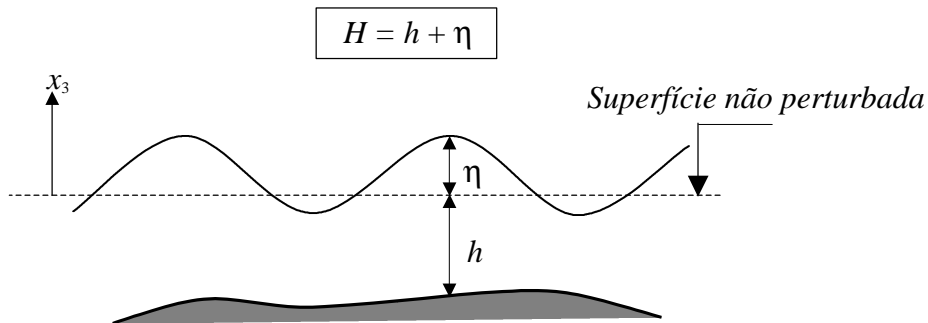


Figura 1. Profundidade ( $h$ ), elevação ( $\eta$ ) e superfície não perturbada.

Estas duas equações, juntas com a equação da continuidade, formam um sistema de três equações e quatro incógnitas  $(u_1, u_2, u_3, \eta)$ . Assim, outra equação é necessária para resolver este sistema. A equação adicional é obtida a partir da integração vertical da equação da continuidade:

$$\int_{-h}^{\eta} \frac{\partial u_1}{\partial x_1} dx_3 + \int_{-h}^{\eta} \frac{\partial u_2}{\partial x_2} dx_3 + \int_{-h}^{\eta} \frac{\partial u_3}{\partial x_3} dx_3 = 0 \quad (13)$$

Aplicando a regra de Leibnitz,

$$\frac{\partial}{\partial s} \int_a^b F(r, s) dr = \frac{\partial}{\partial s} \int_a^b F(r, s) dr - F(b, s) \frac{\partial b}{\partial s} + F(a, s) \frac{\partial a}{\partial s}$$

às duas primeiras integrais em (13), as seguintes relações são obtidas:

$$\int_{-h}^{\eta} \frac{\partial u_1}{\partial x_1} dx_3 = \frac{\partial}{\partial x_1} \int_{-h}^{\eta} u_1 dx_3 - u_1(\eta) \frac{\partial \eta}{\partial x_1} - u_1(-h) \frac{\partial h}{\partial x_1} \quad (14)$$

$$\int_{-h}^{\eta} \frac{\partial u_2}{\partial x_2} dx_3 = \frac{\partial}{\partial x_2} \int_{-h}^{\eta} u_2 dx_3 - u_2(\eta) \frac{\partial \eta}{\partial x_2} - u_2(-h) \frac{\partial h}{\partial x_2} \quad (15)$$

Como as velocidades devem ser zero no fundo, temos que

$$\int_{-h}^{\eta} \frac{\partial u_1}{\partial x_1} dx_3 = \frac{\partial}{\partial x_1} \int_{-h}^{\eta} u_1 dx_3 - u_1(\eta) \frac{\partial \eta}{\partial x_1} \quad (16)$$

$$\int_{-h}^{\eta} \frac{\partial u_2}{\partial x_2} dx_3 = \frac{\partial}{\partial x_2} \int_{-h}^{\eta} u_2 dx_3 - u_2(\eta) \frac{\partial \eta}{\partial x_2} \quad (17)$$

A integração do terceiro termo de (13) leva a

$$\int_{-h}^{\eta} \frac{\partial u_3}{\partial x_3} dx_3 = u_3(\eta) - u_3(-h) = u_3(\eta) \quad (18)$$

Somando-se estes resultados, a equação da continuidade após a integração vertical é dada por

$$\frac{\partial}{\partial x_1} \int_{-h}^{\eta} u_1 dx_3 + \frac{\partial}{\partial x_2} \int_{-h}^{\eta} u_2 dx_3 - u_1(\eta) \frac{\partial \eta}{\partial x_1} - u_2(\eta) \frac{\partial \eta}{\partial x_2} + u_3(\eta) = 0 \quad (19)$$

Identificando na equação anterior a velocidade vertical na superfície livre como sendo

$$u_3(\eta) = \frac{d\eta}{dt} = \frac{\partial \eta}{\partial t} + u_1(\eta) \frac{\partial \eta}{\partial x_1} + u_2(\eta) \frac{\partial \eta}{\partial x_2} \quad (20)$$

a quarta equação necessária para completar o sistema é obtida:

$$\boxed{\frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x_1} \int_{-h}^{\eta} u_1 dx_3 + \frac{\partial}{\partial x_2} \int_{-h}^{\eta} u_2 dx_3 = 0} \quad (21)$$

Resumindo, o modelo de águas rasas em três dimensões é representado por duas equações da quantidade de movimento (11, 12), a equação da continuidade (4) e a equação representativa da superfície livre (21), que devem ser resolvidas para as incógnitas  $u_i, \eta$ . Comparando com o modelo tridimensional completo, notamos que se o escoamento for considerado quase horizontal, a pressão  $p$  pode ser substituída pela elevação  $\eta$ , e ao invés da equação da quantidade de movimento na direção de  $x_3$  teremos a equação representativa da superfície livre.

## 2.2 MODELO BASEADO NA MÉDIA VERTICAL (MODELO 2DH)

O modelo 2DH é obtido pela integração vertical das equações da quantidade de movimento (11, 12), e da equação da continuidade (4). Também deve ser considerado que a coluna de água é homogênea (bem misturada), isto é, existe pouca ou nenhuma estratificação. Podemos então definir as componentes horizontais da velocidade baseada na média vertical  $U_i$  ( $i = 1, 2$ ) como sendo

$$U_i(x_1, x_2, t) = \frac{1}{H} \int_{-h}^{\eta} u_i(x_i, t) dx_3 \quad (22)$$

de tal forma que

$$u_i(x_i, t) = U_i(x_1, x_2, t) + u'_i(x_i, t) \quad (23)$$

Nas equações acima,  $u'_i$  corresponde ao desvio das velocidades médias  $U_i$  (Figura 2) e a seguinte relação deve ser satisfeita:

$$\int_{-h}^{\eta} u'_i(x_i, t) dx_3 = 0 \quad (24)$$

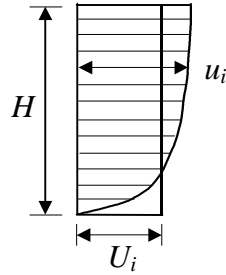


Figura 2. Velocidade Média Vertical.

No modelo 2DH as condições de contorno do fundo e da superfície livre devem ser levadas em consideração. Portanto, as seguintes definições devem ser observadas:

A superfície livre pode ser descrita por

$$S(x_1, x_2, x_3, t) = x_3 - \eta(x_1, x_2, t) = 0 \quad (25)$$

e a sua normal externa como

$$\mathbf{n}^S = (n_1^S, n_2^S, n_3^S) = \frac{\nabla S}{|\nabla S|}; \quad \nabla S = \left( -\frac{\partial \eta}{\partial x_1}, -\frac{\partial \eta}{\partial x_2}, 1 \right) \quad (26)$$

Similarmente, para o fundo podemos escrever que

$$B(x_1, x_2, x_3) = x_3 + h(x_1, x_2) = 0 \quad (27)$$

$$\mathbf{n}^B = (n_1^B, n_2^B, n_3^B) = \frac{\nabla B}{|\nabla B|}; \quad \nabla B = \left( \frac{\partial h}{\partial x_1}, \frac{\partial h}{\partial x_2}, 1 \right) \quad (28)$$

Na superfície livre, as forças atmosféricas por unidade de área (por exemplo, as forças do vento) e as tensões do fluido devem estar em equilíbrio:

$$\tau_{11}n_1^S + \tau_{21}n_2^S + \tau_{31}n_3^S = \tau_1^S \quad (29)$$

$$\tau_{12}n_1^S + \tau_{22}n_2^S + \tau_{32}n_3^S = \tau_2^S \quad (30)$$

e como mencionado anteriormente, a velocidade vertical é igual a

$$u_3(\eta) = \frac{\partial \eta}{\partial t} + u_1(\eta) \frac{\partial \eta}{\partial x_1} + u_2(\eta) \frac{\partial \eta}{\partial x_2} \quad (31)$$

No fundo temos que

$$u_i(-h) = 0 \quad (i = 1, 2, 3) \quad (32)$$

e o equilíbrio de tensões resulta que

$$\tau_{11}n_1^B + \tau_{21}n_2^B + \tau_{31}n_3^B = \tau_1^B \quad (33)$$

$$\tau_{12}n_1^B + \tau_{22}n_2^B + \tau_{32}n_3^B = \tau_2^B \quad (34)$$

Procedendo com a integração vertical da equação da quantidade de movimento (11) na direção de  $x_3$  e integrando o primeiro termo temos

$$\int_{-h}^{\eta} \frac{\partial u_1}{\partial t} dx_3 = \frac{\partial}{\partial t} \int_{-h}^{\eta} u_1 dx_3 - u_1(\eta) \frac{\partial \eta}{\partial t} - u_1(-h) \frac{\partial h}{\partial t} \quad (35)$$

Levando-se em consideração a relação (22) e as condições de contorno (32), a expressão acima leva a

$$\int_{-h}^{\eta} \frac{\partial u_1}{\partial t} dx_3 = \frac{\partial}{\partial t} (HU_1) - u_1(\eta) \frac{\partial \eta}{\partial t} \quad (36)$$

A integração dos termos convectivos conduz a

$$\int_{-h}^{\eta} \frac{\partial}{\partial x_1} (u_1 u_1) dx_3 = \frac{\partial}{\partial x_1} \int_{-h}^{\eta} u_1 u_1 dx_3 - u_1(\eta) u_1(\eta) \frac{\partial \eta}{\partial x_1} - u_1(-h) u_1(-h) \frac{\partial h}{\partial x_1} \quad (37)$$

$$\int_{-h}^{\eta} \frac{\partial}{\partial x_2} (u_2 u_1) dx_3 = \frac{\partial}{\partial x_2} \int_{-h}^{\eta} u_2 u_1 dx_3 - u_2(\eta) u_1(\eta) \frac{\partial \eta}{\partial x_2} - u_2(-h) u_1(-h) \frac{\partial h}{\partial x_2} \quad (38)$$

$$\int_{-h}^{\eta} \frac{\partial}{\partial x_3} (u_3 u_1) dx_3 = u_3(\eta) u_1(\eta) - u_3(-h) u_1(-h) \quad (39)$$

Usando as equações (22-24) e as condições de contorno (32) chegamos a

$$\int_{-h}^{\eta} \frac{\partial}{\partial x_1} (u_1 u_1) dx_3 = \frac{\partial}{\partial x_1} (H U_1 U_1) + \frac{\partial}{\partial x_1} \int_{-h}^{\eta} u_1' u_1' dx_3 - u_1(\eta) u_1(\eta) \frac{\partial \eta}{\partial x_1} \quad (40)$$

$$\int_{-h}^{\eta} \frac{\partial}{\partial x_2} (u_2 u_1) dx_3 = \frac{\partial}{\partial x_2} (H U_2 U_1) + \frac{\partial}{\partial x_2} \int_{-h}^{\eta} u_2' u_1' dx_3 - u_2(\eta) u_1(\eta) \frac{\partial \eta}{\partial x_2} \quad (41)$$

$$\int_{-h}^{\eta} \frac{\partial}{\partial x_3} (u_3 u_1) dx_3 = u_3(\eta) u_1(\eta) \quad (42)$$

A integração das tensões viscosas resulta em

$$\int_{-h}^{\eta} \frac{\partial \tau_{11}}{\partial x_1} dx_3 = \frac{\partial}{\partial x_1} \int_{-h}^{\eta} \tau_{11} dx_3 - \tau_{11}(\eta) \frac{\partial \eta}{\partial x_1} - \tau_{11}(-h) \frac{\partial h}{\partial x_1} \quad (43)$$

$$\int_{-h}^{\eta} \frac{\partial \tau_{21}}{\partial x_2} dx_3 = \frac{\partial}{\partial x_2} \int_{-h}^{\eta} \tau_{21} dx_3 - \tau_{21}(\eta) \frac{\partial \eta}{\partial x_2} - \tau_{21}(-h) \frac{\partial h}{\partial x_2} \quad (44)$$

$$\int_{-h}^{\eta} \frac{\partial \tau_{31}}{\partial x_3} dx_3 = \tau_{31}(\eta) - \tau_{31}(-h) \quad (45)$$

Para os termos restantes podemos escrever que

$$\int_{-h}^{\eta} g \frac{\partial \eta}{\partial x_1} dx_3 = gH \frac{\partial \eta}{\partial x_1} = gH \frac{\partial H}{\partial x_1} - gH \frac{\partial h}{\partial x_1} \quad (46)$$

$$\int_{-h}^{\eta} f u_2 dx_3 = f H U_2 \quad (47)$$

Somando e rearranjando todos os termos integrados temos que

$$\begin{aligned} & \frac{\partial}{\partial t} (H U_1) + \frac{\partial}{\partial x_1} (H U_1 U_1) + \frac{\partial}{\partial x_2} (H U_2 U_1) + gH \frac{\partial H}{\partial x_1} = \\ & u_1(\eta) \left( \frac{\partial \eta}{\partial t} + u_1(\eta) \frac{\partial \eta}{\partial x_1} + u_2(\eta) \frac{\partial \eta}{\partial x_2} - u_3(\eta) \right) + \\ & \frac{1}{\rho} \left( \frac{\partial}{\partial x_1} \int_{-h}^{\eta} (\tau_{11} - \rho u'_1 u'_1) dx_3 + \frac{\partial}{\partial x_2} \int_{-h}^{\eta} (\tau_{21} - \rho u'_2 u'_1) dx_3 \right) + gH \frac{\partial h}{\partial x_1} + f H U_2 + \\ & - \frac{1}{\rho} \left( \tau_{11}(\eta) \frac{\partial \eta}{\partial x_1} + \tau_{21}(\eta) \frac{\partial \eta}{\partial x_2} - \tau_{31}(\eta) \right) - \frac{1}{\rho} \left( \tau_{11}(-h) \frac{\partial h}{\partial x_1} + \tau_{21}(-h) \frac{\partial h}{\partial x_2} + \tau_{31}(-h) \right) \end{aligned} \quad (48)$$

Inserindo as condições de contorno da superfície livre e do fundo chegamos a

$$\begin{aligned} & \frac{\partial}{\partial t} (H U_1) + \frac{\partial}{\partial x_1} (H U_1 U_1) + \frac{\partial}{\partial x_2} (H U_2 U_1) + gH \frac{\partial H}{\partial x_1} = \\ & \frac{1}{\rho} \left( \frac{\partial}{\partial x_1} \int_{-h}^{\eta} (\tau_{11} - \rho u'_1 u'_1) dx_3 + \frac{\partial}{\partial x_2} \int_{-h}^{\eta} (\tau_{21} - \rho u'_2 u'_1) dx_3 \right) + \\ & gH \frac{\partial h}{\partial x_1} + f H U_2 + \frac{1}{\rho} (\tau_1^S |\nabla S| - \tau_1^B |\nabla B|) \end{aligned} \quad (49)$$

Fazendo da mesma forma para a equação da quantidade de movimento (12) na direção de  $x_3$ , as seguintes equações são obtidas:

$$\begin{aligned} & \frac{\partial}{\partial t} (H U_2) + \frac{\partial}{\partial x_1} (H U_1 U_2) + \frac{\partial}{\partial x_2} (H U_2 U_2) + gH \frac{\partial H}{\partial x_2} = \\ & \frac{1}{\rho} \left( \frac{\partial}{\partial x_1} \int_{-h}^{\eta} (\tau_{12} - \rho u'_1 u'_2) dx_3 + \frac{\partial}{\partial x_2} \int_{-h}^{\eta} (\tau_{22} - \rho u'_2 u'_2) dx_3 \right) + \\ & gH \frac{\partial h}{\partial x_2} - f H U_1 + \frac{1}{\rho} (\tau_2^S |\nabla S| - \tau_2^B |\nabla B|) \end{aligned} \quad (50)$$

As tensões no fundo  $\tau_i^B$  podem ser modeladas por

$$\tau_i^B |\nabla B| = \rho \gamma U_i \quad (51)$$

onde  $\gamma$  é o coeficiente de fricção. Este coeficiente é expresso pela fórmula de Chézy

$$\gamma = \frac{g(U_1^2 + U_2^2)^{1/2}}{C^2} \quad (52)$$

onde  $C$  é o coeficiente de Chézy.

As componentes horizontais das tensões cisalhantes do vento  $\tau_i^w$  podem ser modeladas por

$$\tau_i^w = \tau_i^s |\nabla S| = \rho_a C_D u_i^0 \quad (53)$$

onde  $\rho_a$  é a massa específica do ar,  $C_D$  é o coeficiente de arraste do vento e  $u_i^0$  são as componentes horizontais da velocidade do vento medidas num ponto 10 metros acima da superfície livre da água [12].

Levando-se em conta a turbulência, o modelo utiliza-se de um incremento no termo de difusão horizontal das equações da quantidade de movimento, o que é feito através do conceito de viscosidade turbulenta  $\nu_t$ . Esse parâmetro, por sua vez, é estimado levando-se em conta as características do escoamento, por exemplo, através do conceito de comprimento de mistura [13]. Sendo assim as equações da quantidade de movimento podem ser reescritas como

$$\begin{aligned} \frac{\partial}{\partial t} (HU_1) + \frac{\partial}{\partial x_i} (HU_i U_1) + gH \frac{\partial H}{\partial x_1} = \\ \frac{1}{\rho} \left( \frac{\partial}{\partial x_i} \left( \nu_t H \frac{\partial U_1}{\partial x_i} \right) \right) + gH \frac{\partial h}{\partial x_1} + fHU_2 - \gamma U_1 + \frac{\tau_1^w}{\rho} \end{aligned} \quad (54)$$

$$\begin{aligned} \frac{\partial}{\partial t} (HU_2) + \frac{\partial}{\partial x_i} (HU_i U_2) + gH \frac{\partial H}{\partial x_2} = \\ \frac{1}{\rho} \left( \frac{\partial}{\partial x_i} \left( \nu_t H \frac{\partial U_2}{\partial x_i} \right) \right) + gH \frac{\partial h}{\partial x_2} - fHU_1 - \gamma U_2 + \frac{\tau_2^w}{\rho} \end{aligned} \quad (55)$$

onde  $\nu_t$  é a viscosidade turbulenta.

A equação da continuidade, após a integração vertical, já determinada anteriormente e repetida aqui:

$$\int_{-h}^{\eta} \frac{\partial u_i}{\partial x_i} dx_3 = \frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x_1} \int_{-h}^{\eta} u_1 dx_3 + \frac{\partial}{\partial x_2} \int_{-h}^{\eta} u_2 dx_3 = 0 \quad (56)$$

Usando a expressão (22) e assumindo que o fundo não muda com o tempo, o que é equivalente a dizer que

$$\frac{\partial \eta}{\partial t} \equiv \frac{\partial H}{\partial t} \quad (57)$$

e a equação (56) pode ser reescrita como

$$\frac{\partial H}{\partial t} + \frac{\partial}{\partial x_i} (HU_i) = 0 \quad (58)$$

Finalmente, o modelo 2DH pode ser escrito na forma de três equações e três incógnitas ( $HU_1, HU_2, H$ ):

$$\boxed{\begin{aligned} \frac{\partial}{\partial t} (HU_1) + \frac{\partial}{\partial x_i} (HU_i U_1) + gH \frac{\partial H}{\partial x_1} = \\ \frac{1}{\rho} \left( \frac{\partial}{\partial x_i} \left( \nu_t H \frac{\partial U_1}{\partial x_i} \right) \right) + gH \frac{\partial h}{\partial x_1} + fHU_2 - \gamma U_1 + \frac{\tau_1^w}{\rho} \end{aligned}} \quad (59)$$

$$\boxed{\begin{aligned} \frac{\partial}{\partial t} (HU_2) + \frac{\partial}{\partial x_i} (HU_i U_2) + gH \frac{\partial H}{\partial x_2} = \\ \frac{1}{\rho} \left( \frac{\partial}{\partial x_i} \left( \nu_t H \frac{\partial U_2}{\partial x_i} \right) \right) + gH \frac{\partial h}{\partial x_2} - fHU_1 - \gamma U_2 + \frac{\tau_2^w}{\rho} \end{aligned}} \quad (60)$$

$$\boxed{\frac{\partial H}{\partial t} + \frac{\partial}{\partial x_i} (HU_i) = 0} \quad (61)$$

## 2.2.1 FORMA DIVERGENTE DAS EQUAÇÕES DE ÁGUAS RASAS

As equações (59-61) podem ser escritas de forma mais compacta como

$$\mathbf{U}_{,t} + \mathbf{F}_{i,i} = \mathbf{F} \quad (62)$$

onde  $\mathbf{U}^T = [HU_1 \quad HU_2 \quad H]$  denota o vetor das variáveis conservativas,  $\mathbf{F}_i$  ( $i = 1, 2$ ) são os fluxos hidráulicos:

$$\mathbf{F}_1^T = \left[ HU_1^2 + \frac{1}{2}gH^2 \quad HU_1U_2 \quad HU_1 \right] \quad (63)$$

$$\mathbf{F}_2^T = \left[ HU_1U_2 \quad HU_2^2 + \frac{1}{2}gH^2 \quad HU_2 \right] \quad (64)$$

a vírgula inferior representa a diferenciação parcial e  $\mathbf{F}$  é o termo fonte correspondendo aos termos do lado direito das equações (59-61). O sistema (62) é escrito na chamada forma divergente ou forma conservativa e representa um sistema de equações parabólicas incompletas. Por outro lado, a equação reduzida

$$\mathbf{U}_{,t} + \mathbf{F}_{i,i} = \mathbf{0} \quad (65)$$

representa um sistema de equações hiperbólicas para o qual o termo convectivo é expresso na forma divergente.

## 2.2.2 FORMA ADVECTIVA DAS EQUAÇÕES DE ÁGUAS RASAS

Alternativamente à forma divergente (62), as equações de águas rasas podem ser escritas na forma advectiva se a regra da cadeia for aplicada. Das definições de fluxos hidráulicos (63, 64) temos que

$$\mathbf{F}_{i,i}(\mathbf{U}) = \frac{\partial \mathbf{F}_i}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial x_i} = \mathbf{A}_i \mathbf{U}_{,i} \quad (66)$$

onde  $\mathbf{U}_{,i} = \frac{\partial \mathbf{U}}{\partial x_i}$  e  $\mathbf{A}_i$  são as matrizes jacobianas de fluxo:

$$\mathbf{A}_1 = \begin{bmatrix} 2U_1 & 0 & gH - U_1^2 \\ U_2 & U_1 & -U_1U_2 \\ 1 & 0 & 0 \end{bmatrix}; \quad \mathbf{A}_2 = \begin{bmatrix} U_2 & U_1 & -U_1U_2 \\ 0 & 2U_2 & gH - U_2^2 \\ 0 & 1 & 0 \end{bmatrix} \quad (67)$$

Usando estas definições, a equação (62) pode ser escrita na forma matricial como

$$\mathbf{U}_{,t} + \mathbf{A}_i \mathbf{U}_{,i} = \mathbf{F} \quad (68)$$

Outra maneira de representar a equação acima é

$$\mathbf{U}_{,t} + \mathbf{A} \cdot \nabla \mathbf{U} = \mathbf{F} \quad (69)$$

com

$$\mathbf{U}_{,t} = \frac{\partial \mathbf{U}}{\partial t}; \quad \mathbf{A}^T = [\mathbf{A}_1 \quad \mathbf{A}_2]; \quad \nabla \mathbf{U} = \begin{bmatrix} \mathbf{U}_{,1} \\ \mathbf{U}_{,2} \end{bmatrix}; \quad \mathbf{A} \cdot \nabla \mathbf{U} = \mathbf{A}^T \nabla \mathbf{U} \quad (70)$$

A notação do produto escalar  $\mathbf{A} \cdot \nabla \mathbf{U}$  é usada para representar o produto matricial em analogia à equação de transporte escalar. Este termo funciona como um termo advectivo generalizado. Como as matrizes  $\mathbf{A}_i$  são não-simétricas, a equação (69) representa um sistema não-simétrico parabólico incompleto, escrito em função das variáveis  $(HU_1, HU_2, H)$ .

Qualquer mudança de variáveis  $\mathbf{U} \rightarrow \mathbf{V}$  pode ser obtida através das seguintes relações:

$$\mathbf{U}_{,t} = \frac{\partial \mathbf{U}}{\partial \mathbf{V}} \frac{\partial \mathbf{V}}{\partial t} = \mathbf{A}_0 \mathbf{V}_{,t} \quad (71)$$

$$\mathbf{F}_{i,i} = \frac{\partial \mathbf{F}_i}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{V}} \frac{\partial \mathbf{V}}{\partial x_i} = \mathbf{A}_i \mathbf{A}_0 \mathbf{V}_{,i} \quad (72)$$

Substituindo estes resultados na equação (68) chegamos a

$$\mathbf{A}_0 \mathbf{V}_{,t} + \tilde{\mathbf{A}}_i \mathbf{V}_{,i} = \mathbf{F} \quad (73)$$

onde  $\tilde{\mathbf{A}}_i = \mathbf{A}_i \mathbf{A}_0$ . Pré-multiplicando por  $\mathbf{A}_0^{-1}$  obtemos o seguinte sistema

$$\mathbf{V}_{,t} + \bar{\mathbf{A}}_i \mathbf{V}_{,i} = \bar{\mathbf{F}} \quad (74)$$

com  $\bar{\mathbf{A}}_i = \mathbf{A}_0^{-1} \mathbf{A}_i \mathbf{A}_0$  e  $\bar{\mathbf{F}} = \mathbf{A}_0^{-1} \mathbf{F}$ . Note que  $\mathbf{A}_0^{-1} = \mathbf{V}_{,U}$ .

Por exemplo, a forma advectiva é frequentemente descrita em função das variáveis primitivas  $\mathbf{V} = (U_1, U_2, H)$ . Neste caso temos que

$$\mathbf{A}_0 = \begin{bmatrix} H & 0 & U_1 \\ 0 & H & U_2 \\ 0 & 0 & 1 \end{bmatrix}; \quad \mathbf{A}_0^{-1} = \begin{bmatrix} 1/H & 0 & -U_1/H \\ 0 & 1/H & -U_2/H \\ 0 & 0 & 1 \end{bmatrix} \quad (75)$$

$$\bar{\mathbf{A}}_1 = \begin{bmatrix} U_1 & 0 & g \\ 0 & U_1 & 0 \\ H & 0 & U_1 \end{bmatrix}; \quad \bar{\mathbf{A}}_2 = \begin{bmatrix} U_2 & 0 & 0 \\ 0 & U_2 & g \\ 0 & H & U_2 \end{bmatrix} \quad (76)$$

$$\bar{\mathbf{F}} = \begin{bmatrix} g \frac{\partial h}{\partial x_1} + fU_2 - \frac{\gamma}{H} U_1 + \frac{1}{\rho H} \tau_1^s + \frac{1}{\rho H} \left( \frac{\partial}{\partial x_i} \left( vH \frac{\partial U_1}{\partial x_i} \right) \right) \\ g \frac{\partial h}{\partial x_2} - fU_1 - \frac{\gamma}{H} U_2 + \frac{1}{\rho H} \tau_2^s + \frac{1}{\rho H} \left( \frac{\partial}{\partial x_i} \left( vH \frac{\partial U_2}{\partial x_i} \right) \right) \\ 0 \end{bmatrix} \quad (77)$$

### 2.2.3 FORMAS SIMÉTRICAS DAS EQUAÇÕES DE ÁGUAS RASAS

As formas simétricas são bastante desejadas. Primeiro porque possuem propriedades de estabilidade [14, 15] caso sejam derivadas do ponto de vista da entropia e segundo porque os métodos estabilizados, por exemplo, o método *Streamline Upwind Petrov-Galerkin (SUPG)*, podem ser aplicados após a devida transformação de variáveis [16].

