

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E DE ESTATÍSTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**ALGORITMO PARA SIMULAÇÃO NUMÉRICA DAS EQUAÇÕES DO
MOVIMENTO PELO MÉTODO DOS VOLUMES FINITOS USANDO
DIAGRAMAS DE VORONOI**

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA
CATARINA PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA
COMPUTAÇÃO

por

FABIAN CORRÊA CARDOSO

Prof. Sérgio Peters

Orientador

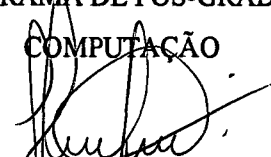
Florianópolis, fevereiro de 1997.

**ALGORITMO PARA SIMULAÇÃO NUMÉRICA DAS EQUAÇÕES DO
MOVIMENTO PELO MÉTODO DOS VOLUMES FINITOS USANDO
DIAGRAMAS DE VORONOI**

FABIAN CORRÊA CARDOSO

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA A OBTENÇÃO DO TÍTULO DE
MESTRE EM CIÊNCIA DA COMPUTAÇÃO

ÁREA DE CONCENTRAÇÃO MATEMÁTICA COMPUTACIONAL, E APROVADA EM SUA
FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA
COMPUTAÇÃO

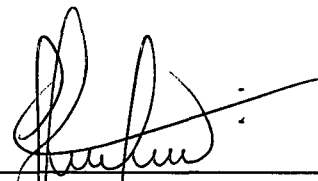


Prof. SÉRGIO PETERS, Dr. Eng. - ORIENTADOR



Prof. MURILO SILVA DE CAMARGO, Dr. - COORDENADOR

BANCA EXAMINADORA



Prof. SÉRGIO PETERS, Dr. Eng. Mec - PRESIDENTE



Prof. ÁLVARO TOUBES PRATA, Ph.D.



Prof. CLÓVIS RAIMUNDO-MALISKA, Ph. D.



Prof. JULIO FELIPE SZEREMETA, M.Sc.

A Deus e a Nossa Senhora, por mais esta fase concluída.

A meus queridos pais, Idelfonso e Marlecy, pelo apoio, incentivo,
e minha formação, que foram o ponto de partida

**A minha adorada esposa, Valéria, pelo amor, apoio e dedicação
que me entregou, sem reservas, em todos os momentos.**

AGRADECIMENTOS

Agradeço a meu orientador, Sérgio Peters, pela orientação e por todo auxílio prestado até a conclusão deste trabalho, na discretização das equações, na correção do trabalho, no desenvolvimento dos programas, bem como, por saber dividir seu conhecimento.

Agradeço a minha colega, Viviana Cocco Mariani, pelo auxílio que prestou na discretização das equações, na correção do trabalho individual, no desenvolvimento dos programas, e por ter compartilhado seu conhecimento matemático.

Agradeço a Clóvis Raimundo Maliska Jr. pelo seu desprendimento em ter fornecido os fontes de seu programa gerador de Voronoi, do visualizador para malha de Voronoi, e, também, pelo tempo dispendido em explicar e adaptar estes programas.

Agradeço ao Prof. Raul Sidnei Waslawick pelas contribuições e sugestões apresentadas no Trabalho Individual.

Agradeço aos membros da Banca examinadora Profs. Alvaro T. Prata, Clovis R. Maliska e Julio F. Szeremeta, pelo tempo dispendido na leitura deste trabalho e pelas valiosas sugestões apresentadas.

Agradeço ao Prof. Murilo, Verinha e Valdete pela administração diária do cotidiano do curso de pós-graduação em Ciência da Computação.

Agradeço ao Conselho Nacional de Pesquisa (Cnpq), pelo apoio financeiro.

"O amor nada dá senão de si próprio e nada recebe senão de si próprio."

Sumário

AGRADECIMENTOS	vi
Sumário	vii
Lista de Figuras	ix
Lista de Tabelas	xi
Simbologia	xii
Resumo	xiv
Abstract	xvi
1. Introdução	1
2. Diagrama de Voronoi	4
2.1. Introdução	4
2.2. Características dos Diagramas de Voronoi	6
3. Discretização das Equações de Navier-Stokes	9
3.1. Introdução	9
3.2. Discretização das Equações de Navier-Stokes	10
3.3. Avaliação do Gradiente de Pressão	23
3.4. Acoplamento Pressão-Velocidade	38
3.4.1. Projeção das componentes nodais de velocidades na direção normal \hat{n}_{ij}	38
3.4.2. Geração de uma equação evolutiva para as componentes de velocidades normais as faces W_{ij} a partir das equações de w_i e w_j vizinhas da face ij	39
3.4.3. Geração de um campo de pressão que satisfaça a conservação da massa	43
3.5. Acoplamento SIMPLEC Pressão-Velocidade	46
3.6. Algoritmo Iterativo	53
4. Resultados e Discussões	56
4.1. Introdução	56
4.2. Formulação do Problema	57
4.3. Malhas Utilizadas	59
4.4. Resultados Obtidos	62

5. Conclusões	77
5.1. Contribuições	77
5.2. Perspectivas Futuras	78
Referências	80
Apêndice A - Avaliação de d_{ij}^w	83
Apêndice B - Programa Computacional Implementado	86

Lista de Figuras

Figura 2.1 - Características dos Diagramas de Voronoi. _____	8
Figura 3.1 - Caso particular de Malha Estruturada com Arranjo Colocalizado. _____	10
Figura 3.2 - Exemplo de Malha não Estruturada de Pontos Nodais Aleatórios. _____	11
Figura 3.3 - Exemplo de Malha não Estruturada Hexagonal. _____	11
Figura 3.4 - Volume de controle genérico i. _____	14
Figura 3.5 - Volume de controle genérico dividido em triângulos _____	15
Figura 3.6 - Malha uniforme com distribuição linear de pressão _____	24
Figura 3.7 - Algoritmo do programa. _____	53
Figura 4.1 - Parede superior deslizante sobre cavidade quadrada. _____	56
Figura 4.2 - Malha estruturada retangular uniforme gerada por DV (100 pontos nodais). _____	60
Figura 4.3a - Malha de volumes hexagonais (95 pontos nodais). _____	60
Figura 4.3b - Malha de volumes hexagonais (95 pontos nodais). _____	61
Figura 4.4 - Malha de pontos aleatórios (100 pontos nodais). _____	62
Figura 4.5 - Componente horizontal de velocidade em $x=5$, $Re=100$ (malha retangular de 100 pontos nodais). _____	63
Figura 4.6 - Componente vertical de velocidade em $y=5$, $Re=100$ (malha retangular com 100 pontos nodais). _____	63
Figura 4.7 - Componente horizontal de velocidade em $x=5$, $Re=100$ (malha de volumes hexagonais com 95 pontos nodais, Fig. (4.3a)). _____	65
Figura 4.8 - Componente vertical de velocidade em $y=5$, $Re=100$ (malha de volumes hexagonais com 95 pontos nodais, Fig. (4.3b)). _____	66
Figura 4.9 - Componente horizontal de velocidade em $x=5$, $Re=100$ (malha aleatória com 100 pontos nodais). _____	67
Figura 4.10 - Componente vertical de velocidade em $y=5$, $Re=100$ (malha aleatória com 100 pontos nodais). _____	67

Figura 4.11 - Componente horizontal de velocidade em $x=5$, $Re=100$ (malha com volumes hexagonais de 613 pontos nodais)..	70
Figura 4.12 - Componente vertical de velocidade em $y=5$, $Re=100$ (malha com volumes hexagonais de 613 pontos nodais).	71
Figura 4.13 - Componente horizontal de velocidade em $x=5$, $Re=400$ (malha com volumes hexagonais de 613 pontos nodais).	71
Figura 4.14 - Componente vertical de velocidade em $y=5$, $Re=400$ (malha com volumes hexagonais de 613 pontos nodais).	72
Figura 4.15 - Componente vertical de velocidade em $y=5$, $Re=400$ (malha com volumes hexagonais de 613 pontos nodais).	73
Figura 4.16 - Componente horizontal de velocidade em $x=5$, $Re=1000$ (malha com volumes hexagonais de 613 pontos nodais).	74
Figura 4.17 - Componente vertical de velocidade em $y=5$, $Re=1000$ (malha com volumes hexagonais de 613 pontos nodais).	74

Lista de Tabelas

Tabela 3.1 - Expressões para ϕ , S^ϕ , Γ^ϕ , conforme a equação representativa.	13
Tabela 3.2 - Fator peso $F(Pe)$ para vários esquemas de interpolação.	21
Tabela 4.1 - Condições de contorno.	58
Tabela 4.2 - Avaliação do tempo de CPU (Estação SPARC 10 da SUN).	64
Tabela 4.3 - Comparação entre os gradientes de pressão.	68
Tabela 4.4 - Comparação entre os gradientes de pressão (malha hexagonal com 95 pontos nodais).	69
Tabela 4.5- Comparação entre o tipo de malha e o cálculo do gradiente através dos resultados obtidos neste trabalho.	75

Simbologia

c_p	Calor específico.
CPU	Unidade Central de Processamento.
DV	Diagrama(s) de Voronoi.
$d\bar{n}$ ou $d\bar{s}_a$	Área de cada face do volume de controle gerado pelo DV.
EDPs	Equações Diferenciais Parciais.
\hat{e}_{ij}	Vetor normal unitário à interface ij.
$e_{x_{ij}}$	Componente do vetor normal unitário à interface ij, na direção x.
$e_{y_{ij}}$	Componente do vetor normal unitário à interface ij, na direção y.
\vec{J}	Fluxo total de ϕ , forma vetorial.
k	Conductividade térmica.
L_{ij}	Distância entre o ponto nodal central i e o ponto nodal vizinho j.
MVF	Método dos Volumes Finitos.
N_V	Número de Vizinhos.
\hat{n}_{ij}	Vetor normal à interface ij.
P	Campo de pressão do escoamento.
\bar{q}	Fluxo de calor bi-dimensional.
S_{ij}	Dimensão da face.
S^ϕ	Termo fonte genérico.
T	Temperatura.
u	Componente de velocidade na direção x.
v	Componente de velocidade na direção y.
\vec{V}	Vetor velocidade no sistema de coordenadas escolhido.
VC	Volume de Controle.
x	Abscissa.
y	Ordenada.

Símbolos Gregos

α_{u_v}	Subrelaxação para as velocidades u e v, neste trabalho $\alpha_{u_v}=0,3$
$\alpha_{p'} = 0,5$	Subrelaxação para a correção da pressão, neste trabalho $\alpha_{p'} = 0,5$.
α_p	Subrelaxação para a pressão, neste trabalho $\alpha_p=0,5$
ϕ	Variável genérica.
Γ	Coefficiente do termo difusivo, equivalente à viscosidade.
μ	Viscosidade dinâmica do fluido.
ρ	Massa específica do fluido.
\mathcal{V}	Volume para integração sobre o VC.

Resumo

O uso de malhas não-estruturadas está, normalmente, associado ao Método dos Elementos Finitos (MEF), e utilização de tais malhas aliadas ao Método dos Volumes Finitos (MVF) é recente, tendo sido explorada em Mecânica dos Fluidos com grandes vantagens sobre as malhas estruturadas. Uma das características importantes do MVF é a validade das leis de conservação das equações diferenciais ao nível dos volumes de controle finitos, ou seja, todos os princípios de conservação podem ser analisados em uma malha não refinada.

No presente trabalho, são utilizadas malhas não-estruturadas geradas por Diagramas de Voronoi (DV). Os DV ganham a cada dia mais espaço em simulações numéricas de fenômenos físicos dadas as suas características de construção. Estas malhas possibilitam que fluxos sejam facilmente avaliados nas fronteiras dos volumes de controle, modelando, exatamente, os limites geométricos do domínio a ser discretizado, permitindo o refinamento de regiões específicas em geometrias complexas. Nestas malhas cada volume de controle preserva as propriedades de tais diagramas.

A geração das malhas não-estruturadas, realizada por DV, dá origem a volumes cujo segmento de ligação, entre o ponto nodal gerador do DV e o ponto vizinho, é ortogonal à face comum com o volume vizinho. Por possuírem esta propriedade os DV se apresentam como excelente alternativa para discretização não-estruturada em problemas de escoamento de fluidos, pois preservam a ortogonalidade local da malha na região entre pontos nodais vizinhos. Por isso, o MVF aplicado à malhas não-estruturadas geradas por DV alia simplicidade com versatilidade.

O objetivo principal do presente trabalho é discretizar as equações de Navier-Stokes e resolvê-las através do uso de malhas não-estruturadas geradas por DV em problemas envolvendo escoamentos de fluidos, bem como testar alguns dos gradientes de pressão presentemente disponíveis na literatura quanto a seu

desempenho, como também sugerir uma nova forma de avaliar tal gradiente de pressão.

Abstract

The use of unstructured grids is, generally, associated to the Finite Element Method (FEM), and the utilization of such grids in conjunction to the Finite Volume Method (FVM) is new, and has been explored in Fluids Mechanics with great advantages over the structured grids. One of the most important characteristics of the FVM is the Differential Equations conservation laws validity in terms of finites control volumes, i.e., all conservation principles can be analised in a non refined grid.

In this work, is used unstructured grid generated by Voronoi Diagrams (VD). The VD has received more attention in numerical simulations of physical phenomena because of its construction characteristics. These grids permit that flows be easily approximated in the boundary of the control volumes, modelling, exactaly, the geometrical limits of the domain to be discretized, allowing the refinement of specific regions in complex geometries. In these grids each control volume preserves the properties of such diagrams.

The unstructured grids generation, performed by VD, gives rise to volumes in which the connection segment, between the generator nodal point of the VD and the neighbor point, is orthogonal to the common face with the neighboring volume. For having this properties the VD presents itself as an excellent alternative for the non structured discretization in problems of fluid flow, because they preserve the grid local orthogonality in the region between the neighboring nodal points. Because of this, the FVM applied to the unstructured grids generated by VD combines simplicity with versatility.

The main objective of the present work is to discretize and solve the Navier-Stokes equations through the use of unstructured grids generated by Voronoi Diagrams, and to test some pressure gradients schemes, nowadays in use in literature, and has been suggested a new form to avaliate this gradient.

1. Introdução

Nos últimos anos, com o avanço da tecnologia, os métodos numéricos também tiveram um grande avanço. Entre eles têm-se os métodos de resolução das equações de conservação da massa e da quantidade de movimento (Equações de Navier-Stokes), que regem problemas de mecânica dos fluidos. Dentre estes destaca-se o Método dos Volumes Finitos (MVF) (Patankar, 1980 e Maliska, 1995), que é aplicado para a resolução de diversos processos físicos e químicos, envolvendo, transferência de calor e massa, reações químicas entre outros.

O grande mérito do MVF foi o tratamento simplificado conseguido na resolução das equações de Navier-Stokes, que permitiu uma democratização no uso destas equações, tanto por Engenheiros, como por Matemáticos, passando pelos Cientistas da Computação.

Presentemente, é proposta a implementação de uma variação do MVF, fazendo uso de malha não estruturada (não vinculada a um sistema de coordenadas cartesianas), baseada em Diagramas de Voronoi (DV), para discretizar as equações de Navier-Stokes e de conservação da massa. Tal discretização apresenta dois aspectos particularmente interessantes, o primeiro é devido ao uso da malha do tipo não estruturada, o que permite a captura de contornos geométricos arbitrários, além de permitir o refinamento da malha em regiões específicas do domínio. O segundo aspecto está associado à ortogonalidade local dos VC gerados pelos DV, o que facilita o processo de discretização numérica.

Para se resolver problemas de escoamentos de fluidos, usam-se as equações de conservação da quantidade de movimento, chamadas de equações de Navier-Stokes, acopladas à equação de conservação da massa. Este sistema de equações diferenciais parciais (EDPs) permite a modelação de uma imensa variedade de problemas

físicos. Possuindo, tais equações, a seguinte forma conservativa dimensional em coordenadas cartesianas para fluido newtoniano incompressível em regime permanente, (Bejan, 1984):

$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho uu)}{\partial x} + \frac{\partial(\rho vu)}{\partial y} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x} \left(\frac{\mu \partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(\frac{\mu \partial u}{\partial y} \right) \quad (1.1)$$

$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial(\rho uv)}{\partial x} + \frac{\partial(\rho vv)}{\partial y} = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial x} \left(\frac{\mu \partial v}{\partial x} \right) + \frac{\partial}{\partial y} \left(\frac{\mu \partial v}{\partial y} \right) \quad (1.2)$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (1.3)$$

onde, u e v são as velocidades locais do fluido nas direções x e y , respectivamente, p é a pressão local do fluido, μ é a viscosidade dinâmica e ρ é a massa específica do fluido, ambas propriedades assumidas como constantes, e t é o tempo.

As Eqs. (1.1) a (1.3), que regem problemas de mecânica dos fluidos, são baseadas em balanços da mecânica clássica, que envolvem impulso e variação de quantidade de movimento do fluido. Esta análise dinâmica é feita em nível de volumes elementares considerando variações de quantidade de movimento devido à advecção e devido às tensões internas ao fluido. Chama-se de advecção, o transporte devido ao movimento do próprio fluido de uma região para outra do domínio físico, carregando desta forma uma grandeza ϕ qualquer, que no caso de Eq. (1.1) é a própria componente u de velocidade, por exemplo. Em transferência de calor, denomina-se de convecção ao transporte advectivo mais difusivo de calor. Por simplicidade, em geral, a palavra convecção é usada como sinônimo de advecção. As tensões envolvidas neste balanço referem-se, em parte, ao atrito devido ao cisalhamento do fluido em movimento relativo a ele mesmo, o que caracteriza o transporte por difusão. Por outro lado, existem tensões normais envolvidas, o que dá origem a variações de pressão no interior do fluido. Deste

balanço de variação de quantidade de movimento em volumes elementares surge, então, a equação diferencial vetorial de Navier-Stokes, que associada à equação da conservação da massa compõem um sistema de EDPs, que regem escoamentos de fluidos.

Neste trabalho, resolvem-se, a equação da massa e da quantidade de movimento correspondendo a um sistema de EDPs que serão discretizadas pelo MVF em malhas não estruturadas geradas por DV. O problema físico resolvido é o da cavidade quadrada sujeita a uma parede deslizando (Ghia et alli, 1982).

No próximo capítulo é apresentada uma visão geral dos Diagramas de Voronoi.

2. Diagramas de Voronoi

2.1 Introdução

O Diagrama de Voronoi é um dos elementos mais interessantes em geometria computacional (Preparata e Shamos, 1985), por resolver com simplicidade problemas de proximidade. E tem sido uma poderosa ferramenta na resolução de uma variedade de problemas, tais como, otimização, mapeamentos geográficos, computação gráfica, robótica, reconhecimento de padrões, projetos auxiliados por computador, resolvendo-os de forma eficiente e elegante.

O Diagrama de Voronoi, primeiramente, definido pelos matemáticos, logo após foi redescoberto pelos físicos (Winterfeld *et alli*, 1981) e meteorologistas. Por consequência é conhecido por diversos nomes, tais como, Dirichlet *tesselations*, polihedro de Voronoi, polígonos de Thiessen em duas dimensões, células ou domínios de Wigner-Seitz (Avis e Bhattacharya, 1983).

Com o uso deste diagrama podem ser resolvidos o problema do vizinho mais próximo, da construção do casco convexo, do menor caminho através da árvore de busca, e do maior círculo vazio. Geógrafos e meteorologistas usam o polígono de Thiessen. Cristalógrafos reconhecem inúmeros modelos para crescimento de cristais (Mackay, 1972). Em três dimensões o DV é de interesse dos físicos moleculares, dos bioquímicos, dos cientistas da matéria, e dos físico-químicos, possuindo aplicação também na caracterização de rochas (Pathak *et alli*, 1980). A construção do DV é, também, de interesse dos astrofísicos em conexão com a fragmentação de corpos celestes (Avis e Bhattacharya, 1983).

Também com o DV pode-se simplificar o plano de um caminho livre de colisões para um robô entre obstáculos, chamado de problema do movimento generalizado (Canny e Donald, 1988).

Recentemente os DV tem sido aplicados a problemas de mecânica dos fluidos por Taniguchi *et alli* (1991a e 1991b), o qual resolveu as equações do movimento através de MVF, com apenas uma forma de avaliação do ∇p (mínimo resíduo quadrático), e, também, a problemas de reservatório de petróleo por Marcondes *et alli* (1994), o qual resolveu equações de transporte de massa por difusão considerando um modo bifásico: óleo-água usando o MVF em malhas híbridas geradas por DV, não estruturada e com forma cilíndrica.

Presentemente, as malhas geradas por DV tem sido utilizadas na área de mecânica dos fluidos, onde existem dificuldades na simulação numérica de escoamentos geometricamente complicados, uma delas é a adaptatividade da malha aos seus contornos e outra é o tempo computacional requerido para se obter uma solução confiável. Ao gerarem-se malhas estruturadas, dificilmente é possível captar com precisão o contorno do domínio de cálculo para geometrias complicadas, uma vez que a malha é estruturada sobre um sistema de coordenadas fixo. Por exemplo, usando VC na forma retangular, associados ao sistema de coordenadas cartesiano, como é possível encaixá-los em um contorno curvilíneo?

O MVF aplicado a malhas estruturadas é simples em discretizações com malhas ortogonais, que facilitam os balanços dos fluxos nas faces dos volumes de controle.

Por outro lado, temos as discretizações com malhas estruturadas em coordenadas generalizadas, que permitem boas aproximações de contornos arbitrários, mas a um alto custo na discretização dos termos convectivos e difusivos devido as não ortogonalidades presentes nas interfaces dos VC.

Embora métodos baseados em malhas estruturadas apresentem facilidades na solução dos sistemas de equações lineares, que são do tipo banda, o MVF aqui implementado, baseado em malha não estruturada, gera sempre um sistema de equações

lineares com estrutura matricial esparsa não do tipo banda. De qualquer forma adota-se uma solução iterativa, independentemente de a malha ser estruturada ou não, pois o problema principal é resolver um sistema de EDPs não-lineares, gerando-se um sistema de equações lineares através de um processo de linearização.

Assim, objetivando-se unir a simplicidade do MVF associado a malhas ortogonais e, ao mesmo tempo, eliminar qualquer tipo de restrição geométrica, tanto de captura de contornos arbitrários, quanto de refinamentos locais de malha, utilizam-se métodos baseados em malhas não-estruturadas. Neste caso, o ideal é que as faces dos volumes de controle gerados sejam sempre ortogonais ao segmento que une dois pontos nodais vizinhos quaisquer. O DV permite que isto aconteça, devido a suas características construtivas.