### 2.2.3.1 VARIÁVEIS DE ENTROPIA

As formas simétricas dos sistemas hiperbólicos podem ser construídas uma vez que a correspondente função de entropia para os sistemas hiperbólicos não-simétricos originais tenha sido derivada. Para as equações de águas rasas [16, 17, 18], a função de entropia  $\mathbf{U}(\mathbf{U})$  pode ser definida como

$$\mathbf{U}(\mathbf{U}) = \frac{H}{2}(c^2 + U_1^2 + U_2^2) \quad (78)$$

onde  $c = \sqrt{gH}$  representa a velocidade de propagação da onda gravitacional. Nesta situação, a mudança do vetor  $\mathbf{U}$  das variáveis conservativas para as chamadas variáveis de entropia é dada por  $\mathbf{V}^T(\mathbf{U}) = \mathbf{U}_{,\mathbf{U}}$ :

$$\mathbf{V}^T = \left[ U_1 \quad U_2 \quad \left( c^2 - \frac{1}{2}(U_1^2 + U_2^2) \right) \right] \quad (79)$$

A matriz Hessiana  $\mathbf{U}_{,\mathbf{UU}}$  possui a propriedade de ser positiva definida, conseqüentemente a transformação de coordenadas é bem definida. Isto nos permite escrever que

$$\mathbf{U}_{,\mathbf{UU}} = \mathbf{A}_0^{-1} = \mathbf{V}_{,\mathbf{U}} = \frac{1}{H} \begin{bmatrix} 1 & 0 & -U_1 \\ 0 & 1 & -U_2 \\ -U_1 & -U_2 & (c^2 + U_1^2 + U_2^2) \end{bmatrix} \quad (80)$$

$$(\mathbf{U}_{,\mathbf{UU}})^{-1} = \mathbf{A}_0 = \mathbf{U}_{,\mathbf{V}} = \frac{1}{g} \begin{bmatrix} c^2 + U_1^2 & U_1 U_2 & U_1 \\ U_1 U_2 & c^2 + U_2^2 & U_2 \\ U_1 & U_2 & 1 \end{bmatrix} \quad (81)$$

O sistema pode ser simetrizado se  $\mathbf{U}$  for uma função convexa e os correspondentes fluxos entrópicos  $\sigma_i$  ( $i = 1, 2$ ) existirem e satisfizerem a relação:

$$\frac{\partial \sigma_i}{\partial \mathbf{U}} = \mathbf{V}^T \mathbf{A}_i \quad (82)$$

Para a função de entropia definida em (78) os fluxos entrópicos são

$$\sigma_i = HU_i \left( c^2 + \frac{1}{2}(U_1^2 + U_2^2) \right) \quad (83)$$

Com as definições acima, obtemos o seguinte sistema:

$$\mathbf{A}_0 \mathbf{V}_{,i} + \tilde{\mathbf{A}}_i \mathbf{V}_{,i} = \mathbf{F} \quad (84)$$

onde

$$\tilde{\mathbf{A}}_1 = \frac{1}{g} \begin{bmatrix} U_1(3c^2 + U_1^2) & U_2(c^2 + U_1^2) & c^2 + U_1^2 \\ U_2(c^2 + U_1^2) & U_1(c^2 + U_2^2) & U_1 U_2 \\ c^2 + U_1^2 & U_1 U_2 & U_1 \end{bmatrix} \quad (85)$$

$$\tilde{\mathbf{A}}_2 = \frac{1}{g} \begin{bmatrix} U_2(c^2 + U_1^2) & U_1(c^2 + U_2^2) & U_1 U_2 \\ U_1(c^2 + U_2^2) & U_2(3c^2 + U_2^2) & c^2 + U_2^2 \\ U_1 U_2 & c^2 + U_2^2 & U_2 \end{bmatrix} \quad (86)$$

### 2.2.3.2 VARIÁVEIS DE VELOCIDADE

Outra forma simétrica advectiva pode ser obtida utilizando variáveis de velocidade [3, 4, 5]. Definamos a mudança de variáveis  $\mathbf{U} \rightarrow \mathbf{V} = (U_1, U_2, \theta)$ , onde  $\theta = 2c$ . Para esta mudança de variáveis podemos escrever que

$$\mathbf{A}_0 = \mathbf{U}_{,v} = \frac{c}{g} \begin{bmatrix} c & 0 & U_1 \\ 0 & c & U_2 \\ 0 & 0 & 1 \end{bmatrix}; \quad \mathbf{A}_0^{-1} = \mathbf{V}_{,u} = \frac{g}{c} \begin{bmatrix} 1/c & 0 & -U_1/c \\ 0 & 1/c & -U_2/c \\ 0 & 0 & 1 \end{bmatrix} \quad (87)$$

$$\tilde{\mathbf{A}}_1 = \mathbf{A}_1 \mathbf{A}_0 = \frac{c}{g} \begin{bmatrix} 2cU_1 & 0 & c^2 + U_1^2 \\ cU_2 & cU_1 & U_1 U_2 \\ c & 0 & U_1 \end{bmatrix} \quad (88)$$

$$\tilde{\mathbf{A}}_2 = \mathbf{A}_2 \mathbf{A}_0 = \frac{c}{g} \begin{bmatrix} cU_2 & cU_1 & U_1U_2 \\ 0 & 2cU_2 & c^2 + U_2^2 \\ 0 & c & U_2 \end{bmatrix} \quad (89)$$

$$\bar{\mathbf{A}}_1 = \begin{bmatrix} U_1 & 0 & c \\ 0 & U_1 & 0 \\ c & 0 & U_1 \end{bmatrix}; \quad \bar{\mathbf{A}}_2 = \begin{bmatrix} U_2 & 0 & 0 \\ 0 & U_2 & c \\ 0 & c & U_2 \end{bmatrix} \quad (90)$$

Portanto, as equações de águas rasas escritas em função de variáveis de velocidade/celeridade são dadas por

$$\mathbf{V}_{,i} + \bar{\mathbf{A}}_i \mathbf{V}_{,i} = \bar{\mathbf{F}} \quad (91)$$

com

$$\bar{\mathbf{F}} = \begin{bmatrix} g \frac{\partial h}{\partial x_1} + fU_2 - \frac{\gamma}{H} U_1 + \frac{1}{\rho H} \tau_1^s + \frac{1}{\rho H} \left( \frac{\partial}{\partial x_i} \left( \nu H \frac{\partial U_1}{\partial x_i} \right) \right) \\ g \frac{\partial h}{\partial x_2} - fU_1 - \frac{\gamma}{H} U_2 + \frac{1}{\rho H} \tau_2^s + \frac{1}{\rho H} \left( \frac{\partial}{\partial x_i} \left( \nu H \frac{\partial U_2}{\partial x_i} \right) \right) \\ 0 \end{bmatrix} \quad (92)$$

Por conveniência, serão usadas as notações  $\mathbf{U} = \mathbf{V}$ ,  $\mathbf{A} = \bar{\mathbf{A}}$ ,  $\mathbf{F} = \bar{\mathbf{F}}$  e desta forma as equações de águas rasas serão reescritas em função das variáveis de velocidade como

$$\mathbf{U}_{,i} + \mathbf{A} \cdot \nabla \mathbf{U} = \mathbf{F} \quad (93)$$

Os termos viscosos incluídos em  $\mathbf{F}$  podem ser escritos sob a forma do operador de difusão  $\nabla \cdot (\mathbf{K} \nabla \mathbf{U})$  e tratados implicitamente do lado esquerdo das equações. Isto pode ser feito escrevendo os termos viscosos como

$$\frac{1}{\rho H} \left( \frac{\partial}{\partial x_i} \left( \nu H \frac{\partial U_j}{\partial x_i} \right) \right) = \frac{1}{\rho} \left( \frac{1}{H} \nabla H \cdot (\nu \nabla U_j) + \nabla \cdot (\nu \nabla U_j) \right) \quad (94)$$

Assumindo que as quantidades  $\frac{1}{H} \nabla H \cdot (\nu \nabla U_i)$  são pequenas comparadas aos termos convectivos, a equação (93) toma a forma

$$\mathbf{U}_{,t} + \mathbf{A} \cdot \nabla \mathbf{U} - \nabla \cdot (\mathbf{K} \nabla \mathbf{U}) = \mathbf{F} \quad (95)$$

onde  $\mathbf{K}$  é a matriz de difusividade do sistema

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix}; \quad \mathbf{K}_{ij} = \delta_{ij} \frac{\nu}{\rho} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (96)$$

sendo que  $\delta_{ij}$  denota o delta de Kronecker, isto é,  $\delta_{ij} = 1$  para  $i = j$  e  $\delta_{ij} = 0$  para os outros casos.

Os termos fonte são agora dados por

$$\mathbf{F} = \begin{bmatrix} g \frac{\partial h}{\partial x_1} + f U_2 - \frac{\gamma}{H} U_1 + \frac{1}{\rho H} \tau_1^s \\ g \frac{\partial h}{\partial x_2} - f U_1 - \frac{\gamma}{H} U_2 + \frac{1}{\rho H} \tau_2^s \\ 0 \end{bmatrix} \quad (97)$$

### 3. FORMULAÇÃO SEMI-DISCRETA ESTABILIZADA DO MÉTODO DOS ELEMENTOS FINITOS

A formulação de elementos finitos mais freqüentemente utilizada é a versão semi-discreta. De fato, este é o método onde somente as variáveis espaciais são tratadas por aproximações de elementos finitos, enquanto que as variáveis temporais são aproximadas por operadores de diferenças finitas. Para problemas auto-adjuntos o método de Galerkin apresenta a melhor aproximação. No entanto, na presença de termos convectivos o método não é capaz de evitar oscilações espúrias. Em outras palavras, falta estabilidade para o método de Galerkin. O método *Streamline Upwind Petrov-Galerkin (SUPG)*, proposto em [6] apresenta uma excelente melhora sobre a formulação pura de Galerkin. É uma maneira consistente e eficiente de introduzir a quantidade necessária de difusão, requerida para eliminar as oscilações espúrias. Este método foi inicialmente desenvolvido para as equações escalares de convecção-difusão. Posteriormente foi generalizado para sistemas multi-dimensionais [19]. O operador adicionado no método *SUPG* controla as derivadas ao longo das linhas de corrente, proporcionando boas propriedades de estabilidade e precisão além de melhorar a convergência sobre o método puro de Galerkin. Contudo, na vizinhança de regiões com elevados gradientes, a solução aproximada pode apresentar flutuações. Faz-se necessário um operador extra, atuando nas regiões de mais alto gradiente. Este operador deve ser proporcional ao resíduo e desaparecer nas regiões onde a solução é suave. O método proposto em [2], *Consistent Approximate Upwind (CAU)*, é um destes operadores de captura de choque ou captura de descontinuidade. Considere o domínio bidimensional  $\Omega$  com fronteira  $\Gamma$  (Figura 3) e o intervalo de tempo  $[0, T] \in \mathfrak{R}^+$ . Conforme visto, o modelo 2DH de águas rasas, escrito em termos das variáveis de velocidade e celeridade é dado pelas seguintes equações:

$$\mathbf{U}_{,t} + \mathbf{A} \cdot \nabla \mathbf{U} - \nabla \cdot (\mathbf{K} \nabla \mathbf{U}) = \mathbf{F} \quad \text{em} \quad \Omega \times [0, T] \quad (98)$$

onde  $\mathbf{A} \cdot \nabla \mathbf{U}$  funciona como um termo de convecção generalizado e  $\nabla \cdot (\mathbf{K} \nabla \mathbf{U})$  funciona como um operador de difusão generalizado.

Na formulação semi-discreta o domínio espacial é particionado em  $nel$  elementos  $\Omega^e$  de tal forma que

$$\Omega = \bigcup_{e=1}^{nel} \Omega^e; \quad \Omega_n^i \cap \Omega_n^j = \emptyset \quad \text{para} \quad i \neq j \quad (99)$$

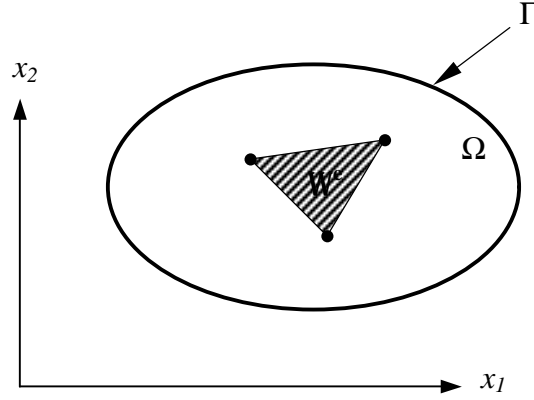


Figura 3. Domínio bidimensional.

Os subespaços de elementos finitos são dados por

$$\hat{U}_n^h \equiv \left\{ \hat{U}^h; \hat{U}^h \in (C^0(\Omega))^3; \hat{U}^h|_{\Omega^e} \in (P^k(\Omega^e))^3; \hat{U}^h|_{\Gamma} = \mathbf{0} \right\} \quad (100)$$

$$U_n^h \equiv \left\{ U^h; U^h \in (C^0(\Omega))^3; U^h|_{\Omega^e} \in (P^k(\Omega^e))^3; U^h|_{\Gamma} = \mathbf{g} \right\} \quad (101)$$

onde  $\hat{U}_n^h$  é o subespaço das funções de peso e  $U_n^h$  é o subespaço das funções admissíveis, ambos considerados no tempo  $t = t_n$ .  $P^k$  representa os polinômios de grau menor ou igual a  $k$ ,  $C^0$  representa o conjunto de funções cuja 1ª derivada é contínua e  $\mathbf{g}$  são as condições de contorno prescritas.

O método de Galerkin consiste em que para um dado  $\mathbf{U}^h(t_0)$  e para cada  $t = t_n$ ,  $n = 1, 2, \dots$  seja encontrado  $\mathbf{U}^h \in U_n^h$  tal que para todo  $\hat{\mathbf{U}}^h \in \hat{U}_n^h$  a seguinte equação variacional seja satisfeita:

$$\int_{\Omega} \mathbf{R} \cdot \hat{\mathbf{U}}^h d\Omega = 0 \quad (102)$$

onde  $\mathbf{R}$  é o resíduo associado com a solução aproximada  $\mathbf{U}^h$ :

$$\mathbf{R}(\mathbf{U}^h) = \mathbf{U}_{,i}^h + \mathbf{A}(\mathbf{U}^h) \cdot \nabla \mathbf{U}^h - \nabla \cdot (\mathbf{K} \nabla \mathbf{U}^h) - \mathbf{F}(\mathbf{U}^h) \quad (103)$$

Conforme mencionado previamente, para problemas dominados pela convecção, o método puro de Galerkin apresenta oscilações espúrias devido à falta de estabilidade da correspondente formulação variacional. A seguir são apresentadas algumas técnicas de estabilização da formulação semi-discreta de Galerkin.

O problema das oscilações espúrias pode ser contornado utilizando a formulação de elementos finitos estabilizada *SUPG*. Este procedimento de estabilização pode ser obtido adicionando à formulação pura de Galerkin o seguinte termo definido como:

$$\mathbf{T}_{SUPG} = \sum_{e=1}^{nel} \int_{\Omega^e} \mathbf{R} \cdot (\hat{\mathbf{0}} \mathbf{A} \cdot \nabla \hat{\mathbf{U}}^h) d\Omega \quad (104)$$

onde  $\hat{\mathbf{0}}$  é uma matriz (3 x 3) simétrica positiva. Esta matriz controla a quantidade necessária de difusão a ser adicionada ao sistema. Este é o ponto chave para o sucesso do método *SUPG*. Usando a definição encontrada em [20] temos que

$$\hat{\mathbf{0}} = \left[ \frac{\partial \xi_i}{\partial x_j} \frac{\partial \xi_i}{\partial x_k} \mathbf{A}_j \mathbf{A}_k + \left( \frac{\partial \xi_i}{\partial x_k} \frac{\partial \xi_j}{\partial x_l} \frac{\partial \xi_i}{\partial x_m} \frac{\partial \xi_j}{\partial x_n} \right) \mathbf{K}_{kl} \mathbf{K}_{mn} \right]^{-1/2} \quad (105)$$

onde  $x_i (i=1,2)$  refere-se às coordenadas globais,  $\xi_i (i=1,2)$  refere-se às coordenadas locais de  $\Omega^e$ ,  $\mathbf{A}_k$  são as matrizes Jacobianas de fluxo e  $\mathbf{K}_{kl}$  são as matrizes difusivas. Como pode ser visto pela equação (104) o operador *SUPG* atua sobre linhas de corrente generalizadas.

Para soluções descontínuas (choques) de problemas hiperbólicos, ou mesmo para soluções aplicadas às fortes camadas limites, relacionadas a problemas parabólicos incompletos altamente dominados pela convecção, é necessário um termo adicional de estabilização caso se queira uma precisão maior na aproximação numérica. No contexto dos métodos variacionais consistentes de resíduos ponderados de Petrov-Galerkin, isto pode ser obtido através da utilização do método *CAU*. Para este método, um termo de captura de choque é adicionado ao método *SUPG*.

$$\mathbf{T}_{CAU} = \sum_{e=1}^{nel} \int_{\Omega^e} \mathbf{R} \cdot (\mathbf{t}_c (\mathbf{A} - \hat{\mathbf{A}}) \cdot \nabla \hat{\mathbf{U}}^h) d\Omega \quad (106)$$

onde  $\mathbf{t}_c$  é uma matriz (3 x 3) simétrica positiva.

Na implementação do método *CAU*, a matriz auxiliar  $\hat{\mathbf{A}}(\mathbf{U}^h)$  é construída de tal forma que

$$(\mathbf{A} - \hat{\mathbf{A}}) \cdot \nabla \hat{\mathbf{U}}^h = \mathbf{R} \quad \mathbf{U}^h \xrightarrow{h \rightarrow 0} \mathbf{U} \Leftrightarrow \hat{\mathbf{A}} \xrightarrow{h \rightarrow 0} \mathbf{A} \quad (107)$$

Esta construção garante que sempre o termo quadrático positivo dos resíduos ponderados seja adicionado à forma variacional, visto que para  $\hat{\mathbf{U}}^h = \mathbf{U}^h$  temos que

$$\mathbf{T}_{CAU} = \|\mathbf{R}\|_{\tau_c}^2 = \sum_{e=1}^{nel} \int_{\Omega^e} \mathbf{R}^T \mathbf{t}_c \mathbf{R} \, d\Omega \quad (108)$$

As definições para  $\mathbf{t}_c$  e para as matrizes  $(\mathbf{A} - \hat{\mathbf{A}})$  são deduzidas para as equações de *Euler* e *Navier-Stokes* em [21] e de forma semelhante nos conduz, no presente caso, à forma reduzida

$$\mathbf{T}_{CAU} = \sum_{e=1}^{nel} \int_{\Omega_e} \bar{\tau} \nabla \mathbf{U}^h \cdot \nabla \hat{\mathbf{U}}^h \, d\Omega \quad (109)$$

onde

$$\bar{\tau} = \begin{cases} \max \left( 0, \frac{|\mathbf{R}|}{|\nabla_{\xi} \mathbf{U}^h|} - \frac{(\mathbf{U}^h + \mathbf{A} \cdot \nabla \mathbf{U}^h)^T \hat{\mathbf{O}} \mathbf{R}}{|\nabla \mathbf{U}^h|^2} \right), & \text{se } \nabla \mathbf{U}^h \neq \mathbf{0} \\ 0 & , \text{ se } \nabla \mathbf{U}^h = \mathbf{0} \end{cases} \quad (110)$$

Na expressão acima,  $\nabla_{\xi}$  representa o gradiente generalizado escrito em variáveis locais:

$$\nabla_{\xi} \mathbf{U}^h = \begin{bmatrix} \mathbf{U}^h_{,\xi_1} \\ \mathbf{U}^h_{,\xi_2} \end{bmatrix} \quad (111)$$

e  $\hat{\mathbf{O}}$  é a matriz definida na equação (105).

### 3.1 ASPECTOS COMPUTACIONAIS DA APROXIMAÇÃO SEMI-DISCRETA

Na versão da formulação semi-discreta, os subespaços de elementos finitos dependem somente das variáveis espaciais. Dessa forma, para um dado tempo  $t = t_n$ , as funções de aproximação são dadas por

$$\mathbf{U}^h(t_n) = \sum_{j=1}^{nnode} \phi_j(x_1, x_2) \mathbf{U}_{j,n}^h \quad (112)$$

$$\hat{\mathbf{U}}^h(t_n) = \sum_{i=1}^{nnode} \phi_i(x_1, x_2) \hat{\mathbf{U}}_{i,n}^h \quad (113)$$

onde  $\mathbf{U}_{j,n}^h$  são as incógnitas de  $\mathbf{U}^h(t_n)$  para o nó  $j$ ,  $\hat{\mathbf{U}}_{i,n}^h$  são os valores nodais de  $\hat{\mathbf{U}}^h(t_n)$  e  $\phi_j$  é a função de interpolação para o nó  $j$ . O índice subscrito  $n$  refere-se ao tempo  $t = t_n$  e  $nnode$  representa o número total de nós.

Substituindo as aproximações (112 e 113) na formulação variacional (102), o seguinte sistema de equações diferenciais ordinárias é obtido:

$$\mathbf{M} \dot{\mathbf{U}}_n + \mathbf{K} \mathbf{U}_n = \mathbf{F} \quad (114)$$

onde  $\mathbf{U}_n$  é o vetor dos valores nodais de  $\mathbf{U}^h(t_n)$ ,  $\dot{\mathbf{U}}_n$  representa a derivada temporal

$\left( \frac{\partial \mathbf{U}^h}{\partial t} \right)_{t=t_n}$  e  $\mathbf{M}$ ,  $\mathbf{K}$ ,  $\mathbf{F}$  são iguais a

$$\mathbf{M} = \mathbf{M}^G + \mathbf{M}^{PG} \quad (\text{Matriz de massa}) \quad (115)$$

$$\mathbf{K} = \mathbf{K}^G + \mathbf{K}^{PG} + \mathbf{K}^{DC} \quad (\text{Matriz de rigidez}) \quad (116)$$

$$\mathbf{F} = \mathbf{F}^G + \mathbf{F}^{PG} \quad (\text{Vetor de forças}) \quad (117)$$

Os índices sobrescritos  $G$ ,  $PG$ ,  $DC$ , referem-se às parcelas provenientes dos termos de Galerkin, Petrov-Galerkin e  $CAU$  respectivamente. Observe que, enquanto todo o sistema é afetado pelas funções ponderadas  $SUPG$ , o operador de captura de descontinuidade atua somente sobre a matriz  $\mathbf{K}$ .

As derivadas no tempo podem ser aproximadas pelo método das diferenças finitas utilizando uma regra trapezoidal:

$$\mathbf{U}_{n+1} = \mathbf{U}_n + \Delta t \dot{\mathbf{U}}_{n+\alpha} \quad (118)$$

$$\dot{\mathbf{U}}_{n+\alpha} = (1-\alpha) \dot{\mathbf{U}}_n + \alpha \dot{\mathbf{U}}_{n+1} \quad (119)$$

O parâmetro  $\alpha$  controla a precisão e estabilidade do método. A Tabela 1 mostra que dependendo do valor de  $\alpha$ , os seguintes métodos são obtidos:

Tabela 1. Métodos disponíveis.

$\alpha$	Método
0	Euler Forward
$\frac{1}{2}$	Crank Nicolson
1	Euler Backward

Para  $\alpha = 0$ , o método é considerado explícito, caso contrário o método é implícito.

### 3.2 SOLUÇÃO DO SISTEMA DE EQUAÇÕES NÃO-LINEARES

O sistema de equações (114) pode ser resolvido utilizando o algoritmo preditor multi-corretor, derivado dos métodos de Newton-Raphson. Para este sistema de equações este algoritmo tem a seguinte forma

$$\left. \begin{aligned} \mathbf{U}_{n+1}^0 &= \mathbf{U}_n + \Delta t (1-\alpha) \dot{\mathbf{U}}_n \\ \dot{\mathbf{U}}_{n+1}^0 &= \mathbf{0} \end{aligned} \right\} \text{ (fase preditora)} \quad (120)$$

Para  $i = 0, 1, 2, \dots$

$$\left. \begin{aligned} \mathbf{R}^i &= \mathbf{F}_{n+1} - \mathbf{M} \dot{\mathbf{U}}_{n+1}^i - \mathbf{K} \mathbf{U}_{n+1}^i \\ \mathbf{M}^* \Delta \dot{\mathbf{U}}_{n+1}^i &= \mathbf{R}^i \\ \dot{\mathbf{U}}_{n+1}^{i+1} &= \dot{\mathbf{U}}_{n+1}^i + \Delta \dot{\mathbf{U}}_{n+1}^i \\ \mathbf{U}_{n+1}^{i+1} &= \mathbf{U}_{n+1}^0 + \alpha \Delta t \dot{\mathbf{U}}_{n+1}^{i+1} \end{aligned} \right\} \text{ (fase multi-corretora)} \quad (121)$$

onde

$$\mathbf{M}^* = \mathbf{M} + \alpha \Delta t \mathbf{K} \quad (122)$$

Como pode ser visto, cada iteração não-linear envolve três etapas:

$$\text{Avaliação do resíduo não-linear: } \mathbf{R}^i(\mathbf{U}^i) \quad (123)$$

$$\text{Solução do sistema: } \mathbf{K}(\mathbf{U}^i) \Delta \mathbf{U}^i = \mathbf{R}^i \quad (124)$$

$$\text{Atualização da solução: } \mathbf{U}^{i+1} = \mathbf{U}^i + \Delta \mathbf{U}^i \quad (125)$$

Dada a tolerância  $\varepsilon$ , o processo não-linear encerra quando

$$\|\mathbf{R}^i\| < \varepsilon \|\mathbf{R}^0\| \quad (126)$$

Os solucionadores iterativos são especialmente adequados para este tipo de algoritmo, pois a precisão requerida na resolução da equação (124) depende do resíduo não-linear  $\mathbf{R}^i$ . O número de iterações lineares pode ser bastante reduzido utilizando uma variável de tolerância  $\tilde{\alpha}$  para controlar o resíduo linear:

$$(\mathbf{K} \Delta \mathbf{U}^i - \mathbf{R}^i) < \tilde{\alpha} \mathbf{R}^i \quad (127)$$

Este procedimento leva aos *Métodos de Newton Inexatos* [22, 23]. A variável de tolerância  $\tilde{\alpha}$  pode ser avaliada de acordo com a seguinte expressão [24]:

$$\tilde{\alpha} = \max \left( \tilde{\alpha}_{\min}, \min \left( \tilde{\alpha}_{\max}, \left( \frac{\|\mathbf{R}^i\|}{\|\mathbf{R}^0\|} \right)^{1/2} \right) \right) \quad (128)$$

onde  $\tilde{\alpha}_{\min}$  e  $\tilde{\alpha}_{\max}$  representam os valores mínimo e máximo para  $\tilde{\alpha}$ , respectivamente. Neste trabalho, para a solução do método iterativo utilizado na resolução do sistema linear, foi adotado internamente  $\tilde{\alpha}_{\max}$  como sendo igual a  $10^{-1}$  e como parâmetro de entrada  $\tilde{\alpha}_{\min}$  igual a  $10^{-3}$ .

#### 4. ESTRUTURA DE DADOS

Conforme visto no capítulo anterior, a formulação semi-discreta conduz a um sistema de equações não-lineares as quais podem ser resolvidas por uma seqüência de sistemas lineares. Os métodos diretos baseados na eliminação de Gauss podem ser usados para resolver estes sistemas. No entanto, tão logo o número de equações aumenta, estes métodos tornam-se caros do ponto de vista de processamento e memória quando comparados com os métodos iterativos. Entre estes últimos, um dos mais conhecidos e eficientes métodos iterativos é o *GMRES (Generalized Minimum Residual)*, definido para sistemas não-simétricos. Este método, desenvolvido por Saad e Schultz [7], consiste em minimizar a norma do resíduo e é baseado na geração dos espaços de Krilov.

O desempenho de qualquer método iterativo pode ser significativamente melhorado através da utilização de algum tipo de preconditionador. Os preconditionadores simplesmente transformam o sistema de equações originais em outro sistema, com a mesma solução, porém mais fácil de ser solucionado. Uma completa e detalhada discussão sobre métodos iterativos e preconditionadores pode ser encontrada na referência [25]. Outra maneira de melhorar o desempenho é através da otimização do código, reduzindo a demanda de memória, o número de operações de ponto flutuante (*flops*) e o número de operações de endereçamentos indiretos (*i/a*).

O núcleo computacional do método *GMRES* é baseado em três operações principais:

$$\textit{Atualização do Vetor} : \quad \mathbf{x} = \mathbf{x} + \alpha \mathbf{p} \quad (129)$$

$$\textit{Produto Escalar} : \quad \alpha = \mathbf{r}^T \mathbf{r} \quad (130)$$

$$\textit{Produto Matriz – Vetor} : \quad \mathbf{p} = \mathbf{K} \mathbf{u} \quad (131)$$

As operações dos tipos (129 e 130) podem ser executadas de maneira eficiente em máquinas paralelas/vetoriais da mesma forma que em máquinas escalares. Já a operação do tipo (131) é a que apresenta a maior demanda computacional. Principalmente em problemas não-lineares dependentes do tempo, como é o caso das equações de águas rasas, esta operação é repetida muitas vezes. Sendo assim, uma atenção especial deve ser dada a este aspecto. Esta operação é usualmente efetuada numa abordagem baseada por “elementos”, porém uma melhora significativa pode ser obtida caso seja utilizada uma estrutura de dados baseada por “arestas”.

## 4.1 ESTRUTURA DE DADOS BASEADA EM ELEMENTO E EM ARESTA

Como a matriz  $\mathbf{K}$ , resultado da discretização dos elementos finitos, é esparsa, podemos, tirando vantagem dessa característica, não mais montar a matriz global  $\mathbf{K}$ . Alternativamente, podemos armazenar somente as matrizes de elementos. A utilização das matrizes de elementos garante que somente os coeficientes não nulos sejam armazenados. Diferentemente do armazenamento global, o armazenamento em nível de elemento não depende da largura de banda do sistema. Utilizando o armazenamento em nível de elemento, o produto matriz-vetor pode ser realizado elemento por elemento da seguinte forma [8]:

$$\mathbf{p} = \mathbf{A} \sum_{e=1}^{nel} \mathbf{K}^e \mathbf{u}^e \quad (132)$$

Na expressão acima,  $\mathbf{p}$  é o vetor global,  $\mathbf{A}$  representa o operador de montagem,  $\mathbf{K}^e$  são as matrizes de elementos,  $\mathbf{u}^e$  são os componentes de  $\mathbf{u}$  relativos aos graus de liberdade local e  $nel$  é o número total de elementos.

Para este trabalho, uma malha composta por  $nel$  elementos **triangulares**,  $nnode$  pontos nodais e  $nedge$  arestas foi considerada, além da seguinte notação:

$neq$  : número total de equações

$nen$  : número de nós por elemento ( $nen = 3$ )

$ndf$  : número de graus de liberdade por nó ( $ndf = 3$ )

$nd$  : número de graus de liberdade por elemento ( $nd = 9$ )

O número de coeficientes armazenados por elemento é igual a 81 ( $nd^2$ ), para matrizes não-simétricas. Para cada passo do *loop* de elementos, o número de operações de ponto flutuante é igual a 162 ( $2 \times nd^2$ ) e o número de endereçamentos indiretos é igual a 27 ( $3 \times nd$ ). O desempenho do algoritmo do produto matriz-vetor elemento por elemento pode ser melhorado utilizando a técnica proposta por Gijzen [26]. Esta técnica consiste em separar o produto matriz-vetor em

$$\mathbf{p} = (\mathit{diag} \mathbf{K}) \mathbf{u} + \mathbf{A} \sum_{e=1}^{nel} (\mathbf{K}^e - \mathit{diag} \mathbf{K}^e) \mathbf{u}^e \quad (133)$$

onde  $(diag \mathbf{K})$  é a diagonal de  $\mathbf{K}$ .

Extraindo a diagonal das matrizes de elemento, o número de coeficientes armazenados em cada elemento é reduzido para 72  $((nd-1) \times nd)$  e para cada passo do *loop* de elementos, o número de operações de ponto flutuante é reduzido para 144  $(2 \times (nd-1) \times nd)$  e o número de endereçamentos indiretos não se altera.

Assim sendo, a estrutura de dados por elemento é armazenada de forma que  $(\mathbf{K}^e - diag \mathbf{K}^e)$  é uma matriz  $AL(nce, nel)$ , onde  $nce$  é o número de coeficientes por elemento e a diagonal global  $(diag \mathbf{K})$  é uma matriz  $AD(1, neq)$ .

Com as técnicas acima, a esparsidade do sistema é completamente explorada pelo armazenamento em nível de elemento, sendo que a montagem da diagonal global permite a redução da quantidade de memória necessária assim como o número de operações de ponto flutuante.

Contudo, os coeficientes que se relacionam através de dois nós diferentes ainda permanecem distribuídos na contribuição de cada elemento associado (Figura 4). Nesta Figura, os dois elementos (**A** e **B**) contribuem para o coeficiente global  $S(i, j)$ , e ambas as contribuições  $S_A(i, j)$  e  $S_B(i, j)$  são armazenadas nas matrizes dos elementos correspondentes. Se a aresta que conecta os nós  $i$  e  $j$  for usada para armazenar o coeficiente  $S(i, j)$ , o espalhamento da contribuição em nível de elemento é evitada. Isto pode ser feito realizando um *loop* nos elementos e montando os coeficientes por aresta.

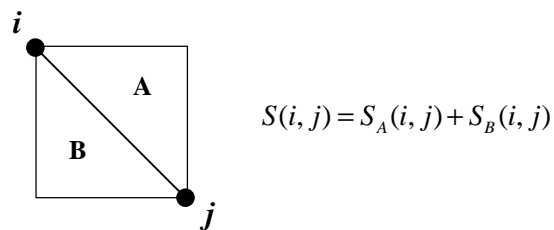


Figura 4. Contribuição de cada elemento para os coeficientes fora da diagonal.

Utilizando matrizes por aresta, o produto matriz-vetor dado pela equação (133) é substituído por

$$\mathbf{p} = (\text{diag } \mathbf{K})\mathbf{u} + \sum_{a=1}^{nedge} (\mathbf{K}^a - \text{diag } \mathbf{K}^a)\mathbf{u}^a \quad (134)$$

onde  $(\text{diag } \mathbf{K})$  é a mesma diagonal da equação (133),  $(\mathbf{K}^a - \text{diag } \mathbf{K}^a)$  são as matrizes por aresta contendo somente os coeficientes fora da diagonal e  $nedge$  representa o número total de arestas. O número de coeficientes armazenados por aresta é igual a 30  $(2ndf \times (2ndf - 1))$ , para matrizes não-simétricas. Para cada passo do *loop* de arestas, o número de operações de ponto flutuante é igual a 60  $(4ndf \times (2ndf - 1))$  e o número de endereçamentos indiretos é igual a 18  $(6 \times ndf)$ .

Assim sendo, a estrutura de dados por aresta é bastante semelhante a de elemento e é armazenada de forma que  $(\mathbf{K}^a - \text{diag } \mathbf{K}^a)$  é uma matriz  $AL(nca, nedge)$ , onde  $nca$  é o número de coeficientes por aresta e a diagonal global  $(\text{diag } \mathbf{K})$  é uma matriz  $AD(1, neq)$ .

Uma característica importante associada à otimização da estrutura de dados deve levar em consideração o acesso aos dados na memória. Inicialmente deve ser observado se os dados usados em seqüência estão armazenados na memória física perto uns dos outros. A Figura 5 mostra a localização das equações de cada elemento ou aresta e que a distância entre elas é importante. O comprimento  $L$  deve ser o menor possível e de preferência constante para todos os elementos ou arestas [27]. Além disso, é interessante que as equações sejam acessadas em ordem ascendente ou descendente [28].

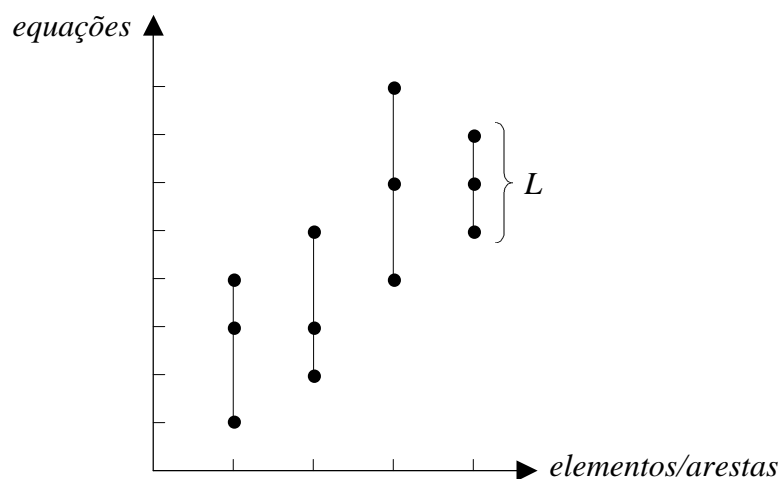


Figura 5. Localização dos dados para o produto matriz-vetor.

## 5. IMPLEMENTAÇÃO EM PARALELO

Para a implementação paralela, seja o domínio original  $\Omega$ , subdividido em  $nel$  elementos  $\Omega^e$ . Este domínio  $\Omega$  é então particionado em  $N$  subdomínios locais  $\Omega_{L_i}$ ,  $i = 0, 1, 2, \dots, N-1$ , tal que  $N$  represente o número de processadores (Figura 6). Cada subdomínio  $\Omega_{L_i}$  é então subdividido em  $nelloc$  elementos  $\Omega^e$  de forma que

$$\Omega = \bigcup_{i=0}^{N-1} \Omega_{L_i} \quad \Omega_{L_j} \cap \Omega_{L_k} = \emptyset \quad \text{para } j \neq k \quad (135)$$

$$\Omega_{L_i} = \bigcup_{e=1}^{nelloc} \Omega_{L_i}^e \quad \Omega_{L_i}^j \cap \Omega_{L_i}^k = \emptyset \quad \text{para } j \neq k \quad (136)$$

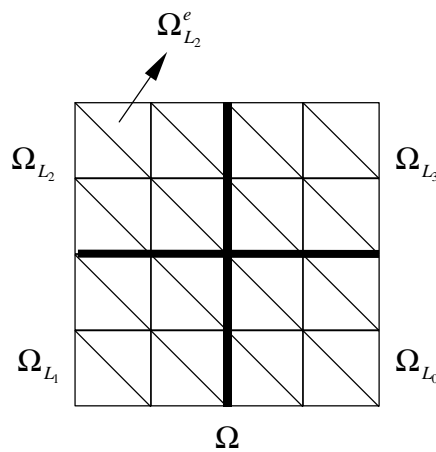


Figura 6. Particionamento para  $N = 4$ .

O problema da decomposição de domínios tem sido estudado há muito tempo. Com o surgimento dos computadores dotados de múltiplos processadores ou mesmo vários computadores trabalhando simultaneamente (*clusters*), é possível calcular os diversos subdomínios concorrentemente a fim de diminuir o tempo de processamento. Ao utilizar-se de um ambiente paralelo, podemos enviar cada subdomínio para ser resolvido em um processador diferente. Sendo assim, é importante que se tenha um algoritmo que seja rápido, automático, eficiente e que permita a divisão da malha original, identificando os nós internos e a(s) fronteira(s) entre cada subdomínio. Na verdade, não existe uma solução única e perfeita, porém um bom esquema de decomposição de domínio deve apresentar as seguintes características:

- Possuir carga de trabalho balanceada entre os processadores. Cada processador deverá ter aproximadamente o mesmo número de elementos.
- Minimização da comunicação entre os processadores. Para isso é conveniente diminuir o número total de nós na fronteira e evitar partições desconexas.
- Permitir o tratamento de geometrias irregulares.

Entre os principais pacotes de programas para o particionamento de grafos disponíveis em domínio público, foi selecionado o METIS, que será descrito a seguir.

## 5.1 PROGRAMA METIS

O programa METIS [10], desenvolvido por George Karypis e Vipin Kumar da Universidade de Minnesota (EUA), é um software utilizado para o particionamento de malhas não estruturadas. Malhas não estruturadas [9] são aquelas que apresentam uma estrutura irregular, de tal forma que um determinado nó pode estar conectado a um número arbitrário de elementos. Ao contrário das malhas estruturadas, as malhas não estruturadas não podem ter seus dados acessados na forma de um *array* do tipo  $i$   $j$   $k$ . A Figura 7 mostra um exemplo de cada tipo de malha.

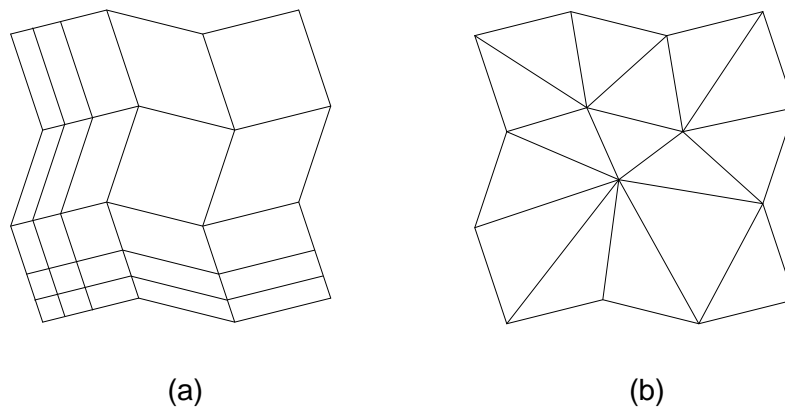


Figura 7. (a) Malha estruturada e (b) Malha não estruturada.

O programa METIS prevê que o particionamento da malha seja feito utilizando-se um de seus dois métodos (Dual ou Nodal) disponíveis, de forma a obter  $n$  subdomínios de tamanhos aproximadamente iguais. O programa, inicialmente, converte a malha em um grafo, para em seguida particionar este grafo. A diferença entre os dois métodos utilizados reside no fato de que um deles converte a malha em um grafo dual (cada **elemento** da malha torna-se um vértice do grafo) e o outro converte a malha em um grafo nodal (cada **nó** da malha torna-se um vértice do grafo).

Para os dados de entrada, o METIS requer além do tipo do elemento utilizado, a matriz de conectividades nodais de cada elemento. O resultado é o particionamento da malha tanto por elementos quanto por nós. Este resultado é obtido na forma de dois arquivos (por elementos ou por nós) numerados seqüencialmente, onde cada linha do arquivo representa o domínio (*rank*) a que pertence, variando de 0 a  $n-1$ , onde  $n$  é o número de domínios ou processadores. O programa METIS suporta quatro diferentes tipos de elementos: triangular, tetraedro, hexaedro e quadrilátero, sendo que neste trabalho foram utilizados somente elementos triangulares. Nas figuras a seguir são mostrados a malha original (Figura 8) e os particionamentos realizados pelo METIS utilizando 2, 4, 6, 8, 10, 12, 14 e 16 partições (Figuras 9-16) utilizando o método **dual**.

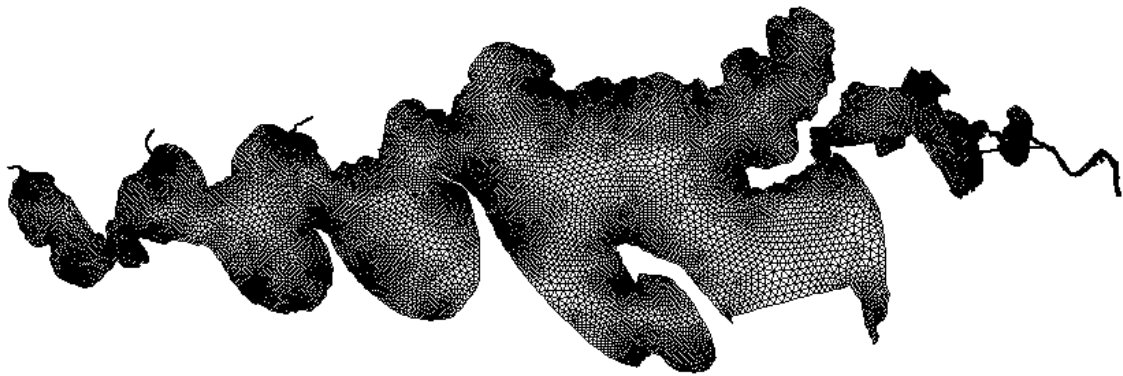


Figura 8. Malha original.



Figura 9. Malha particionada em 2 subdomínios.

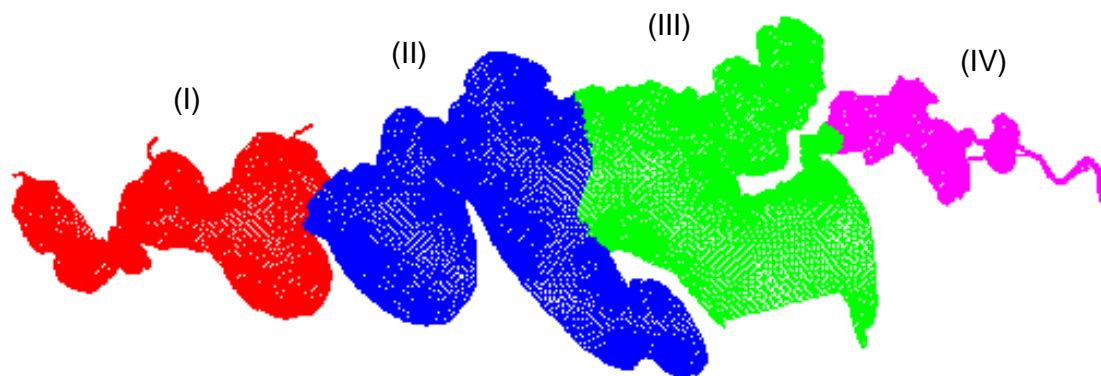


Figura 10. Malha particionada em 4 subdomínios.



Figura 11. Malha particionada em 6 subdomínios.



Figura 12. Malha particionada em 8 subdomínios.



Figura 13. Malha particionada em **10** subdomínios.



Figura 14. Malha particionada em **12** subdomínios.



Figura 15. Malha particionada em **14** subdomínios.



Figura 16. Malha particionada em **16** subdomínios.

A Tabela 2 mostra o resultado do METIS para o particionamento em 4 subdomínios (Figura 10). Embora esta seja uma malha com muitos nós, neste caso específico, o número total de nós nas fronteiras entre cada subdomínio foi de apenas 71 nós. Podemos verificar também que o número de nós e elementos de cada subdomínio, embora parecidos, não são exatamente iguais. Observe que neste particionamento não foi gerado nenhum subdomínio desconexo.

Tabela 2. Resultado do particionamento em 4 subdomínios.

	Malha Original	Malha Particionada			
		(I)	(II)	(III)	(IV)
<b>Elementos</b>	36.300	9.067	9.282	8.928	9.023
<b>Nós</b>	19.732	4.976	4.917	4.793	5.046

## 5.2 TRATAMENTO DAS FRONTEIRAS

Conforme visto em itens anteriores, o particionamento em vários subdomínios gera uma fronteira entre dois subdomínios adjacentes de tal forma que os elementos pertencentes à vizinhança dessa fronteira permanecem em subdomínios diferentes, enquanto que os nós e as arestas são compartilhados pelos dois subdomínios. A Figura 17 ilustra em nível de elemento como os subdomínios se apresentam após esse particionamento.

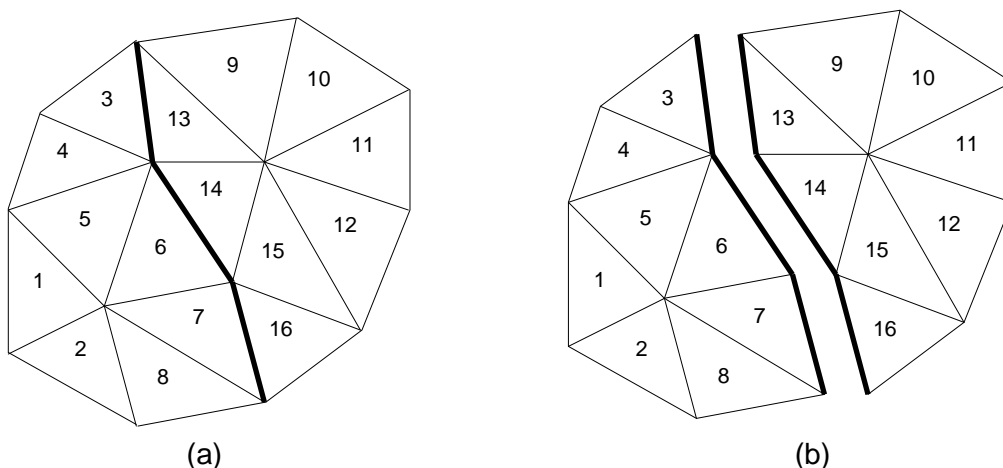


Figura 17. Malha original (a) e Malha particionada (b).

A Figura 18 ilustra mais detalhadamente a malha original da Figura 17a onde foi feito foco em alguns elementos (6, 7, 14, 15, 16) na região do particionamento. Neste exemplo, o “nó central” tomado como referência, circundado por estes elementos, possui o número de graus de liberdade por nó igual a 3 (direção de  $x$ , direção de  $y$  e elevação na direção de  $z$ ). Utilizando-se uma abordagem onde a estrutura de dados é baseada em elementos, cada um dos elementos adjacentes a este nó central contribui com sua parcela para o coeficiente global de cada uma das equações associada ao seu respectivo grau de liberdade.

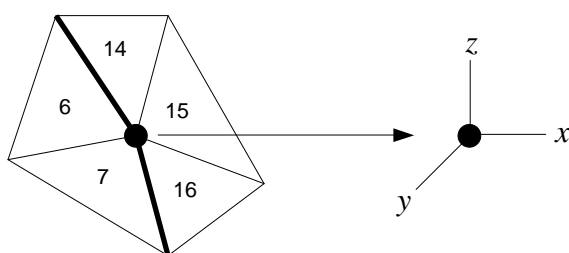


Figura 18. Representação de um nó mostrando suas equações numa abordagem por elemento.

Ao ser particionado o domínio original, cada elemento adjacente ao “nó central” continuará contribuindo, só que agora para o coeficiente global de cada uma das equações da fronteira de seu **subdomínio**. Observe que, para que o coeficiente global de cada equação na malha original, associada à mesma equação da fronteira na malha particionada, seja obtido, é necessário que os coeficientes globais de cada

equação da fronteira de cada subdomínio sejam somados. A Figura 19 mostra o acoplamento entre os nós da fronteira.

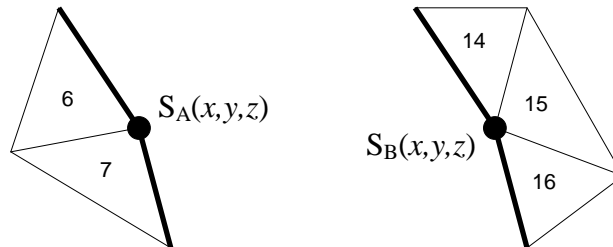


Figura 19. Contribuição de cada subdomínio na estrutura por elemento.

Da mesma forma que a estrutura de dados por elemento, a estrutura de dados baseada em arestas apresenta um comportamento semelhante. A diferença básica consiste em que na estrutura de dados por aresta o coeficiente global associado a cada equação é composto pelas parcelas de cada aresta adjacente ao nó em questão (Figura 20).

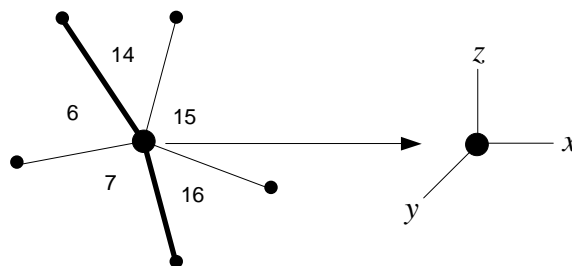


Figura 20. Representação de um nó mostrando suas equações numa abordagem por aresta.

Semelhantemente ao verificado na estrutura de dados por elemento, após o particionamento do domínio original, cada aresta adjacente ao “nó central” contribuirá com sua parcela para o coeficiente global de cada equação da fronteira do seu **subdomínio** (Figura 21). Igualmente, para que o coeficiente global de cada equação da malha original, associada à mesma equação da fronteira na malha particionada, seja obtido, é necessário que os coeficientes globais de cada equação da fronteira de cada subdomínio sejam somados.

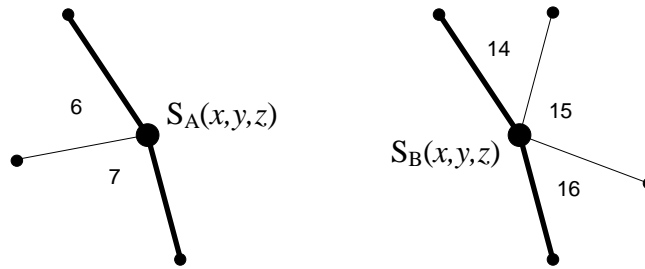


Figura 21. Contribuição de cada subdomínio na estrutura por aresta.

### 5.3 COMUNICAÇÃO ENTRE PROCESSADORES

Como o processamento paralelo deve ser realizado de forma distribuída entre os vários processadores, e que estes processadores devem estar conectados por algum tipo de rede de comunicação de dados, torna-se necessário um pacote de rotinas que seja capaz de gerenciar e controlar as diversas máquinas do sistema [29]. Tais programas permitem o gerenciamento do sistema, além de fornecer bibliotecas que efetuam a comunicação entre os processadores. Existem no mercado diversos módulos que desempenham esta função, por exemplo, PICL [30], PVM [31, 32], PARMACS [33], p4 [34, 35, 36], Chameleon [37], Zipcode [38], MPI [11], TCGMSG [39], Express [40], alguns ainda amplamente utilizados e de domínio público. Dentre estes podemos destacar dois [41] dos mais utilizados no processamento paralelo: PVM (Parallel Virtual Machine) e MPI (Message Passing Interface). O PVM é um projeto das Universidades de Emory e do Tennessee juntamente com o Laboratório Nacional de Oak Ridge e constitui-se num pacote integrado de ferramentas de software e bibliotecas que possui como características principais a interoperacionalidade e a alteração no número de processadores de um programa em execução, dinamicamente. Esta interoperacionalidade possibilita a conexão de máquinas heterogêneas, não somente diferentes em sua capacidade computacional, mas também diferentes em sua arquitetura, permitindo a sua utilização, inclusive com sistemas operacionais distintos (*Unix/Linux e/ou Windows, etc.*). Estas máquinas podem trabalhar interconectadas através de uma rede e ser utilizadas como uma grande máquina paralela virtual.

Da mesma forma, o padrão MPI constitui-se também numa biblioteca de funções cujo objetivo principal é, como o próprio nome diz, explorar a existência de múltiplos processadores através da troca de mensagens entre eles. Foi desenvolvido no início

da década de 90 por um grupo de representantes da indústria, governo e do meio acadêmico para que pudesse se tornar um padrão na passagem de mensagens. Uma das motivações da época era que, como cada fabricante desenvolvia seu próprio programa, isso reduzia enormemente a portabilidade das rotinas. Por ter sido elaborada em conjunto com a indústria, tornou-se a biblioteca nativa utilizada por muitos fabricantes, e portanto, não só permite a sua portabilidade como também garante o uso de rotinas mais otimizadas para cada máquina. Embora esteja disponível uma grande quantidade de rotinas para a troca de mensagens tanto ponto-a-ponto quanto a nível global, é possível implementar muitos programas com um número básico de funções. Os programas escritos em FORTRAN, C ou C++ ao utilizarem o pacote MPI, que não é uma linguagem e sim uma biblioteca, especificam nomes, declarações e funções que são inseridos no código. Posteriormente estes programas são compilados e lincados com a biblioteca MPI.

No modelo de passagem de mensagem utilizado em computação paralela, os processos que estão sendo executados em paralelo possuem endereços diferentes. A comunicação se dá quando o endereço do espaço de um processo é copiado para o endereço do espaço de outro processo. Esta operação é dita cooperativa e ocorre quando o primeiro processo executa uma operação de “enviar” e o segundo processo executa uma operação de “receber”. Para o processo que envia deve ser especificado o dado a ser comunicado e o processo de destino para o qual o dado deve ser enviado. A maneira pela qual o dado é descrito é especificando o seu endereço inicial e o tamanho (em bytes), já a identificação do processo de destino pode ser feita através de um número inteiro. Pelo lado do processo que recebe a mensagem, os argumentos mínimos são: o endereço e o comprimento de uma área na memória local na qual a variável recebida deve ser alocada, além de uma variável que indique qual processo enviou o dado. Embora a utilização destes dados mínimos pareça adequada para algumas aplicações, outros parâmetros geralmente são necessários a fim de tornar a passagem de mensagem mais precisa e completa.

Na implementação deste trabalho foi adotado o padrão MPI, visto que este já era o padrão instalado nas máquinas onde todos os exemplos foram executados. Para demonstrar a simplicidade de se escrever programas usando a biblioteca MPI, é aqui mostrada com um pouco mais de detalhes a lista das funções indispensáveis, aquelas que todo programador não deve deixar de usar. As funções básicas são em número de seis. Com somente estas seis funções um grande número de programas pode ser desenvolvido. Embora existam diversas outras funções além destas, elas servem para permitir ao programador mais flexibilidade (tipos de dados), robustez (funções *send/receive* não-blocadas), eficiência, modularidade (grupos) ou mesmo

conveniência (operações coletivas). Para cada uma destas seis e das demais funções existem argumentos que por ora não serão objeto de discussão em profundidade, mas que podem ser consultados com mais detalhes em literatura específica [11, 42, 43].