Presentemente, propõe-se a discretização das equações de Navier-Stokes, para escoamento de fluidos, em regime laminar, pelo MVF usando-se as malhas não-estruturadas geradas por DV em domínio de cálculo bi-dimensional.

2.2. Características do Diagrama de Voronoi

No esquema de discretização numérica a ser explorado, cada volume de controle é gerado em torno de um ponto nodal previamente definido, e deve obedecer a algumas propriedades fundamentais dos DV, que facilitam a realização dos balanços de conservação sobre cada volume de controle.

A Fig. (3.1) apresenta um DV, onde se pode notar as seguintes propriedades (Preparata e Shamos, 1985):

- √ A face \overline{AB} é normal a qualquer segmento \overline{ij} . Esta propriedade é muito importante para avaliar as derivadas normais às interfaces, que podem envolver, apenas o ponto central i e um vizinho j .
- √ A face \overline{AB} corta o segmento \overline{ij} , exatamente, ao meio. Assim, as interpolações para obter as propriedades nas interfaces (\overline{AB} , por exemplo) podem ser

realizadas como se a malha fosse uniforme, pois o segmento \bar{ij} é seccionado ao meio.

- √ Qualquer ponto, dentro do DV centrado em i , está mais perto de i do que de qualquer outro ponto nodal j vizinho.
- √ Cada vértice do DV coincide com o centro da circunferência que passa pelos três (ou mais) pontos nodais que circundam tal vértice.

No DV cada volume de controle é constituído pelas regiões do domínio que estiverem mais próximas do ponto nodal central (Preparata e Shamos, 1985), ou seja,

$$V(x^i) = \bigcap_{j \neq i}^{NV} \{x / d(x^i, x) \leq d(x^j, x)\} \quad (2.1)$$

onde, $V(x^i)$ é a região definida pelo volume de controle associado ao ponto nodal x^i , x é um ponto interno qualquer ao volume de controle x^i , e $d(x^i, x)$ é a distância entre x^i e x , x^j é um ponto nodal qualquer diferente de x^i . Logo, um ponto x pertence ao VC, associado ao ponto mãe x^i , quando ele estiver mais próximo de x^i do que de outro ponto nodal qualquer x^j .

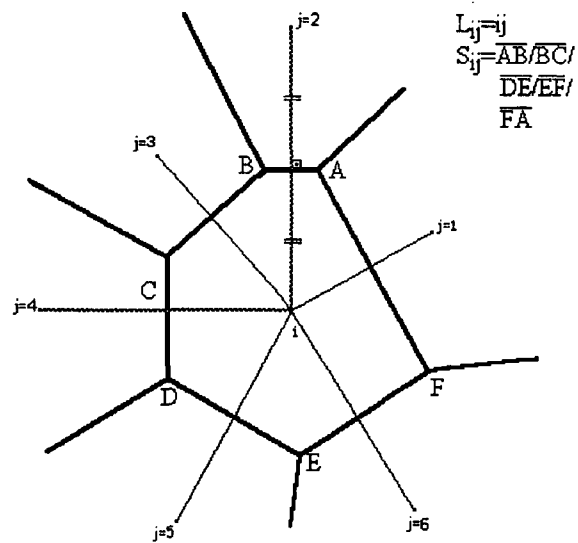


Figura 2.1 - Características dos Diagramas de Voronoi.

Presentemente, foi usado um gerador de DV incremental, ou seja, que utiliza a técnica de incrementação de volumes, isto quer dizer que tendo-se um diagrama inicial como sendo todo o espaço onde deseja-se fazer a geração vai-se incluindo todos os outros geradores da lista neste diagrama (Maliska Jr., 1993).

No próximo capítulo é apresentada a discretização das equações de Navier-Stokes.

3. Discretização das Equações de Navier-Stokes

3.1. Introdução

O MVF surgiu nos estudos numéricos de transferência de calor e mecânica dos fluidos, procurando associar a física envolvida e a metodologia numérica existente (Patankar, 1980). Permitiu simplificação das modelações matemáticas por EDPs, no que se refere a implementação de condições de contorno.

O método consiste na integração numérica das equações diferenciais em VC finitos promovendo o balanço da grandeza física envolvida neste VC.

Este método é mais eficiente quando se faz a discretização numérica das equações a partir da sua forma conservativa., o que permite estender a conservação de volumes elementares a níveis de volumes finitos. Na verdade, as equações diferenciais são resultado de balanços efetuados em volumes elementares. E, no MVF, tem-se uma macro conservação em níveis de volumes finitos das grandezas físicas envolvidas. Assim, com o MVF, tem-se um passo intermediário entre o balanço global e o balanço infinitesimal, o que é um passo preliminar para se atingir a conservação promovida pelas equações diferenciais efetuada em níveis de volumes elementares. Métodos baseados em elementos finitos ou diferenças finitas se preocupam em representar as derivadas envolvidas nas equações diferenciais de forma discreta, sem preocupação com o balanço conservativo proposto pela equação diferencial.

3.2. Discretização das Equações de Navier-Stokes

A seguir é apresentada a discretização das Equações de Navier-Stokes, e de conservação da massa, Eqs. (1.1) a (1.3), pelo Método dos Volumes Finitos.

Dentro do contexto do MVF, propõe-se a integração de cada equação diferencial em todos os VC do domínio discretizado, integrando-se as equações para u e v no volume de controle principal, usando arranjo colocalizado, onde todas as variáveis são armazenadas no ponto nodal central do volume.

As Fig. (3.1) a (3.3) apresentam alguns tipos de VC gerados com DV e o respectivo arranjo colocalizado:

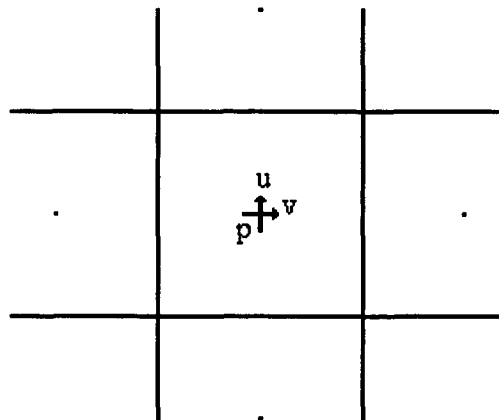


Figura 3.1 - Caso particular de Malha Estruturada com Arranjo Colocalizado.

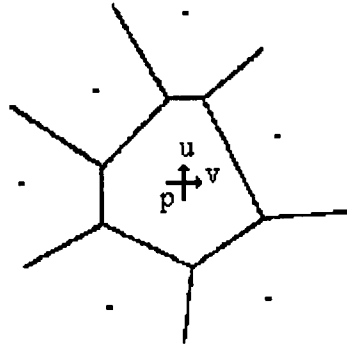


Figura 3.2 - Exemplo de Malha não Estruturada de Pontos Nodais Aleatórios.

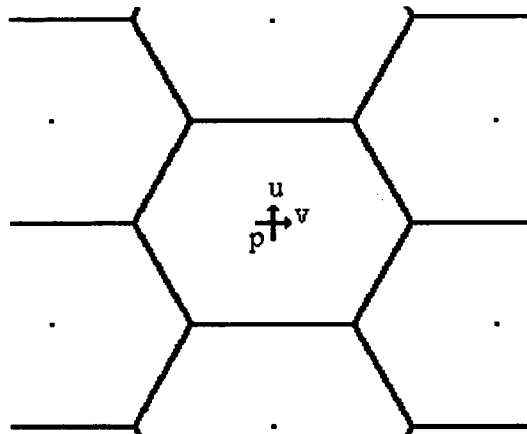


Figura 3.3 - Exemplo de Malha não Estruturada Hexagonal.

As equações diferenciais parciais não-lineares, Eqs. (1.1) a (1.3), que governam diversos fenômenos físicos, podem ser escritas, genericamente, na forma vetorial, para facilitar o trabalho de integração nos VC. Considerando, genericamente, a equação de convecção-difusão de ϕ , em regime transiente, para um ϕ que pode ser: u , v , ou T , por exemplo, tem-se

$$\frac{\partial(\rho\phi)}{\partial t} + \vec{\nabla} \cdot \vec{J} = S^* \quad (3.1)$$

onde \vec{J} é o fluxo convectivo-difusivo de ϕ através de uma superfície, dado pela equação abaixo:

$$\vec{J} = \rho \vec{V} \phi - \Gamma \vec{\nabla} \phi \quad (3.2)$$

onde \vec{V} é o vetor velocidade no sistema de coordenadas escolhido, ρ é a massa específica do fluido, Γ é o coeficiente de difusão de ϕ no meio, S^ϕ é o termo fonte de geração de ϕ no VC e $\vec{\nabla}$ é o operador vetorial (nabla) do sistema de coordenadas correspondentes.

No presente trabalho, adotou-se como sistema base, as coordenadas cartesianas, com versores i e j , para as direções x e y , respectivamente.

Para um sistema cartesiano bidimensional, tem-se

$$\vec{V} = u \hat{i} + v \hat{j} \quad (3.3)$$

onde u é a velocidade na direção x , v é a velocidade na direção y e o operador vetorial nabla é dado por

$$\vec{\nabla} = \frac{\partial}{\partial x} \hat{i} + \frac{\partial}{\partial y} \hat{j} \quad (3.4)$$

No caso específico da equação de conservação da massa, da quantidade de movimento em fluidos e da energia, tem-se um valor específico de ϕ , S^ϕ e Γ^ϕ , conforme Tab. (3.1),

Tabela 3.1 - Expressões para ϕ , S^ϕ , Γ^ϕ , conforme a equação representativa.

Equações	ϕ	S^ϕ	Γ^ϕ
Conservação da massa	1	0	0
Conservação de quantidade de movimento na direção x	u	$-(\partial p / \partial x)$	μ
Conservação de quantidade de movimento na direção y	v	$-(\partial p / \partial y)$	μ
Transferência de calor por condução	T	q''' / c_p	k / c_p

onde p é a pressão local envolvida, μ é a viscosidade absoluta, T é a temperatura, k é a condutividade térmica e c_p é o calor específico.

Os valores de \bar{J} e S^ϕ , representativos de cada equação substituídos na Eq. (3.2) resultam no sistema de EDPs não-lineares, apresentado nas Eqs. (1.1) a (1.3).

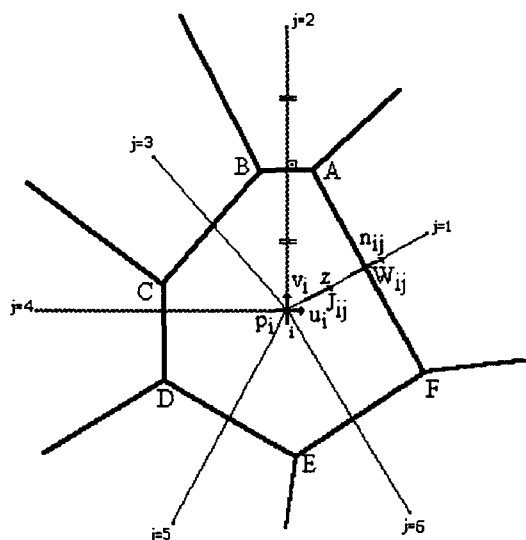


Figura 3.4 - Volume de controle genérico i.

Na Fig. (3.4) é apresentado um VC finito para um ponto nodal i genérico, conjuntamente com os seus respectivos vizinhos j e parâmetros envolvidos.

Procedendo-se a integração numérica, da equação vetorial de ϕ , que pode ser u, v, T ou outra, sobre cada VC i do domínio definido no DV, conforme Fig. (3.4), e sendo \mathcal{V} o volume de integração para cada VC, tem-se

$$\iiint_{\mathcal{V}} \frac{\partial(\rho\phi)}{\partial t} d\mathcal{V} + \iiint_{\mathcal{V}} (\vec{\nabla} \cdot \vec{J}) d\mathcal{V} = \iiint_{\mathcal{V}} S^{\phi} d\mathcal{V} \quad (3.5)$$

Pelo teorema de divergência, pode-se reduzir uma integração volumétrica de um divergente para uma integração de superfície no contorno do volume de controle em questão,

$$\iiint_{\mathcal{V}} (\vec{\nabla} \cdot \vec{J}) d\mathcal{V} = \iint_{S_n} \vec{J} \cdot d\vec{n} \quad (3.6)$$

$$\iiint_{\mathcal{V}} \frac{\partial(\rho\phi)}{\partial t} d\mathcal{V} = \frac{(\rho_i\phi_i^{t+\Delta t} - \rho_i\phi_i^t)}{\Delta t} \Delta\mathcal{V}_i \quad (3.8)$$

onde $\Delta\mathcal{V}_i$ é o volume de controle do ponto nodal i na malha discretizada. Neste ponto é interessante apresentar a forma como é calculado $\Delta\mathcal{V}_i$ em um VC gerado por DV, vide Fig. (3.5),

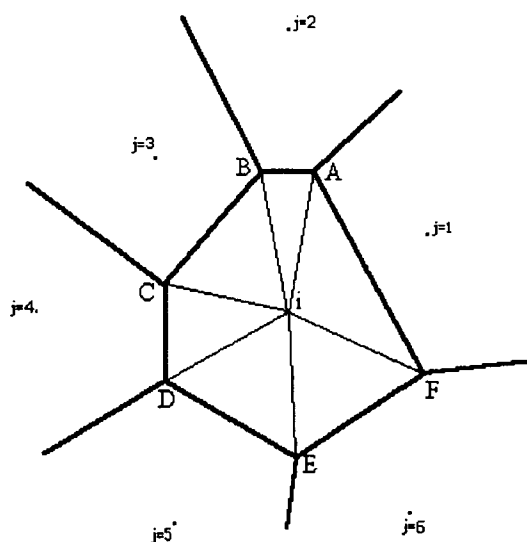


Figura 3.5 - Volume de controle genérico dividido em triângulos.

Observando-se a Fig. (3.5) percebe-se que, devido as propriedades dos DV, o volume $\Delta\mathcal{V}_i$ é formado por triângulos que ligam cada vértice ao ponto nodal i , então, pode-se facilmente calcular o volume somando-se os volumes correspondentes a cada triângulo, que são calculados através do determinante da matriz formada por uma coluna de 1 e seus três vértices, da seguinte forma:

$$\text{Area}_{j,j+1,i} = ((y_i * x_{j+1}) + (y_{j+1} * x_j) + (y_j * x_i) - ((y_j * x_{j+1}) + (y_{j+1} * x_i) + (y_i * x_j))) * 0.5 \quad (3.9)$$

onde x_i, y_i são as coordenadas do ponto nodal i , x_j, y_j são as coordenadas do primeiro vértice do polígono de Voronoi i em relação ao vizinho j (sentido de caminamento anti-horário) e x_{j+1}, y_{j+1} são as coordenadas do vértice $j+1$ subsequente, logo $\Delta\vartheta_i$ é dado por

$$\Delta\vartheta_i = \sum_{j=1}^{nv} \text{Area}_{j,j+1,i} \quad (3.10)$$

Retornando-se a dedução das equações, convencionou-se denotar $t + \Delta t = 1$ e $t = 0$, logo, tem-se

$$\iiint_{\vartheta} \frac{\partial(\rho\phi)}{\partial t} d\vartheta = (\phi_i^1 - \phi_i^0) A_i^0 \quad (3.11)$$

onde

$$A_i^0 = \frac{\rho_i \Delta\vartheta_i}{\Delta t} \quad (3.12)$$

Nos demais termos é necessário estabelecer qual o instante de tempo em que se avaliam as variáveis ϕ . No presente trabalho adota-se a formulação totalmente implícita, onde todas as variáveis são estimadas no instante novo $t + \Delta t$ (denotado por 1). Para simplificar a notação eliminam-se todos os índices 1, por aparecerem em grande maioria.

Assim,

$$\iiint_{\mathcal{V}} \frac{\partial(\rho\phi)}{\partial t} d\mathcal{V} = (\phi_i - \phi_i^0) A_i^0 \quad (3.13)$$

Integrando-se o termo (c), tem-se

$$\iiint_{\mathcal{V}} S d\mathcal{V} = \bar{S} \Delta \mathcal{V}_i \quad (3.14)$$

onde \bar{S} é o valor médio de S no VC i .

As integrações na superfície do VC, do termo (b), podem ser efetuadas de forma discreta somando-se as contribuições de cada face, necessitando-se, para isto, de valores de J estimados em cada face do mesmo (J_{ij}). O cálculo deste valor na interface do VC, no caso do armazenamento colocalizado das variáveis, é feito através de um esquema de interpolação.

Assim,

$$\iint_{S_n} \vec{J} \cdot d\vec{S}_n = \sum_{j=1}^{nv} J_{ij} S_{ij} \quad (3.15)$$

Note-se que $J_{ij} S_{ij}$ representa, exatamente, o fluxo J líquido que passa na face ij . Considerando W como a velocidade normal a face ij , responsável pela advecção de ϕ , e n a direção normal a cada face ij , tem-se que J_{ij} é dado por

$$J_{ij} = \left(\rho W \phi - \Gamma \frac{\partial \phi}{\partial n} \right)_{ij} \quad (3.16)$$

e,

$$\sum_{j=1}^{nv} J_{ij} S_{ij} = \sum_{j=1}^{nv} (\rho W \phi - \Gamma \frac{\partial \phi}{\partial n})_{ij} S_{ij} \quad (3.17)$$

onde,

$$\rho_{ij} = \frac{\rho_i + \rho_j}{2} \quad (3.18)$$

$$\Gamma_{ij} = \frac{2\Gamma_i \Gamma_j}{\Gamma_i + \Gamma_j} \quad (3.19)$$

e

$$W_{ij} = \frac{w_i + w_j}{2} \quad (3.20)$$

Para o cálculo da velocidade normal à face, W_{ij} , optou-se por uma média simples, onde w_i e w_j são as velocidades projetadas na direção normal a face ij (\hat{n}_{ij}), avaliadas, respectivamente, sobre os pontos nodais i e j . Assim,

$$w_i = (u_i \hat{i} + v_i \hat{j}) \cdot \hat{e}_{ij} \quad (3.21)$$

e

$$w_j = (u_j \hat{i} + v_j \hat{j}) \cdot \hat{e}_{ij} \quad (3.22)$$

onde \hat{e}_{ij} é o versor normal a face ij , dado por

$$\hat{e}_{ij} = \frac{\hat{\mathbf{n}}_{ij}}{\|\hat{\mathbf{n}}_{ij}\|} = e_{x_{ij}} \hat{i} + e_{y_{ij}} \hat{j} \quad (3.23)$$

sendo que,

$$\hat{\mathbf{n}}_{ij} = (x_j - x_i) \hat{i} + (y_j - y_i) \hat{j} \quad (3.24)$$

e

$$\|\hat{\mathbf{n}}_{ij}\| = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \quad (3.25)$$

Efetuando o produto escalar dado na Eq. (3.21), tem-se

$$\mathbf{w}_i = u_i \mathbf{e}_{x_{ij}} + v_i \mathbf{e}_{y_{ij}} \quad (3.26)$$

e

$$\mathbf{w}_j = u_j \mathbf{e}_{x_{ij}} + v_j \mathbf{e}_{y_{ij}} \quad (3.27)$$

A componente do gradiente de ϕ normal a face ij pode ser avaliada através dos valores ϕ_j e ϕ_i armazenados sobre a direção normal ij . Portanto,

$$\left(\frac{\partial\phi}{\partial n}\right)_{ij} = \frac{\phi_j - \phi_i}{L_{ij}} \quad (3.28)$$

onde L_{ij} é a distância entre i e j , equivalente a $\left\|\hat{n}_{ij}\right\|$.

Observando-se a Eq. (3.17) verifica-se que falta uma avaliação para o valor interpolado de ϕ_{ij} , ou seja, a grandeza representada por ϕ e cujo resultado exato é obtido pela solução da equação envolvida. Porém isto não faz sentido, pois o objetivo da interpolação é justamente a resolução das equações diferenciais. Então, para se resolver este problema, surgiram várias propostas, mas dentre elas a fisicamente mais consistente foi a de Patankar (1980), que propôs a resolução da equação de transporte unidimensional, na direção de interpolação. Ou seja,

$$\frac{dJ}{dz} = 0 \text{ ou } \frac{d}{dz}(\rho W\phi)_{ij} = \frac{d}{dz}\left(\Gamma \frac{d\phi}{dz}\right)_{ij} \quad (3.29)$$

com $\phi(z=0) = \phi_i$ e $\phi(z=L_{ij}) = \phi_j$, onde z é uma coordenada auxiliar da direção normal ij , conforme Fig. (3.4).

Esta é uma simplificação da equação de convecção e difusão completa, considerando parâmetros constantes na região entre i e j . Resolvendo-se a Equação Diferencial Ordinária Linear proposta por Patankar, considerando ρ_{ij} , W_{ij} e Γ_{ij} constantes ao longo de ij , tem-se

$$\frac{\phi(z) - \phi_i}{\phi_j - \phi_i} = \frac{e^{Pe \cdot z/L_{ij}} - 1}{e^{Pe} - 1} \quad (3.30)$$

onde,

$$Pe = \left(\frac{\rho W L}{\Gamma} \right)_{ij} \quad (3.31)$$

é o número de Peclet do VC i em relação a face j .

A partir da definição de $\phi(z)$ é possível calcular o fluxo J_{ij} que atravessa a face ij com $\phi_{ij} = \phi(z = \frac{L_{ij}}{2})$, como segue:

$$J_{ij} = (\rho W)_{ij} \left[\phi_i + \frac{\phi_i - \phi_j}{e^{Pe} - 1} \right] \quad (3.32)$$

A expressão acima ficou conhecida como Esquema Exponencial, mas devido ao alto custo computacional para sua avaliação, propuseram-se algumas simplificações, a mais conhecida é a seguinte:

$$J = - \left(\frac{\Gamma_{ij}}{L_{ij}} \right) F(Pe) (\phi_j - \phi_i) + \rho_{ij} W_{ij} \phi^{UP} \quad (3.33)$$

onde $\phi^{UP} = \begin{cases} \phi_i & \text{para } W_{ij} > 0 \\ \phi_j & \text{para } W_{ij} < 0 \end{cases}$ e $F(Pe)$ assume valores conforme o esquema escolhido,

upwind de primeira ordem, diferença central, lei da potência, conforme Tab. (3.2).