- **MPI\_INIT:** É a primeira função a ser usada e deve ser chamada antes de qualquer outra função. Esta função serve para inicializar a biblioteca MPI e deve ser chamada uma única vez por todos os processos.
- **MPI\_COMM\_SIZE:** Esta função retorna em um de seus argumentos a quantidade de processos (número de processadores) que está executando o programa.
- **MPI\_COMM\_RANK:** Esta função retorna em um de seus argumentos o *rank* do processo em curso. Em um grupo contendo  $n$  processos, estes são identificados através do seu *rank*, os quais são representados por números inteiros variando de 0 a  $n-1$ .
- **MPI\_SEND:** Esta função é utilizada quando um processo deseja enviar uma mensagem a outro processo. Alguns dos argumentos utilizados por esta função são: *message* (endereço inicial da mensagem), *count* (tamanho da mensagem), *datatype* (tipo da mensagem), *dest* (processo de destino da mensagem) e *tag* (parâmetro inteiro especificado pelo usuário para distinguir mensagens enviadas a um mesmo processo).
- **MPI\_RECV:** Esta função é utilizada quando um processo deseja receber uma mensagem de outro processo. Alguns dos argumentos utilizados por esta função são: *message* (endereço inicial da mensagem), *count* (tamanho da mensagem), *datatype* (tipo da mensagem), *source* (processo que enviou a mensagem), *tag* (parâmetro inteiro especificado pelo usuário para distinguir mensagens recebidas de um mesmo processo) e *status* (parâmetro complementar ao argumento *source*).
- **MPI\_FINALIZE:** Esta função deve ser chamada ao final do programa para finalizar o uso da biblioteca MPI. Esta chamada “limpa” algum evento ainda não terminado e deixado pelo MPI. Assim como MPI\_INIT, esta função deve ser chamada por todos os processos.

Conforme já mencionado anteriormente, as funções de uso coletivo que envolvem dois ou mais processos, podem ser representadas por uma composição das funções básicas vistas acima, de forma a tornar a programação mais clara e eficiente. A seguir algumas destas funções coletivas são mostradas:


- **MPI\_BCAST:** Esta é uma função que envolve todos os processos. *Broadcast* é uma troca de mensagens coletiva em que um único processo envia a mesma mensagem para todos os outros processos. Alguns dos argumentos utilizados por esta função são: *buffer* (endereço inicial da mensagem), *count* (tamanho da mensagem), *datatype* (tipo da mensagem) e *root* (processo que envia as mensagens). Simplesmente é enviada uma cópia do *buffer* do processo *root* para cada um dos outros processos. Esta função é chamada por todos os processos com o mesmo argumento para *root*.
- **MPI\_REDUCE:** Esta função é utilizada quando todos os processos desejam realizar uma mesma operação (por exemplo, máximo, mínimo, soma, produto, etc...) armazenando o resultado num determinado processo. Alguns dos argumentos utilizados por esta função são: *operand* (variável a ser operada), *result* (resultado da operação), *count* (tamanho da variável), *datatype* (tipo da variável), *op* (operação a ser realizada) e *root* (processo que armazena o resultado). Os operandos armazenados em *operand* são combinados usando a operação *op* e o resultado é armazenado em *result* no processo *root*. Esta função é chamada por todos os processos com o mesmo argumento para *op*.
- **MPI\_ALLREDUCE:** Esta é uma função coletiva e pode ser descrita como uma combinação das funções MPI\_REDUCE e MPI\_BCAST. Esta função é utilizada quando todos os processos desejam realizar uma mesma operação e o resultado ser distribuído também entre todos os processos. Alguns dos argumentos utilizados por esta função são: *operand* (variável a ser operada), *result* (resultado da operação), *count* (tamanho da variável), *datatype* (tipo da variável), *op* (operação a ser realizada). Os operandos armazenados em *operand* são combinados usando a operação *op* e o resultado é armazenado em *result* em todos os processos. Esta função é chamada por todos os processos com o mesmo argumento para *op*.
- **MPI\_GATHER:** Esta função possui a propriedade de concatenação das mensagens de cada um dos processos, armazenando o seu resultado num

determinado processo. Alguns dos argumentos utilizados por esta função são: *send\_buf* (variável a ser enviada por cada processo), *send\_count* (tamanho da variável), *send\_type* (tipo da variável), *recv\_buf* (variável com o resultado da concatenação), *recv\_count* (número de itens recebido de cada processo), *recv\_type* (tipo da variável), *root* (processo que recebe as variáveis) Cada processo envia o conteúdo de seu *buffer* para o processo *root*. O processo *root* concatena as variáveis recebidas em *recv\_buf* na seqüência do *rank*, isto é, o dado do processo 0 é seguido do dado do processo 1 que é seguido pelo dado do processo 2 e assim por diante. Os argumentos *recv* são significativos apenas no processo *root*.

- **MPI\_SCATTER:** Esta função funciona de forma inversa à função anterior e apresenta os mesmos argumentos. O processo com o *rank root* distribui o conteúdo de seu *send\_buf* por todos os outros processos. O conteúdo do *send\_buf* é dividido em *n* segmentos, cada um consistindo de *send\_count* itens. O primeiro segmento é enviado ao processo 0, o segundo ao processo 1, o terceiro ao processo 2 e assim por diante. Os argumentos *send* são significativos apenas no processo *root*.

Considerando o exposto no subitem 5.2, temos que para a montagem das matrizes de elemento e aresta, os coeficientes correspondentes às equações da fronteira entre cada subdomínio devem ser somados e o seu resultado novamente distribuído entre cada subdomínio. Observando as rotinas disponíveis na biblioteca MPI, algumas delas listadas acima, observa-se que o comando **MPI\_ALLREDUCE** possui exatamente esta função. Assim sendo, pode-se afirmar que apenas com a utilização deste comando, acrescido dos comandos básicos (MPI\_INIT, MPI\_COMM\_SIZE, MPI\_COMM\_RANK, MPI\_FINALIZE), obviamente, é possível realizar toda a implementação em paralelo.

## 5.4 FLUXOGRAMA

A Figura 22 ilustra de forma simplificada o fluxo do cálculo utilizado na implementação em paralelo. O *loop* mais externo indica o avanço da simulação ao longo do tempo e o mais interno a resolução do sistema não-linear. O *solver* é composto pelo método iterativo GMRES que através do produto matriz-vetor determina a sua solução. O símbolo  indica que a implementação é feita em paralelo.

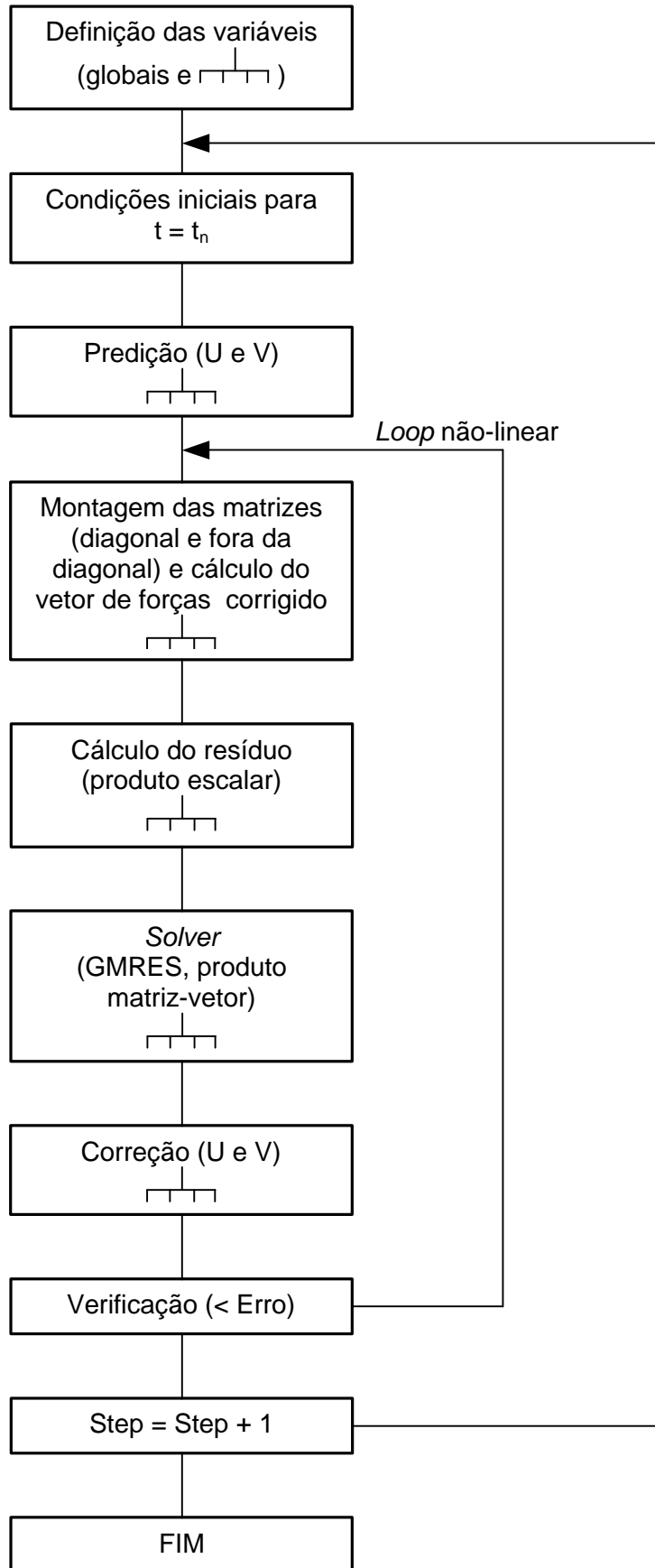


Figura 22. Fluxograma da implementação em paralelo.

## 6. DADOS E RESULTADOS

### 6.1 DADOS GEOMÉTRICOS

A fim de ilustrar o desempenho da implementação em paralelo do método dos elementos finitos para as equações de águas rasas foi utilizado um exemplo numérico que simula o comportamento de maré da Lagoa de Araruama, Rio de Janeiro, Brasil. Para a representação da sua geometria foram utilizadas três malhas diferentes, denominadas aqui de malha pequena, média e grande, sendo que para cada uma destas malhas foram utilizados somente elementos triangulares. As Figuras 23 e 24 ilustram as dimensões do domínio e mostram a malha menor, caracterizando o seu tamanho e batimetria aproximados. Observe também que a malha apresenta no seu lado direito, na entrada do canal, uma fronteira externa aberta na qual é prescrita a função do comportamento da maré. No restante da malha, em toda a extensão da fronteira externa fechada, ambas as velocidades nas direções de  $x$  e  $y$  são nulas.

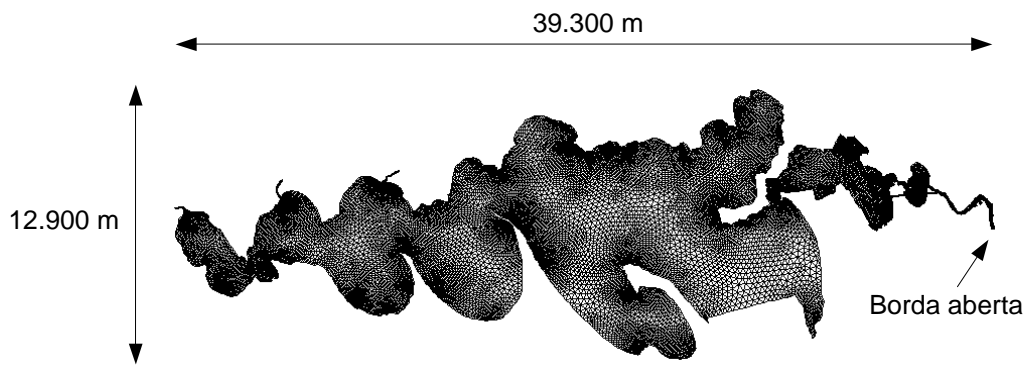


Figura 23. Malha pequena.

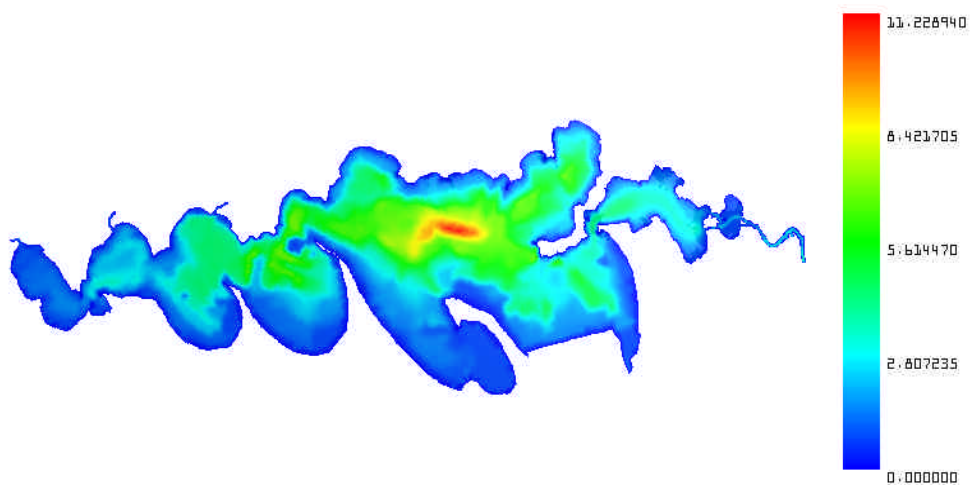


Figura 24. Batimetria (em metros) da malha pequena.

As malhas média e grande foram obtidas a partir do refinamento uniforme da malha pequena e da malha média, respectivamente. Nesse refinamento, cada um dos elementos foi transformado em outros quatro novos elementos através da criação de três novos nós interligados entre si e localizados no meio de cada uma das arestas do elemento original (Figura 25).

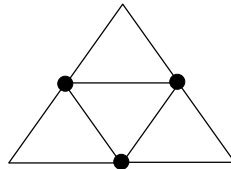


Figura 25. Refinamento de um elemento tipo.

A Tabela 3 mostra as características topológicas das três malhas utilizadas. Considerando as relações geométricas de Euler entre o número de nós (vértices), faces (elementos) e arestas para a implementação da triangularização em duas dimensões, observamos que de acordo com Carey [44] ou Lohner [45],  $n_{el} \approx 2 \times n_{nós}$  e  $n_{edges} \approx 1,5 \times n_{el} \approx 3 \times n_{nós}$ , onde  $n_{el}$  é o número total de elementos,  $n_{nós}$  é o número total de nós e  $n_{edges}$  é o número total de arestas. Para essa estimativa foi considerado que o número de nós na borda da malha pode ser desprezível comparado com o número de nós no seu interior.

Como já foi mencionado anteriormente, para este trabalho foram adotadas três equações para cada nó, uma para cada grau de liberdade ( $x, y, z$ ), sendo que os nós da fronteira externa fechada são prescritos nas direções de  $x$  e  $y$  e os da fronteira externa aberta são prescritos na direção de  $z$ .

Tabela 3. Dados topológicos para as três malhas da Lagoa de Araruama.

Malha	Nós	Elementos	Arestas	Equações
Pequena	19.732	36.300	56.035	52.859
Média	75.767	145.200	220.970	214.628
Grande	296.737	580.800	877.540	864.866

Um parâmetro importante no controle do intervalo de tempo utilizado na simulação é a condição de estabilidade de Courant-Friedrichs-Levy (CFL) [46]. Esta condição de CFL

é uma restrição no tamanho do passo de tempo  $\Delta t$  usado no “andamento numérico” da simulação. Este passo de tempo deve ser menor que o tempo mínimo sobre o qual algo significativo possa ocorrer no sistema em questão e mesmo assim a simulação continue numericamente estável. A condição de CFL depende tanto do sistema físico modelado quanto das características utilizadas no método de discretização. No contexto do sistema descrito neste trabalho foi utilizada a condição de CFL como sendo igual a unidade conforme a seguinte equação:

$$CFL = \frac{|u| \Delta t}{S} = 1 \quad (137)$$

onde  $|u|$  é a velocidade da onda gravitacional dada por  $|u| = \sqrt{g h}$ ,  $g$  é a aceleração da gravidade,  $h$  é a profundidade média do elemento e  $S$  é o tamanho característico do elemento, calculado como sendo igual a  $\sqrt{\text{área do elemento}}$ . Desta forma a equação (137) pode ser reescrita como

$$\Delta t = \sqrt{\frac{\text{área do elemento}}{g h}} \quad (138)$$

Como as malhas possuem elementos de diferentes tamanhos, temos que para cada malha diversos passos de tempo podem ser calculados. A Tabela 4 mostra os menores e os maiores passos de tempo para cada uma das três malhas utilizadas. Sendo assim, neste trabalho, a avaliação do desempenho da implementação em paralelo, medido através do ganho, foi realizada utilizando-se passos de tempo com tamanhos de: 1, 10, 20, 30, 40 e 50s. O processamento foi encerrado após terem sido decorridos 10 passos de tempo.

Tabela 4. Passo de tempo mínimo e máximo.

Malha	$\Delta t$ mín (s)	$\Delta t$ máx (s)
Pequena	1,14	70,32
Média	0,61	38,72
Grande	0,26	21,15

## 6.2 DADOS AMBIENTAIS

A variação do nível do mar na entrada do canal, limitado pela fronteira externa aberta, foi modelada através das curvas harmônicas da maré. Os parâmetros utilizados para a geração destas curvas foram fornecidos pelo Departamento de Hidrografia e Navegação da Marinha do Brasil. Dentre as diversas curvas, foram selecionadas as dez curvas que apresentaram as maiores amplitudes (Tabela 5).

Tabela 5. Principais parâmetros das curvas de maré.

Nome	Amplitude Máx. (m)	Período (s)	Fase (rad)
M2	0,3256	44.714,1600	1,370607062
S2	0,1718	43.200,0000	1,539729466
O1	0,0999	92.949,6290	1,524894167
K1	0,0504	86.164,0900	2,568426527
K2	0,0467	43.082,0450	1,553517567
N2	0,0418	45.570,0500	1,609891702
Q1	0,0267	96.726,0800	1,312313065
L2	0,0222	43.889,8300	1,611986097
M4	0,0193	22.357,0821	0,429176463
P1	0,0167	86.637,2045	2,490235777

Cada conjunto de parâmetros gera uma das curvas de maré as quais obedecem ao comportamento da seguinte equação do movimento periódico [47]:

$$A = A_0 \cos(\omega t + \phi) \quad (139)$$

sendo  $A$  a amplitude no instante  $t$ ,  $A_0$  a máxima amplitude ( $m$ ),  $\omega$  a velocidade angular ( $rad/s$ ) e  $\phi$  o ângulo de fase ( $rad$ ). A velocidade angular também pode ser expressa

por  $\omega = 2\pi f$  ou  $\omega = \frac{2\pi}{T}$ , onde  $f$  é a frequência ( $Hz$ ) e  $T$  é o período ( $s$ ).

Considerando cada curva gerada (Figura 26) e realizando suas superposições, obtemos uma nova curva (Figura 27) usada como condição de contorno de elevação do nível do mar na fronteira externa aberta. Observe que, embora cada uma das curvas geradoras tenha um comportamento periódico bem definido, a curva resultante apresenta um comportamento semelhante a um batimento. Uma aproximação

razoável seria considerar esta periodicidade diária, o que a tornaria compatível com o nível das marés.

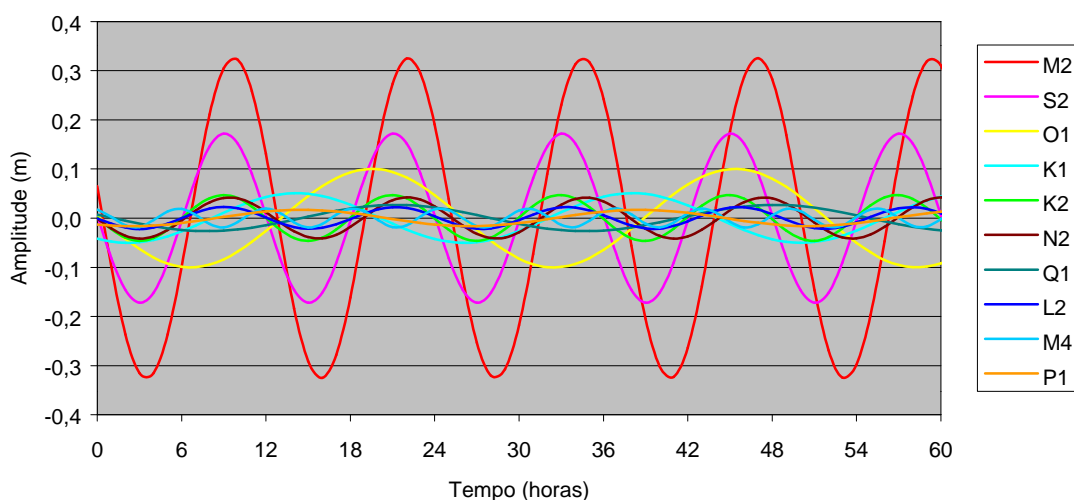


Figura 26. Curvas representativas das marés.

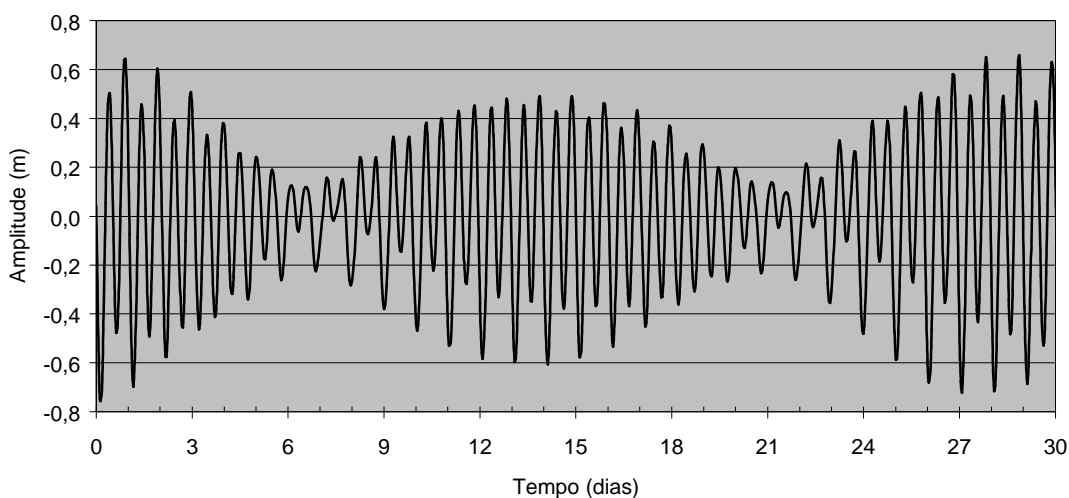


Figura 27. Somatório das curvas representativas das marés.

Lembrando que para a avaliação do desempenho, medido através do ganho, a simulação aqui denominada de SCD (Simulação de Curta Duração), somente utilizou 10 passos de tempo sendo que o maior passo de tempo adotado foi de 50s, o que corresponde a um tempo total simulado de 500s, no maior dos casos. Dessa forma, considerando a condição de contorno adotada pela curva acima, apenas o efeito do primeiro declínio foi utilizado nesta primeira avaliação. Já na análise da extensão da condição de contorno adentrando através do canal, aqui denominada de simulação SLD (Simulação de Longa Duração), foram utilizados 21.000 passos de tempo sendo

que em cada passo de tempo foi utilizado um tamanho de 30s, correspondendo ao tempo total de simulação de 630.000s ou aproximadamente 7,3 dias. Observando o comportamento da curva da Figura 27, vemos que este tempo decorrido de simulação corresponde ao tempo no qual a amplitude do “batimento” sai de seu valor máximo até atingir o valor mínimo. Este declínio também pode ser observado nas curvas obtidas a partir da medição de alguns dos nós internos da malha.

A fim de auxiliar a avaliação dos resultados, foram feitos gráficos a partir de medições temporais em pontos selecionados ao longo da malha. Estes pontos possuem suas posições definidas de acordo com o mapa da Figura 28 e estão associados aos nós definidos pela Tabela 6.



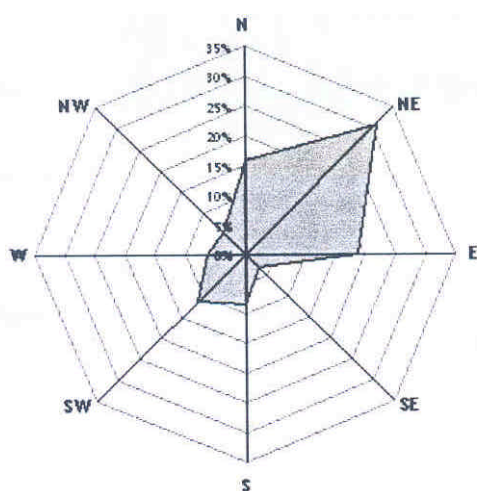
Figura 28. Pontos de medição das séries temporais.

Tabela 6. Localização dos pontos.

Ponto	Nó
1	5.322
2	5.158
3	4.811
4	4.083
5	2.360
6	1.156
7	186

Os ventos que sopram sobre a Lagoa de Araruama podem apresentar forte influência em sua dinâmica. Por isso, para que se tenham melhores resultados qualitativos, a simulação deve levar em consideração os parâmetros associados à direção e velocidade dos ventos. A Figura 29 mostra a direção predominante através de medições realizadas durante um período de 21 anos (1976-1997). Os dados compõem uma série temporal de medições de velocidade e direção do vento referenciado ao norte. Com base nestes resultados observa-se a predominância dos ventos na direção NE, sendo que aproximadamente 2/3 de todas as ocorrências estão entre as direções N-NE-E. Já a intensidade apresentou uma variação de 0 a 10 m/s.

Embora a opção pela utilização das condições do vento esteja disponível no simulador, esta opção não foi utilizada, visto que para o objetivo do presente trabalho o seu uso não acrescentaria nenhum benefício relevante.



Direção	Ocorrência
N	15,9 %
NE	31,0 %
E	18,8 %
SE	3,2 %
S	8,4 %
SW	11,3 %
W	6,4 %
NW	5,0 %

Figura 29. Estatística da direção do vento.

### 6.3 OUTROS DADOS

Na resolução do sistema de equações foi adotado o método iterativo **GMRES**, que na sua implementação utiliza-se da geração dos espaços de Krylov. Para a geração destes espaços, foi utilizado o número de vetores igual a 10. O parâmetro  $\alpha$  (Tabela 1), utilizado no controle do método numérico foi fixado igual a 1 (Euler Backward), desta forma, o método numérico é considerado implícito. A tolerância da condição de convergência tanto do *loop* não-linear quanto do próprio método iterativo GMRES foi estabelecida como sendo igual a  $10^{-3}$ . As simulações foram realizadas em dois *clusters* diferentes, denominados de “*cluster* MC” e “*cluster* SISMOS”, nomes estes referentes às gerências da PETROBRAS/CENPES aos quais pertencem. Cada um dos *clusters* apresenta as seguintes características:

**CLUSTER SISMOS**: 72 Processadores Pentium III *Single*, 550 MHz, 768 MB de memória RAM cada, Sistema Operacional Linux Red Hat 7.2, Biblioteca MPICH, Compilador Fortran 90 Portland e Fast-Ethernet.

**CLUSTER MC**: 39 Processadores Pentium III *Dual*, 1 GHz, 1 GB de memória RAM cada, Sistema Operacional Linux Red Hat 7.2, Biblioteca LAM-MPI, Compilador Fortran 90 Lahey e Fast-Ethernet.

Como no início do desenvolvimento o *cluster* MC ainda não se apresentava plenamente operacional, as simulações concentraram-se no *cluster* SISMOS (550 MHz), que pelo fato de possuir o pacote da Portland instalado, dispunha de algumas importantes ferramentas de controle para programas paralelos. Entre estas, o *profile*, uma poderosa e prática ferramenta utilizada na determinação do custo de processamento de cada rotina para todos os processadores.

Tão logo o *cluster* MC foi disponibilizado, toda simulação foi então direcionada para o novo *cluster*, que embora apresentasse um melhor desempenho não possuía as mesmas facilidades quanto às ferramentas disponíveis. Portanto, para a análise de desempenho (SCD) do simulador assim como as curvas de alcance das condições iniciais num tempo longo (SLD) foram obtidas a partir do *cluster* MC, restando para o *cluster* SISMOS somente os resultados da análise de custos de processamento.

## 6.4 RESULTADOS

Todos os resultados apresentados foram validados a partir do programa utilizado na implementação do trabalho de Ribeiro *et al* [8]. A seguir são mostrados os resultados referentes à simulação das três malhas (pequena, média e grande) utilizando a estrutura de dados por **elemento**. Numa primeira abordagem, apenas **um** processador foi utilizado em ambos os *clusters* (Tabelas 7-12). Estes resultados servirão de referência mais adiante na determinação do desempenho da simulação paralela.

Tabela 7. *Cluster* **SISMOS** - Malha **Pequena**.

10 Passos de Tempo (s)	Tempo Total (s)	Tempo Solver (s)	Iterações Não-Lineares	Iterações GMRES
10	103	45	45	181
100	184	87	75	370
200	234	125	85	565
300	297	176	94	817
400	352	221	102	1051
500	399	264	105	1278

Tabela 8. *Cluster* **SISMOS** - Malha **Média**.

<b>10 Passos de Tempo (s)</b>	<b>Tempo Total (s)</b>	<b>Tempo Solver (s)</b>	<b>Iterações Não-Lineares</b>	<b>Iterações GMRES</b>
10	394	200	40	217
100	862	522	70	613
200	1389	969	87	1181
300	1757	1295	96	1609
400	1926	1474	93	1858
500	2403	1920	100	2445

Tabela 9. *Cluster* **SISMOS** - Malha **Grande**.

<b>10 Passos de Tempo (s)</b>	<b>Tempo Total (s)</b>	<b>Tempo Solver (s)</b>	<b>Iterações Não-Lineares</b>	<b>Iterações GMRES</b>
10	2081	1158	49	315
100	5673	4283	74	1296
200	9603	7957	88	2472
300	12790	11001	96	3447
400	14948	13131	97	4139
500	19468	17503	105	5556

Tabela 10. *Cluster* **MC** - Malha **Pequena**.

<b>10 Passos de Tempo (s)</b>	<b>Tempo Total (s)</b>	<b>Tempo Solver (s)</b>	<b>Iterações Não-Lineares</b>	<b>Iterações GMRES</b>
10	56	17	45	182
100	97	32	75	370
200	119	46	85	567
300	145	63	94	814
400	170	81	103	1066
500	186	95	105	1273

Tabela 11. *Cluster MC* - Malha Média.

10 Passos de Tempo (s)	Tempo Total (s)	Tempo Solver (s)	Iterações Não-Lineares	Iterações GMRES
10	214	81	40	217
100	446	213	70	612
200	685	397	87	1182
300	848	530	96	1613
400	921	610	94	1876
500	1116	785	100	2447

Tabela 12. *Cluster MC* - Malha Grande.

10 Passos de Tempo (s)	Tempo Total (s)	Tempo Solver (s)	Iterações Não-Lineares	Iterações GMRES
10	1052	437	49	316
100	2541	1614	74	1299
200	4118	3008	88	2472
300	5360	4159	96	3451
400	6170	4955	97	4140
500	7902	6589	105	5555

Estes resultados (Tabelas 7-12) podem ser agrupados de outra forma e melhor visualizados através de gráficos (Figuras 30-32).

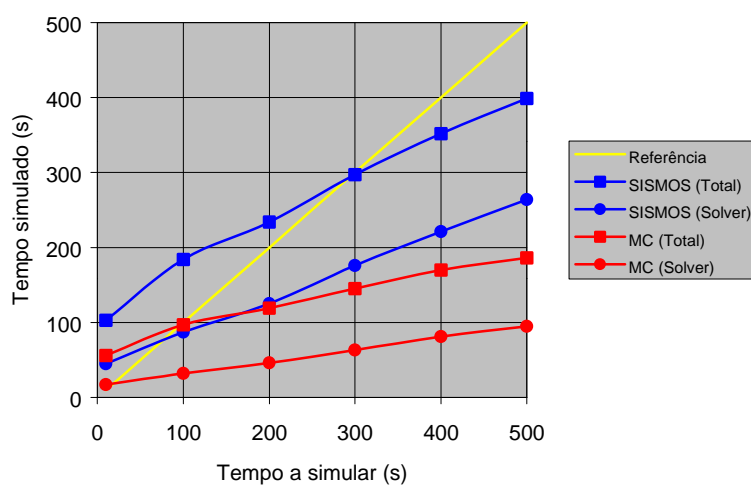


Figura 30. Malha Pequena.

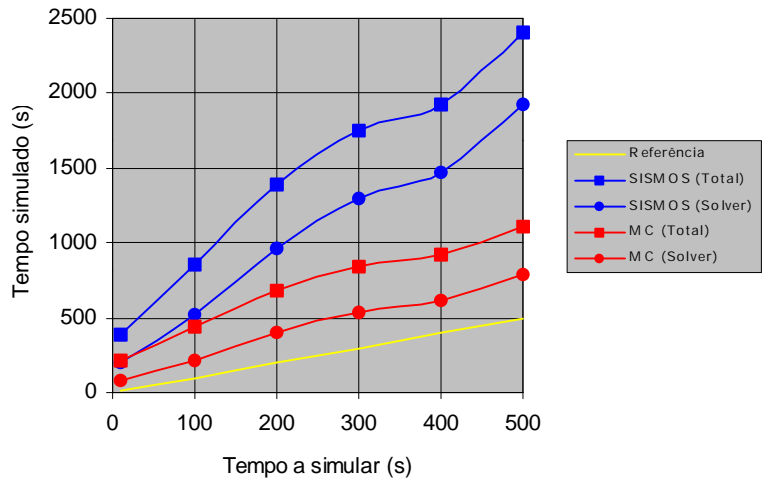


Figura 31. Malha Média.

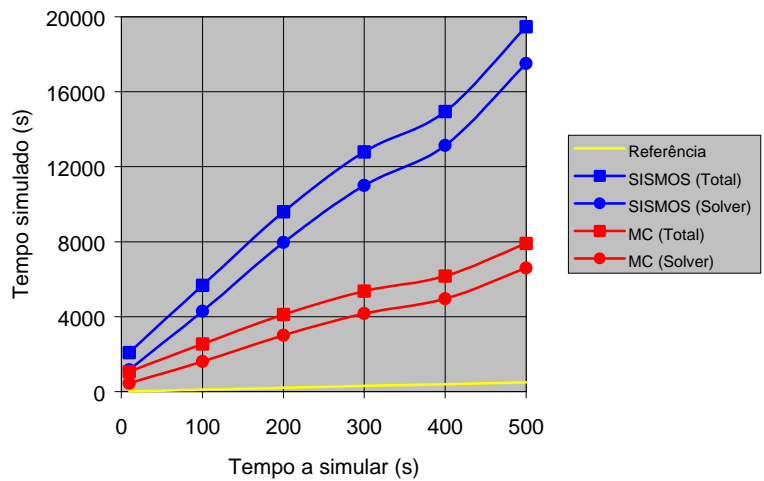


Figura 32. Malha Grande.

Observando os dados preliminares obtidos na simulação utilizando a estrutura de dados por elemento, vemos que para as malhas média e grande (Figuras 31 e 32), a simulação utilizando somente um processador torna-se praticamente inviável devido ao grande esforço computacional apresentado. Observe que nestes dois casos os resultados estão todos acima da linha amarela, representando um tempo simulado maior que o tempo de relógio, isto é, a simulação numérica demanda mais tempo que o comportamento normal do fenômeno.

Para as análises comparativas entre ambas as estruturas de dados (por elemento e por aresta), também foram realizadas simulações com a estrutura de dados por **aresta** utilizando-se apenas **um** processador, porém, neste caso, apenas o *cluster* MC foi

utilizado. Os resultados referentes a esta simulação para as três malhas (pequena, média e grande) são mostrados nas Tabelas 13-15 e os gráficos correspondentes nas Figuras 33-35.

Tabela 13. *Cluster MC* - Malha Pequena.

10 Passos de Tempo (s)	Tempo Total (s)	Tempo Solver (s)	Iterações Não-Lineares	Iterações GMRES
10	55	13	45	182
100	95	25	75	370
200	115	36	85	567
300	138	50	94	814
400	160	64	103	1066
500	174	76	105	1283

Tabela 14. *Cluster MC* - Malha Média.

10 Passos de Tempo (s)	Tempo Total (s)	Tempo Solver (s)	Iterações Não-Lineares	Iterações GMRES
10	206	60	40	217
100	417	163	70	612
200	621	305	87	1182
300	758	410	96	1615
400	790	451	94	1876
500	944	584	100	2448

Tabela 15. *Cluster MC* - Malha Grande.

10 Passos de Tempo (s)	Tempo Total (s)	Tempo Solver (s)	Iterações Não-Lineares	Iterações GMRES
10	1017	349	49	316
100	2339	1328	74	1297
200	3681	2484	88	2470
300	4756	3450	96	3451
400	5456	4125	97	4141
500	6907	5475	105	5557

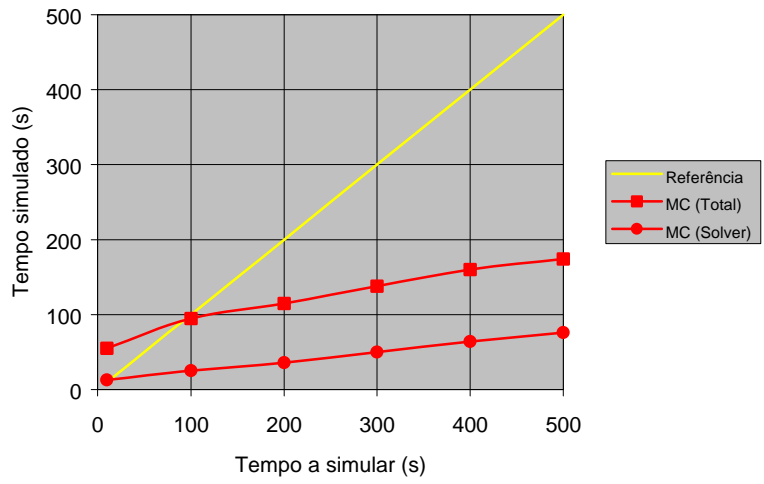


Figura 33. Malha Pequena.

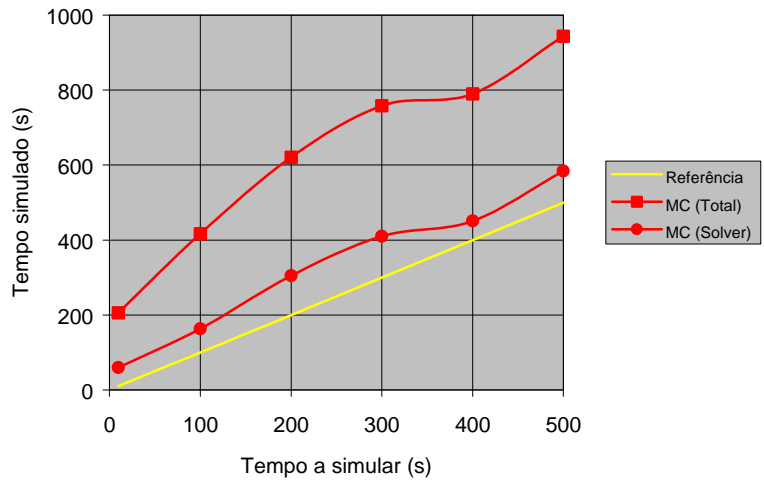


Figura 34. Malha Média.

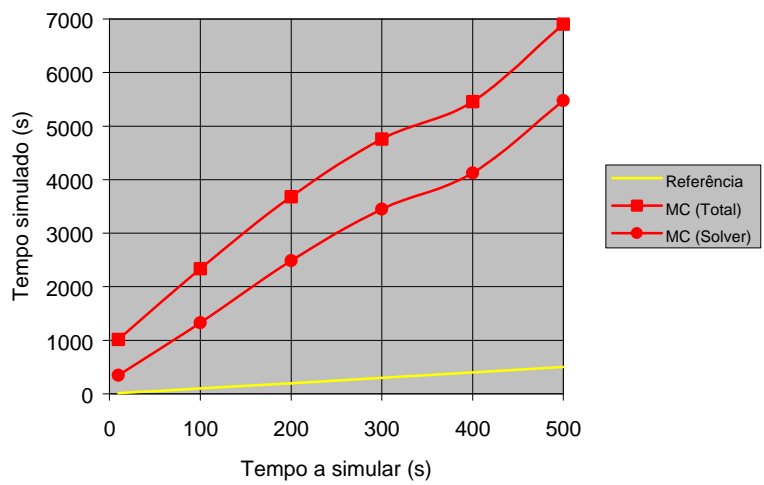


Figura 35. Malha Grande.

Note que para a malha pequena (Figura 33), utilizando a estrutura por aresta, seria viável a sua utilização em apenas um processador, embora não recomendável.

A seguir será feita uma breve descrição sobre o conceito da terminologia usada na medição do desempenho de programas paralelos. Uma das questões fundamentais a respeito de um algoritmo executado em um sistema paralelo, consiste em saber se “este algoritmo é executado rápido, e caso seja, quão rápido é”. Idealmente, se um dado problema é resolvido usando-se  $N$  processadores, então o tempo de execução será reduzido por um fator de  $N$ . Isto nos leva a uma definição básica do ganho, o qual representa a relação entre o tempo de execução em um processador e o tempo de execução em  $N$  processadores.

$$S_p = \frac{\text{Tempo de execução em um processador}}{\text{Tempo de execução em } N \text{ processadores}} \quad (140)$$

Tal medida considera que no tempo de execução observado está embutido a sobrecarga existente no sistema paralelo que pode ser representado pela “quebra” do programa em tarefas paralelas além da comunicação entre elas. A comparação entre o tempo em um processador e o tempo em  $N$  processadores pode, ~~às~~ vezes, levar a um erro. Embora este procedimento dê uma medida precisa da escalabilidade do algoritmo utilizado, ainda não responde a questão de como resolver o problema mais rapidamente usando  $N$  processadores onde o algoritmo paralelo freqüentemente apresenta sobrecargas que não estão presentes nos algoritmos seqüenciais. Ortega e Voigt [48] definiram *speed-up* como sendo a relação entre o tempo de execução do melhor algoritmo seqüencial e aquele utilizado pelo algoritmo paralelo.

$$S_p = \frac{\text{Tempo de execução do melhor algoritmo seqüencial em um processador}}{\text{Tempo de execução do algoritmo paralelo em } N \text{ processadores}} \quad (141)$$

Nos anos 60, Amdahl [49], notou que o *speed-up* era limitado pela porção do programa que não podia ser executado em paralelo. Segundo Amdahl, supondo que um programa que é executado em 10,0 s tenha uma subrotina “chave” que responda por 80% do tempo total de execução e os 20% restantes deste tempo total para outras tarefas. Então, se usarmos uma versão mais eficiente desta subrotina “chave” de tal forma que ela execute duas vezes mais rápido que sua versão anterior, o tempo total de execução cairá para 6,0 s (8,0/2 segundos para a subrotina mais 2,0 segundos para o restante do programa). Se encontrarmos uma subrotina paralela que seja

executada de maneira ideal, em uma máquina paralela com 8 processadores, o tempo total cairá para 3,0 s (8,0/8 segundos para a subrotina paralela mais 2,0 segundos para o restante do programa). Da mesma forma, se a execução for feita idealmente em 100 processadores o tempo de execução será de 2,08 s. À medida que o número de processadores aumenta, o tempo de execução se aproxima do tempo de execução da porção seqüencial, nunca menos do que isso. Neste exemplo, por mais rápido que o programa seja, o tempo total nunca será inferior a 2,0 s, o que representa um *speed-up* máximo de 5,0.

Se normalizarmos a fórmula que define o *speed-up* atribuindo o valor unitário ao tempo de execução da parte seqüencial, e expressando os outros tempos como uma porcentagem do tempo seqüencial, teremos a seguinte formulação da lei de Amdahl para o processamento paralelo:

$$S = \frac{1}{(1-a) + \frac{a}{S_p}} \quad (142)$$

onde  $a$  representa a fração do programa que pode ser paralelizada, e conseqüentemente  $(1-a)$  representará a fração seqüencial. O denominador é o tempo gasto na execução do programa paralelo, isto é, a soma do tempo gasto na parte seqüencial mais o tempo gasto na parte paralela, sendo que o tempo da parte paralela é função do fator de *speed-up* da sua parte paralela. Se a parte paralela apresenta um *speed-up* ideal, isto é, um fator de *speed-up* de  $N$  quando executado em  $N$  processadores, então a equação (142) pode ser reescrita como sendo

$$S = \frac{1}{(1-a) + \frac{a}{N}} \quad (143)$$

Já a eficiência de um programa paralelo pode ser descrita como sendo a relação entre o *speed-up* e o número de processadores usados para obter aquele *speed-up*.

$$E_p = \frac{S_p}{N} \quad (144)$$

Por exemplo, se 10 processadores são usados e o programa é executado 10 vezes mais rápido, então o *speed-up* máximo foi alcançado, e todos os processadores estão

sendo usados em suas máximas potencialidades. Se o *speed-up* é 5,0 então a eficiência do programa é de 50%, significando que metade do poder de processamento está sendo perdido na sobrecarga (comunicação ou sincronização).

Nesta etapa, todas as simulações apresentadas são de curta duração (SCD) e foram realizadas a partir do *cluster* MC. Os resultados foram obtidos de forma a permitir comparações entre as estruturas de dados (por **elemento** e por **aresta**) e entre os métodos de particionamento das malhas (**dual** e **nodal**).

Nas Tabelas 16-18 podem ser vistos os dados obtidos considerando a estrutura de dados por **elemento** e o particionamento **dual (ED)**.

Nas Tabelas 19-21 podem ser vistos os dados obtidos considerando a estrutura de dados por **elemento** e o particionamento **nodal (EN)**.

Nas Tabelas 22-24 podem ser vistos os dados obtidos considerando a estrutura de dados por **aresta** e o particionamento **dual (AD)**.

Nas Tabelas 16-24 o valor informado na 2ª linha do cabeçalho refere-se ao número total de equações da(s) fronteira(s) entre os subdomínios. Adiante serão mostradas análises comparativas a partir destes valores.

Tabela 16. Estrutura por **Elemento** - Particionamento **Dual** - Malha **Pequena**.

	<b>2 P</b>		<b>4 P</b>		<b>6 P</b>		<b>8 P</b>		<b>10 P</b>		<b>12 P</b>		<b>14 P</b>		<b>16 P</b>		
(1)	119 EQ		201 EQ		454 EQ		624 EQ		1430 EQ		971 EQ		1712 EQ		1626 EQ		(2)
	T(3)	S(4)	T	S	T	S	T	S	T	S	T	S	T	S	T	S	
<b>10</b>	30,0	9,5	16,0	5,0	12,0	4,5	10,0	4,5	12,0	6,5	9,5	5,0	12,5	7,4	11,7	7,1	182
<b>100</b>	51,0	18,0	27,0	9,5	21,0	8,5	18,0	8,5	22,0	13,0	17,0	9,5	22,5	14,2	21,4	13,7	370
<b>200</b>	63,0	26,0	33,0	14,0	26,0	12,0	23,0	12,0	29,0	19,0	23,0	14,0	30,5	21,0	28,9	20,3	567
<b>300</b>	78,0	36,0	40,0	19,0	32,0	17,0	29,0	17,0	37,0	26,0	29,0	20,0	39,9	29,5	38,5	28,9	814
<b>400</b>	92,0	46,0	48,0	24,0	38,0	22,0	36,0	23,0	46,0	34,0	36,0	26,0	50,5	38,9	48,1	37,5	1066
<b>500</b>	101	55,0	53,0	29,0	42,0	26,0	40,0	27,0	53,0	40,0	41,0	31,0	57,3	45,7	55,5	44,7	1283

(1) 10 Passos de Tempo (s), (2) Iterações do GMRES, (3) Tempo Total (s), (4) Tempo do *Solver* (s)

Tabela 17. Estrutura por **Elemento** - Particionamento **Dual** - Malha **Média**.

	<b>4 P</b>		<b>6 P</b>		<b>8 P</b>		<b>12 P</b>		<b>16 P</b>		
(1)	1383 EQ		1609 EQ		1298 EQ		1892 EQ		2572 EQ		(2)
	T(3)	S(4)	T	S	T	S	T	S	T	S	
<b>10</b>	62	26	45	19	35	15	29	14	30	16	217
<b>100</b>	133	69	96	52	74	41	63	37	67	44	612
<b>200</b>	209	130	152	98	118	77	102	71	112	83	1182
<b>300</b>	262	175	191	132	148	103	129	95	143	112	1613
<b>400</b>	286	202	210	152	164	119	143	110	160	130	1872
<b>500</b>	349	261	258	197	201	154	178	142	200	168	2445

(1) 10 Passos de Tempo (s), (2) Iterações do GMRES, (3) Tempo Total (s), (4) Tempo do *Solver* (s)

Tabela 18. Estrutura por **Elemento** - Particionamento **Dual** - Malha **Grande**.

	<b>8 P</b>		<b>12 P</b>		<b>16 P</b>		<b>20 P</b>		<b>24 P</b>		
(1)	2871 EQ		3401 EQ		4929 EQ		5989 EQ		7171 EQ		(2)
	T(3)	S(4)	T	S	T	S	T	S	T	S	
<b>10</b>	169	79	126	64	116	63	126	75	136	85	316
<b>100</b>	426	293	331	238	313	235	350	279	386	317	1297
<b>200</b>	703	547	554	444	535	443	661	546	756	638	2471
<b>300</b>	935	765	741	621	710	616	869	760	1002	902	3451
<b>400</b>	1085	920	864	742	839	738	1010	917	1214	1120	4140
<b>500</b>	1421	1235	1121	989	1088	985	1332	1230	1590	1503	5558

(1) 10 Passos de Tempo (s), (2) Iterações do GMRES, (3) Tempo Total (s), (4) Tempo do *Solver* (s)

Tabela 19. Estrutura por **Elemento** - Particionamento **Nodal** - Malha **Pequena**.

	<b>2 P</b>		<b>4 P</b>		<b>6 P</b>		<b>8 P</b>		<b>10 P</b>		<b>12 P</b>		<b>14 P</b>		<b>16 P</b>		
(1)	199 EQ		282 EQ		652 EQ		1023 EQ		929 EQ		1322 EQ		1419 EQ		1599 EQ		(2)
	T(3)	S(4)	T	S	T	S	T	S	T	S	T	S	T	S	T	S	
<b>10</b>	30,0	9,5	16,0	5,0	12,0	4,5	12,0	5,5	10,0	5,0	11,0	6,0	11,2	6,4	11,7	7,1	182
<b>100</b>	52,0	18,0	27,0	10,0	21,0	9,0	21,0	10,0	18,0	10,0	20,0	12,0	20,4	12,5	21,6	13,8	370
<b>200</b>	64,0	26,0	34,0	14,0	27,0	13,0	27,0	15,0	24,0	14,0	27,0	17,0	26,7	18,0	28,7	20,2	567
<b>300</b>	79,0	37,0	41,0	20,0	34,0	18,0	35,0	22,0	30,0	20,0	35,0	24,0	35,8	26,0	38,2	28,7	814
<b>400</b>	93,0	47,0	49,0	25,0	41,0	24,0	42,0	28,0	37,0	26,0	43,0	32,0	43,9	33,3	48,6	37,9	1066
<b>500</b>	102	56,0	54,0	29,0	45,0	28,0	48,0	33,0	42,0	31,0	49,0	38,0	50,4	39,5	55,0	44,4	1283

(1) 10 Passos de Tempo (s), (2) Iterações do GMRES, (3) Tempo Total (s), (4) Tempo do *Solver* (s)

Tabela 20. Estrutura por **Elemento** - Particionamento **Nodal** - Malha **Média**.

	<b>4 P</b>		<b>6 P</b>		<b>8 P</b>		<b>12 P</b>		<b>16 P</b>		
(1)	1120 EQ		1668 EQ		1828 EQ		2260 EQ		3923 EQ		(2)
	T(3)	S(4)	T	S	T	S	T	S	T	S	
<b>10</b>	62	25	45	20	36	16	30	15	37	22	217
<b>100</b>	132	68	97	53	78	43	66	40	84	59	612
<b>200</b>	207	128	154	100	124	81	108	76	141	111	1182
<b>300</b>	260	172	194	134	156	109	137	102	182	149	1613
<b>400</b>	284	199	212	155	171	126	151	118	207	174	1872
<b>500</b>	347	257	261	200	211	163	188	153	261	226	2445

(1) 10 Passos de Tempo (s), (2) Iterações do GMRES, (3) Tempo Total (s), (4) Tempo do *Solver* (s)

Tabela 21. Estrutura por **Elemento** - Particionamento **Nodal** - Malha **Grande**.

	<b>8 P</b>		<b>12 P</b>		<b>16 P</b>		<b>20 P</b>		<b>24 P</b>		
(1)	4863 EQ		6717 EQ		8137 EQ		7876 EQ		9806 EQ		(2)
	T(3)	S(4)	T	S	T	S	T	S	T	S	
<b>10</b>	188	97	166	96	161	102	155	101	187	131	316
<b>100</b>	498	360	468	366	481	393	471	391	591	507	1297
<b>200</b>	835	673	807	684	843	739	832	737	1058	958	2471
<b>300</b>	1109	932	1085	952	1138	1026	1129	1026	1440	1332	3451
<b>400</b>	1292	1113	1272	1137	1342	1228	1326	1226	1704	1594	4140
<b>500</b>	1677	1483	1656	1513	1768	1645	1751	1638	2251	2133	5558

(1) 10 Passos de Tempo (s), (2) Iterações do GMRES, (3) Tempo Total (s), (4) Tempo do *Solver* (s)

Tabela 22. Estrutura por **Aresta** - Particionamento **Dual** - Malha **Pequena**.

	<b>2 P</b>		<b>4 P</b>		<b>6 P</b>		<b>8 P</b>		<b>10 P</b>		<b>12 P</b>		<b>14 P</b>		<b>16 P</b>		
(1)	119 EQ		201 EQ		454 EQ		624 EQ		1430 EQ		971 EQ		1712 EQ		1626 EQ		(2)
	T(3)	S(4)	T	S	T	S	T	S	T	S	T	S	T	S	T	S	
<b>10</b>	30,0	7,4	15,9	3,9	11,9	3,6	10,4	3,9	12,2	6,0	9,6	4,6	12,5	7,1	11,5	6,7	182
<b>100</b>	52,2	14,3	27,6	7,6	20,8	7,0	18,2	7,6	21,7	11,8	17,2	9,1	22,4	13,7	21,1	13,0	370
<b>200</b>	64,0	21,1	33,5	11,0	25,7	10,4	23,2	11,1	28,6	17,4	22,6	13,5	30,3	20,4	28,3	19,3	567
<b>300</b>	77,1	29,6	40,3	15,4	31,6	14,6	29,2	15,8	37,0	24,5	29,3	19,0	40,0	29,0	37,7	27,9	814
<b>400</b>	90,5	38,2	47,1	19,9	37,5	18,9	35,4	20,6	45,5	31,9	35,9	24,8	49,7	37,6	46,9	36,1	1066
<b>500</b>	98,6	45,7	51,3	23,8	41,3	22,6	39,4	24,6	51,8	38,1	40,7	29,7	56,4	44,9	54,0	43,3	1283

(1) 10 Passos de Tempo (s), (2) Iterações do GMRES, (3) Tempo Total (s), (4) Tempo do *Solver* (s)

Tabela 23. Estrutura por **Aresta** - Particionamento **Dual** - Malha **Média**.

	<b>4 P</b>		<b>6 P</b>		<b>8 P</b>		<b>12 P</b>		<b>16 P</b>		
(1)	1383 EQ		1609 EQ		1298 EQ		1892 EQ		2572 EQ		(2)
	T(3)	S(4)	T	S	T	S	T	S	T	S	
<b>10</b>	63,9	21,2	45,2	16,2	35,1	13,0	29,3	12,7	30,2	15,0	217
<b>100</b>	132,6	58,7	95,1	44,8	74,1	35,8	63,9	34,7	66,8	41,4	612
<b>200</b>	204,3	112,5	147,8	85,5	115,5	67,7	102,4	65,8	110,1	78,8	1182
<b>300</b>	254,9	153,6	184,9	116,1	144,4	91,8	126,9	89,5	140,6	107,1	1613
<b>400</b>	274,9	175,5	199,1	131,6	156,9	105,7	139,5	101,2	156,1	121,3	1872
<b>500</b>	333,7	229,0	247,5	175,6	191,4	138,0	172,3	131,4	193,5	157,7	2445

(1) 10 Passos de Tempo (s), (2) Iterações do GMRES, (3) Tempo Total (s), (4) Tempo do *Solver* (s)

Tabela 24. Estrutura por **Aresta** - Particionamento **Dual** - Malha **Grande**.

	<b>8 P</b>		<b>12 P</b>		<b>16 P</b>		<b>20 P</b>		<b>24 P</b>		(2)
	2871 EQ		3401 EQ		4929 EQ		5989 EQ		7171 EQ		
	(1)										
	T(3)	S(4)	T	S	T	S	T	S	T	S	
<b>10</b>	170,6	68,6	127,9	55,8	117,0	56,8	121,7	66,4	137,4	77,8	316
<b>100</b>	418,5	261,8	321,0	213,4	311,6	218,1	335,7	253,3	384,5	298,9	1297
<b>200</b>	673,2	493,2	525,7	401,5	507,5	409,5	562,4	477,2	633,5	563,3	2471
<b>300</b>	893,0	693,7	700,5	560,9	687,1	574,9	773,0	667,9	895,2	790,2	3451
<b>400</b>	1034,5	834,4	811,8	673,7	803,3	691,6	898,6	801,2	1038,7	948,3	4140
<b>500</b>	1329,2	1112,5	1050,9	899,5	1023,4	922,9	1158,0	1068,5	1325,2	1265,3	5558

(1) 10 Passos de Tempo (s), (2) Iterações do GMRES, (3) Tempo Total (s), (4) Tempo do *Solver* (s)

Os resultados obtidos nas Tabelas 16-24 podem ser agrupados de outra forma e melhor visualizados através de gráficos (Figuras 36-44).