Tabela 3.2 - Fator peso $F(Pe)$ para vários esquemas de interpolação.

Esquema	$F(Pe)$
Upwind de 1ª ordem	1
Diferença Central	$1 - 0,5 Pe $
Lei da Potência	$\text{máx}[0, (1 - 0,1 Pe)^5]$

Alternativamente, existem interpolações quadráticas, como o QUICK envolvendo três pontos nodais, o qual não é baseado na física do problema. Em malhas com distribuição aleatória de pontos nodais tem-se dificuldades na sua implementação. Presentemente, o esquema utilizado foi o esquema da Lei da Potência, o qual simula o esquema exponencial, com economia de tempo computacional.

Retornando a Eq. (3.15), referente ao termo (b) da Eq. (3.7), tem-se

$$\iint_{S_n} \bar{J} \cdot d\bar{S}_n = \sum_{j=1}^{nv} J_{ij} S_{ij} = \sum_{j=1}^{nv} \left[\left(\frac{\Gamma_{ij} S_{ij}}{L_{ij}} \right) F(Pe)(\phi_j - \phi_i) + \rho_{ij} W_{ij} S_{ij} \phi^{UP} \right] \quad (3.34)$$

onde

$$\rho_{ij} W_{ij} S_{ij} \phi^{UP} = \max[0, +F_{ij}] \phi_i - \max[0, -F_{ij}] \phi_j \quad (3.35)$$

e

$$F_{ij} = (\rho WS)_{ij} \quad (3.36)$$

é a vazão mássica, e $\max[0, \pm F_{ij}]$ toma o maior dentre os dois valores,

$$D_{ij} = \left(\frac{\Gamma S}{L} \right)_{ij} \quad (3.37)$$

é o coeficiente de difusão,

$$Pe = \frac{F_{ij}}{D_{ij}} \quad (3.38)$$

é o número de Peclet.

Assim, o termo (b) da Eq. (3.7) toma a seguinte forma:

$$\iint_{S_n} \bar{J} \cdot d\bar{S}_n = \sum_{j=1}^{nv} \left(-D_{ij} \cdot F(Pe) (\phi_j - \phi_i) + \max[0, +F_{ij}] \phi_i - \max[0, -F_{ij}] \phi_j \right) \quad (3.39)$$

3.3 - Avaliação do gradiente de pressão

Cabe lembrar que os termos fonte (termo (c)) das equações de u e v possuem avaliações de gradientes de pressão, que podem ser obtidas de várias formas:

Caso 1: Média Ponderada entre os Gradientes Normais Projetados

O gradiente de pressão pode ser obtido por médias ponderadas de seus componentes sobre cada face (Maliska, 1995). Assim,

$$\nabla p_x = \frac{\sum_{j=1}^{nv} \left[\left(\bar{\nabla} p_{ij} \cdot \hat{i} \right) L_{ij} \right]}{\sum_{j=1}^{nv} L_{ij}} \quad (3.40)$$

$$\nabla p_y = \frac{\sum_{j=1}^{nv} \left[\left(\bar{\nabla} p_{ij} \cdot \hat{j} \right) L_{ij} \right]}{\sum_{j=1}^{nv} L_{ij}} \quad (3.41)$$

onde $\bar{\nabla} p_{ij}$ é o gradiente de pressão normal à interface ij , e é dado por

$$\bar{\nabla} p_{ij} = \left(\frac{p_j - p_i}{L_{ij}} \right) \cdot \hat{e}_{ij} = \left(\frac{p_j - p_i}{L_{ij}} \right) \cdot \left(e_{x_{ij}} \hat{i} + e_{y_{ij}} \hat{j} \right) \quad (3.42)$$

Mas, estas expressões não avaliam adequadamente os gradientes nos pontos centrais i , e isto já foi corrigido por Maliska (1995). Agora isto será mostrado através de um exemplo. Supõe-se que um determinado campo de pressão satisfaça a seguinte distribuição linear, conforme Fig. (3.6).

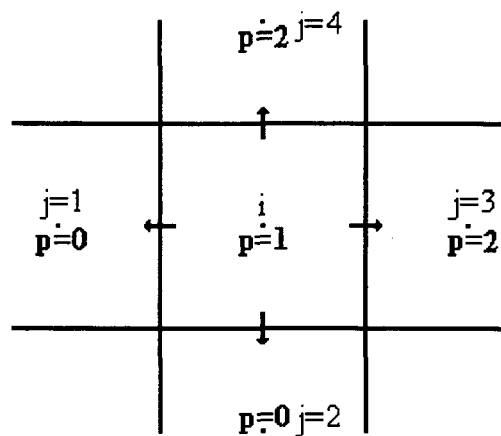


Figura 3.6 - Malha uniforme com distribuição linear de pressão.

Neste caso, o denominador das Eqs. (3.40) e (3.41) considera todas as distâncias que separam o ponto nodal i de seus vizinhos j . Logo, supondo uma malha uniforme, com distâncias nodais unitárias, conforme o volume apresentado na Fig. (3.6), e calculando-se o gradiente de pressão para x ∇p_x , como proposto por Maliska, obtem-se o seguinte:

$$\nabla p_x = \frac{\begin{matrix} j=1 & j=2 & j=3 & j=4 \\ (0-1)(-1) & (0-1)(0) & (2-1)(1) & (2-1)(0) \end{matrix}}{1+1+1+1} = \frac{1+1}{4} = \frac{1}{2}$$

Porém, verifica-se que o gradiente exato é 1, e, no entanto, desta forma obtivemos a metade do resultado correto, isto em uma malha estruturada, igualmente espaçada, conforme Fig. (3.6).

Conforme o próprio Maliska (1995, errata) as Eqs. (3.40) e (3.41) eram uma proposta, e estas equações, particularmente, não haviam sido testadas e, aparentemente, supunha-se que somente fossem válidas para polígonos irregulares ao mesmo tempo em que sugeria a expressão de Jameson e Mavriplis (discutida mais adiante). No entanto, isto permitiu que, no presente trabalho, se fizesse uma reflexão sobre as Eqs. (3.40) e (3.41) apresentadas, de onde pode-se concluir, por exemplo, que o somatório de distâncias L_{ij} se consideradas as influências das distâncias L_{ij} em x e em y sem distinção as projeções em y nada acrescentavam ao cálculo do gradiente de pressão ∇p_x , pois no caso do volume quadrado elas, na verdade, dividem o valor por dois. Então, propõe-se que para o cálculo do gradiente de pressão x considere-se apenas as distâncias L_{ij} projetadas em x , da seguinte forma:

$$\nabla p_x = \frac{\sum_{j=1}^{nv} \left[\left(\bar{\nabla} p_{ij} \cdot \hat{i} \right) L_{ij} \right]}{\sum_{j=1}^{nv} (L_{ij} |e_{x_{ij}}|)} \quad (3.43)$$

analogamente para y

$$\nabla p_y = \frac{\sum_{j=1}^{nv} \left[\left(\bar{\nabla} p_{ij} \cdot \hat{j} \right) L_{ij} \right]}{\sum_{j=1}^{nv} (L_{ij} |e_{y_{ij}}|)} \quad (3.44)$$

Os denominadores das Eqs. (3.40) e (3.41) consideram apenas as distâncias que contribuem com gradiente projetado. Portanto, as projeções do gradiente em x, por exemplo, consideram apenas as distâncias L_{ij} projetadas em x. Então, refazendo os cálculos para o exemplo anterior, tem-se:

$$\nabla p_x = \frac{\overset{j=1}{(0-1)(-1)} + \overset{j=2}{(0-1)(0)} + \overset{j=3}{(2-1)(1)} + \overset{j=4}{(2-1)(0)}}{1|-1| + 1|0| + 1|1| + 1|0|} = \frac{1+1}{2} = 1$$

Assim, conseguiu-se obter o valor exato do gradiente de pressão proposto na Fig. (3.6).

Tratamento especial foi dispensado aos pontos nodais ligados às fronteiras do domínio geométrico, cujo campo de pressão do contorno não é conhecido. Assim, optou-se por considerar apenas as contribuições do campo de pressão interno, pois se fosse calculado o campo do contorno, através de extrapolação do campo interno, não se alcançaria informação adicional alguma ao se usar este campo de pressão extrapolado para o contorno. Toda informação estaria baseada somente no campo interno.

Caso 2: Mínimo Resíduo Quadrático

O vetor gradiente de pressão é avaliado sobre cada ponto nodal i do domínio e possui duas componentes (caso bidimensional),

$$\vec{\nabla} p_i = (\nabla p)_x \hat{i} + (\nabla p)_y \hat{j} \quad (3.45)$$

Mas, o que se tem disponível são apenas as componentes do gradiente nas direções normais às faces e avaliadas sobre estas faces,

$$(\nabla p)_{ij} = \frac{p_j - p_i}{L_{ij}} \quad (3.46)$$

Portanto, existe uma diferença entre a avaliação sobre cada face e a projeção do gradiente na direção normal \hat{n}_{ij} avaliado sobre o ponto nodal central. Esta diferença δp^{ij} é dada por

$$\delta p^{ij} = \frac{p_j - p_i}{L_{ij}} - (\tilde{\nabla} p_i) \cdot \hat{e}_{ij} \quad (3.47)$$

Conforme Taniguchi (1991a e 1991b), propõe-se uma função global Ψ_i de minimização das diferenças δp_{ij} , dada por

$$\Psi_i = \sum_{j=1}^{nv} \left(g^{ij} \delta p^{ij} \delta p^{ij} \right) \quad (3.48)$$

substituindo a Eq. (3.47) em (3.48), tem-se

$$\Psi_i = \sum_{j=1}^{nv} \left\{ g^{ij} \left[(\nabla p)_{ij} - \left(\nabla p_x \hat{i} + \nabla p_y \hat{j} \right) \cdot \hat{e}_{ij} \right]^2 \right\} \quad (3.49)$$

logo,

$$\Psi_i = \sum_{j=1}^{nv} \left\{ g^{ij} \left[(\nabla p)_{ij} - \left(\nabla p_x e_{x_{ij}} + \nabla p_y e_{y_{ij}} \right) \right]^2 \right\} \quad (3.50)$$

onde g^{ij} é uma função peso do gradiente normal de cada face. Neste caso, adotou-se g^{ij} como sendo proporcional ao ângulo de visão de cada face ij em relação ao ponto central i (Taniguchi, 1991a e 1991b). Este valor é equivalente a razão entre a área de passagem S_{ij} e a distância L_{ij} . Portanto, g^{ij} é dado por

$$g^{ij} = \frac{S_{ij}}{L_{ij}} \quad (3.51)$$

Fazendo-se uma minimização de ψ_i em função de cada componente do vetor gradiente de pressão, ∇p_x e ∇p_y , obtém-se duas equações para as duas componentes incógnitas, quais sejam

$$\frac{\partial \Psi_i}{\partial (\nabla p_x)} = \sum_{j=1}^{nv} \left\{ 2g^{ij} \left[(\nabla p)_{ij} - \left((\nabla p)_x e_{x_{ij}} + (\nabla p)_y e_{y_{ij}} \right) \right] e_{x_{ij}} \right\} = 0 \quad (3.52)$$

$$\frac{\partial \Psi_i}{\partial (\nabla p_y)} = \sum_{j=1}^{nv} \left\{ 2g^{ij} \left[(\nabla p)_{ij} - \left((\nabla p)_x e_{x_{ij}} + (\nabla p)_y e_{y_{ij}} \right) \right] e_{y_{ij}} \right\} = 0 \quad (3.53)$$

A partir das equações (3.52) e (3.53) pode ser obtido um sistema de equações lineares para ∇p_x e ∇p_y , da seguinte forma:

$$\begin{cases} \left[\sum_{j=1}^{nv} g^{ij} (e_{x_{ij}})^2 \right] \nabla p_x + \left[\sum_{j=1}^{nv} g^{ij} (e_{y_{ij}} e_{x_{ij}}) \right] \nabla p_y = \sum_{j=1}^{nv} \left[g^{ij} \nabla p_{ij} e_{x_{ij}} \right] \\ \left[\sum_{j=1}^{nv} g^{ij} (e_{x_{ij}} e_{y_{ij}}) \right] \nabla p_x + \left[\sum_{j=1}^{nv} g^{ij} (e_{y_{ij}})^2 \right] \nabla p_y = \sum_{j=1}^{nv} \left[g^{ij} \nabla p_{ij} e_{y_{ij}} \right] \end{cases} \quad (3.54)$$

Resolvendo-se o sistema dado pela Eq. (3.54), tem-se

$$\nabla p_x = \frac{\left[\sum_{j=1}^{nv} g^{ij} \nabla p_{ij} e_{x_{ij}} \right] \left[\sum_{j=1}^{nv} g^{ij} (e_{y_{ij}})^2 \right] - \left[\sum_{j=1}^{nv} g^{ij} (e_{y_{ij}} e_{x_{ij}}) \right] \left[\sum_{j=1}^{nv} g^{ij} \nabla p_{ij} e_{y_{ij}} \right]}{\left[\sum_{j=1}^{nv} g^{ij} (e_{x_{ij}})^2 \right] \left[\sum_{j=1}^{nv} g^{ij} (e_{y_{ij}})^2 \right] - \left[\sum_{j=1}^{nv} g^{ij} (e_{x_{ij}} e_{y_{ij}}) \right]^2} \quad (3.55)$$

e

$$\nabla p_y = \frac{\left[\sum_{j=1}^{nv} g^{ij} (e_{x_{ij}})^2 \right] \left[\sum_{j=1}^{nv} g^{ij} \nabla p_{ij} e_{y_{ij}} \right] - \left[\sum_{j=1}^{nv} g^{ij} (e_{x_{ij}} e_{y_{ij}}) \right] \left[\sum_{j=1}^{nv} g^{ij} \nabla p_{ij} e_{x_{ij}} \right]}{\left[\sum_{j=1}^{nv} g^{ij} (e_{x_{ij}})^2 \right] \left[\sum_{j=1}^{nv} g^{ij} (e_{y_{ij}})^2 \right] - \left[\sum_{j=1}^{nv} g^{ij} (e_{x_{ij}} e_{y_{ij}}) \right]^2} \quad (3.56)$$

Nesta avaliação também foram consideradas apenas as contribuições dos gradientes internos ao domínio geométrico, sem considerar-se o campo de pressão sobre o contorno geométrico.

Caso 3: Avaliação segundo Jameson e Mavriplis (1986)

A avaliação do gradiente de pressão pode ser obtido pela eq.(3.57) (Maliska, 1995), segundo proposta de Jameson e Mavriplis (1986),

$$\Delta p_x = \sum_{j=1}^{nv} \frac{\left(\frac{p_j S_{ij} e_{x_{ij}}}{2} \right)}{\Delta \mathfrak{S}_i} \quad (3.57)$$

Para um melhor entendimento das limitações impostas à Eq. (3.57), apresentada por Jameson e Mavriplis (1986), procurou-se reproduzir a sua dedução (Peters, 1997).

Tomando a equação da conservação da quantidade de movimento de Cauchy geral (Serrin, 1959), na sua forma tensorial, conforme Eq. (3.58), a seguir:

$$\rho \frac{d\bar{u}}{dt} = \rho \bar{f} + \nabla \cdot \bar{\bar{T}} \quad (3.58)$$

onde ρ é a massa específica, \bar{u} é o vetor velocidade, $\frac{d\bar{u}}{dt}$ é a derivada material de \bar{u} , \bar{f} é uma força de corpo genérica $\bar{\bar{T}}$ é o tensor que avalia as relações entre tensão e deformação, no presente caso, para fluidos newtonianos, dado pela Eq. (3.59), conforme segue:

$$\bar{\bar{T}} = -p \cdot \bar{\bar{I}} + \left(\beta - \frac{2\mu}{3} \right) \bar{\bar{I}} \nabla \cdot \bar{u} + 2\mu \bar{\bar{D}} \quad (3.59)$$

No tensor tensão $\bar{\bar{T}}$ estão constituídas as relações entre tensão e deformação, incluindo as contribuições da tensão normal, dada pela pressão p , e das tensões devidas às deformações de volume e também devidas ao atrito gerado nas deformações por cisalhamento (Serrin, 1959).

O termo $\bar{\bar{I}}$ é o tensor identidade (Aris, 1962), definido conforme a Eq. (3.60):

$$\bar{\bar{I}} = \delta_{mn} \hat{h}_m \cdot \hat{h}_n \quad (3.60)$$

onde δ_{mn} é o delta de Kronecker, \hat{h}_m e \hat{h}_n são os versores ortonormais de um sistema genérico de coordenadas.

Focalizando o interesse deste trabalho somente no termo de pressão do tensor tensão $\bar{\bar{T}}$, integrando-o em um volume de controle $\Delta\vartheta_i$, conforme esquema do MVF, tem-se a avaliação do gradiente de pressão $\bar{\nabla}p$ no VC $\Delta\vartheta_i$, conforme Eq. (3.61):

$$\bar{\nabla}p.\Delta\vartheta_i = \iiint_{\vartheta_i} \bar{\nabla}.(-p\bar{\bar{I}})d\vartheta \quad (3.61)$$

Pelo teorema da divergência, pode-se reduzir uma integral de volume de um divergente para uma integral de superfície, conforme Eq. (3.62):

$$\bar{\nabla}p.\Delta\vartheta_i = \iiint_{\vartheta_i} \bar{\nabla}.(-p\bar{\bar{I}})d\vartheta = \iint_{S_{ij}} -p_s\bar{\bar{I}}.\hat{n} dS_n \quad (3.62)$$

onde \hat{n} é o versor normal à superfície de integração, ou seja, é o versor de cada face S_{ij} do VC gerado por DV, conforme Fig. (2.1), dS_n é a área de integração de cada face, e p_s é a pressão avaliada em cada face S_{ij} do VC. Assim, o versor \hat{n} será denotado conforme Eq. (3.63):

$$\hat{n} = n_k^{ij}.\hat{h}_k \quad (3.63)$$

onde n_k^{ij} são as k componentes do versor ortonormal \hat{n} vinculadas a cada face ij.

Substituindo as Eqs. (3.60) e (3.63) na Eq. (3.62), tem-se

$$\begin{aligned} \bar{\nabla}p.\Delta\vartheta_i &= \iint_{S_{ij}} -\left(p_s\delta_{mn}\hat{h}_m\hat{h}_n\right)\left(n_k^{ij}\hat{h}_k\right)dS_n = \\ &= \sum_{j=1}^{nv} \left(-p_s\delta_{mn}\hat{h}_m\hat{h}_n\right)\left(n_k^{ij}\hat{h}_k\right)S_{ij} \end{aligned} \quad (3.64)$$

Efetuada-se o produto vetorial existente na Eq. (3.64) (Aris, 1962), tem-se

$$\bar{\nabla} p \Delta \vartheta_i = - \sum_{j=1}^{nv} (p_s n_k^{ij} S_{ij}) \hat{h}_k \quad (3.65)$$

Avaliando a pressão p_s na face ij do VC i , através de média aritmética simples entre os valores p_i e p_j , conforme Eq. (3.66), e substituindo-a na eq. (3.65), obtém-se a Eq. (3.67),

$$p_s = \frac{p_i + p_j}{2} \quad (3.66)$$

$$\bar{\nabla} p \Delta \vartheta_i = - \sum_{j=1}^{nv} \left(\frac{p_i}{2} n_k^{ij} S_{ij} \right) \hat{h}_k - \sum_{j=1}^{nv} \left(\frac{p_j}{2} n_k^{ij} S_{ij} \right) \hat{h}_k \quad (3.67)$$

Analisando a primeira parcela da Eq. (3.67), nota-se que se trata de uma integração cíclica de superfície em um polígono fechado, e nestes casos tem-se uma integração nula, conforme Eq. (3.68):

$$- \sum_{j=1}^{nv} \left(\frac{p_i}{2} n_k^{ij} S_{ij} \right) \hat{h}_k = - \frac{p_i}{2} \sum_{j=1}^{nv} n_j^{ij} S_{ij} \hat{h}_k = 0 \quad (3.68)$$

Assim,

$$\bar{\nabla} p = - \frac{1}{2 \Delta \vartheta_i} \sum_{j=1}^{nv} p_j n_k^{ij} S_{ij} \hat{h}_k \quad (3.69)$$

Exemplificando esta avaliação para a direção x ($k=1$, $\hat{h}_1 = \hat{i}$, $n_1^{ij} = e_{x_{ij}}$) do sistema de coordenadas cartesianas, tem-se a seguinte componente horizontal do gradiente de pressão:

$$\nabla p_x = -\frac{1}{2\Delta g_i} \sum_{j=1}^{nv} p_j S_{ij} e_{x_{ij}} \quad (3.70)$$

Na avaliação dada pela Eq. (3.70) tem-se problemas com avaliações do gradiente de pressão em pontos nodais adjacentes às fronteiras, onde a pressão p_s , dos contornos do domínio, não é conhecida. Além do que p_s não pode ser avaliada através de média entre p_i e p_j (vizinho), pois p_j seria a própria pressão do contorno, conforme pode ser visto na Fig. (1) do apêndice A.

Nota-se na Fig. (1) do Apêndice A, que a pressão no vizinho do contorno do ponto nodal $i=91$ é a própria pressão p_s do contorno, portanto a avaliação do gradiente de pressão nestes pontos de fronteira do domínio computacional deve ser feita através da Eq. (3.65), onde p_s deve assumir avaliações diferenciadas em cada face ij do VC.

Então, uma desvantagem desta formulação é a necessidade de avaliação de todas as pressões sobre as fronteiras do domínio computacional, o que desestimulou a aplicação desta formulação no presente trabalho, em favor da aplicação da avaliação da média ponderada entre os gradientes normais projetados e da proposta de Taniguchi *et alli*, que não necessitam da avaliação das pressões nas fronteiras do domínio.

Obtidas numericamente as duas componentes do gradiente de pressão, segundo algum dos casos apresentados, pode-se avaliar as componentes de velocidade u e v a partir de um campo inicial de u^* , v^* e p^* .

Juntando os termos (a), (b) e (c), segundo as Eqs. (3.13), (3.14) e (3.39), e rearranjando as variáveis, tem-se

$$\begin{aligned} & (\phi_i - \phi_i^0) A_i^0 + \sum_{j=1}^{nv} \left[-D_{ij} \cdot F(\text{Pe}) (\phi_j - \phi_i) + \max[0, +F_{ij}] \phi_i - \max[0, -F_{ij}] \phi_j \right] \\ & = \bar{S}^\phi \Delta \theta_i \end{aligned} \quad (3.71)$$

Soma-se a equação da continuidade discretizada (para escoamento incompressível, $\rho = \text{cte}$) multiplicada por ϕ_i , à Eq. (3.71).

Assim, tomando a equação da continuidade, tem-se

$$\iiint_{\mathcal{V}} (\bar{\nabla} \cdot \bar{\mathbf{J}}) d\mathcal{V} = 0 \quad (3.72)$$

com $\bar{\mathbf{J}} = \rho \bar{\mathbf{V}}$.

Pelo teorema da divergência, tem-se

$$\iint_{S_n} \rho \bar{\mathbf{V}} \cdot d\bar{\mathbf{S}}_n = 0 \quad (3.73)$$

$$\iint_{S_n} \rho \bar{\mathbf{V}} \cdot d\bar{\mathbf{S}}_n = \sum_{j=1}^{nv} (\rho_{ij} W_{ij} S_{ij}) = \sum_{j=1}^{nv} F_{ij} = 0 \quad (3.74)$$

Reescrevendo a equação (3.74) multiplicada por ϕ_i , obtém-se a seguinte expressão,

$$\sum_{j=1}^{nv} F_{ij} \phi_i = \sum_{j=1}^{nv} \left\{ \max[0, +F_{ij}] \phi_i - \max[0, -F_{ij}] \phi_j \right\} = 0 \quad (3.75)$$

Subtraindo a Eq. (3.75), que é nula, da Eq. (3.71), tem-se

$$\begin{aligned}
 & (\phi_i - \phi_i^0)A_i^0 + \\
 & \sum_{j=1}^{nv} \left[-D_{ij} \cdot F(\text{Pe})(\phi_j - \phi_i) + \max[0, +F_{ij}] \phi_i - \max[0, -F_{ij}] \phi_j \right] - \\
 & \sum_{j=1}^{nv} \left\{ \max[0, +F_{ij}] \phi_i - \max[0, -F_{ij}] \phi_j \right\} = \bar{S}^\phi \Delta \vartheta_i - 0
 \end{aligned} \tag{3.76}$$

Rearranjando os termos, tem-se

$$(\phi_i - \phi_i^0)A_i^0 + \sum_{j=1}^{nv} \left[\begin{array}{l} -D_{ij} \cdot F(\text{Pe})(\phi_j - \phi_i) - \\ \max[0, -F_{ij}](\phi_j - \phi_i) \end{array} \right] = \bar{S}^\phi \Delta \vartheta_i \tag{3.77}$$

$$(\phi_i - \phi_i^0)A_i^0 + \sum_{j=1}^{nv} \left\{ \begin{array}{l} [\max[0, -F_{ij}] + D_{ij} \cdot F(\text{Pe})] \phi_i - \\ [\max[0, -F_{ij}] + D_{ij} \cdot F(\text{Pe})] \phi_j \end{array} \right\} = \bar{S}^\phi \Delta \vartheta_i \tag{3.78}$$

Alocando os termos em ϕ_j , no lado direito da equação e reorganizando-os,
tem-se

$$\left\{ A_i^0 + \sum_{j=1}^{nv} \max[0, -F_{ij}] + D_{ij} \cdot F(\text{Pe}) \right\} \phi_i =$$

$$\sum_{j=1}^{nv} \left\{ \left[\max[0, -F_{ij}] + D_{ij} \cdot F(\text{Pe}) \right] \phi_j \right\} + \bar{S}^\phi \Delta \vartheta_i + \phi_i^0 A_i^0 \quad (3.79)$$

Agrupando os termos, tem-se

$$A_i^P \phi_i = \sum_{j=1}^{nv} \left\{ A_{ij} \phi_j \right\} + b_i^\phi \quad (3.80)$$

onde

$$A_{ij} = \left[\max[0, -F_{ij}] + D_{ij} \cdot F(\text{Pe}) \right] \quad (3.81)$$

$$A_i^P = A_i^0 + \sum_{j=1}^{nv} A_{ij} \quad (3.82)$$

e

$$b_i^\phi = S^\phi \Delta \vartheta_i + \phi_i^0 A_i^0 \quad (3.83)$$

Lembra-se que para $=u$ ou $=v$, tem-se um termo fonte auxiliar de gradiente de pressão, logo,

$$A_i^P u_i = \sum_{j=1}^{nv} \{A_{ij} u_j\} + b_i^u - \Delta \vartheta_i (\nabla p)_x \quad (3.84)$$

e

$$A_i^P v_i = \sum_{j=1}^{nv} \{A_{ij} v_j\} + b_i^v - \Delta \vartheta_i (\nabla p)_y \quad (3.85)$$

Até o presente momento tem-se apenas uma equação evolutiva para u e v , ou para um ϕ que seja regido por uma equação de transporte do tipo convectiva e difusiva.

A pressão é uma incógnita que aparece apenas como uma variável das equações gerais, não se tem uma equação evolutiva específica para a variável pressão. Para resolver este problema adota-se um acoplamento entre pressão e velocidade, através da equação de conservação da massa. A pressão não necessitaria ser calculada, uma vez

que poderia se resolver o sistema para u e v usando apenas os gradientes de pressão na direção x e y .

3.4 - Acoplamento SIMPLE pressão-velocidade

Uma maneira computacionalmente econômica de resolver este problema é usar a equação da conservação da massa como equação para pressão, adotando um acoplamento entre pressão e velocidade. Trata-se de uma resolução segregada das variáveis envolvidas: u , v e p , resolvidas uma de cada vez.

Adota-se aqui o Método SIMPLE de acoplamento (Semi-Implicit Method for Pressure Linked Equations), modificado para malhas colocadas (Peric *et alli*, 1988), conforme descrição a seguir.

A idéia central do método é encontrar um campo de pressão que satisfaça a equação da conservação da massa, Eq. (3.74).

Os passos a serem seguidos são descritos nos próximos subitens:

3.4.1 - Projeção das componentes nodais de velocidades na direção normal \hat{n}_{ij}

Substituindo as equações de u e v , equações (3.84) e (3.85), na equação (3.26), tem-se

$$A_i^p w_i = \sum_{k=1}^{nv} \{A_{ik} w_k\} + b_i^w - \Delta \theta_i (\nabla p)_w^i \quad (3.86)$$

onde k indica o índice dos vizinhos de i e b_i^w é dado por

$$b_i^w = b_i^u e_{x_{ij}} + b_i^v e_{y_{ij}} \quad (3.87)$$

e

$$(\nabla p)_w^i = (\nabla p)_x e_{x_{ij}} + (\nabla p)_y e_{y_{ij}} \quad (3.88)$$

A forma análoga da Eq. (3.86) só é possível em malhas colocalizadas, onde os coeficientes A_{ij} e A_i são os mesmos para u_i , v_i e w_i .

Note-se que w_i é avaliado sobre o ponto nodal i , e precisa-se do valor sobre a face ij dado por W_{ij} , dada por uma média simples entre w_i e w_j , Eq. (3.20), para satisfazer a equação da conservação da massa. Portanto é necessária a avaliação de w_j por equação análoga a Eq. (3.86).

3.4.2 - Geração de uma equação evolutiva para as componentes de velocidades normais às faces W_{ij} a partir das equações de w_i e w_j vizinhas da face ij

Tomando-se as equações, para w_i e w_j , tem-se

$$A_i^p w_i = \sum_{k=1}^{nv} \{A_{ik} w_k\} + b_i^w - \Delta \vartheta_i (\nabla p)_w^i \quad (3.89)$$

$$A_i^p w_j = \sum_{k=1}^{nv} \{A_{jk} w_k\} + b_j^w - \Delta \vartheta_j (\nabla p)_w^j \quad (3.90)$$

de onde obtém-se W_{ij} através de uma média aritmética simples, que é uma boa aproximação para malhas geradas por DV, vide Eq. (3.20).

Substituindo w_i e w_j , dadas pelas equações (3.89) e (3.90), em W_{ij} dada por (3.20), tem-se

$$W_{ij} = \left\{ \left[\frac{\sum_{k=1}^{nv} [A_{ik} w_k] + b_i^w}{A_i^p} \right] + \left[\frac{\sum_{k=1}^{nv} [A_{jk} w_k] + b_j^w}{A_j^p} \right] \right\} / 2 - d_{ij}^w (\nabla p)_w^{ij} \quad (3.91)$$

onde d_{ij}^w é dado por

$$d_{ij}^w = \frac{1}{2} \left[\frac{\Delta \vartheta_i}{A_i^p} + \frac{\Delta \vartheta_j}{A_j^p} \right] \quad (3.92)$$

e $(\nabla p)_w^{ij}$ pode ser dado por

$$(\nabla p)_w^{ij} = \frac{1}{2} \left[(\nabla p)_w^i + (\nabla p)_w^j \right] \quad (3.93)$$

Uma análise a respeito da avaliação de d_{ij}^w pela Eq. (3.93) pode ser encontrada no Apêndice A. Optou-se por esta forma porque os valores obtidos para d_{ij}^w nos contornos são mais próximos dos obtidos por d_{ij}^w internos, do que em relação a outras formas conhecidas na literatura (Maliska, 1995).

Agrupando os termos da Eq. (3.92), tem-se

$$W_{ij} = \hat{W}_{ij} - d_{ij}^w (\nabla p)_w^{ij} \quad (3.94)$$

onde

$$\hat{W}_{ij} = \left\{ \left[\frac{\sum_{k=1}^{nv} \{A_{ik} w_k\} + b_i^w}{A_i^p} \right] + \left[\frac{\sum_{k=1}^{nv} \{A_{jk} w_k\} + b_j^w}{A_j^p} \right] \right\} / 2 \quad (3.95)$$

Deve-se notar que o gradiente de pressão normal à face, avaliado sobre a face ij , pode ser avaliado mais consistentemente através dos pontos nodais i e j vizinhos a face ij , por isto adotou-se a Eq. (3.96), conforme segue, vide Fig. (3.4),

$$(\nabla p)_w^{ij} = \frac{p_j - p_i}{L_{ij}} \quad (3.96)$$

Esta avaliação do gradiente de pressão é análoga ao esquema de malha desencontrada.

Através da equação (3.95), com gradiente avaliado pela equação (3.96), e de um campo estimado u^* , v^* e p^* , podemos avaliar W_{ij}^* estimado, mas que ainda não conserva a massa, ou seja,

$$W_{ij}^* = \hat{W}_{ij}^* - d_{ij}^w (\nabla p^*)_{ij}^w \quad (3.97)$$

Deve-se agora encontrar um campo de pressão p que venha a satisfazer a equação da conservação da massa, equação (3.74).

Repetindo, tem-se

$$\iint_{s_n} \rho \vec{V} \cdot d\vec{S}_n = \sum_{j=1}^{nv} (\rho_{ij} W_{ij} S_{ij}) = \sum_{j=1}^{nv} F_{ij} = 0 \quad (3.98)$$

3.4.3 - Geração de um campo de pressão que satisfaça a conservação de massa

Supondo que a Eq. (3.99) a seguir, satisfaça corretamente a conservação da massa

$$W_{ij} = \hat{W}_{ij} - d_{ij}^w (\nabla p)_w^{ij} \quad (3.99)$$

então, deve haver uma correção de velocidades W'_{ij} que corrija W^*_{ij} e que conduza a esta conservação.

Assim,

$$W_{ij} = W^*_{ij} + W'_{ij} \quad (3.100)$$

ou

$$W'_{ij} = W_{ij} - W_{ij}^* \quad (3.101)$$

Logo, subtraindo a equação (3.97) de (3.99), de acordo com a equação (3.101), tem-se a equação de correção W'_{ij} , dada por

$$W'_{ij} = \hat{W}'_{ij} - d_{ij}^w (\nabla p')_{ij}^w \quad (3.102)$$

onde

$$p = p^* + p' \quad (3.103)$$

ou

$$p' = p - p^* \quad (3.104)$$

Assim, substituindo-se a equação (3.102) em (3.100), tem-se

$$W_{ij} = W_{ij}^* + \hat{W}_{ij} - d_{ij}^w (\nabla p')_{ij}^w \quad (3.105)$$

Pode-se aplicar uma equação simplificada para correção de W_{ij}^* , pois após o processo convergido todas as correções impostas serão nulas. Pode-se desprezar as correções \hat{W}_{ij} da equação (3.102) gerando,

$$W_{ij}' = -d_{ij}^w (\nabla p')_{ij}^w \quad (3.106)$$

E o valor correto de W_{ij} dado pela equação (3.105), após simplificada, torna-se

$$W_{ij} = W_{ij}^* - d_{ij}^w \left(\frac{p_j' - p_i'}{L_{ij}} \right). \quad (3.107)$$

3.5 - Acoplamento SIMPLEC pressão-velocidade

Alternativamente, por questões de estabilidade na convergência numérica, pode-se adotar o Método SIMPLEC de Van Doormaal e Raithby (1984). O método SIMPLEC é o SIMPLE corrigido, diferindo, apenas, nas equações de correção das velocidades. A seguir, o método SIMPLEC será detalhado.

Tomando as Eqs. (3.89) e (3.90) para calcular os valores estimados de w_i^* e w_j^* , conforme as Eqs. (3.108) e (3.109), a seguir

$$A_i^p w_i^* = \sum_{k=1}^{nv} \{A_{ik} w_k^*\} + b_i^{w^*} - \Delta V_i (\nabla_p^*)_w^i \quad (3.108)$$

e

$$A_j^p w_j^* = \sum_{k=1}^{nv} \{A_{jk} w_k^*\} + b_j^{w^*} - \Delta \mathcal{V}_j (\nabla_p^*)_w^j \quad (3.109)$$

e tomando as Eqs. (3.89) e (3.90), como sendo os valores corretos de w_i e w_j , que devem satisfazer a conservação da massa, e subtraindo as Eqs. (3.108) e (3.109) das Eqs. (3.89) e (3.90), obtém-se os valores das correções w_i' e w_j'

$$A_i^p w_i' = \sum_{k=1}^{nv} \{A_{ik} w_k'\} + b_i^{w'} - \Delta \mathfrak{G}_i (\nabla p')_w^i \quad (3.110)$$

e

$$A_j^p w_j' = \sum_{k=1}^{nv} \{A_{jk} w_k'\} + b_j^{w'} - \Delta \mathfrak{G}_j (\nabla p')_w^j \quad (3.111)$$

O valor de $b_i^{w'}$ pode ser desprezado, pois normalmente este termo independe de w_i , por isso propõe-se um valor para correção w_i' independente de $b_i^{w'}$.

Subtraindo o termo $\sum_{k=1}^{nv} \{A_{ik} w_k'\}$ de ambos os membros da Eq. (3.110) e, analogamente, repetindo este procedimento para a Eq. (3.111), geram-se as seguintes equações para correção de w_i' e w_j' ,

$$\left(A_i^p - \sum_{k=1}^{nv} A_{ik} \right) w_i' = \sum_{k=1}^{nv} \{A_{ik} (w_k' - w_i')\} - \Delta \mathfrak{G}_i (\nabla p')_w^i \quad (3.112)$$

e

$$\left(A_j^p - \sum_{k=1}^{nv} A_{jk} \right) w_j' = \sum_{k=1}^{nv} \left\{ A_{jk} (w_k' - w_j') \right\} - \Delta \mathcal{G}_j (\nabla p')_w^j \quad (3.113)$$

Desprezando-se, consistentemente, o termo $\sum_{k=1}^{nv} \{A_{ik}(w_k' - w_i')\}$ da Eq.

(3.112), por se tratar da diferença entre as correções dos vizinhos e do ponto central i .

Note-se que no caso do acoplamento SIMPLE despreza-se o termo $\sum_{k=1}^{nv} \{A_{ik}(w_k')\}$, que

é de maior magnitude. Assim,

$$w_i' = - \left(\frac{\Delta \mathcal{G}_i}{A_i^p - \sum_{k=1}^{nv} A_{ik}} \right) (\nabla p')_w^i \quad (3.114)$$

analogamente para j , tem-se

$$w'_j = - \left(\frac{\Delta \vartheta_j}{A_j^p - \sum_{k=1}^{nv} A_{jk}} \right) (\nabla p')_w^j \quad (3.115)$$

No método SIMPLE, W_{ij} é calculado por uma média entre w_i e w_j , e no SIMPLEC estende-se esta média para os valores das correções de velocidade.

Assim, fazendo-se a média a partir das Eqs. (3.114) e (3.115), tem-se

$$W'_{ij} = -d_{ij}^w (\nabla p)_w^{ij} \quad (3.116)$$

onde

$$d_{ij}^w = \frac{1}{2} \left[\frac{\Delta \vartheta_i}{A_i^p - \sum_{k=1}^{nv} A_{ik}} + \frac{\Delta \vartheta_j}{A_j^p - \sum_{k=1}^{nv} A_{jk}} \right] \quad (3.117)$$

Novamente $(\nabla p)_w^{ij}$ é avaliado da forma mais consistente possível, conforme Eq. (3.96).

Como pode-se notar a única diferença entre o método SIMPLE e o método

SIMPLEC é o denominador $A_i^p - \sum_{k=1}^{nv} A_{ik}$, ao invés de A_i^p . Este denominador de menor

magnitude no SIMPLEC faz com que d_{ij}^w tenha um valor de maior magnitude no decorrer das iterações. Isto conduz a correções de pressão p_i' menores, tornando a convergência mais estável. Uma discussão mais detalhada sobre o método SIMPLEC e o SIMPLE e porque utilizar-se um ou outro pode ser visto no Apêndice A deste trabalho.

Para que W_{ij} realmente satisfaça a conservação da massa substitui-se a Eq. (3.107), ou a Eq. (3.116), na Eq. (3.101) e em seguida substitui-se W_{ij} corrigido na equação da conservação da massa (3.99), de onde calcula-se um campo de pressão p_i' correto:

$$\sum_{j=1}^{nv} \left[\rho \left(W_{ij}^* - d_{ij}^w \left(\frac{p_j' - p_i'}{L_{ij}} \right) \right) S_{ij} \right] = 0 \quad (3.118)$$

$$\sum_{j=1}^{nv} \left[\rho_{ij} W_{ij}^* S_{ij} - \frac{\rho_{ij} S_{ij} d_{ij}^w}{L_{ij}} (p_j' - p_i') \right] = 0 \quad (3.119)$$

Rearranjando os termos, tem-se

$$a_i^p p_i' = \sum_{j=i}^{nv} a_{ij} p_j' - \sum_{j=i}^{nv} \rho_{ij} W_{ij}^* S_{ij} \quad (3.120)$$

onde

$$a_{ij} = \frac{\rho_{ij} S_{ij} d_{ij}^w}{L_{ij}} \quad (3.121)$$

$$a_i^p = \sum_{j=i}^{nv} a_{ij} \quad (3.122)$$

Obtidos os valores de p' , através da resolução do sistema linear dado pela Eq. (3.120) pode-se corrigir p^* , através da Eq. (3.103), e obter-se o campo correto de pressão p , que satisfaz a equação da conservação da massa. Então, toma-se o p correto como o p^* do próximo nível iterativo.

3.6 - Algoritmo iterativo

Na Fig. (3.7) é apresentado o fluxograma das funções implementadas para resolução do problema proposto, definindo os passos do algoritmo utilizado. Maiores informações poderão ser obtidas no Apêndice B no final deste trabalho, contendo os fontes do programa implementado em linguagem C.

Como pode ser visto na Fig. (3.7), o programa foi desenvolvido de uma forma estruturada, a princípio, sem preocupações quanto a se utilizar a orientação a

objetos. O programa preocupa-se, essencialmente, em resolver as equações de Navier-Stokes, sem a utilização da parte chamada em informática de .perfumaria., ou seja, telas e outras tantas. Os fontes do programa podem ser vistos no Apêndice B.

Para resolver-se o sistema foi usado, conforme já mencionado, o método de Gauss-Seidel, por se ter sempre um sistema com matriz de diagonal dominante e irredutível, que tem convergência garantida, por estes motivos não foram usados métodos diretos de resolução, mesmo para malha quadrada estruturada, uma vez que tinham-se poucos pontos nodais nesta malha, apenas 100 e não compensaria implementar um método específico de resolução.

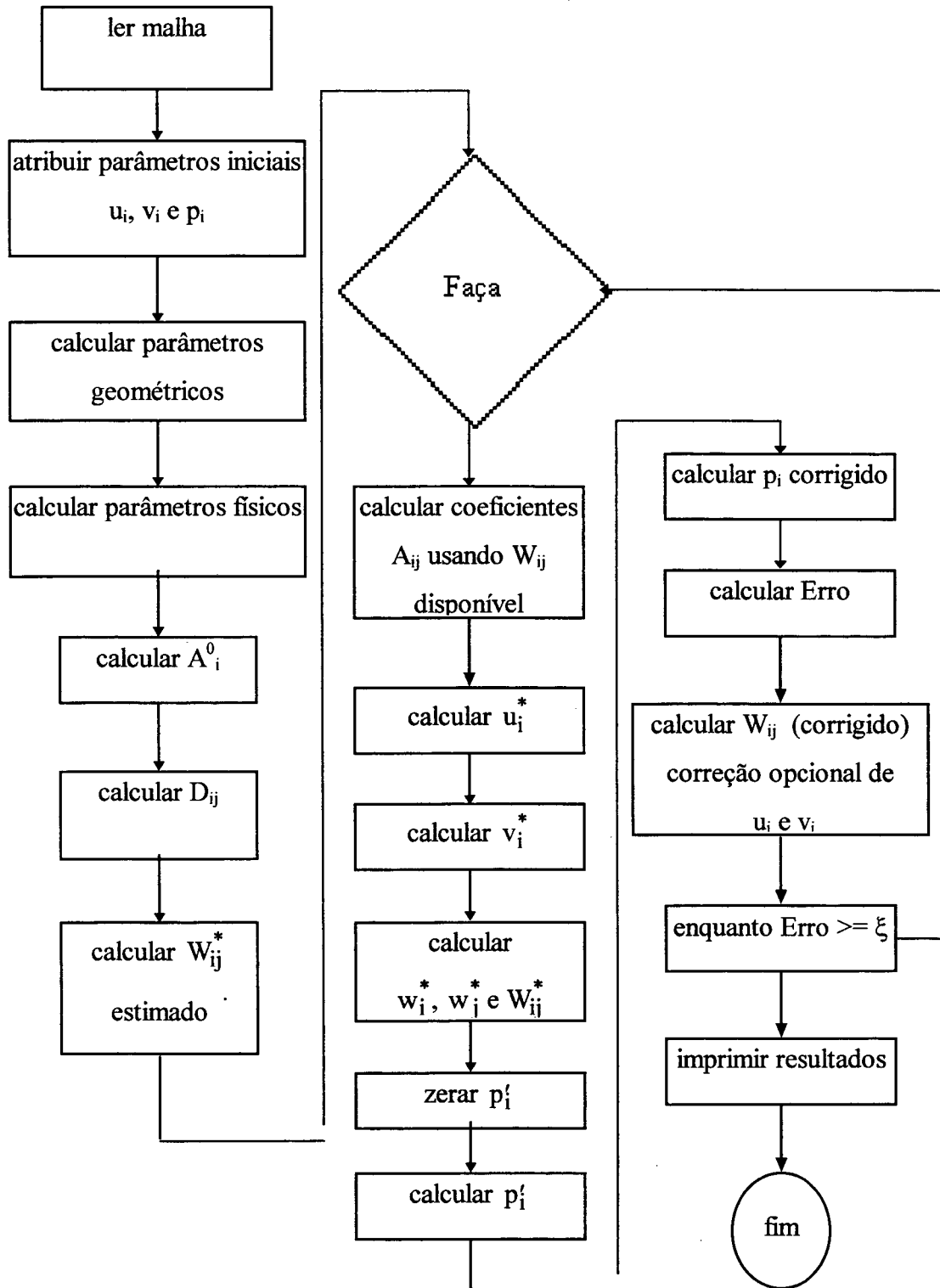


Figura 3.7 - Algoritmo implementado.