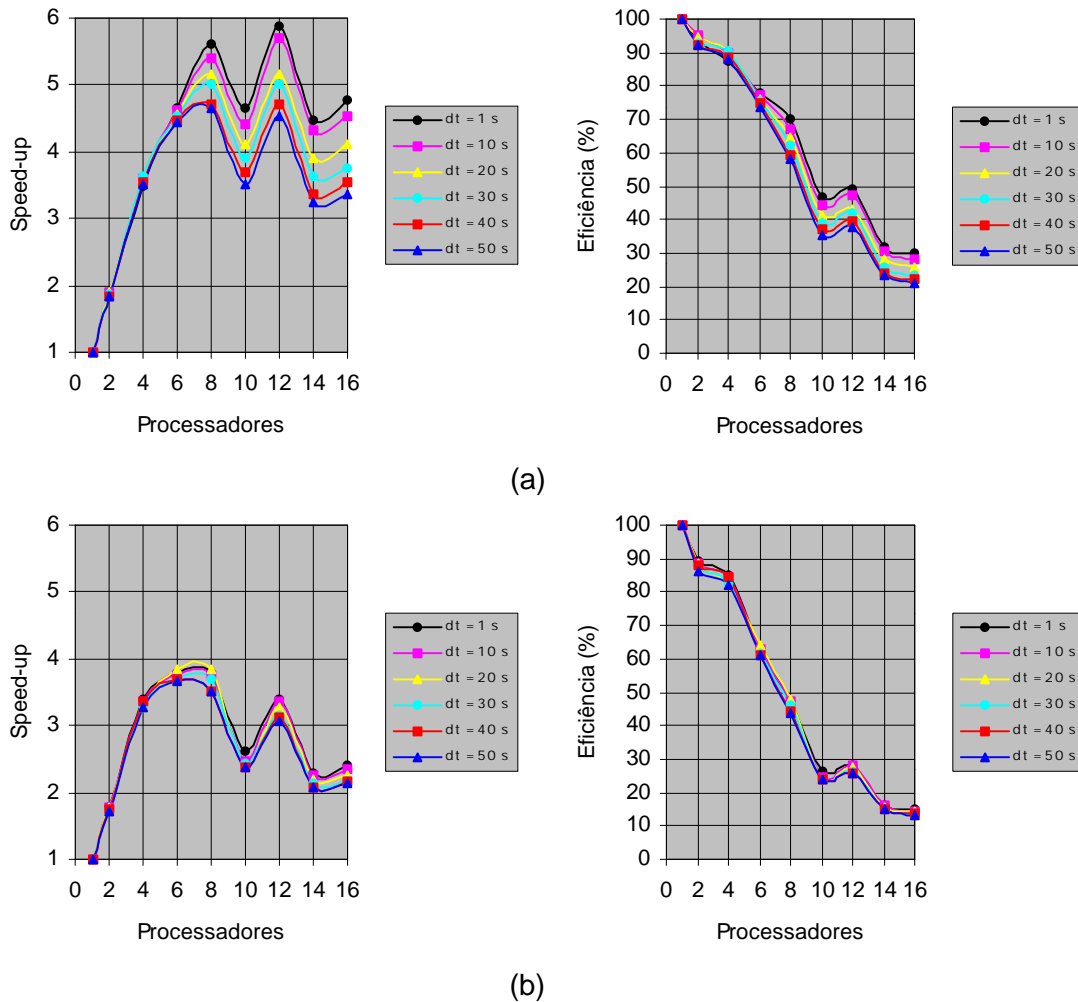
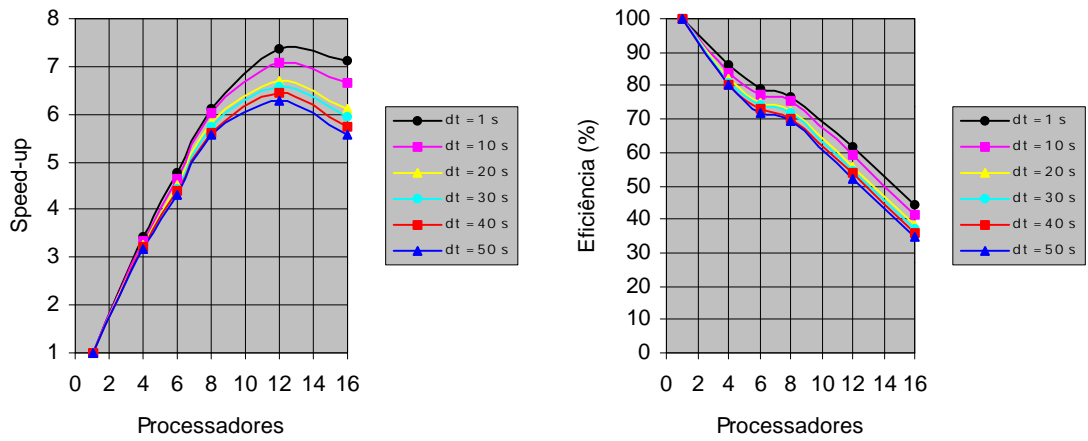
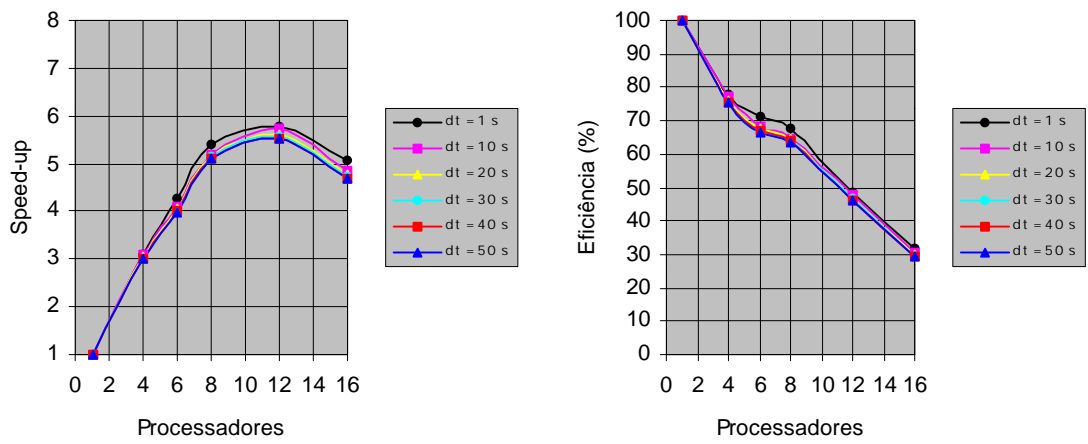


Figura 36. Estrutura por **Elemento** - Particionamento **Dual** - Malha **Pequena**

(a) Tempo Total (b) Tempo do *Solver*.



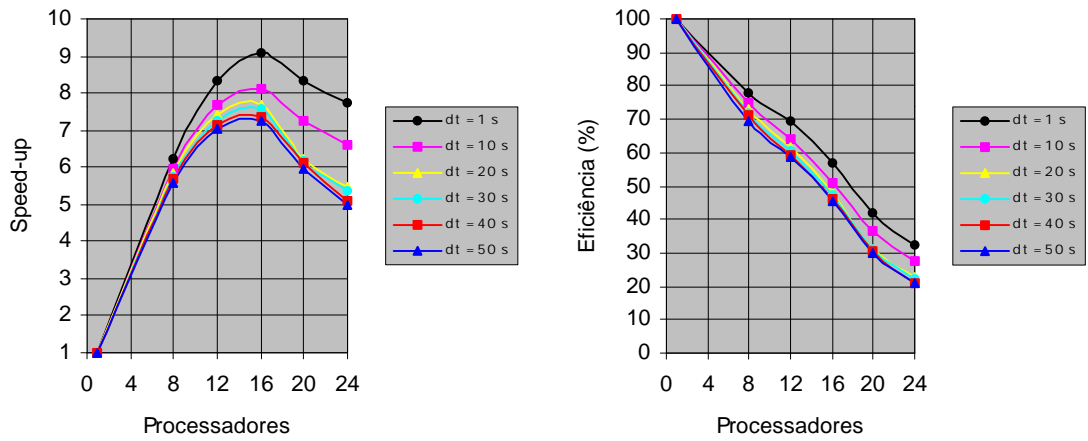
(a)



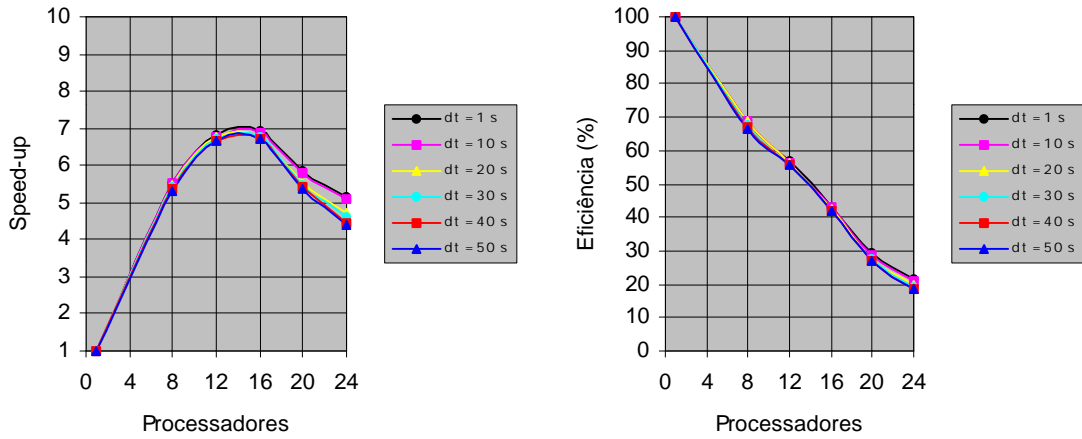
(b)

Figura 37. Estrutura por **Elemento** - Particionamento **Dual** - Malha **Média**

(a) Tempo Total (b) Tempo do *Solver*.



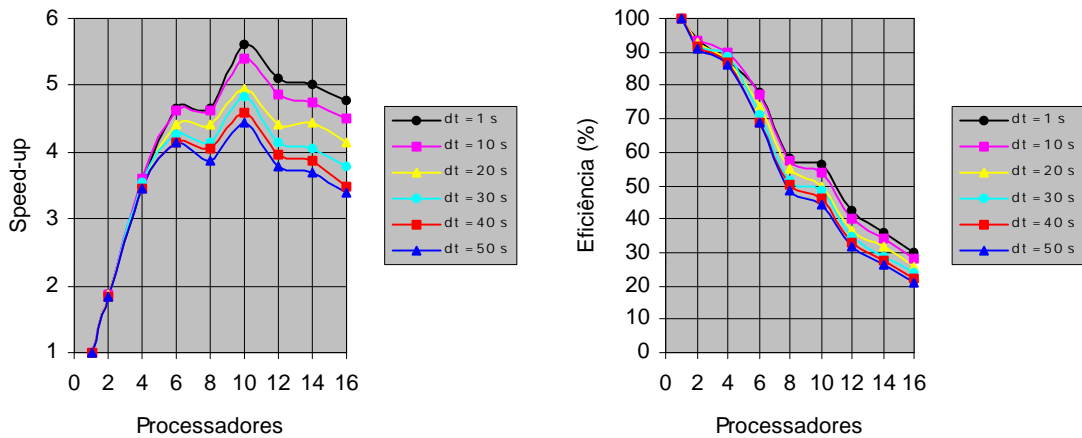
(a)



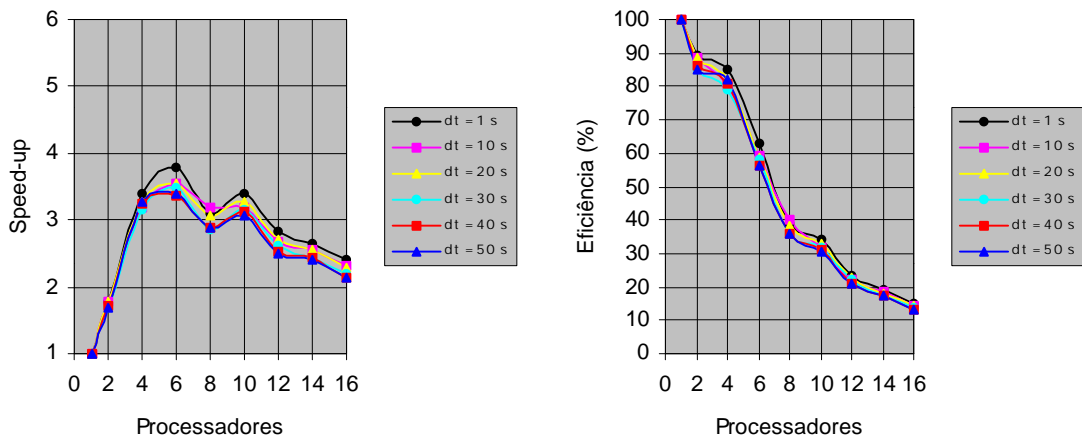
(b)

Figura 38. Estrutura por **Elemento** - Particionamento **Dual** - Malha **Grande**

(a) Tempo Total (b) Tempo do *Solver*.



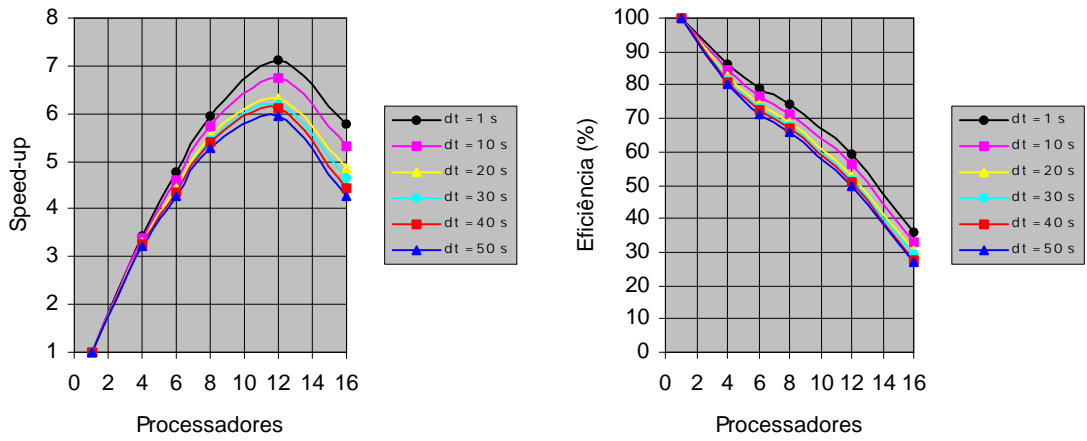
(a)



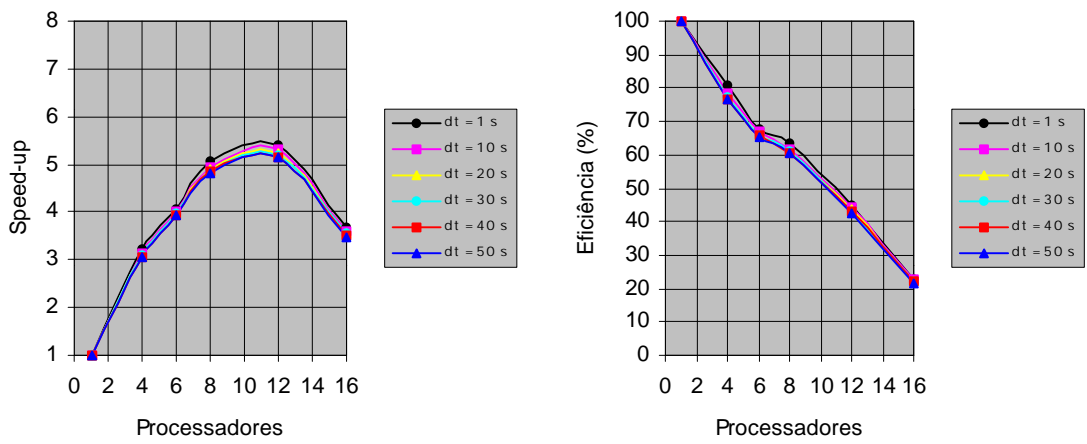
(b)

Figura 39. Estrutura por **Elemento** - Particionamento **Nodal** - Malha **Pequena**

(a) Tempo Total (b) Tempo do *Solver*.



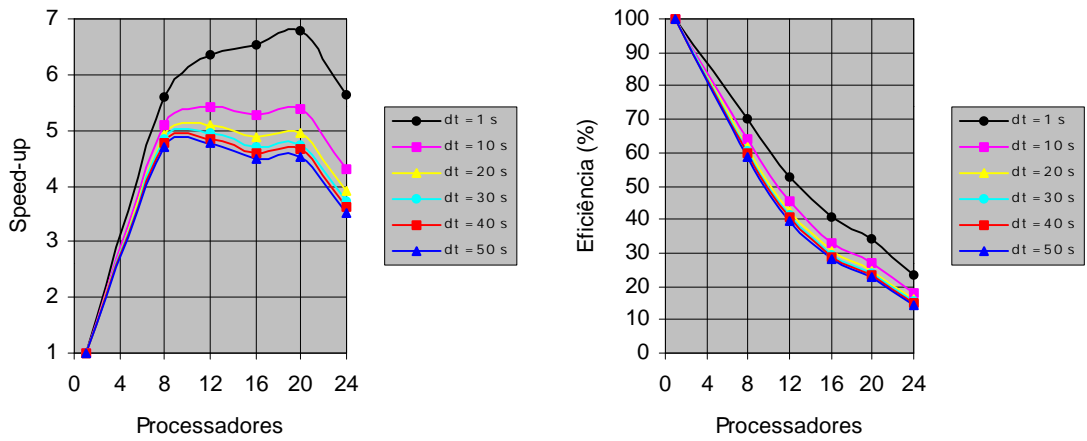
(a)



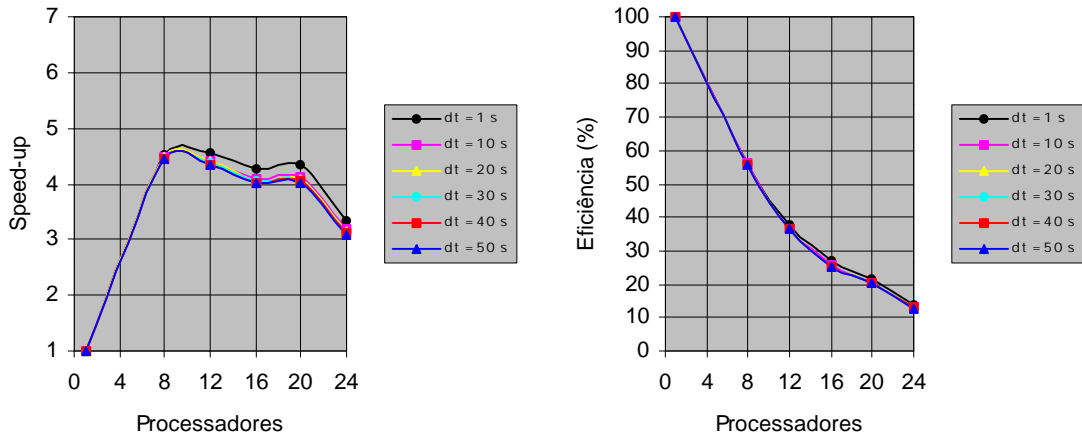
(b)

Figura 40. Estrutura por **Elemento** - Particionamento **Nodal** - Malha **Média**

(a) Tempo Total (b) Tempo do *Solver*.



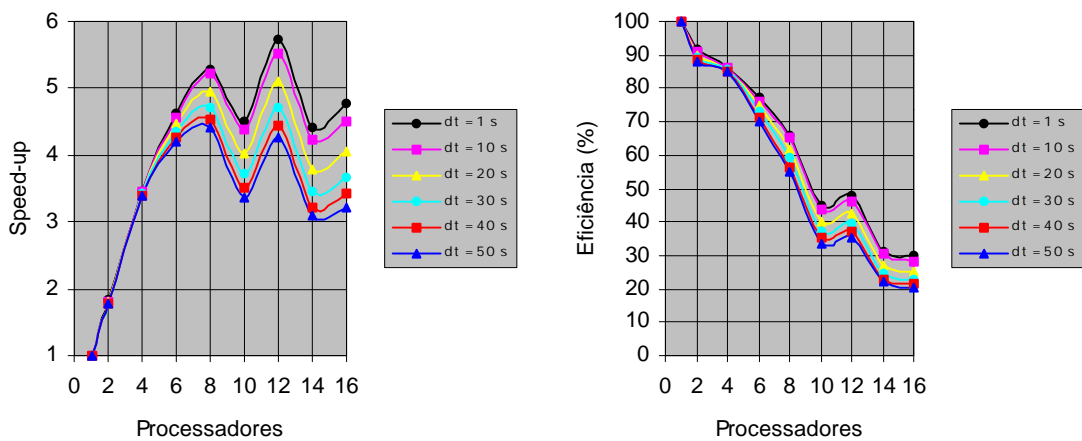
(a)



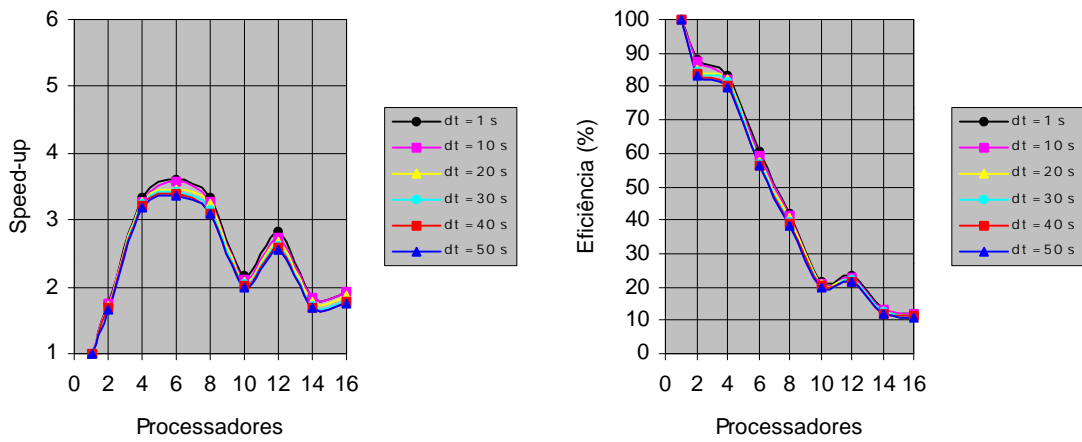
(b)

Figura 41. Estrutura por **Elemento** - Particionamento **Nodal** - Malha **Grande**

(a) Tempo Total (b) Tempo do *Solver*.



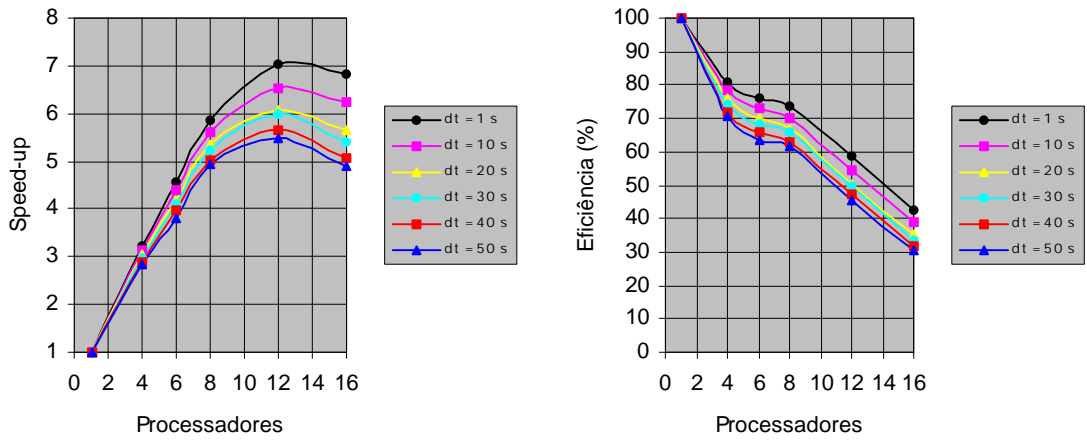
(a)



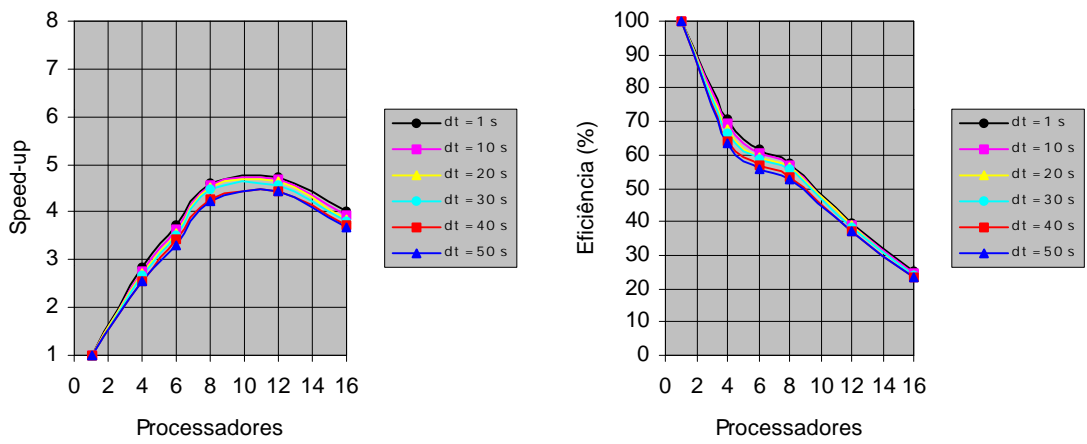
(b)

Figura 42. Estrutura por **Aresta** - Particionamento **Dual** - Malha **Pequena**

(a) Tempo Total (b) Tempo do *Solver*.



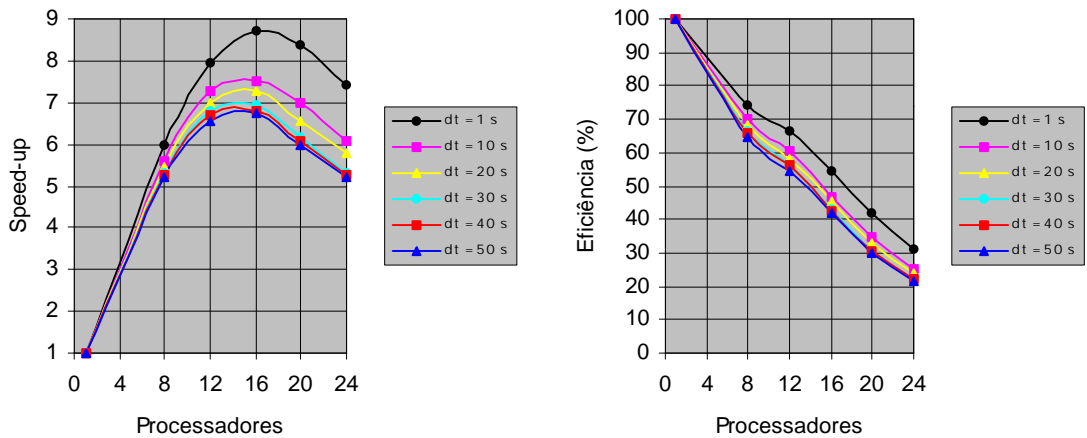
(a)



(b)

Figura 43. Estrutura por **Aresta** - Particionamento **Dual** - Malha **Média**

(a) Tempo Total (b) Tempo do *Solver*.