O método de Gauss-Seidel foi implementado usando-se a ordem de visitação natural dos índices dos pontos nodais, ou seja, primeiro encontra-se a solução para o ponto nodal número 1, em seguida para o número 2, e assim sucessivamente até o número total de pontos nodais. Isto é feito sequencialmente para u , v e p' com 10 iterações internas para cada variável, uma iteração completa corresponde ao ciclo de 10 iterações para u , v e p' . Para facilitar o processo de convergência dos sistemas de equações linearizadas foram usados fatores de subrelaxação, que foram implementados nas próprias equações do sistema (Patankar, 1980), usando-se $\alpha_u = 0,3$, para as equações correspondentes a velocidade u e a velocidade v , $\alpha_p = 0,5$, para as equações da correção de pressão, e $\alpha_p = 0,5$, para a atualização da pressão corrigida, como segue:

$$A_i^p \phi_i = \sum_{j=1}^{nv} \{A_{ij} \phi_j\} + b_i^p - \Delta \theta_i \nabla p + A_i^p (1 - \alpha) \phi_i \quad (3.123)$$

onde A_i^p é dado por

$$A_i^p = \frac{\sum_{j=1}^{nv} (A_{ij} + A_i^0)}{\alpha_{u,v}} \quad (3.124)$$

onde ∇p pode ser u ou v e ∇p pode ser p_x e p_y , respectivamente, para u e v o sistema de equações para p' foi resolvido pela Eq. (3.125), com aplicação direta do fator de subrelaxação $\alpha p'$

$$a_i^p p_i' = \left[\sum_{j=i}^{nv} (a_{ij} p_j') - \sum_{j=i}^{nv} (\rho_{ij} W_{ij}^* S) \right] \alpha p_i' + p_i' (1 - \alpha p') \quad (3.125)$$

e a pressão corrigida é obtida por

$$p_i = p_i + p' \alpha p \quad (3.126)$$

No próximo capítulo serão apresentados os resultados obtidos e os testes feitos para resolução das Equações de Navier-Stokes.

4. Resultados e Discussões

4.1 Introdução

Na validação das estruturas implementadas para solução das equações de Navier-Stokes é importante usar-se modelos já testados para corroboração dos resultados obtidos, por este motivo resolveu-se adotar o cálculo do escoamento laminar incompressível em uma cavidade quadrada de profundidade infinita, cuja parede superior se move a uma velocidade constante (*shear-driven cavity flow*), e utilizar-se do artigo de Ghia *et alli* (1982) para comparação dos resultados obtidos. Para uma melhor compreensão a cavidade pode ser vista na Fig. (4.1).

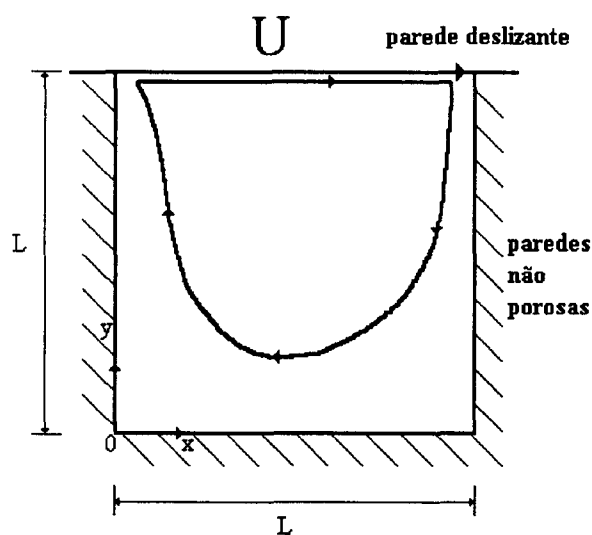


Figura 4.1 - Parede superior deslizante sobre cavidade quadrada.

Este problema da cavidade quadrada, sujeita a uma parede superior deslizando modela matematicamente, por exemplo, uma esteira deslizando constantemente sobre uma cavidade quadrada com um fluido dentro, o qual pode ser óleo, água, ar, etc. Sendo que a parede inferior e as paredes laterais não permitem entrada ou saída de massa do interior da cavidade, ou seja, são não-porosas.

4.2 Formulação do problema

O problema proposto, conforme já foi mencionado, baseia-se na resolução de um sistema de equações diferenciais parciais não-lineares, que simula o escoamento na cavidade quadrada citada, cujo fluido é newtoniano, incompressível, com escoamento laminar. As equações governantes são as seguintes:

$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u u)}{\partial x} + \frac{\partial(\rho v u)}{\partial y} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x} \left(\frac{\mu \partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(\frac{\mu \partial u}{\partial y} \right) \quad (4.1)$$

$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial(\rho u v)}{\partial x} + \frac{\partial(\rho v v)}{\partial y} = -\frac{\partial p}{\partial y} + \frac{\partial}{\partial x} \left(\frac{\mu \partial v}{\partial x} \right) + \frac{\partial}{\partial y} \left(\frac{\mu \partial v}{\partial y} \right) \quad (4.2)$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (4.3)$$

onde p é o campo de pressão do escoamento, u é a componente de velocidade na direção x , v é a componente de velocidade na direção y , x é a abscissa e y é a ordenada. O escoamento na cavidade quadrada está sujeito as condições de contorno apresentadas na Tab. (4.1).

Tabela 4.1 - Condições de contorno.

Velocidade u	Velocidade v
$u(x, y=L)=U$	$v(x, y=L)=0$
$u(x, y=0)=0$	$v(x, y=0)=0$
$u(x=0, y)=0$	$v(x=0, y)=0$
$u(x=L, y)=0$	$v(x=L, y)=0$

Atribuiu-se as variáveis u, v e p as seguintes condições iniciais: $u=v=p=0$.

Os testes do presente trabalho foram efetuados simulando regime estacionário de escoamento, embora o programa implementado possibilite a discretização temporal.

Foi adotado $L=10$, $U=1$, $\rho=1$ e μ variando conforme o número de Reynolds desejado.

Para avaliar os resultados numéricos, realizados com o programa implementado, adotou-se o seguinte critério de convergência

$$E_r^{(n+1)} = \max \left(\sum_{j=1}^{nv} F_{ij} \right) < \xi \quad (4.4)$$

onde $E_r^{(n+1)}$ é o resíduo absoluto da equação da conservação da massa na iteração atual e ξ é a precisão desejada, tendo sido adotado $\xi=10^{-4}$. Utilizou-se o método de Gauss-Seidel (Claudio, 1989) com subrelaxação para resolver o sistema de equações algébricas lineares gerados na discretização numérica. Adotou-se, os seguintes critérios para repetição do laço interno na resolução iterativa do sistema de equações pelo método de Gauss-Seidel com sub-relaxação, 20 repetições para o laço da pressão e 10 repetições para os laços u e v, respectivamente. O acoplamento pressão-velocidade adotado foi o SIMPLEC, pois, a princípio, acreditava-se que apresentaria uma melhor convergência. Foram testados números de Reynolds de 100, 400 e 1000, calculados como segue:

$$Re = \rho \frac{U.L}{\mu} \quad (4.5)$$

sendo U a velocidade de movimento da parede superior, L a largura da cavidade e, ρ é a massa específica do fluido, μ é a viscosidade dinâmica do fluido variando conforme o número de Reynolds desejado.

Adotou-se resolver o sistema por Gauss-Seidel com sub-relaxação, aplicada a cada variável, embora este não seja um método de convergência rápida, mas por motivos de praticidade na resolução do sistema, uma vez que já se tem uma boa experiência com este método e por este não ser o objetivo principal deste trabalho. Além do mais, outros métodos poderão ser encontrados na dissertação de mestrado de Viviana Cocco Mariani (1997).

4.3 Malhas utilizadas

Vários foram os testes realizados para verificar qual tipo de malha melhor se adapta para resolução do problema da cavidade quadrada, bem como qual a melhor forma de calcular o gradiente de pressão nas equações de Navier-Stokes. Para tal, foram realizados testes com malhas estruturadas uniforme (gerados por DV, também), Fig. (4.2), malhas com volumes tipo colméia, Fig. (4.3a) e (4.3b), e malha com pontos nodais aleatórios, Fig. (4.4), todas geradas a partir do gerador de DV de Maliska Jr. (1993). Também foram testados os gradientes de pressão propostos por Taniguchi *et alli* (1991a e 1991b) e uma variação do gradiente de pressão apresentado por Maliska (1995).

As condições de contorno foram implementadas no programa computacional, através da atribuição de um índice nodal específico para cada parede, onde cada ponto nodal correspondente aos pontos da fronteira foram conectados por laço de vizinhança com o respectivo índice do contorno.

10	20	30	40	50	60	70	80	90	100
9	19	29	39	49	59	69	79	89	99
8	18	28	38	48	58	68	78	88	98
7	17	27	37	47	57	67	77	87	97
6	16	26	36	46	56	66	76	86	96
5	15	25	35	45	55	65	75	85	95
4	14	24	34	44	54	64	74	84	94
3	13	23	33	43	53	63	73	83	93
2	12	22	32	42	52	62	72	82	92
1	11	21	31	41	51	61	71	81	91

Figura 4.2 - Malha estruturada retangular uniforme gerada por DV (100 pontos nodais).

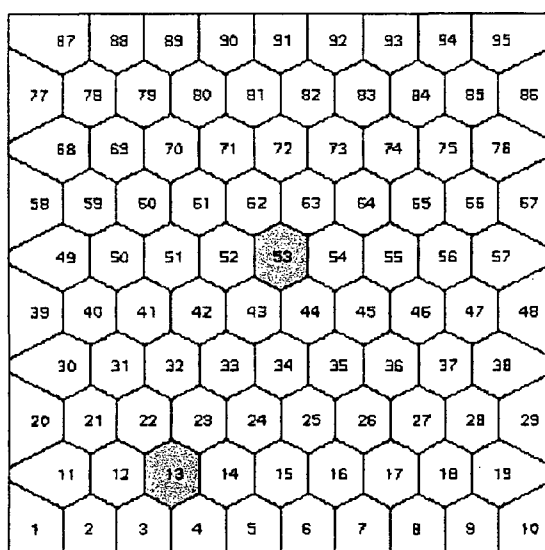


Figura 4.3a - Malha de volumes hexagonais (95 pontos nodais).

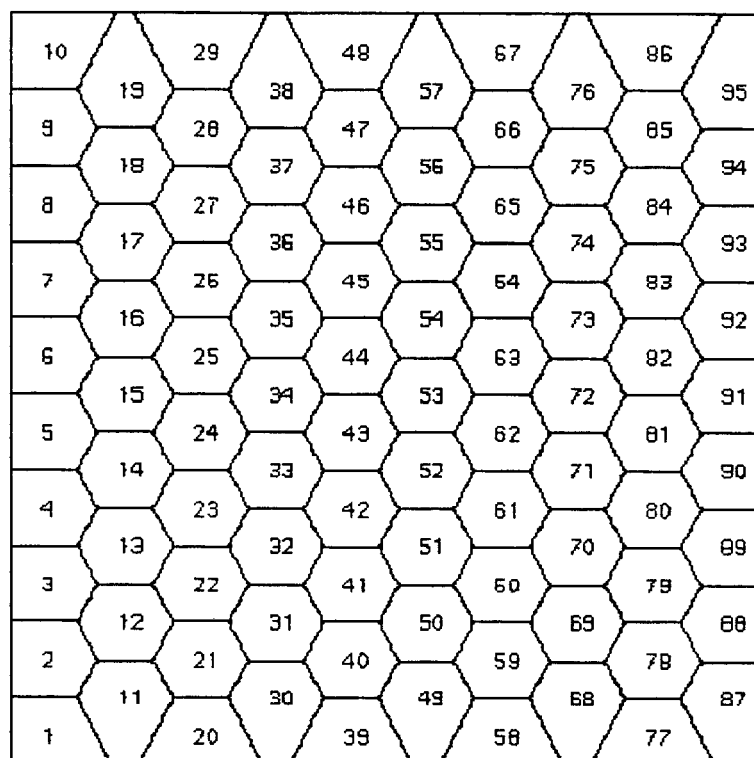


Figura 4.3b - Malha de volumes hexagonais (95 pontos nodais).

As malhas das Figs. (4.3a) e (4.3b) foram geradas, respectivamente, para comparação, com o artigo de Ghia *et alli* (1982), em que a velocidade u é plotada em $x=5$ para vários y da cavidade, analogamente plota-se v em $y=5$ variando o x .

A malhas utilizadas foram geradas pelo programa gerador de Clóvis Raimundo Maliska Jr., conforme já citado. Tal programa lia os pontos nodais geradores da malha de um arquivo previamente gerado, manualmente, e, então, criava as associações entre os pontos nodais, determinando seus vértices e vizinhos. Para o contorno, porém, o programa permitia apenas a utilização de um único vizinho para todos os pontos nodais ligados a uma parede, o qual foi bastante útil para a resolução do sistema proposto. Convém salientar, no entanto, que as condições de contorno poderiam ser inseridas nas próprias equações discretizadas, apenas isto não foi feito pois não se necessitava de mais de uma condição por lado da malha.

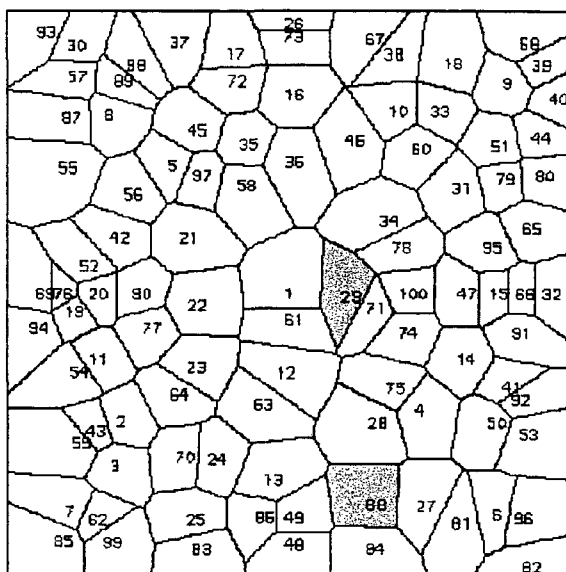


Figura 4.4 - Malha de pontos aleatórios (100 pontos nodais).

O objetivo de se gerar estes vários tipos de malha é o de comparar o desempenho do programa, quanto ao tempo de execução, bem como verificar qual é a melhor forma de cálculo dos gradientes de pressão. Convém salientar, que a forma de resolução das equações, nestes casos, foi a mesma adotada para todos os casos, pois as malhas são todas não-estruturadas e com todas as variáveis armazenadas no ponto nodal central (malha colocada).

4.4 Resultados obtidos

A seguir apresenta-se os resultados obtidos para u e v com malha estruturada de 100 pontos nodais, Fig. (4.2), utilizando duas formas de cálculo: caso 1 da média ponderada entre os gradientes normais projetados e o caso 2 da avaliação dos gradientes buscando o mínimo resíduo quadrático.

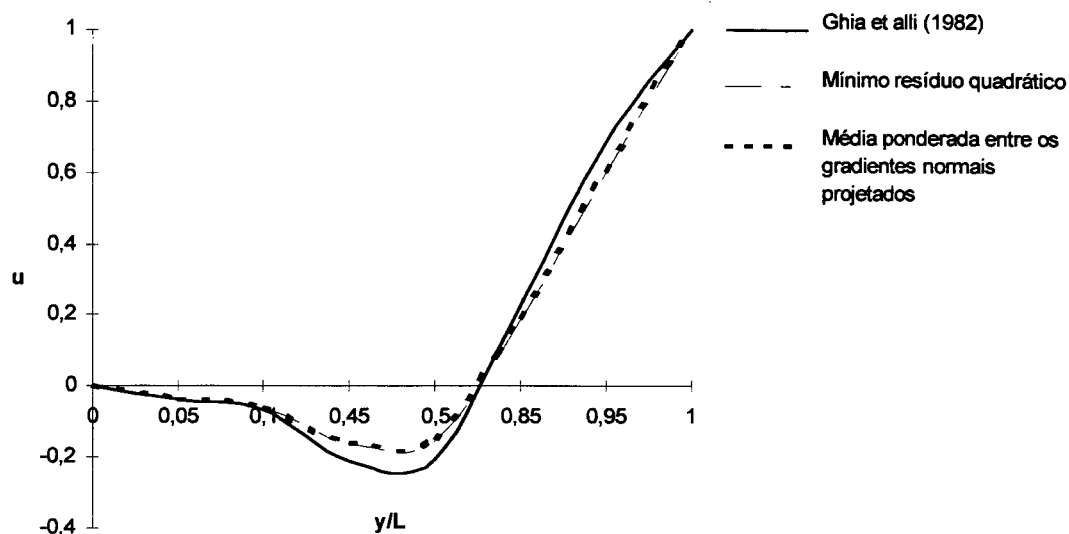


Figura 4.5 - Componente horizontal de velocidade em $x=5$, $Re=100$ (malha retangular com 100 pontos nodais).

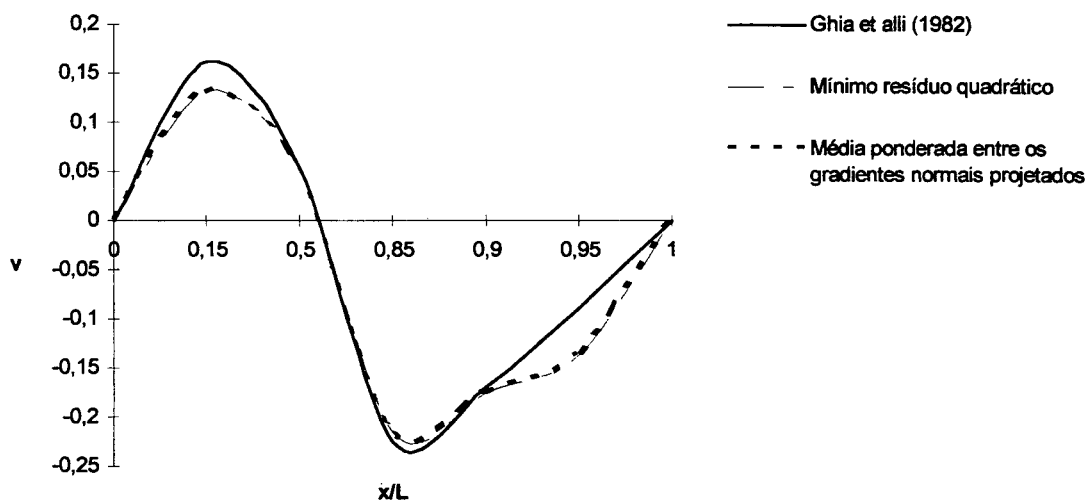


Figura 4.6 - Componente vertical de velocidade em $y=5$, $Re=100$ (malha retangular com 100 pontos nodais).

Neste teste, em malha retangular, as formas de avaliação do gradiente de pressão foram idênticas, tanto como o calculado segundo Taniguchi *et alli* (1991a e

1991b), pelo mínimo resíduo quadrático, como com o gradiente de pressão proposto neste trabalho, pela média ponderada entre os gradientes normais projetados, geraram curvas sobrepostas, vide Figs. (4.5) e (4.6). Também pode-se notar que a aproximação do resultado obtido com o de Ghia *et alli* (1982) não é das melhores. Porém, isto já era previsto devido ao pequeno número de pontos da malha. Portanto, a avaliação do gradiente de pressão nos dois casos testados foi equivalente com o uso da malha estruturada, Fig. (4.2). Isto se deve a uniformidade da malha, em que os cálculos de cada componente do gradiente de pressão sofrem apenas as respectivas influências unidirecionais da pressão envolvida.

Também é interessante comparar os tempos de execução do programa usando-se o mínimo resíduo quadrático e a média ponderada entre os gradientes normais projetados, cujos resultados obtidos são apresentados na Tab. (4.2). Nesta tabela apresentam-se as médias de tempo de CPU avaliadas entre vários casos testados em uma estação SPARC 10 da SUN com utilização de memória compartilhada.

Tabela 4.2 - Avaliação do tempo de CPU (Estação SPARC 10 da SUN).

	mínimo resíduo quadrático	média ponderada entre os gradientes normais projetados
Intervalo de tempo médio	5,53s	5,05s
Número de iterações	101	101

Observa-se da Tab. (4.2), que a diferença de tempo entre uma e outra forma de calcular os gradientes não é muito sensível, neste caso é mais interessante usar o mínimo resíduo quadrático, por ser um cálculo mais consistente, visto que toma todas as influências dos pontos nodais vizinhos para cálculo do gradiente de pressão. Resta conferir se este método apresenta os mesmos resultados em outros tipos de malhas. Para tal, testou-se ambos os métodos para malha com 95 pontos e volumes hexagonais, Figs.

(4.3a) e (4.3b), e para malha com 100 pontos aleatórios, Fig. (4.4), onde foram obtidos os seguinte resultados:

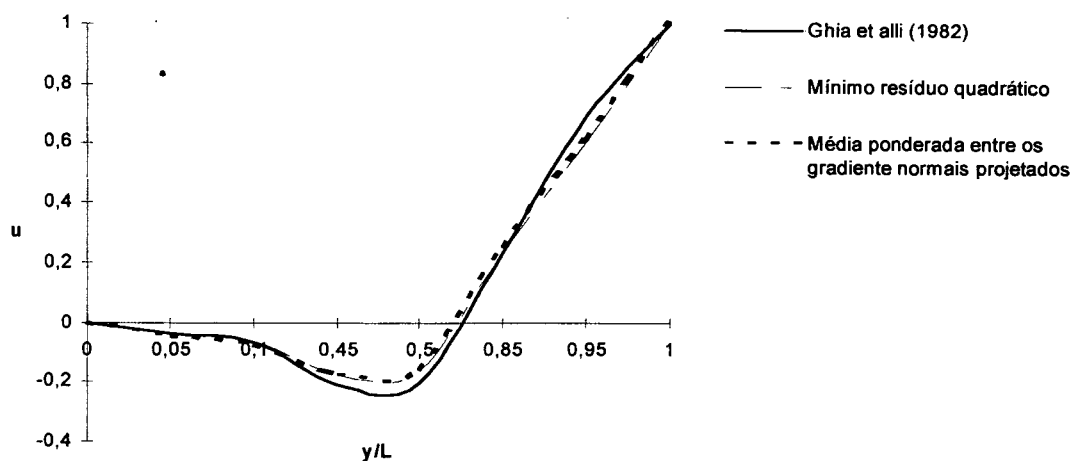


Figura 4.7 - Componente horizontal de velocidade em $x=5$, $Re=100$ (malha de volumes hexagonais com 95 pontos nodais, Fig. (4.3a)).

Os resultados mostrados na Fig. (4.7) foram calculados usando-se a malha apresentada na Fig. (4.3a), tendo sido alcançados em 233 iterações para o mínimo resíduo quadrático e 383 iterações para a média ponderada entre os gradientes normais projetados.

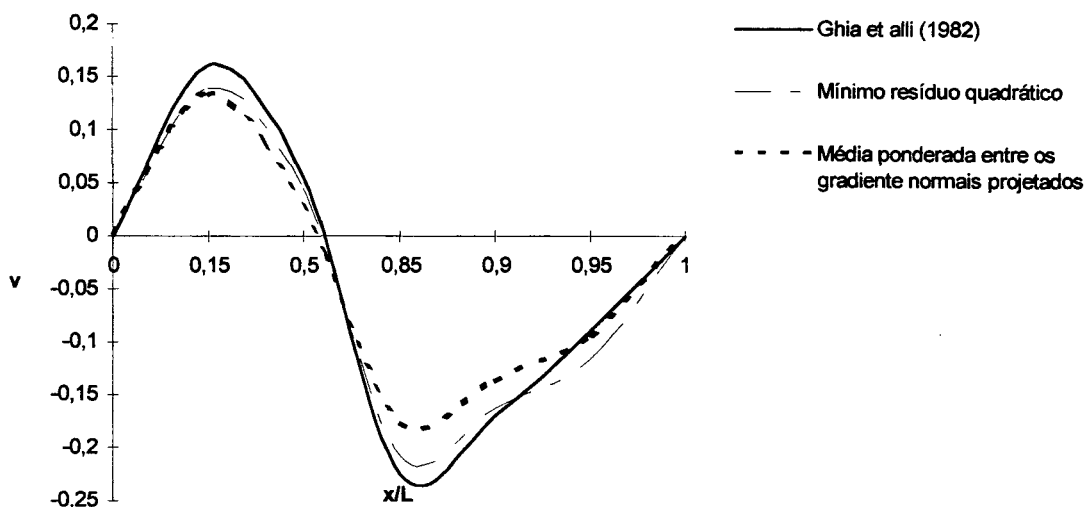


Figura 4.8 - Componente vertical de velocidade em $y=5$, $Re=100$ (malha de volumes hexagonais com 95 pontos nodais, Fig. (4.3b)).

O resultado mostrado na Fig. (4.8) foi calculado usando-se a malha apresentada na Fig. (4.3b), tendo sido alcançado em 239 iterações para o mínimo resíduo quadrático e 253 iterações para a média ponderada entre os gradientes normais projetados.

Portanto, o cálculo do gradiente de pressão através do mínimo resíduo quadrático foi mais eficiente quanto ao número de iterações necessárias, embora o tempo de CPU seja ligeiramente maior, além de ser mais próximo do resultado padrão de Ghia et alii (1982).

A seguir apresentam-se os resultados com a malha aleatória, Fig. (4.4), com 100 pontos nodais.

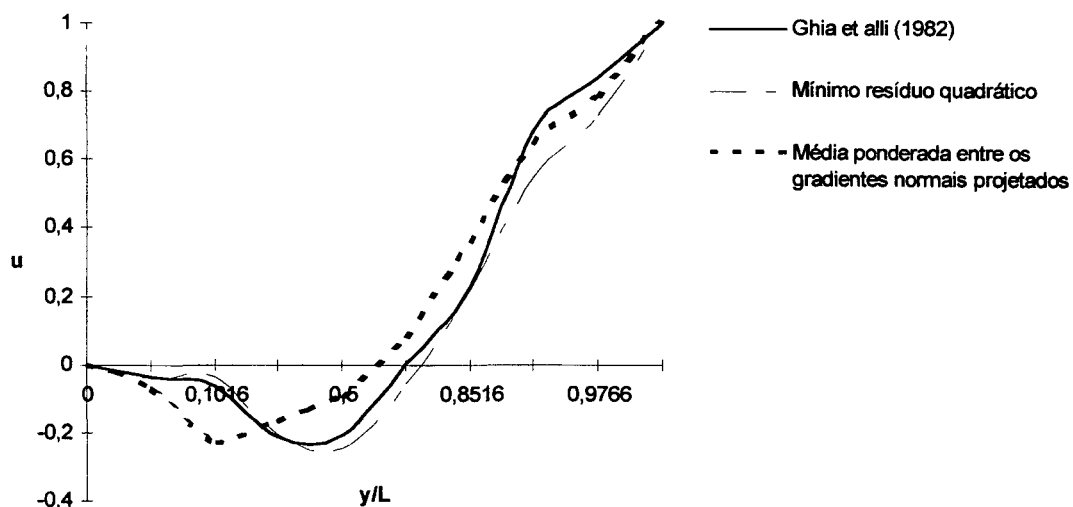


Figura 4.9 - Componente horizontal de velocidade em $x=5$, $Re=100$ (malha aleatória com 100 pontos nodais).

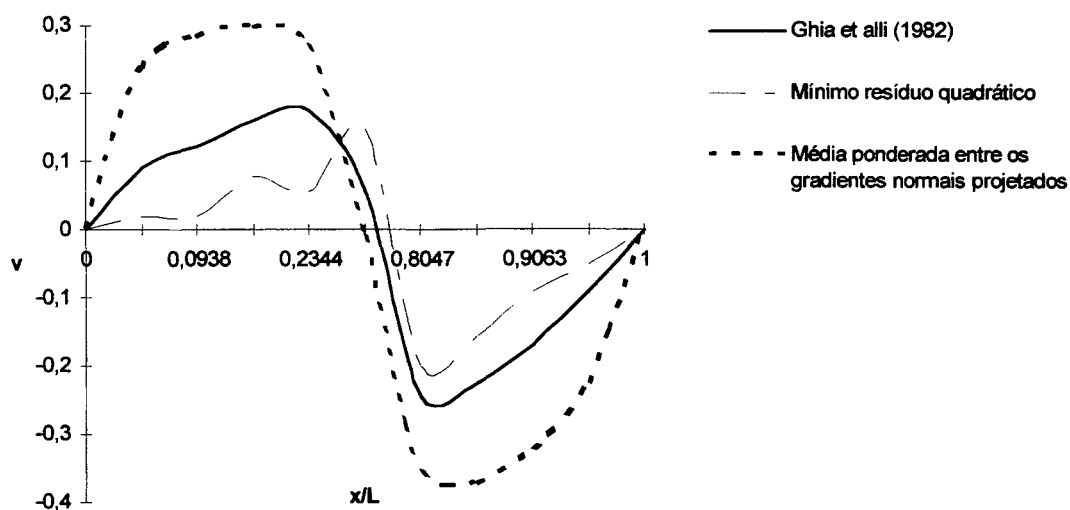


Figura 4.10 - Componente vertical de velocidade em $y=5$, $Re=100$ (malha aleatória com 100 pontos nodais).

Os resultados apresentados para malha aleatória nas Figs. (4.9) e (4.10) foram alcançados em 265 iterações para o mínimo resíduo quadrático e

1931 iterações para a média ponderada entre os gradientes normais projetados. Tais resultados não são conclusivos, mas os resultados obtidos com mínimo resíduo quadrático foram mais próximos do resultado padrão de Ghia et alli (1982), embora os resultados obtidos com malha hexagonal, Figs. (4.7) e (4.8) tenham sido superiores aos da malha aleatória utilizada, Figs. (4.9) e (4.10).

Os resultados apresentados na Fig. (4.10) sugerem que se investigue sobre tamanha diferença entre os resultados obtidos pelos dois gradientes de pressão em discussão, para tal a Tab. (4.3) faz uma comparação entre os resultados dos gradientes de pressão obtidos ao final da última iteração para os pontos nodais 29 e 88, apresentados na Fig. (4.4), sendo o primeiro VC bastante irregular e o segundo mais regular.

Tabela 4.3 - Comparação entre os gradientes de pressão (malha aleatória com 100 pontos nodais).

Método de avaliação de ∇p Gradientes de pressão	Mínimo resíduo quadrático	Média ponderada entre os gradientes normais projetados
$\Delta p_{x_{29}}$	-0.007322	0.004558
$\Delta p_{y_{29}}$	0.004256	-0.028159
$\Delta p_{x_{88}}$	0.016569	0.008321
$\Delta p_{y_{88}}$	0.012690	-0.006068

Verifica-se da Tab. (4.3), que as avaliações locais do gradiente de pressão pelos métodos apresentados são bastante diferentes entre si, na malha do tipo aleatória. Provavelmente, em malhas mais refinadas estas avaliações seriam mais próximas.

Uma outra avaliação sobre o cálculo do gradiente de pressão pode ser vista na Tab. (4.4) para malha hexagonal de 95 pontos nodais, Fig. (4.3a), avaliando o escoamento à $Re=100$, nos pontos 13 e 53. Tal avaliação corresponde aos resultados apresentados na Fig. (4.7).

Tabela 4.4 - Comparação entre os gradientes de pressão (malha hexagonal com 95 pontos nodais).

Método de avaliação de ∇p Gradientes de pressão	Mínimo resíduo quadrático	Média ponderada entre os gradientes normais projetados
$\Delta p_{x_{13}}$	0.000968	0.000850
$\Delta p_{y_{13}}$	-0.000496	-0.000718
$\Delta p_{x_{53}}$	-0.000775	-0.002570
$\Delta p_{y_{53}}$	-0.018374	-0.019159

Como pode ser visto na Tab. (4.4), as avaliações do gradiente de pressão com as duas formas de cálculo apresentadas neste trabalho, estão bem mais próximo entre si, do que no caso da malha aleatória, apresentada na Tab. (4.3). Convém lembrar que outros erros, associados a forma de discretização produzida pelas duas malhas analisadas, influem diretamente nestas comparações entre as duas formas de avaliação do gradiente de pressão.

Devido aos resultados apresentados até o momento optou-se por realizar testes em uma malha com volumes hexagonais, usando o mínimo resíduo quadrático, por terem sido obtidos os melhores resultados com esta combinação de tipo de malha e tipo de gradiente de pressão utilizado. Para confirmar tal escolha foram realizados mais algumas avaliações em uma malha mais refinada com 613 pontos nodais, cujos resultados são apresentados nas Figs. (4.11) e (4.12) para $Re=100$, e nas Figs. (4.13) e (4.14) para $Re=400$. No entanto, convém salientar que estes resultados não são conclusivos e determinantes de que a malha hexagonal seja melhor que a malha de pontos aleatórios ou a malha estruturada, nem de que o gradiente de pressão proposto por Taniguchi et ali (1991a e 1991b) seja melhor do que o proposto neste trabalho. Há a necessidade de investigações mais profundas para se chegar a resultados conclusivos.

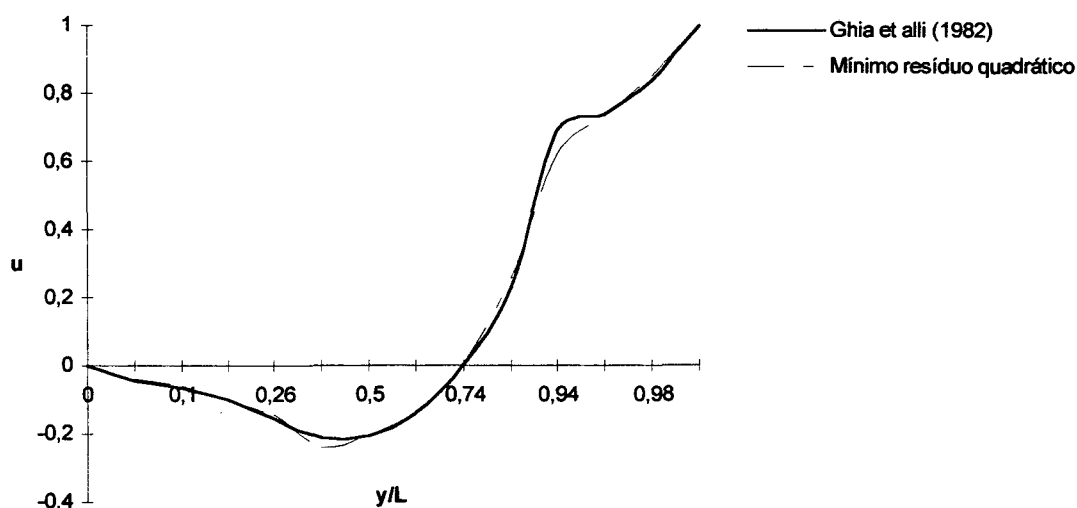


Figura 4.11 - Componente horizontal de velocidade em $x=5$, $Re=100$ (malha de volumes hexagonais com 613 pontos nodais).

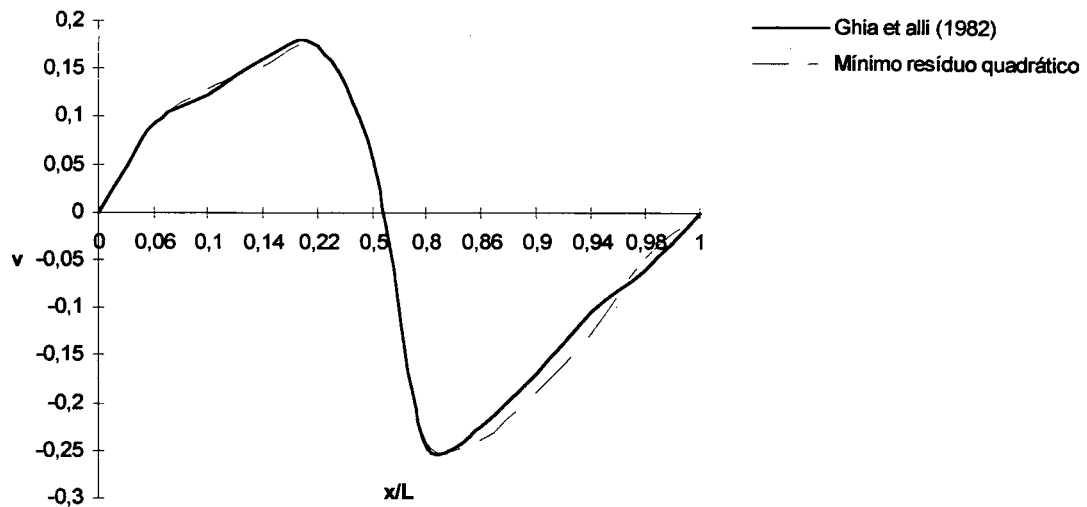


Figura 4.12 - Componente vertical de velocidade em $y=5$, $Re=100$ (malha de volumes hexagonais com 613 pontos nodais).

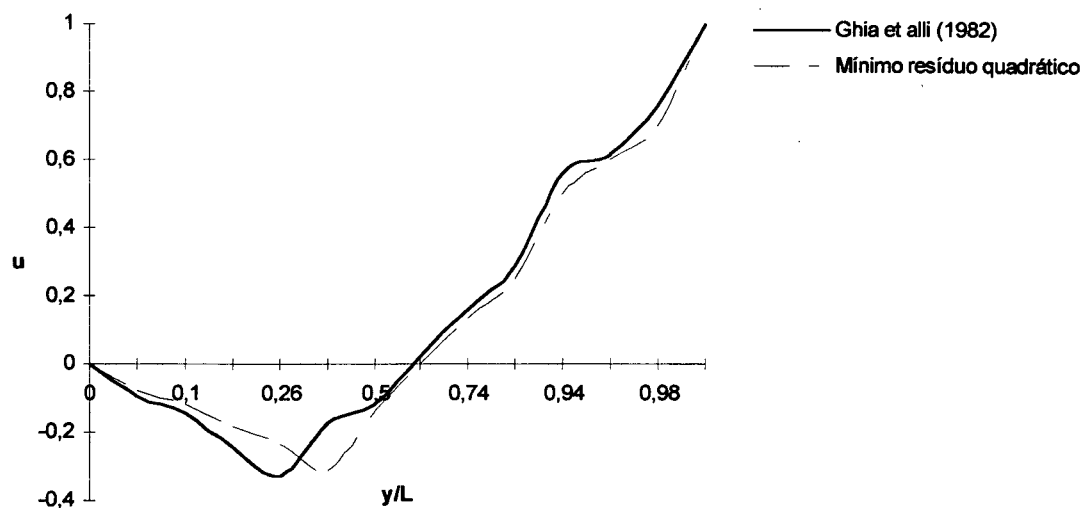


Figura 4.13 - Componente horizontal de velocidade em $x=5$, $Re=400$ (malha de volumes hexagonais com 613 pontos nodais).

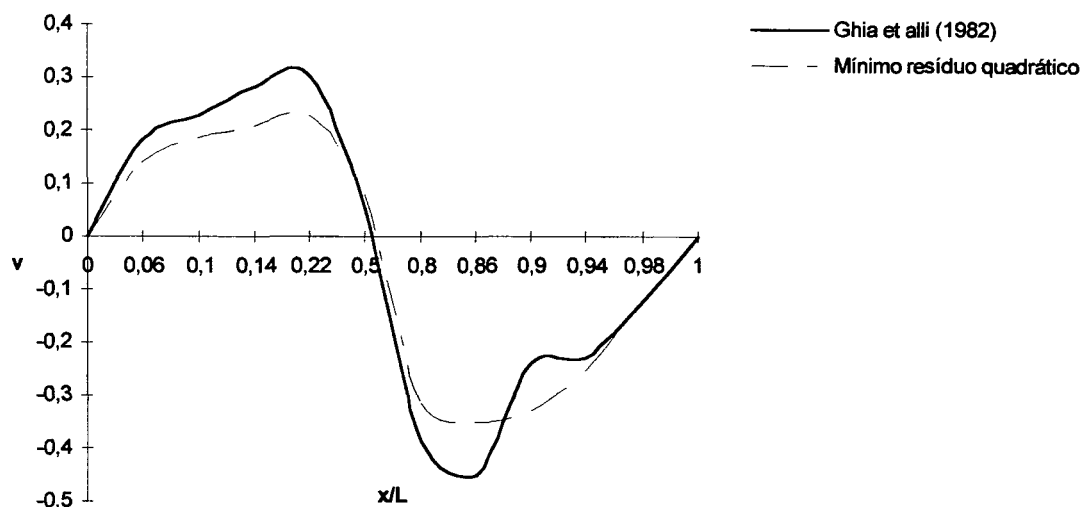


Figura 4.14 - Componente vertical de velocidade em $y=5$, $Re=400$ (malha de volumes hexagonais com 613 pontos nodais).

Pode observar-se pelas Figs. (4.13) e (4.14) que os resultados obtidos estão um pouco aquém do esperado para $Re=400$, isto foi atribuído à malha com 613 pontos nodais considerada, por ainda não ser suficientemente refinada para captar corretamente o escoamento dentro da cavidade, mas espera-se que uma malha com mais pontos consiga uma aproximação melhor do resultado. Além disso é importante salientar que a malha utilizada por Ghia *et alii* (1982) foi gerada com 16641 pontos.

Também foi testado, como última análise, o desempenho do acoplamento Simple entre pressão e velocidade. Como foi visto, até aqui usou-se o SIMPLEC, por acreditar-se ser o melhor entre os dois, no entanto não foi isto o que ocorreu, isto foi verificado para $Re=400$ e $Re=1000$, conforme pode ser visto nas Figs. (4.15) a (4.17), com a malha de 613 pontos nodais.

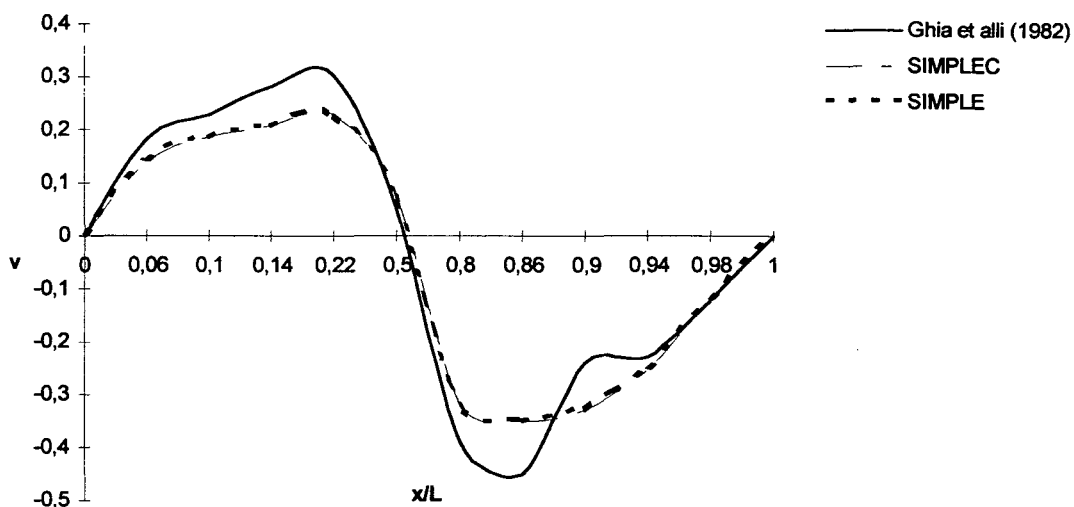


Figura 4.15 - Componente vertical de velocidade em $y=5$, $Re=400$ (malha de volumes hexagonais com 613 pontos nodais).