(a)

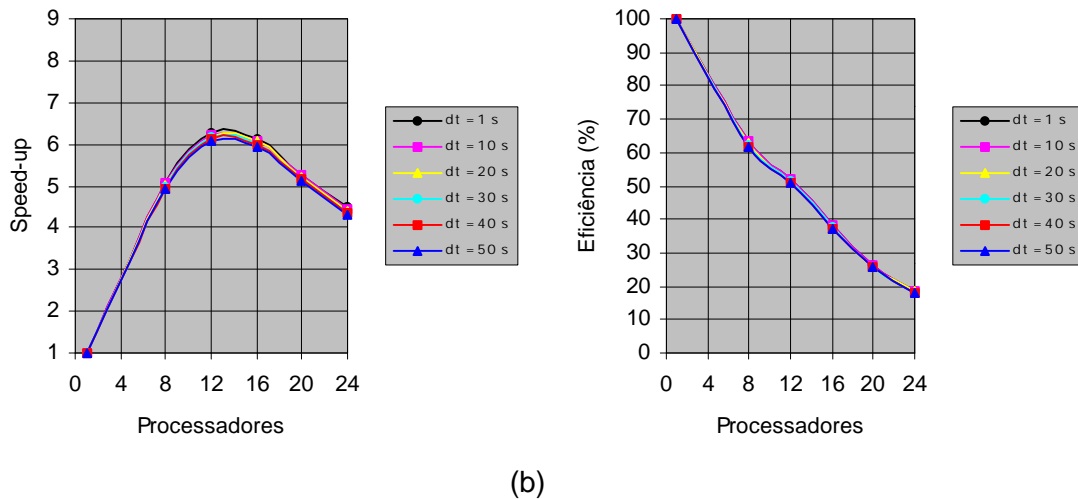


Figura 44. Estrutura por **Aresta** - Particionamento **Dual** - Malha **Grande**

(a) Tempo Total (b) Tempo do *Solver*.

Observando os resultados obtidos, podemos fazer as seguintes análises comparativas para as três situações apresentadas - Estrutura por Elemento com Particionamento Dual (**ED**), Estrutura por Elemento com Particionamento Nodal (**EN**) e Estrutura por Aresta com Particionamento Dual (**AD**):

- As configurações **ED** e **AD** apresentaram aumento no *speed-up* à medida que a malha correspondente foi refinada (Figuras 36-38 e 42-44). Já a configuração **EN** apresentou um aumento no *speed-up* apenas no refinamento da malha pequena para a malha média (Figuras 39 e 40), decaindo o seu valor no refinamento da malha média para a malha grande (Figuras 40 e 41). Este último comportamento pode ser explicado pelo aumento da comunicação provocado pelo excessivo número de equações nas fronteiras da malha grande (Tabela 21) nessa configuração (**EN**) comparado às outras duas configurações (**ED** e **AD**).
- Analisando as três configurações (**ED**, **EN** e **AD**), vemos que para a simulação da malha pequena um comportamento não usual na evolução do *speed-up* pode ser percebido, sendo maior nas configurações **ED** (Figura 36) e **AD** (Figura 42) e menor em **EN** (Figura 39). Observando o número de processadores onde estas perturbações ocorreram, temos que para as configurações **ED** e **AD** houve queda no *speed-up* para o número de processadores igual a 10 e 14, visto que os respectivos números de equações nas fronteiras aumentaram bruscamente (Tabelas 16 e 22). A configuração **EN** apresentou queda no *speed-up*, também devido ao aumento anormal do número de equações em sua fronteira (Tabela 19), só que para 8 e 12 processadores. Analisando o particionamento realizado pelo METIS para 10 e 14 processadores (Figuras 13 e 15), por exemplo, observamos

que além da existência de partições desconexas os subdomínios apresentaram fronteiras muito longas. Desconsiderando estes pontos citados anteriormente, obtemos uma nova curva (à direita), mais representativa do comportamento esperado do desempenho da implementação paralela (Figura 45-47).

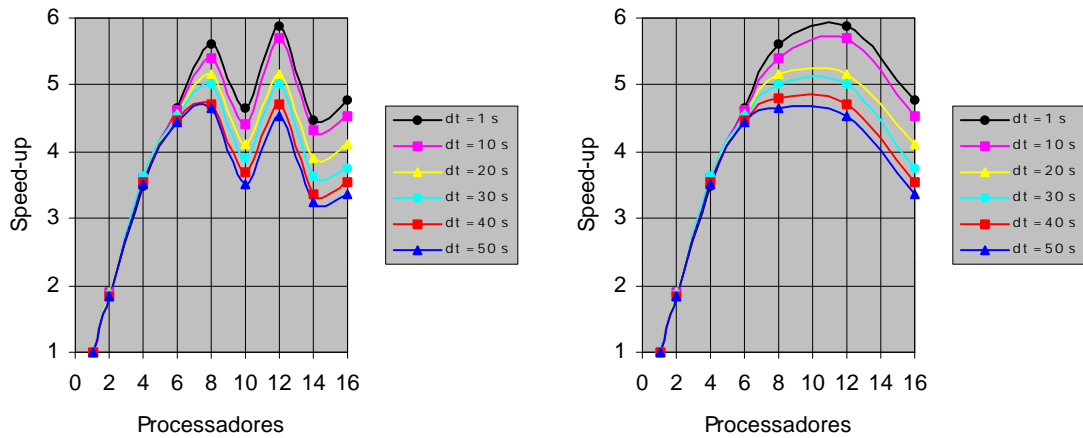


Figura 45. Configuração **ED** – Tempo Total.

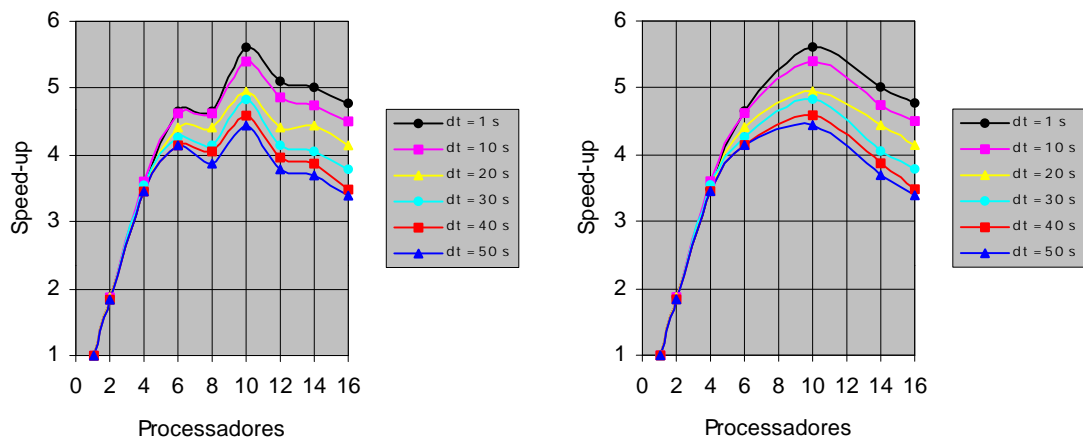


Figura 46. Configuração **EN** – Tempo Total.

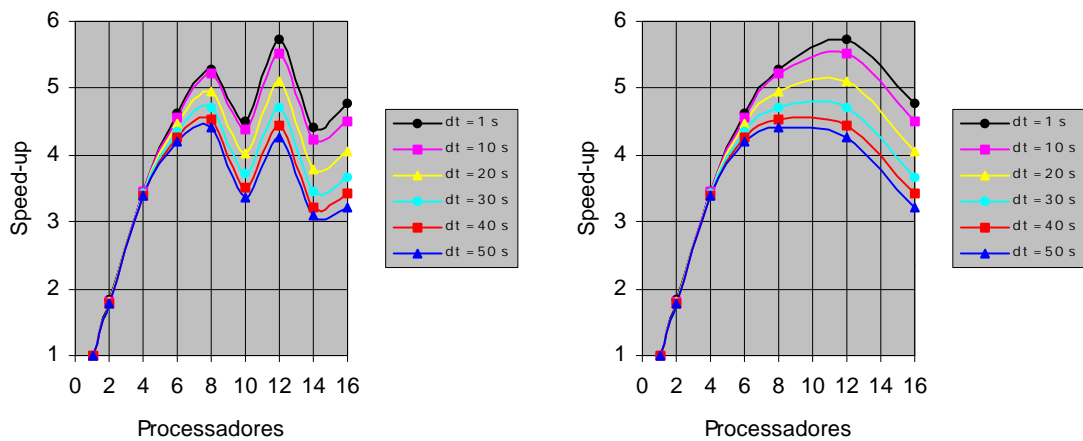


Figura 47. Configuração **AD** – Tempo Total.

- O número ótimo de processadores em todas as simulações variou conforme a configuração utilizada (**ED**, **EN** ou **AD**), o refinamento da malha e o tamanho do passo de tempo. Para a malha pequena, o melhor *speed-up*, para um passo de tempo de 50s, foi obtido com 8 processadores nas configurações **ED** e **AD** e 10 processadores na configuração **EN**. Já para a malha média, o melhor *speed-up*, para um passo de tempo de 50s, foi obtido com 12 processadores em todas as configurações. A malha grande obteve o melhor *speed-up*, para um passo de tempo de 1s, com 16 processadores nas configurações **ED** e **AD** e 20 processadores na configuração **EN**. Em praticamente todas as simulações pode ser observado que o *speed-up* diminui à medida que o passo de tempo aumenta. Podemos associar este comportamento ao aumento do número de iterações do método iterativo GMRES utilizando o tamanho do passo de tempo maior (Figura 48).

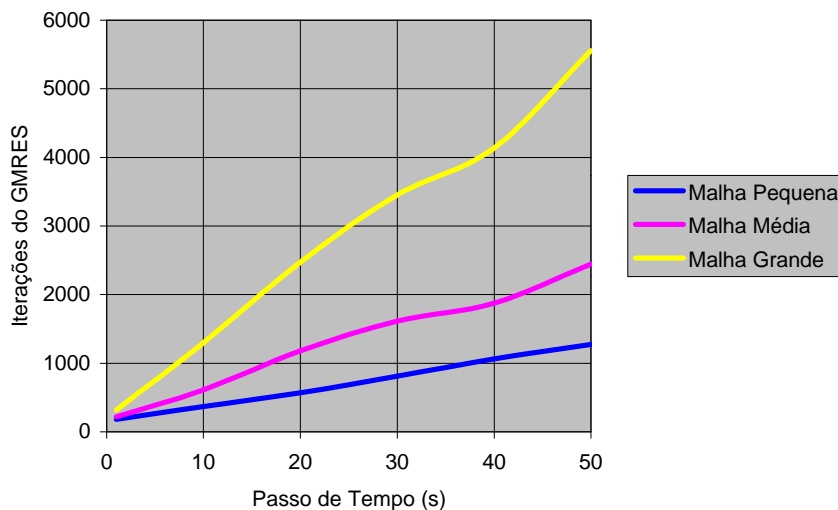


Figura 48. Número de iterações GMRES x Tamanho do passo de tempo.

- Comparando os tipos de particionamentos utilizados, para o passo de tempo de 50s, vemos que para a simulação da malha pequena utilizando 8 processadores, o *solver* da configuração **ED** (Tabela 16) foi executado em 27s enquanto que o *solver* da configuração **EN** (Tabela 19) foi executado em 33s nas mesmas condições, aumentando o custo computacional em 22%. Para a malha média com 12 processadores, o *solver* da configuração **ED** (Tabela 17) foi executado em 142s enquanto que o *solver* da configuração **EN** (Tabela 20) foi executado em 153s nas mesmas condições, gerando um aumento no custo computacional de quase 8%. Já a simulação da malha grande com 16 processadores, o *solver* da configuração **ED** (Tabela 18) foi executado em 985s enquanto que o *solver* da configuração **EN**

(Tabela 21) foi executado em 1.645s, aumentando o custo computacional em 67%. Assim sendo, podemos concluir que o particionamento dual é mais favorável que o nodal, pois via de regra utiliza menos equações na(s) fronteira(s) entre seus subdomínios, o que diminui o custo computacional.

- Comparando as estruturas de dados utilizadas, para o passo de tempo de 50s, vemos que para a simulação da malha pequena com 8 processadores, o *solver* da configuração **ED** (Tabela 16) foi executado em 27s enquanto que o *solver* da configuração **AD** (Tabela 22) foi executado em 24,6s nas mesmas condições, reduzindo o custo computacional em quase 9%. Para a malha média com 12 processadores, o *solver* da configuração **ED** (Tabela 17) foi executado em 142s enquanto que o *solver* da configuração **AD** (Tabela 23) foi executado em 131,4s nas mesmas condições, gerando uma redução no custo computacional de quase 8%. Já a simulação da malha grande com 16 processadores, o *solver* da configuração **ED** (Tabela 18) foi executado em 985s enquanto que o *solver* da configuração **AD** (Tabela 24) foi executado em 922,9s, reduzindo o custo computacional em 6,3%. Esta redução também pode ser vista quando da simulação em apenas um processador (Tabelas 10-15), onde os percentuais de redução são de 20%, 25,6% e 16,9%, para as malhas pequena, média e grande respectivamente. Assim sendo, podemos dizer que a estrutura por aresta é mais favorável que a estrutura por elemento, visto que sua matriz é menor que a matriz por elemento.
- Tomando por base a melhor configuração (Estrutura por **Aresta** e Particionamento **Dual**), podemos dizer que a simulação de 500s (tempo real) foi melhor realizada para a malha pequena utilizando 8 processadores em aproximadamente 40s (Tabela 22), reduzindo o tempo real simulado por um fator de 12,5. Já a malha média foi melhor simulada, nas mesmas condições, com 12 processadores em 172s (Tabela 23), produzindo um fator de redução de 2,9. No entanto, nenhuma redução foi obtida na malha grande, visto que, nas mesmas condições, a melhor simulação em 16 processadores foi realizada em 1.023s (Tabela 24). Se um *cluster* com maior capacidade de processamento for utilizado, reduções semelhantes às anteriores, no tempo de simulação, poderão ser obtidas também para a malha grande.
- Observando os resultados obtidos nas Tabelas 22-24 sob o aspecto da escalabilidade, isto é, como o tempo de simulação é afetado pela utilização de

malhas maiores, vemos que o tempo de simulação em função do refinamento, obedece a uma função que apresenta um comportamento não-linear, onde um dos parâmetros está diretamente associado ao número de iterações do método iterativo. Por exemplo, utilizando novamente a melhor configuração (Estrutura por **Aresta** e Particionamento **Dual**), vemos que o tempo gasto na simulação da malha pequena, para o passo de tempo de 30s, utilizando-se 2 processadores, foi de 77,1s (Tabela 22), enquanto que a malha média, para o mesmo passo de tempo, com 8 processadores (4 x 2), foi executada em 144,4s (Tabela 23), produzindo um “fator de refinamento” do ponto de vista do tempo simulado igual a **1,87** (144,4 / 77,1). Já o “fator de refinamento”, também para um passo de tempo de 30s, entre a malha média, utilizando-se 4 processadores, e a malha grande, utilizando-se 16 processadores (4 x 4), foi igual a **2,70** (687,1 / 254,9). Observando o “fator de refinamento” pelo ponto de vista do número de iterações do método iterativo GMRES (Tabelas 22-24), vemos que, para o passo de tempo de 30s, da malha pequena para a malha média este fator foi igual a **1,98** (1613 / 814) e da malha média para a malha grande foi igual a **2,14** (3451 / 1613). A Figura 49 ilustra este último aspecto, onde o comportamento não-linear do número de iterações do GMRES em função do refinamento afeta a escalabilidade. Outro parâmetro importante na análise da escalabilidade é o tempo gasto na comunicação entre os processos. Esta comunicação aumenta quando se passa da malha pequena utilizando-se 2 processadores para a malha média com 8 (4 x 2) processadores ou da malha média utilizando-se 4 processadores para a malha grande com 16 (4 x 4) processadores.

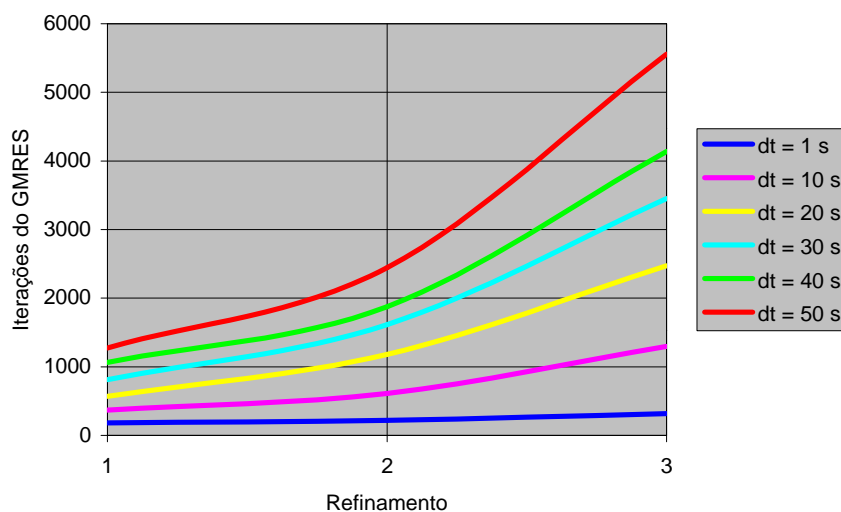


Figura 49. Número de iterações GMRES x Refinamento (1-Malha Pequena, 2-Malha Média, Malha Grande).

A seguir são mostrados os resultados obtidos nas simulações de longa duração (SLD) e que representam as medições temporais dos pontos definidos na Figura 28 (Figuras 50-56). Lembrando que estas medições foram feitas a partir da malha pequena, utilizando 21.000 passos de tempo onde cada passo de tempo era de 30s.

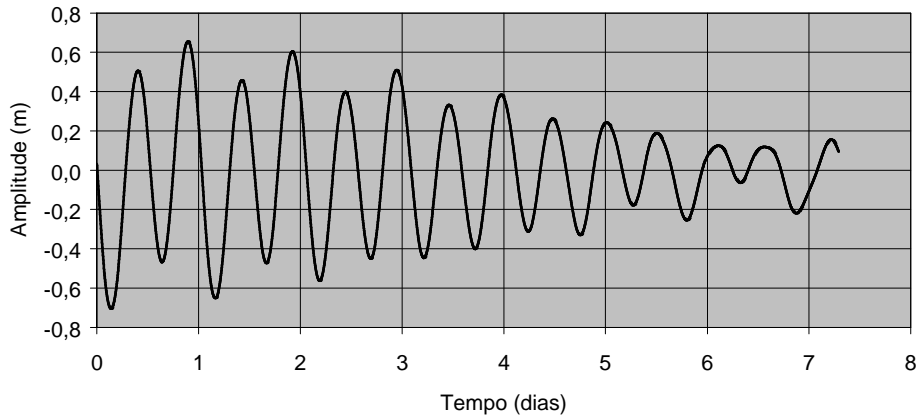


Figura 50. Medição da elevação do nó 5322 (**Ponto 1**).

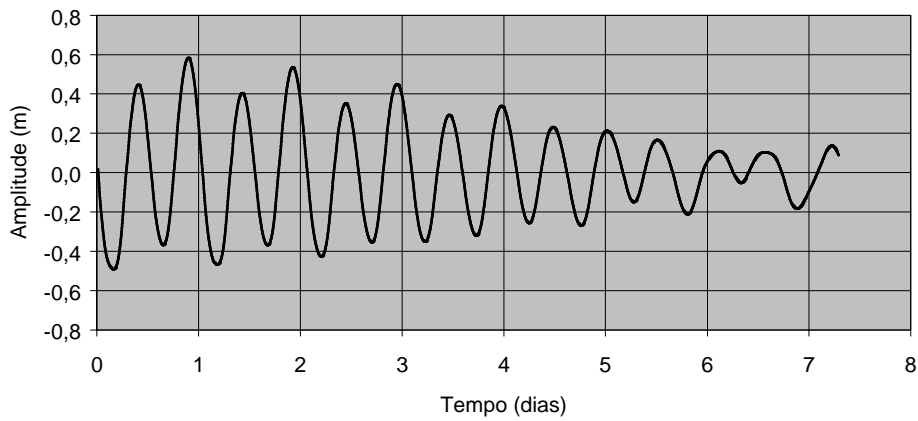


Figura 51. Medição da elevação do nó 5158 (**Ponto 2**).

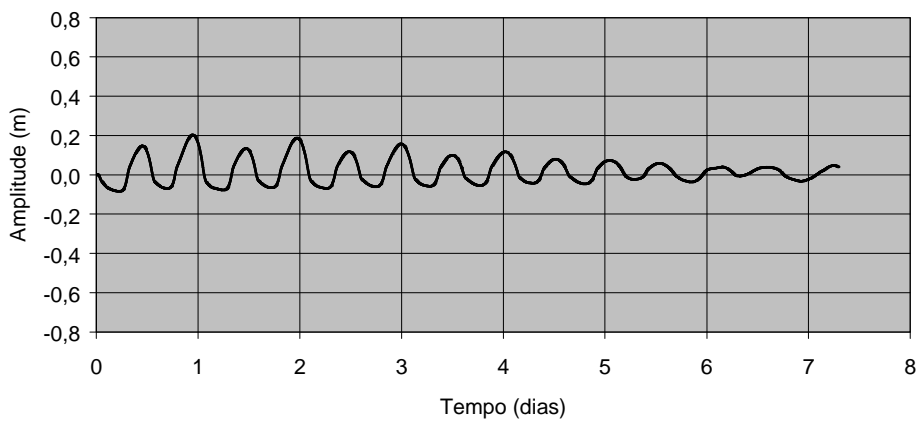


Figura 52. Medição da elevação do nó 4811 (**Ponto 3**).

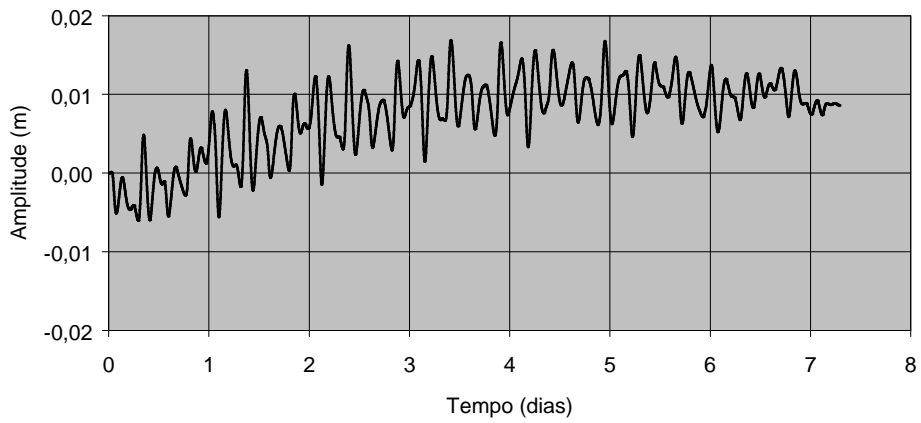


Figura 53. Medição da elevação do nó 4083 (Ponto 4).

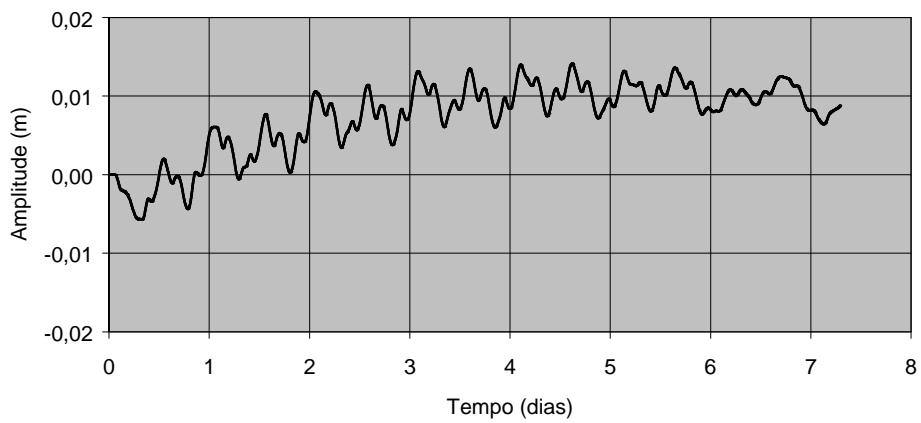


Figura 54. Medição da elevação do nó 2360 (Ponto 5).

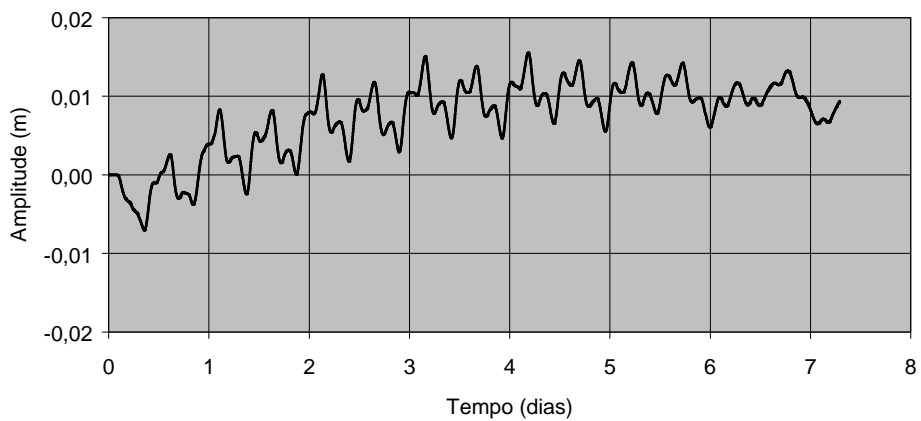


Figura 55. Medição da elevação do nó 1156 (Ponto 6).

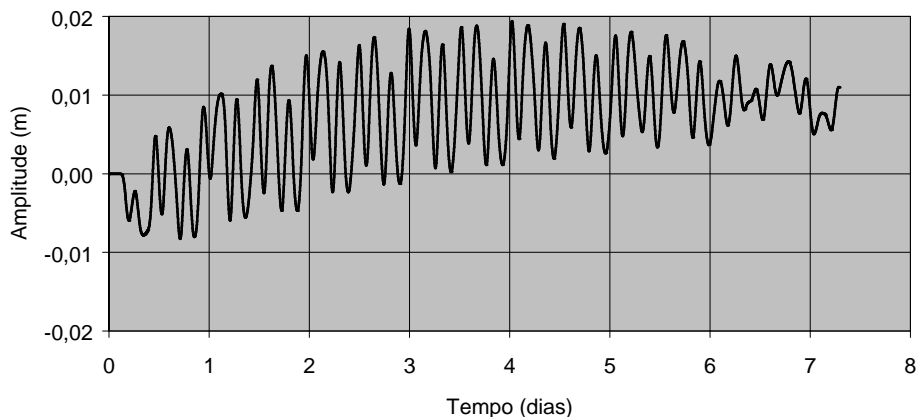


Figura 56. Medição da elevação do nó 186 (**Ponto 7**).

A partir dos gráficos anteriores referentes aos sete pontos de medição temporal, podemos observar que apenas os três primeiros pontos de medição (Figuras 50-52) apresentaram algum efeito, embora atenuado, do comportamento de elevação da maré na entrada do canal. Os quatro últimos pontos de medição (Figuras 53-56), embora apresentem algum comportamento “periódico”, na verdade apenas indicam uma condição inicial (observar as escalas). Mesmo para as maiores amplitudes aplicadas à entrada do canal ao longo de toda a simulação, não foi possível obter energia suficiente para atingir o interior da lagoa. Como o comportamento da maré na entrada do canal apresenta influência apenas na porção inicial da Lagoa, não tendo qualquer influência em seu interior, foi ilustrado a seguir o comportamento hidrodinâmico somente desta porção inicial. Considere inicialmente o fluxo entrando no canal (Figuras 57, 58), e em seguida o fluxo saindo pelo canal (Figuras 59-61), maré alta ( $t = 36.000s$ ) e maré baixa ( $t = 60.000s$ ) respectivamente. A Figura 61 mostra com mais detalhes o escoamento através da garganta do canal, onde pode ser observado o efeito de vórtice, representado pela cor azul.