Como era naturalmente de se esperar, as soluções convergidas com SIMPLE geraram os mesmos resultados obtidos com SIMPLEC, mas diferiram no número de iterações, uma vez que o SIMPLE foi mais rápido, pois resolveu o sistema em 688 iterações e o SIMPLEC em 965 iterações, tendo o SIMPLE resolvido o problema em 29% menos iterações que o SIMPLEC.

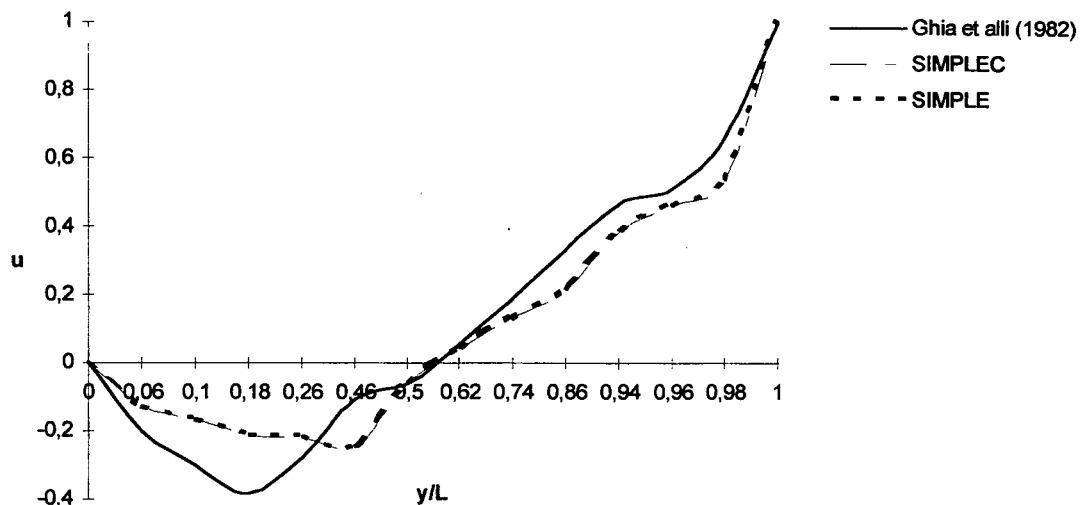


Figura 4.16 - Componente horizontal de velocidade em $x=5$, $Re=1000$ (malha de volumes hexagonais com 613 pontos nodais).

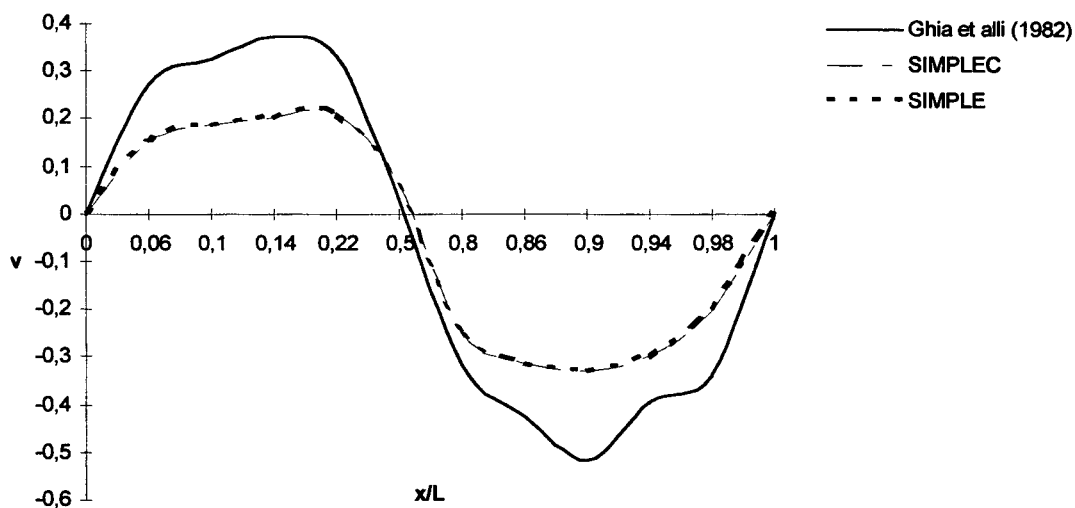


Figura 4.17 - Componente vertical de velocidade em $y=5$, $Re=1000$ (malha de volumes hexagonais com 613 pontos nodais).

Nas Figs. (4.16) e (4.17) é feita uma apresentação dos resultados para $Re=1000$, onde o acoplamento SIMPLE mostrou-se novamente mais rápido que o

acoplamento SIMPLEC, obtendo-se os mesmos resultados em 1153 iterações, enquanto o último realizou-os em 1620 iterações, mantendo o mesmo percentual de 29% menos iterações. Mais uma vez, convém observar que para $Re=1000$ se necessitaria de uma malha com bem mais pontos nodais do que a utilizada para se obter uma aproximação melhor dos resultados obtidos por Ghia *et alli* (1982). As Figs. (4.15) a (4.17) foram geradas com o objetivo de verificação da implementação do programa computacional, para testar o desempenho das formas de acoplamento pressão-velocidades.

A seguir, apresenta-se a Tab. (4.5) para uma melhor comparação entre as várias formas de avaliação do gradiente de pressão e de malhas utilizadas.

Tabela 4.5- Comparação entre o tipo de malha e o cálculo do gradiente através dos resultados obtidos neste trabalho.

Método de Resolução Tipo de malha	Média ponderada entre os gradientes normais projetados (MPGNP)	Mínimo resíduo quadrático
Malha estruturada	Resultados idênticos ao mínimo resíduo quadrático. Processo mais rápido.	Resultados idênticos a MPGNP. Processo mais lento.
Malha aleatória	Resultado ruim, atribuído ao fato de não serem consideradas todas as influências dos vizinhos para cálculo do gradiente	Obteve um resultado razoável, mas inferior aos obtidos com malha hexagonal
Malha hexagonal	Não foi utilizado	Obteve bons resultados para $Re=400$, para $Re=1000$ são necessárias malhas mais refinadas

Com base na Tab. (4.5) podem-se tirar as seguintes conclusões: de todos os casos estudados a melhor combinação de implementações foi a utilização de uma malha com distribuição hexagonal de pontos nodais usando o cálculo do gradiente de pressão com o mínimo resíduo quadrático, proposto por Taniguchi (1991), e além disso, verificou-se um melhor desempenho do acoplamento SIMPLE entre pressão e velocidade. Embora, conforme já mencionado, estas afirmações não sejam finais ou conclusivas, em virtude da pequena gama de testes efetuados.

É importante salientar, que a presente proposta de avaliação do gradiente de pressão com a média ponderada entre os gradientes normais projetados apresentou bons resultados para malhas estruturadas de 100 pontos nodais internos, espera-se que esta avaliação apresente melhores resultados com malhas mais refinadas.

No próximo capítulo são apresentadas as conclusões obtidas do presente trabalho.

5. Conclusões

Neste trabalho foi proposto se resolver as equações de Navier-Stokes em um novo modelo de malha não-estruturada, que é a malha gerada por DV.

No presente trabalho, é abordado o problema do escoamento confinado em uma cavidade quadrada sujeito ao movimento de uma parede deslizante. Tal escoamento é alvo de constantes estudos e é muito utilizado por pesquisadores como padrão de comparação para aferição de seus códigos computacionais.

A resolução das equações de Navier-Stokes é feita em malha não-estruturada gerada por DV usando-se armazenamento colocalizado de variáveis u , v e p , cuja avaliação das velocidades nas faces dos VC, são obtidas por médias apropriadas, conforme Taniguchi et alli (1991), isto traz a vantagem de integrar-se numericamente todas as equações no VC principal.

Este trabalho de dissertação busca fornecer subsídios para que se possa continuar com a construção de códigos computacionais que façam simulação numérica de escoamentos, utilizando o Método dos Volumes Finitos com discretização em malha gerada por Diagramas de Voronoi.

5.1 Contribuições

Entre as contribuições apresentadas, até o momento, estão o desenvolvimento de um caminho para ligação entre os pontos nodais formadores da malha gerada por pontos nodais aleatórios, o que permite uma otimização da esparsidade do sistema de equações, registrada no trabalho individual (Cardoso, 1996).

Entre as contribuições apresentadas, neste trabalho podem ser citadas:

√ a comparação entre malhas estruturadas, malhas com volumes hexagonais e malhas com pontos aleatórios, tendo as malhas com volumes hexagonais ou em colméia se mostrado melhores que a malha estruturada e do que a malha de pontos aleatórios que foi utilizada, pois aproximou melhor os resultados obtidos, com o padrão apresentado por Ghia et ali (1982).

√ o melhor cálculo do gradiente de pressão, foi o proposto por Taniguchi *et ali* (1991), visto que considera todas as influências das pressões dos pontos nodais vizinhos em relação ao ponto nodal central buscando as mínimas diferenças. O gradiente proposto por Jameson e Mavriplis (1986) possui o inconveniente de necessitar da pressão sobre o contorno, que necessita ser estimado por extrapolação dos valores internos.

√ por último foi feita uma análise entre o acoplamento Simple e o acoplamento Simplec, onde foi demonstrado que o acoplamento Simple é mais rápido (29% menos iterações) para resolução do sistema proposto.

5.2. Perspectivas Futuras

Para uma evolução completa deste programa é necessária uma evolução paralela no algoritmo de geração de malha, por exemplo, para geometrias não convexas e com ilhas. O uso do gerador da malha e sua adaptação ao algoritmo proposto foi uma dificuldade bastante sentida no decorrer do trabalho. O gerador se mostrou falho na geração de malhas convexas, tipo L, por exemplo, e, na sua forma original, permite a implementação de apenas uma condição para um mesmo contorno, o que dificulta a prescrição de condições no contorno variadas, que podem ser implementadas em cada equação discretizada, num procedimento manual.

Também, sente-se a necessidade de uma pesquisa permanente em formas alternativas de resolução de sistemas de equações lineares esparsas, gerados da discretização das equações diferenciais. A resolução destes sistemas é a parte do programa computacional que consome maior tempo de CPU, e, por isso, merece um grande investimento em pesquisa. Tal estudo já está em andamento a cargo da colega Viviana Cocco Mariani (1997), a qual utilizará o programa apresentado neste trabalho para testar outros algoritmos de resolução das equações.

Com o programa computacional desenvolvido fica aberto um grande leque de trabalhos e projetos a serem realizados, entre eles novas formas de implementar o gradiente de pressão e o acoplamento pressão-velocidade, além de permitir que sejam testados vários tipos de malhas com funções alternativas de interpolação do fluxo J nas interfaces das malhas.

É interessante que se façam novos testes com o presente programa computacional no sentido de utilizar-se malhas mais refinadas com avaliações de número de Reynolds mais elevadas, para testar o programa a exaustão.

Outro aspecto interessante é a implementação do presente programa sob o prisma da otimização, buscando maior eficiência nos cálculos e no armazenamento das variáveis, neste contexto uma implementação orientada a objetos seria adequada.

Referências Bibliográficas

- ARIS, R., Vectors, Tensors, and the Basic Equations of Fluid Mechanics.** Dover, 1962.
- AVIS, D. e BHATTACHARYA, B. K., Algorithms for Computing d-Dimensional Voronoi Diagrams and their Duals.** Advances in Computing Research, Vol. 1, 159-80, 1983.
- BEJAN, Adrian, Convection Heat Transfer.** John Wiley & Sons (1984).
- CARDOSO, F. C., Construção de um Código Computacional para Simulação Numérica de Escoamentos com o Método dos Volumes Finitos usando Diagramas de Voronoi.** Trabalho individual para encaminhamento de Dissertação de Mestrado, Pós-Graduação em Ciências da Computação, Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, 1996.
- CLAUDIO, D. M. e MARINS, J. M., Cálculo Numérico Computacional.** Atlas, 1989.
- GHIA, U., GHIA, K. N., SHIN, C. T., High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method.** Journal of Computational Physics, 387-411, 1982.
- JAMESON, A. e MAVRIPLIS, D., Finite Volume Solution of the Two-Dimensional Euler Equations on a Regular Triangular Mesh.** AIAA Journal, Vol. 24, 611-18, 1986.
- MACKAY, A. L., Stereological Characteristic of Anatomic Arrangements in Crystals.** Journal of Microscopy, 95, 217-27, 1972.

- MALISKA, C. R., Transferência de Calor e Mecânica dos Fluidos Computacional.** Livros Técnicos e Científicos, 1995.
- MALISKA Jr., C. R., Um Robusto Gerador de Diagramas de Voronoi para Discretização de Domínios Irregulares.** XIV Congresso Ibero Latino-americano sobre Métodos Computacionais para a Engenharia, 548-57, Belo Horizonte, Brasil, 1994.
- MARCONDES, F., ZAMBALDI, M. C. e MALISKA, C. R., Simulação Numérica de Reservatórios de Petróleo Utilizando Malhas de Voronoi.** V ENCIT, 335-38, São Paulo, Brazil, 1994.
- MARIANI, V. C., Dissertação de Mestrado, em andamento, Pós-Graduação em Ciência da Computação, Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, 1997.**
- PATANKAR, S. V., Numerical Heat Transfer and Fluid Flow.** Hemisphere-McGraw-Hill, 1980.
- PATHAK, P., WINTERFELD, P. H., DAVIS, H. T. e SCRIVEN, L. E., Rock Structure and Transport there in: Unifying with Voronoi Models and Percolation Concepts.** paper SPE 8846, First Joint SPE/DOE Symposium on Enhanced Oil Recovery, Tulsa, 1980.
- PERIC, M., KESSLER, R., SCHEUERER, G., Comparison of Finite Volume Numerical Methods with Staggered and Colocated Grids.** Computers & Fluids, Vol. 16, 389-403, 1988.
- PETERS, S., Análise de Artigo para Submissão à Comissão Examinadora do Concurso para Prof. Assistente na Área de Cálculo Numérico.** Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, 1992.
- PETERS, S., Comunicação Pessoal, Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, 1997.**
- PREPARATA, F. P. e SHAMOS, M. I., Computational Geometry, an Introduction.** Springer Verlag, 1985.

- SERRIN, J., Mathematical Principles of Classical Fluid Mechanics. In Handbuck der Physik VIII/1, S. Flugge, 1959.**
- TANIGUCHI, N., ARAKAWA, C. e KOBAYASHI, T., Construction of a Flow-Simulating Method with Finite Volume Based on a Voronoi Diagram. JSME International Journal, Series II, Vol. 34, N° 1, 1991a.**
- TANIGUCHI, N. e KOBAYASHI, T., Finite Volume Method on the Unstructured Grid System. Computers & Fluids, Vol. 19, N° 34, 287-295, 1991b.**
- VAN DOORMAAL, J. P. e RAITHBY, G. D., Enhancements of the Simple Method for Predicting Incompressible Fluid Flow. Numerical Heat Transfer, Vol. 7, 147-63, 1984.**
- WINTERFELD, P. H., SCRIVEN, L. E. e DAVIS, H. T., Percolation and Conductivity of Random Two Dimensional Composites. J. Phys. C: Solid State Phys., 14, 2361-76, 1981.**

Apêndice A - Avaliação de d_{ij}^w

Este apêndice pretende discutir o por que de utilizar-se a média apresentada nas Eqs. (3.80) e (3.105) e não aquela apresentada por Maliska (1995) Pág. (347) para malhas não estruturadas dada pela Eq. (1), a seguir

$$d_{ij}^w = \frac{\overline{\Delta\vartheta}}{A_p} = \frac{\Delta\vartheta_i + \Delta\vartheta_j}{A_i^p + A_j^p}. \quad (1)$$

Nos pontos nodais mais próximos ao contorno, d_{ij}^w assume valores menores, algumas vezes bem menores, que aqueles obtidos nos pontos internos da malha, isto ocorre porque os valores de A_i^p nos pontos nodais das fronteiras são maiores, devido as menores distâncias entre o ponto i e vizinhos sobre a fronteira. Isto dificulta a convergência, pois quanto menor o d_{ij}^w maior será o gradiente de correção da pressão p' , o que traz instabilidade ao processo iterativo, e quando d_{ij}^w é grande as correções p' diminuem, facilitando o processo de convergência.

Procede-se uma análise deste fato através de um exemplo mostrado na Fig. (1) deste apêndice, onde calcula-se d_{ij}^w pelas formas propostas em Maliska (1995) e no presente trabalho, e após, faz-se uma comparação entre as duas.

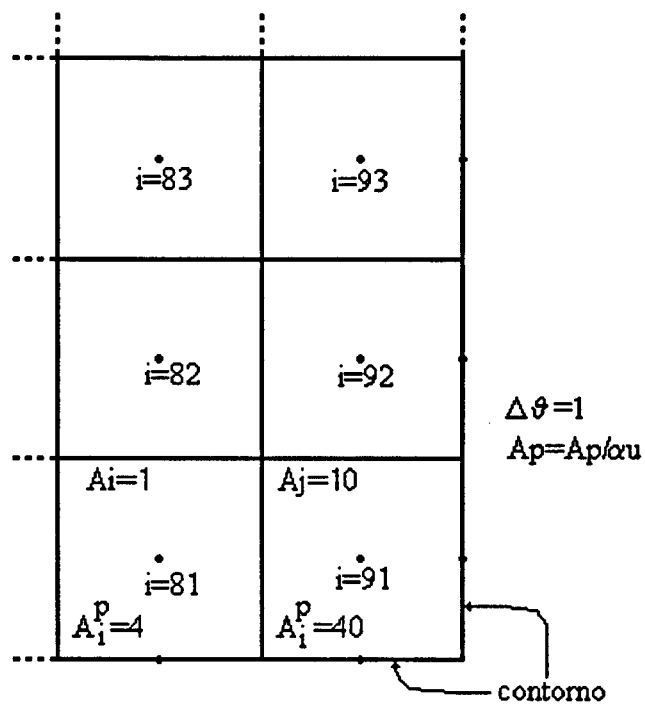


Figura 1 - Dados exemplo para cálculo de d_{ij}^w em malha estruturada para $i=91$.

Realizando-se os cálculos, pelo acoplamento Simple, conforme Eq. (1), de Maliska (1995), e usando-se coeficiente de relaxação $\alpha u=0,5$ (Patankar, 1980), tem-se:

$$\begin{cases} A_{81}^p = 8 \\ A_{91}^p = 80 \end{cases}, \quad (2)$$

$$d_{ij}^w = \left(\frac{(1+1) * 0,5}{(8+80) * 0,5} \right) = 0,0227, \quad (3)$$

e conforme a Eq. (3.80), do presente trabalho, tem-se:

$$d_{ij}^w = \left(\frac{1}{8} + \frac{1}{80} \right) * 0,5 = 0,06875, \quad (4)$$

Realizando-se os cálculos, pelo acoplamento Simplec, tem-se pela Eq. (1)

$$d_{ij}^w = \left(\frac{(1+1) * 0,5}{[(8-4*1) + (80-4*10)] * 0,5} \right) = 0,04\bar{5}, \quad (5)$$

e pela Eq. (3.93), tem-se

$$d_{ij}^w = \left(\frac{1}{8-4*1} + \frac{1}{80-4*10} \right) * 0,5 = 0,1375. \quad (6)$$

Levando-se em conta que um d_{ij}^w maior é melhor, do ponto de vista de estabilidade numérica, então, conclui-se que a Eq. (3.93) utilizada no presente trabalho é a melhor proposta, tanto no acoplamento Simple quanto no acoplamento Simplec.