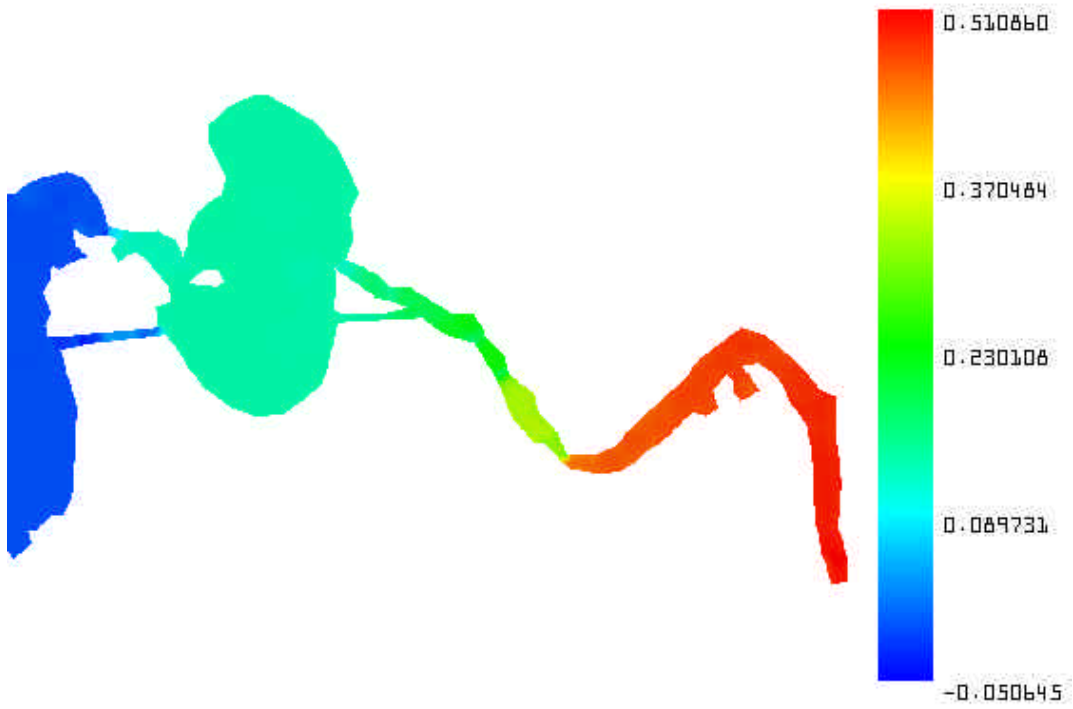


Figura 57. Perfil de elevação (m) para  $t = 36.000s$ .

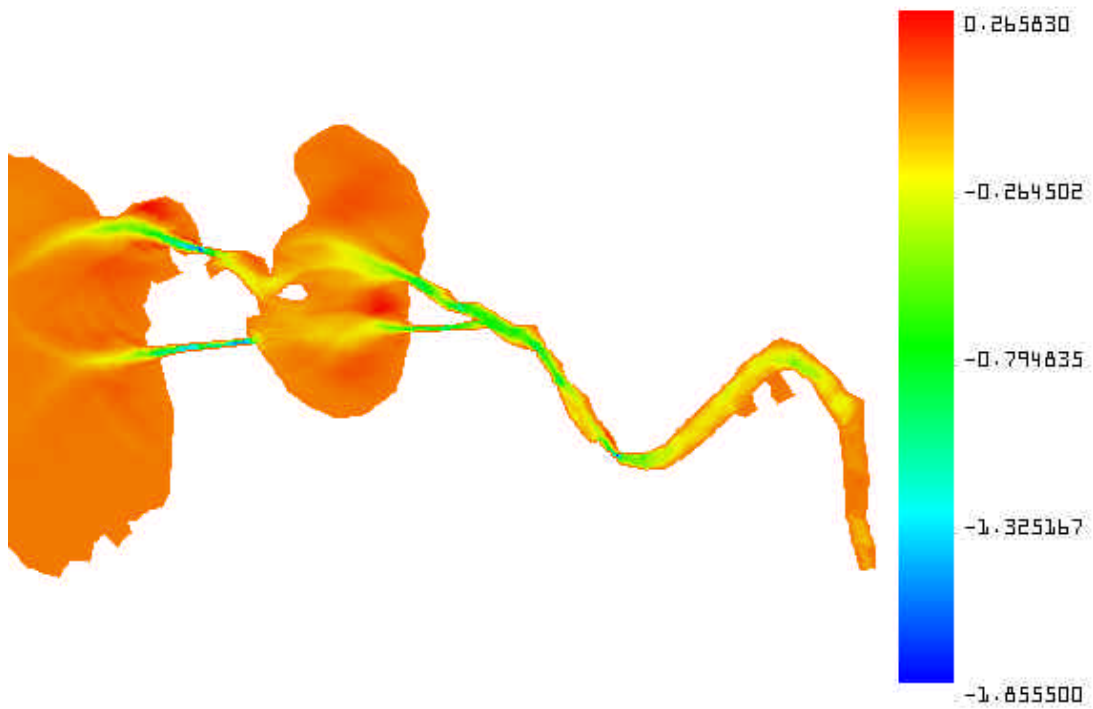


Figura 58. Perfil da velocidade horizontal (m/s) para  $t = 36.000s$ .

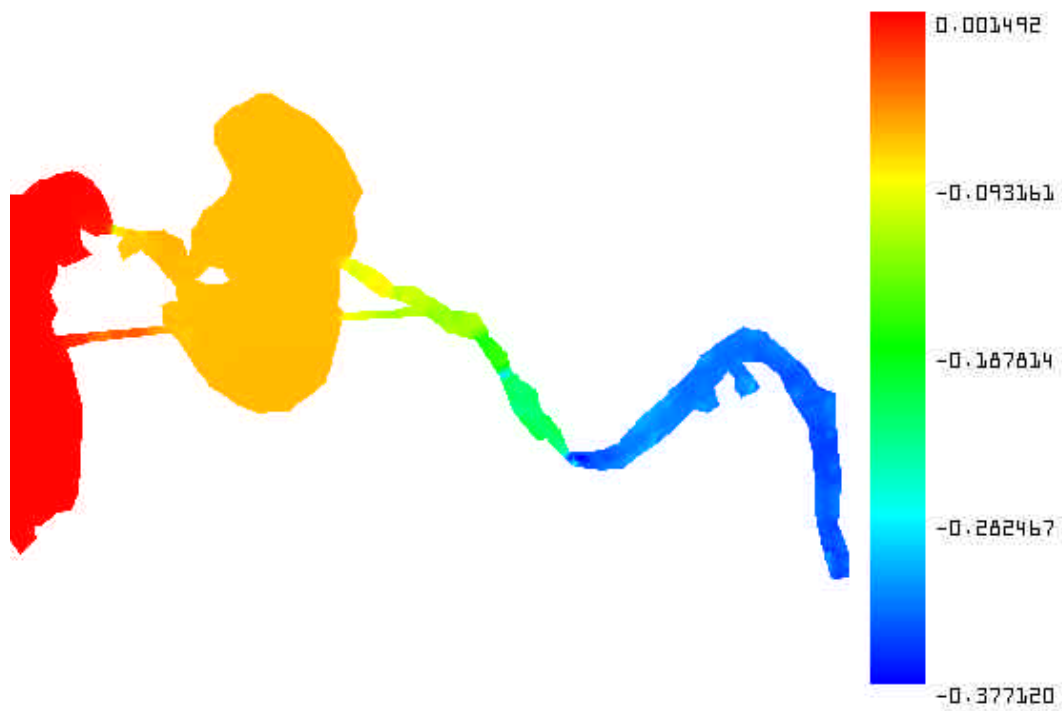


Figura 59. Perfil de elevação (m) para  $t = 60.000s$ .

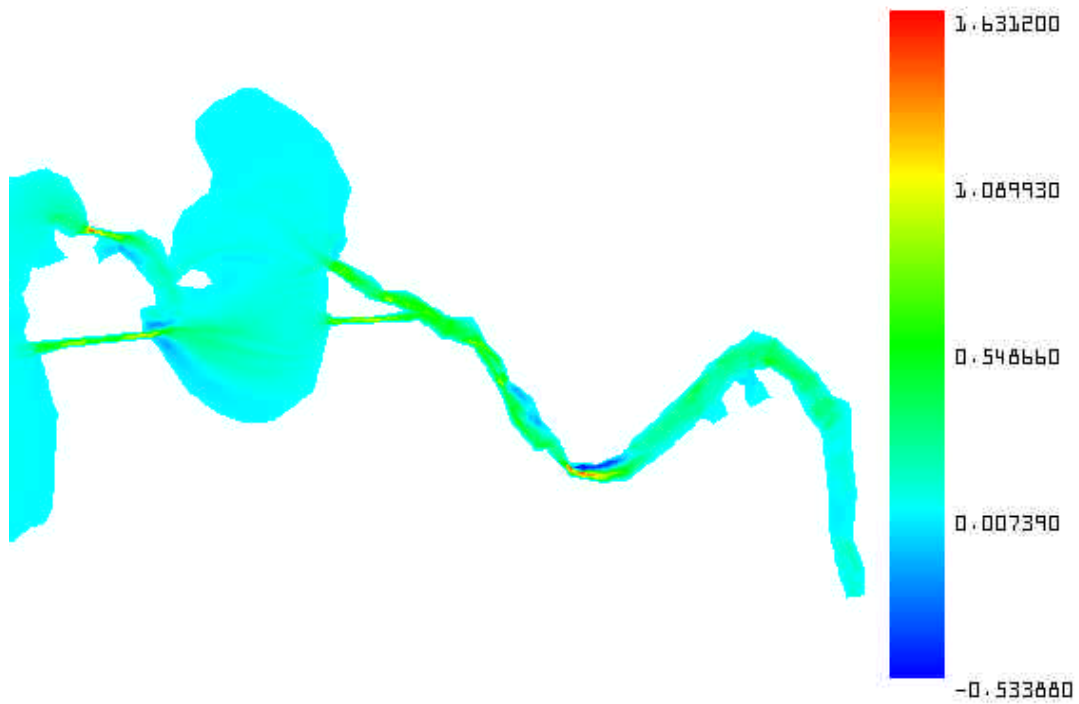


Figura 60. Perfil da velocidade horizontal (m/s) para  $t = 60.000s$ .

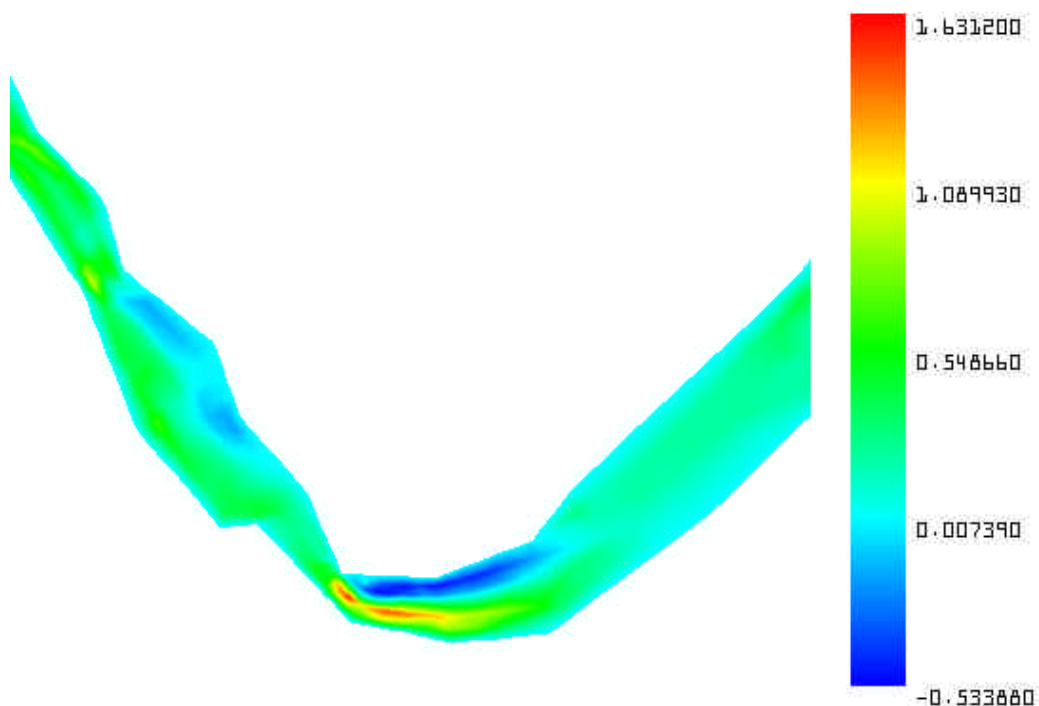


Figura 61. Perfil da velocidade horizontal (m/s) para  $t = 60.000s$  (detalhe do canal).

Considerando o aspecto tempo, a análise do custo computacional das rotinas internas do programa foi realizada a partir de ferramentas disponíveis no pacote da Portland instalado no *cluster* SISMOS. Para esta análise foi escolhida a malha pequena executada em 4 processadores utilizando 10 passos de tempo e a estrutura de dados por elemento. As Tabelas 25-28 mostram apenas as rotinas do programa cujos tempos percentuais alcançaram valores superiores a 1% em cada processador.

Tabela 25. Custo percentual do processador 0.

Rotina	Número de Chamadas	Tempo (%)	Função
<b>Inversa</b>	<b>7.878.560</b>	<b>22,31</b>	<b>Inversa da matriz 3x3</b>
<b>Det2</b>	<b>70.907.040</b>	<b>19,05</b>	<b>Determinante 2x2</b>
<b>Pax</b>	<b>668</b>	<b>15,99</b>	<b>Produto matriz-vetor</b>
Elmt02	742.561	5,84	Matriz do elemento
Pdata	10	5,47	Saída
Pdot	3.402	4,27	Produto escalar
Squareroot	742.560	3,99	Raíz quadrada matriz 3x3
Prediag	80	3,87	Precondicionador
Gmres	80	2,99	Método GMRES
Det3	7.878.560	2,31	Determinante 3x3
Pform	80	1,98	Matriz de rigidez
Formke	742.560	1,95	Matriz de elementos
Tau	742.560	1,88	Matriz Tau
Lku	742.560	1,41	Produto matriz-vetor local

Tabela 26. Custo percentual do processador 1.

<b>Rotina</b>	<b>Número de Chamadas</b>	<b>Tempo (%)</b>	<b>Função</b>
<b>Inversa</b>	<b>7.273.520</b>	<b>20,79</b>	<b>Inversa da matriz 3x3</b>
<b>Det2</b>	<b>65.461.680</b>	<b>17,00</b>	<b>Determinante 2x2</b>
<b>Pax</b>	<b>668</b>	<b>16,25</b>	<b>Produto matriz-vetor</b>
Pform	80	6,85	Matriz de rigidez
Elmt02	725.361	5,73	Matriz do elemento
Pdata	10	5,53	Saída
Pdot	3.402	4,66	Produto escalar
Squareroot	725.360	3,64	Raíz quadrada matriz 3x3
Prediag	80	3,63	Precondicionador
Gmres	80	2,76	Método GMRES
Det3	7.273.520	2,15	Determinante 3x3
Formke	725.360	1,90	Matriz de elementos
Tau	725.360	1,85	Matriz Tau
Lku	725.360	1,42	Produto matriz-vetor local

Tabela 27. Custo percentual do processador 2.

<b>Rotina</b>	<b>Número de Chamadas</b>	<b>Tempo (%)</b>	<b>Função</b>
<b>Inversa</b>	<b>7.336.240</b>	<b>21,08</b>	<b>Inversa da matriz 3x3</b>
<b>Det2</b>	<b>66.026.160</b>	<b>17,16</b>	<b>Determinante 2x2</b>
<b>Pax</b>	<b>668</b>	<b>16,31</b>	<b>Produto matriz-vetor</b>
Pform	80	6,64	Matriz de rigidez
Elmt02	714.241	5,66	Matriz do elemento
Pdata	10	5,54	Saída
Pdot	3.402	4,46	Produto escalar
Prediag	80	3,76	Precondicionador
Squareroot	714.240	3,68	Raíz quadrada matriz 3x3
Gmres	80	2,84	Método GMRES
Det3	7.336.240	2,18	Determinante 3x3
Formke	714.240	1,90	Matriz de elementos
Tau	714.240	1,83	Matriz Tau
Lku	714.240	1,39	Produto matriz-vetor local

Tabela 28. Custo percentual do processador 3.

Rotina	Número de Chamadas	Tempo (%)	Função
<b>Inversa</b>	<b>6.693.839</b>	<b>19,19</b>	<b>Inversa da matriz 3x3</b>
<b>Pax</b>	<b>668</b>	<b>16,45</b>	<b>Produto matriz-vetor</b>
<b>Det2</b>	<b>60.244.551</b>	<b>15,73</b>	<b>Determinante 2x2</b>
Pform	80	10,43	Matriz de rigidez
Elmt02	721.841	5,79	Matriz do elemento
Pdata	10	5,56	Saída
Pdot	3.402	4,44	Produto escalar
Prediag	80	3,82	Precondicionador
Squareroot	721.840	3,35	Raíz quadrada matriz 3x3
Gmres	80	2,74	Método GMRES
Det3	6.693.839	1,99	Determinante 3x3
Formke	721.840	1,91	Matriz de elementos
Tau	721.840	1,86	Matriz Tau
Lku	721.840	1,41	Produto matriz-vetor local

As Tabelas 25-28 mostram as principais rotinas do programa executado em cada processador e que respondem por aproximadamente 95% do tempo total consumido. Dentre estas rotinas, destacamos as três rotinas iniciais, que juntas consomem aproximadamente 55% de todo o processamento. Curiosamente, e diferentemente do esperado, a rotina mais onerosa e, portanto, a que deveria ser a mais importante do ponto de vista de otimização, não foi a que efetuava o cálculo do produto matriz-vetor (**PAX**). Em três dos quatro processadores, duas outras rotinas (**INVERSA** e **DET2**), consumiram mais tempo que o produto matriz-vetor. Estas rotinas são utilizadas fora do *solver*, na montagem das matrizes de elemento e aresta para a determinação da matriz  $\hat{\mathbf{O}}$  utilizada no termo de estabilização *SUPG*. Embora estas rotinas sejam bastante simples, a justificativa provável para o seu alto consumo, deve-se ao fato de ambas serem chamadas muitas vezes. O elevado número de chamadas destas rotinas é função do número de elementos em cada partição e do número de iterações não-lineares do sistema.

## 7. CONCLUSÕES

Neste trabalho, a implementação em paralelo do método dos elementos finitos para as equações de águas rasas foi plenamente atingida evidenciando um ganho significativo no desempenho em relação à implementação seqüencial. Todo o processo computacional associado à montagem e atualização das matrizes em ambas as estruturas, por elemento e por aresta, assim como seus resíduos foi totalmente paralelizado. Da mesma forma, o produto matriz-vetor utilizado como núcleo do solucionador do método iterativo GMRES também teve a sua implementação efetuada em paralelo.

O estudo do comportamento de elevação da maré para um caso real demonstra que as soluções encontradas para um sistema não-linear com muitas equações, podem ser obtidas eficientemente através de máquinas de baixo custo operando de forma paralela.

Entre os métodos de particionamento de domínio disponíveis no METIS, o método dual obteve um desempenho consideravelmente superior ao método nodal, demonstrando que o particionamento por elementos é mais eficiente do que por nós.

Também entre os tipos de estruturas de armazenamento dos dados das matrizes, a estrutura por aresta mostrou-se mais eficiente que a estrutura por elemento, evidenciando que a operação utilizando estruturas com matrizes menores são computacionalmente mais adequadas.

A análise da escalabilidade do sistema paralelo demonstrou que o comportamento não-linear do número de iterações do método iterativo GMRES em função do refinamento da malha aumentava à medida que o tamanho do passo de tempo também aumentava. Este comportamento influenciou diretamente no tempo de simulação e representa um dos parâmetros utilizados na correta determinação da escalabilidade do sistema.

Outras linhas de pesquisa que podem ajudar no aprimoramento deste trabalho são: acoplamento com modelos de transporte (químicos ou biológicos), otimização da implementação em paralelo (parâmetros de compilação, pré-tratamento da malha, maior velocidade da rede, etc.), investigação de outros algoritmos para decomposição de domínio (GREEDY, RCB, RGB, RSB, MRSB, JOSTLE), outras formas de armazenamento da estrutura de dados das matrizes (CSR, por exemplo) e utilização de outros preconditionadores (bloco diagonal, por exemplo).

## 8. REFERÊNCIAS BIBLIOGRÁFICAS

1. Zienkiewicz O. C., Taylor R. L., "The Finite Element Method", Fifth Edition, Butterworth-Heinemann, volume 3: Fluid Dynamics, 2000.
2. Galeão A. C., Carmo E. G. do, "A Consistent Approximate Upwind Petrov-Galerkin Method for Convection-Dominated Problems", *Computer Methods in Applied Mechanics and Engineering*, 32, pp. 199-259, 1988. Harten A., "On the Symmetric Form of Systems of Conservation Laws with Entropy", *Journal of Computational Physics*, 49, pp. 151-164, 1983.
3. Ribeiro F. L. B., Galeão A. C., Landau L., "A Space-Time Finite Element Formulation for Shallow Water Equations", Development and Application of Computer Techniques to Environmental Studies VI, pp. 403-414, Computational Mechanics Publications, 1996.
4. Ribeiro F. L. B., Castro R. G. S., Galeão A. C., Loula A. F. D., Landau L., "A Space-Time Finite Element Formulation for Shallow Water Equations with Shock-Capturing Operator", IV World Congress, Argentina, 1998.
5. Saleri F., "Some Stabilization Techniques in Computational Fluid Dynamics", *Proceedings of the 9<sup>th</sup> International Conference on Finite Elements in Fluids*, Venezia, 1995.
6. Brooks A. N., Hughes T. J., "Streamline Upwind Petrov-Galerkin Formulation for Convection-Dominated Flows with Particular Emphasis on the Incompressible Navier-Stokes Equations", *Computer Methods in Applied Mechanics and Engineering*, 32, pp. 199-259, 1982.
7. Saad Y., Schultz M. H., "GMRES: Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems", *SIAM Journal of Scientific and Statistical Computing*, 7, pp. 856-869, 1986.
8. Ribeiro F. L. B., Galeão A. C., Landau L., "Edge-based Finite Element Method for Shallow Water Equations", *International Journal for Numerical Methods in Fluids*, 36, pp. 659-685, 2001.

9. Peraire J., Morgan K., Peiro J., "Unstructured Grid Methods for Compressible Flows", AGARD Special Course on Unstructured Grid Methods for Advection Dominated Flows, 787, pp. 5.1-5.39, 1992.
10. Karypis G., Kumar V., "A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices", *Technical Report, University of Minnesota, Department of Computer Science*, 1998.
11. Gropp W. D., Lusk E., Skjellum A., "Using MPI – Portable Parallel Programming with the Message-Passing Interface", Second Edition, The MIT Press, Cambridge, Massachusetts, 1999.
12. Ribeiro F. L. B., Galeão A. C., Landau L., "Finite Element Models for Shallow Water Flow", *Relatório Técnico Interno, COPPE-CIVIL/UFRJ*.
13. Schlichting H. "Boundary-Layer Theory", McGraw Hill Book Company, 1968.
14. Harten A., "On the Symmetric Form of Systems of Conservation Laws with Entropy", *Journal of Computational Physics*, 49, pp. 151-164, 1983.
15. Hughes T. J. R., Franca L. P., Mallet M., "A New Finite Element Formulation for Computational Fluid Dynamics: I. Symmetric Forms of the Compressible Euler and Navier-Stokes Equations and the Second Law of Thermodynamics", *Computer Methods in Applied Mechanics and Engineering*, 54, pp. 223-234, 1986.
16. Hauke G., Hughes T. J. R., "A Comparative Study of Different Sets of Variables for Solving Compressible and Incompressible Flows", *Computer Methods in Applied Mechanics and Engineering*, 153, pp. 1-44, 1998.
17. Bova S. W., Carey G. F., "An Entropy Variable Formulation and Applications for the Two-dimensional Shallow Water Equations", *International Journal for Numerical Methods in Fluids*, 23, pp. 29-46, 1996.

18. Hauke G., "A Symmetric Formulation for Computing Transient Shallow Water Flows", *Computer Methods in Applied Mechanics and Engineering*, 163, pp. 111-122, 1998.
19. Hughes T. J., Mallet M., "A New Finite Element Formulation for Computational Fluid Dynamics: III. The Generalized Streamline Operator for Multidimensional Advective-Diffusive Systems", *Computer Methods in Applied Mechanics and Engineering*, 58, pp. 305-328, 1986.
20. Shakib F., "Finite Element Analysis of the Compressible Euler and Navier-Stokes Equations", *Ph.D. Thesis*, Stanford University, 1988.
21. Almeida R. C., Galeão A. C., "An Adaptive Petrov-Galerkin Formulation for the Compressible Euler and Navier-Stokes Equations", *Computer Methods in Applied Mechanics and Engineering*, 129, pp. 157-176, 1996.
22. Dembo R. S., Eisenstat S. C., Steihaug T., "Inexact Newton Methods", *SIAM Journal Numerical Analysis*, 19, pp. 400-408, 1982.
23. Eisenstat S. C., Walker H. F., "Globally Convergent Inexact Newton Methods", *SIAM J. Optimization*, 4, pp. 393-422, 1994.
24. Papadrakakis M., "Solving Large-Scale Problems in Mechanics: The Development and Application of Computational Solution Procedures", John Wiley & Sons, Chichester, 1993.
25. Saad Y., "Iterative Methods for Sparse Linear Systems", PWS Publishing, Boston, 1996.
26. Gijzen M. B. van, "Large Scale Finite Element Computations with GMRES-Like Methods on a Cray-Y-MP", *Applied Numerical Mathematics*, 19, pp. 51-62, 1995.
27. Ribeiro F. L. B., Coutinho A. L. G. A., "Comparison Between Element, Edge and CSR Storage Schemes for Iterative Solutions in Finite Element Analyses", *Computers and Structures*, 2003 (submetido para publicação).

28. Löhner R., "Some Useful Renumbering Strategies for Unstructured Grids", *International Journal for Numerical Methods in Engineering*, 36, pp. 3259-3270, 1993.
29. Silva R. S., "Estratégias de Resolução em Sistemas Distribuídos de Problemas em Dinâmica dos Fluidos Computacional via Elementos Finitos Adaptativos", *Tese de Doutorado em Engenharia Civil*, COPPE/UFRJ, 1998.
30. Geist G. A., Heath M. T., Peyton B. W., Worley P. H., "PICL: A Portable Instrumented Communications Library – C Reference Manual", *Technical Report TM-11130*, Oak Ridge National Laboratory, 1990.
31. Geist G. A., Beguelin A., Dongarra J., Jiang W., Manchek R., Sunderam V., "PVM: Parallel Virtual Machine - A Users' Guide and Tutorial for Networked Parallel Computing", The MIT Press, Cambridge, Massachusetts, 1994.
32. Beguelin A., Dongarra J., Geist G. A., Manchek R., Sunderam V., "A Users' Guide to PVM: Parallel Virtual Machine", *Technical Report TM-11826*, Oak Ridge National Laboratory, 1991.
33. Bomans L., Roose D., Hempel R., "The Argonne/GMD Macros in FORTRAN for Portable Parallel Programming and their Implementation on the Intel iPSC/2", *Parallel Computing*, 15, pp. 119-132, 1990.
34. Boyle J., Butler R., Disz T., Glickfeld B., Lusk E., Overbeek R., Patterson J., Stevens R., "Portable Programs for Parallel Processors", *Holt, Rinehart and Winston*, 1987.
35. Butler R., Lusk E., "Users' Guide to the p4 Parallel Programming System", *Technical Report ANL-92/17*, Argonne National Laboratory, 1992.
36. Butler R., Lusk E., "Monitors, Messages and Clusters: The p4 parallel programming system", *Parallel Computing*, 20, pp. 547-564, 1994.
37. Gropp W. D., Smith B., "Chameleon Parallel Programming Tools Users Manual", *Technical Report ANL-93/23*, Argonne National Laboratory, 1993.

38. Skjellum A., Smith S. G., Still C. H., Leung A. P., Morari M., "The Zipcode Message-Passing System", In Geoffrey C. Fox, Editor, *Parallel Computing Works*, Morgan Kaufman, 1994.
39. Harrison R. J., "Portable Tools and Applications for Parallel Computers", *International Journal of Quantum Chemicals*, 40, pp. 847,1991.
40. Parasoftware Corporation, "Express version 1.0: A Communication Environment for Parallel Computers", 1988.
41. Geist G. A., Kohl J. A., Papadopoulos P. M., "PVM and MPI: A Comparison of Features", *Calculateurs Paralleles*, 8(2), pp. 137-150, 1996.
42. Pacheco P. S., "Programming Parallel with MPI", San Francisco, California, Morgan Kaufman, 1997.
43. Gropp W. D., Lusk E., "Users' Guide for MPICH, a Portable Implementation of MPI – version 1.2.0", Mathematics and Computer Science Division, University of Chicago, 1999.
44. Carey G. F., "Computational Grids: Generation, Adaptation and Solution Strategies", Taylor and Francis, Washington, DC, USA, 1997.
45. Lohner R., "Edges, Stars, Superedges and Chains", *Computer Methods in Applied Mechanics and Engineering*, 111, pp. 255-263, 1994.
46. Courant R., Friedrichs K. O., Levy H., "On the Partial Difference Equations of Mathematical Physics", *IBM Journal*, 11, pp. 215-235, 1967.
47. Prodonoff V., "Vibrações Mecânicas – Simulação e Análise", Maity Comunicação e Editora Ltda, Rio de Janeiro, 1990.
48. Ortega J., Voigt R., "Solution of Partial Differential Equations on Vector and Parallel Computers", SIAM, Philadelphia, PA, 1985.
49. Amdahl G., "The Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities", AFIPS Conf. Proc. 30, pp. 483-485, 1967.