Apêndice B - Programa Computacional Implementado

```
// Programa da Dissertação de Mestrado: voronoi.c
// Ultima correção 3-12-96
// Ultima alteração 16-12-96
// Autor: Fabian Corrêa Cardoso
// Cuidado!!! Programa para malha com 613 pontos nodais, ter atenção nas condições de
contorno.

#include <iostream.h>
#include <string.h>
#include <stdlib.h>
#include <fcntl.h>
#include <stdio.h>
#include <math.h>
/*#include <X11/X.h> // Alguma(s) dessas deve(m) incluir uma ou mais bibliotecas
necessárias para calcular o tempo.
#include <X11/Xos.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>*/

#include "voronoi.h"

// DEFINIÇÃO DAS CONSTANTES GLOBAIS AO SISTEMA
const double ERRO = 1e-4;
```

```

const double DELTA_t = 1e30;
const double ALFA_p_lin = 0.5;
const double ALFA_p = 0.5;
const double ALFA = 0.3;
const double REYNOLDS = 400;

// DECLARAÇÃO DE VARIÁVEIS GLOBAIS AO SISTEMA
double Soma_1[N_PONTOS], Soma_2[N_PONTOS], Soma_3[N_PONTOS],
Soma_4[N_PONTOS],
    Soma_5[N_PONTOS];
double g[N_PONTOS][MAX_VIZ];
int num_pts, num_rep_p=20, num_rep_u_v=10, num_iter=0;
tipo_volume volume[N_PONTOS];
tipo_param param[N_PONTOS];
tipo_coefic coefic[N_PONTOS];
double Soma_A[N_PONTOS];
double S[N_PONTOS];
static int i, j;
int grad_type = 4; // tipo do gradiente a ser utilizado
                    // = 1 -> Maliska
                    // = 2 -> Maliska, por nós modificado
                    // = 3 -> Jameson e Mavriplis
                    // = 4 -> Taniguchi et alli

// DEFINIÇÃO DA FUNÇÃO QUE DEVOLVE O MAIOR VALOR ENTRE DOIS
// VALORES
double max(double val1, double val2){

```

```
    if (val1 > val2)
        return val1;
    else
        return val2;
}

// DEFINIÇÃO DA FUNÇÃO QUE DEVOLVE O VALOR MÉDIO ENTRE DOIS
// VALORES
double media(double med1, double med2){
    double med;
    med = (med1 + med2)*0.5;
    return med;
}

// DEFINIÇÃO DA FUNÇÃO QUE LÊ OS N_PONTOS NODAIS
// Cada linha do arquivo lido são as coordenadas x,y separadas por
// uma tabulação (\t).
void le_coordenadas_pontos_nodais(){
    double pass1, pass2;
    FILE *fp;
    // Lê o vetor de pontos nodais.
    if ((fp = fopen("pontos.vor", "r")) == NULL){
        printf ("\nArquivo pontos.vor não encontrado.\n");
        exit(1);
    }

    fscanf (fp, "%d\n", &num_pts);
    printf ("\nLendo arquivo pontos.vor...");
```



```

// Fica em "loop" lendo as linhas do arquivo.
for (i=1; i<=num_pts; i++){
    fscanf (fp, "%lf\t%lf\n", &pass1, &pass2);
    volume[i].cx = pass1;
    volume[i].cy = pass2;
}
fclose(fp);
}

// DEFINIÇÃO DA FUNÇÃO QUE LÊ OS PONTOS NODAIS VIZINHOS PARA OS
N_PONTOS
// Cada linha do arquivo lido são os vizinhos dos pontos nodais.
void le_vizinhos(){
int pass3, pass4;
FILE *fp;
if ((fp = fopen("vizinhos.vor", "r")) == NULL){
    printf ("\nArquivo vizinhos.vor não encontrado.\n");
    exit(1);
}

printf ("\nLendo arquivo vizinhos.vor...");
// Fica em "loop" lendo as linhas do arquivo.
for (i=1; i<=num_pts; i++){
    // O valor lido %o o número de vizinhos do ponto nodal
    fscanf (fp, "%d",&pass3);
    volume[i].n_pts = pass3;
    for (j=1; j<=volume[i].n_pts; j++){
        // Lê os vizinhos

```

```

        fscanf (fp, "%d ", &pass4);
        volume[i].viz1[j] = pass4;
    }
}
fclose(fp);
}

// DEFINIÇÃO DA FUNÇÃO QUE LÊ OS VÉRTICES DOS VOLUMES DE CADA
PONTO NODAL
// Note-se que no arquivo o primeiro par de vértices repete-se para cada ponto nodal,
// isto ocorre para se fechar o volume do diagrama.
void le_vertices(){
int pass5=0;
double pass6, pass7;
FILE *fp;

if ((fp = fopen("vertices.vor", "r")) == NULL){
    printf ("\nArquivo vertices.vor não encontrado.\n");
    exit(1);
}

printf ("\nLendo arquivo vertices.vor...");
for (i=1; i<=num_pts; i++){
    // lê uma linha com o numero da linha
    fscanf (fp, "%d\n", &pass5);
    // o numero de vértices já foi lido junto com os vizinhos
    for (j=1; j<=volume[i].n_pts+1; j++){
        fscanf (fp, "%lf\t%lf", &pass6, &pass7);
    }
}
}

```

```
        volume[i].x[j] = pass6;  
        volume[i].y[j] = pass7;  
    }  
}  
fclose(fp);  
}
```

// DEFINIÇÃO DA FUNÇÃO QUE LÊ A MALHA

```
void le_malha(){  
    le_coordenadas_pontos_nodais();  
    le_vizinhos();  
    le_vertices();  
}
```

// DEFINIÇÃO DA FUNÇÃO QUE ATRIBUI OS PARÂMETROS INICIAIS

```
void atribui_param_iniciais(tipo_volume volume[], tipo_param param[]){
```

```
    // Condição de contorno
```

```
    volume[710].u = 0.0;
```

```
    volume[720].u = 0.0;
```

```
    volume[730].u = 0.0;
```

```
    volume[740].u = 1.0;
```

```
    volume[710].v = 0.0;
```

```
    volume[720].v = 0.0;
```

```
    volume[730].v = 0.0;
```

```
    volume[740].v = 0.0;
```

```
    volume[710].p = 0.0;
```

```
    volume[720].p = 0.0;
```

```
volume[730].p = 0.0;
volume[740].p = 0.0;
volume[710].p_lin = 0.0;
volume[720].p_lin = 0.0;
volume[730].p_lin = 0.0;
volume[740].p_lin = 0.0;
param[710].gama = 10.0/REYNOLDS;
param[720].gama = 10.0/REYNOLDS;
param[730].gama = 10.0/REYNOLDS;
param[740].gama = 10.0/REYNOLDS;
param[710].rho = 1.0;
param[720].rho = 1.0;
param[730].rho = 1.0;
param[740].rho = 1.0;
S[710] = 0.0;
S[720] = 0.0;
S[730] = 0.0;
S[740] = 0.0;
```

```
// Chute inicial das variáveis
```

```
for (i=1; i<=num_pts; i++){
    volume[i].p = 0.0;
    volume[i].u = 0.5;
    volume[i].v = 0.1;
}
```

```
// Inicializacao de variáveis
```

```
for (i=1; i<=num_pts; i++){
```

```

    param[i].gama = 10.0/REYNOLDS;
    param[i].rho = 1.0;
    S[i] = 0.0;
  }
}

// DEFINIÇÃO DA FUNÇÃO QUE CALCULA A DISTANCIA ENTRE OS PONTOS
// NODAIS E
// A DISTANCIA ENTRE OS PONTOS DOS VÉRTICES DE CADA VOLUME
// (ÁREA DA FACE)
void calcula_L_e_S(tipo_volume volume[], tipo_param param[]){
  for (i=1; i<=num_pts; i++){
    int n_pts = volume[i].n_pts;
    for (j=1; j<=n_pts; j++){
      // Calcula tamanho das arestas entre os vizinhos (face de Voronoi)
      param[i].S[j] = sqrt(pow(volume[i].x[j+1] - volume[i].x[j], 2) +
        pow(volume[i].y[j+1] - volume[i].y[j], 2));
      int viz1 = volume[i].viz1[j];
      // Calcula a distancia de um ponto nodal ate' seus vizinhos
      switch (viz1){
        case BOTTOM_WALL:{
          param[i].L[j] = volume[i].cy;
        }
        break;

        case RIGHT_WALL:{
          param[i].L[j] = 10.0 - volume[i].cx;
        }
      }
    }
  }
}

```

```
        break;

        case LEFT_WALL:{
            param[i].L[j] = volume[i].cx;
        }
        break;

        case TOP_WALL_SLIP:{
            param[i].L[j] = 10.0 - volume[i].cy;
        }
        break;

        default:
            param[i].L[j] = sqrt(pow((volume[viz1].cx - volume[i].cx), 2) +
                                pow((volume[viz1].cy - volume[i].cy), 2));
        }
    }
}
}
```

// DEFINIÇÃO DA FUNÇÃO QUE CALCULA A NORMA DO VETOR NORMAL DE CADA FACE DOS VOLUMES DE CONTROLE

```
void calcula_norma(tipo_volume volume[], tipo_param param[]){
double norma; // CALCULA A NORMA DO VETOR NORMAL DE CADA FACE
double delta_x;
double delta_y;
    for (i=1; i<=num_pts; i++){
        for (j=1; j<=volume[i].n_pts; j++){
```

```

if (volume[i].viz1[j] <= num_pts){
    delta_x = volume[volume[i].viz1[j]].cx - volume[i].cx;
    delta_y = volume[volume[i].viz1[j]].cy - volume[i].cy;
    norma = sqrt((delta_x * delta_x) + (delta_y * delta_y));
    param[i].ex[j] = (delta_x)/norma;
    param[i].ey[j] = (delta_y)/norma;
}
else{
    if (volume[i].viz1[j] == 740){
        param[i].ex[j] = 0.0;
        param[i].ey[j] = 1.0;
    }
    if (volume[i].viz1[j] == 730){
        param[i].ex[j] = 1.0;
        param[i].ey[j] = 0.0;
    }
    if (volume[i].viz1[j] == 720){
        param[i].ex[j] = 0.0;
        param[i].ey[j] = -1.0;
    }
    if (volume[i].viz1[j] == 710){
        param[i].ex[j] = -1.0;
        param[i].ey[j] = 0.0;
    }
}

/*          CUIDADO SE O CONTORNO FOR DEFINIDO POR RETAS
HORIZONTAIS OU VERTICAIS, DARÁ ERRO: DIVISÃO POR ZERO
          angulo_teta = arctan  (-(volume[i].x[j+1] -
volume[i].x[j])/(volume[i].y[j] - volume[i].y[j-1]));

```

```

        param[i].ex[j] = cos (angulo_teta);
        param[i].ey[j] = sin (angulo_teta);          */
    }
}
}
}

```

// DEFINIÇÃO DA FUNÇÃO QUE CALCULA O SOMATÓRIO DO Lij

```

void soma_L(tipo_volume volume[], tipo_param param[]){
    for (i=1; i<=num_pts; i++){
        param[i].Soma_L = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts){
                param[i].Soma_L = param[i].Soma_L + param[i].L[j];
            }
        }
    }
}

```

// DEFINIÇÃO DA FUNÇÃO QUE CALCULA O SOMATÓRIO DO Lij
PROJETADO NA DIREÇÃO x

```

void soma_Lx(tipo_volume volume[], tipo_param param[]){
    for (i=1; i<=num_pts; i++){
        param[i].Soma_Lx = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts){
                param[i].Soma_Lx = param[i].Soma_Lx + (param[i].L[j] *
fabs(param[i].ex[j]));
            }
        }
    }
}

```



```

    }
  }
}

// DEFINIÇÃO DA FUNÇÃO QUE CALCULA O SOMATÓRIO DO Lij
PROJETADO NA DIREÇÃO y
void soma_Ly(tipo_volume volume[], tipo_param param[]){
  for (i=1; i<=num_pts; i++){
    param[i].Soma_Ly = 0.0;
    for (j=1; j<=volume[i].n_pts; j++){
      if (volume[i].viz1[j] <= num_pts)
        param[i].Soma_Ly = param[i].Soma_Ly + (param[i].L[j] *
fabs(param[i].ey[j]));
    }
  }
}

// DEFINIÇÃO DA FUNÇÃO QUE CALCULA A ÁREA DE UM TRIÂNGULO
ATRAVÉS DO DETERMINANTE DE UMA MATRIZ FORMADA PELA
PRIMEIRA COLUNA DE 1S (UNS) E AS OUTRAS PELOS VÉRTICES DE UM
TRIÂNGULO
double determinante(double xa, double ya, double xb, double yb, double xc, double yc){
double A;
// Tentar fatorar para uma expressão mais simples no MAPLE ou MATHEMATICA
  A = ((yc * xb) + (yb * xa) + (ya * xc) - ((ya * xb) + (yb * xc) + (yc * xa))) * 0.5;
  return A;
}

```

// DEFINIÇÃO DA FUNÇÃO QUE CALCULA O VOLUME DE UM DIAGRAMA DE VORONOI DIVIDINDO-O EM TRIÂNGULOS

```
void calcula_volume_diagrama(tipo_volume volume[]){
double Area, x1, x2, x3, y1, y2, y3;

    for (i=1; i<=num_pts; i++){
        volume[i].vol_diag = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            x3 = volume[i].x[j];
            x2 = volume[i].x[j+1];
            x1 = volume[i].cx;
            y3 = volume[i].y[j];
            y2 = volume[i].y[j+1];
            y1 = volume[i].cy;
            Area = determinante(x1, y1, x2, y2, x3, y3);
            volume[i].vol_diag = volume[i].vol_diag + Area;
        }
    }
}
```

// DEFINIÇÃO DA FUNÇÃO QUE CALCULA OS PARÂMETROS GEOMÉTRICOS

```
void calcula_param_geom(tipo_volume volume[], tipo_param param[]){
    // Rotina para calculo do Lij e do Sij
    calcula_L_e_S(volume, param);

    switch (grad_type){
        case 1:{
```

```

        // Rotina para calculo do somatório de Lij
        soma_L(volume, param);
    }
break;

case 2:{
    // Rotina para calculo do somatório de Lij projetado na direção x
    soma_Lx(volume, param);

    // Rotina para calculo do somatório de Lij projetado na direção y
    soma_Ly(volume, param);
}

break;
}

// Calculo do volume de um diagrama de Voronoi
calcula_volume_diagrama(volume);
}

// DEFINIÇÃO DA FUNÇÃO QUE CALCULA A MASSA ESPECIFICA
void calcula_Rho(tipo_volume volume[], tipo_param param[]){
//   printf ("\nCalculando Rhoij...");
    for (i=1; i<=num_pts; i++)
        for (j=1; j<=volume[i].n_pts; j++)
            param[i].Rho[j] = media(param[i].rho, param[volume[i].viz1[j]].rho);
}

```

```
// DEFINIÇÃO DA FUNÇÃO QUE CALCULA O COEFICIENTE DE
CONDUTIVIDADE
```

```
void calcula_Gama(tipo_volume volume[], tipo_param param[]){
    for (i=1; i<=num_pts; i++){
        for (j=1; j<=volume[i].n_pts; j++){
            param[i].Gama[j] = (2 * param[i].gama * param[volume[i].viz1[j]].gama)
                / (param[i].gama + param[volume[i].viz1[j]].gama);
        }
    }
}
```

```
// DEFINIÇÃO DA FUNÇÃO QUE CALCULA O COEFICIENTE TEMPORAL
```

```
void calcula_A0(tipo_volume volume[], tipo_coefic coefic[], tipo_param param[]){
    for (i=1; i<=num_pts; i++){
        coefic[i].A0 = param[i].rho * volume[i].vol_diag/DELTA_t;
    }
}
```

```
// DEFINIÇÃO DA FUNÇÃO QUE CALCULA O FLUXO DIFUSIVO
```

```
void calcula_D(tipo_volume volume[], tipo_param param[], tipo_coefic coefic[]){
    for (i=1; i<=num_pts; i++){
        for (j=1; j<=volume[i].n_pts; j++){
            coefic[i].D[j] = (param[i].Gama[j] * param[i].S[j])/param[i].L[j];
        }
    }
}
```

```
// DEFINIÇÃO DA FUNÇÃO QUE CALCULA OS SOMATÓRIOS PARA
CALCULO DO GRADIENTE NA
```

```
// DIREÇÃO x E y PELO MÉTODO DE TANIGUCHI
```

```
void calcula_soma_Tan(){
```

```

for (i=1; i<=num_pts; i++)
    for (j=1; j<=volume[i].n_pts; j++){
        g[i][j] = param[i].S[j] / param[i].L[j];
    }

for (i=1; i<=num_pts; i++){
    Soma_1[i] = 0.0;
    Soma_2[i] = 0.0;
    Soma_4[i] = 0.0;
    for (j=1; j<=volume[i].n_pts; j++)
        if (volume[i].viz1[j] <= num_pts){
            Soma_1[i] = Soma_1[i] + (g[i][j] * param[i].ex[j]
                * param[i].ex[j]);
            Soma_2[i] = Soma_2[i] + (g[i][j] * param[i].ex[j]
                * param[i].ey[j]);
            Soma_4[i] = Soma_4[i] + (g[i][j] * param[i].ey[j]
                * param[i].ey[j]);
        }
    }
}

```

**// DEFINIÇÃO DA FUNÇÃO QUE CALCULA A VELOCIDADE NODAL NORMAL
A FACEij PARA A PRIMEIRA ITERAÇÃO**

```

void calcula_W(tipo_volume volume[], tipo_param param[]){
double wi;
double wj;
    for (i=1; i<=num_pts; i++){
        for (j=1; j<=volume[i].n_pts; j++){

```

```

        if (volume[i].viz1[j] <= num_pts){
            wi = volume[i].u * (param[i].ex[j]) + volume[i].v * (param[i].ey[j]);
            wj = volume[volume[i].viz1[j]].u * (param[i].ex[j]) +
volume[volume[i].viz1[j]].v * (param[i].ey[j]);
            // Calculo da velocidade normal sobre a faceij
            param[i].W[j] = media (wi, wj);
        }
        else{
//          W nos contornos e' sempre nulo, pois não temos fluxos de massa nos
contornos
            param[i].W[j] = 0.0;
        }
    }
}
}
}

```

```

// DEFINICAO DA FUNÇÃO QUE CALCULA O FLUXO CONVECTIVO
void calcula_F(tipo_volume volume[], tipo_param param[], tipo_coefic coefic[]){
    for (i=1; i<=num_pts; i++){
        for (j=1; j<=volume[i].n_pts; j++){
            coefic[i].F[j] = param[i].Rho[j] * param[i].W[j] * param[i].S[j];
        }
    }
}
}

```

```

// DEFINICAO DA FUNÇÃO QUE CALCULA O FATOR PESO DE AJUSTE DA
FUNÇÃO DE
// INTERPOLAÇÃO LEI DA POTÊNCIA

```

```

double fator_peso (double Pe){
double Fat_Pe;
    Fat_Pe = max(0, pow(1 - (0.1 * fabs(Pe)), 5));
    return Fat_Pe;
}

// DEFINICAO DA FUNÇÃO QUE CALCULA O COEFICIENTE DE LIGAÇÃO
ENTRE i E j
void calcula_A(tipo_volume volume[], tipo_coefic coefic[]){
double FPe;
double Peclet;
    for (i=1; i<=num_pts; i++){
        for (j=1; j<=volume[i].n_pts; j++){
            Peclet = coefic[i].F[j]/coefic[i].D[j];
            FPe = fator_peso(Peclet);
            coefic[i].A[j] = max(0, -coefic[i].F[j]) + (coefic[i].D[j] * FPe);
        }
    }
}

// DEFINICAO DA FUNÇÃO QUE CALCULA O COEFICIENTE CENTRAL
void calcula_Ap(tipo_coefic coefic[]){
    for (i=1; i<=num_pts; i++){
        Soma_A[i] = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            Soma_A[i] = Soma_A[i] + coefic[i].A[j];
        }
    }

    for (i=1; i<=num_pts; i++){

```

```

        coefic[i].Ap = Soma_A[i] + coefic[i].A0;
        coefic[i].Ap = coefic[i].Ap/ALFA;
    }
}

// DEFINICAO DA FUNÇÃO QUE CALCULA O TERMO FONTE PARA A
VELOCIDADE u
void calcula_bu(tipo_volume volume[], tipo_coefic coefic[]){
    for (i=1; i<=num_pts; i++){
        coefic[i].bu = (S[i] * volume[i].vol_diag) + (volume[i].u_old * coefic[i].A0) +
        coefic[i].Ap * (1 - ALFA) * volume[i].u ;
    }
}

// DEFINICAO DA FUNÇÃO QUE CALCULA O TERMO FONTE PARA A
VELOCIDADE v
void calcula_bv(tipo_volume volume[], tipo_coefic coefic[]){
    for (i=1; i<=num_pts; i++){
        coefic[i].bv = (S[i] * volume[i].vol_diag) + (volume[i].v_old * coefic[i].A0) +
        coefic[i].Ap * (1 - ALFA) * volume[i].v;
    }
}

// DEFINICAO DA FUNÇÃO QUE CALCULA OS COEFICIENTES
void calcula_coeficientes(){
    // Rotina para calculo do Fij
    calcula_F(volume, param, coefic);
}

```



```

// Rotina para calculo do Aij
calcula_A(volume, coefic);

// Rotina para calculo do Api
calcula_Ap(coefic);

// Rotina para calculo do bui
calcula_bu(volume, coefic);

// Rotina para calculo do bui
calcula_bv(volume, coefic);
}

// DEFINICAO DA FUNÇÃO QUE CALCULA OS SOMATÓRIOS PARA
CALCULO DO GRADIENTE NA
// DIREÇÃO x E y PELO MÉTODO DE TANIGUCHI, LEVANDO EM CONTA A
PRESSÃO
void calcula_soma_Tan_pressao(){
    for (i=1; i<=num_pts; i++){
        Soma_3[i] = 0.0;
        Soma_5[i] = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts){
                Soma_3[i] = Soma_3[i] + (g[i][j] * param[i].ex[j]
                    * (volume[volume[i].viz1[j]].p - volume[i].p)
                    / param[i].L[j]);
                Soma_5[i] = Soma_5[i] + (g[i][j] * param[i].ey[j]
                    * (volume[volume[i].viz1[j]].p - volume[i].p)

```

```

        / param[i].L[j]);
    }
}
}

// DEFINIÇÕES DAS FUNÇÕES QUE CALCULAM O GRADIENTE DE PRESSÃO
NA DIREÇÃO x
// Método das médias ponderadas dos gradientes existentes nas interfaces
// Maliska, pág. 349, eq. 16.46
void calcula_grad_pressao_x_Mal(tipo_volume volume[], tipo_param param[]){
double Soma_Grad_x[N_PONTOS];
    for (i=1; i<=num_pts; i++){
        Soma_Grad_x[i] = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts){
                Soma_Grad_x[i] = Soma_Grad_x[i] + ((volume[volume[i].viz1[j]].p -
volume[i].p) * param[i].ex[j]);
            }
        }
    }

    for (i=1; i<=num_pts; i++){
        volume[i].Grad_Px = Soma_Grad_x[i]/param[i].Soma_L;
    }
}

// Método das médias ponderadas entre os gradientes normais projetados

```

```

//          Dissertação de mestrado de Fabian Corrêa Cardoso, págs. 20-
23
void calcula_grad_pressao_x_Nos(tipo_volume volume[], tipo_param param[]){
double Soma_Grad_x[N_PONTOS];
    for (i=1; i<=num_pts; i++){
        Soma_Grad_x[i] = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts){
                Soma_Grad_x[i] = Soma_Grad_x[i] + ((volume[volume[i].viz1[j]].p -
volume[i].p)
                    * param[i].ex[j]);
            }
        }
    }

    for (i=1; i<=num_pts; i++){
        volume[i].Grad_Px = Soma_Grad_x[i]/param[i].Soma_Lx;
    }
}

// Método
//          Jameson e Mavriplis (AIAA Journal, Vol. 24, pág. 611-18, 1986)
void calcula_grad_pressao_x_JeM(tipo_volume volume[], tipo_param param[]){
    for (i=1; i<=num_pts; i++){
        volume[i].Grad_Px = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts)

```

```

                                volume[i].Grad_Px    =    volume[i].Grad_Px    +
(((volume[volume[i].viz1[j]].p    *    param[i].S[j]    *    param[i].ex[j])    *
0.5)/volume[i].vol_diag);
        }
    }
}
// Método
//                                Taniguchi et alli (Computer & Fluids, Vol. 19, Nº. 34, 287-95,
1991)
void calcula_grad_pressao_x_Tan(tipo_volume volume[]){
    for (i=1; i<=num_pts; i++){
        volume[i].Grad_Px = ((Soma_3[i] * Soma_4[i]) - (Soma_2[i]
            * Soma_5[i]))/((Soma_1[i] * Soma_4[i])
            - (Soma_2[i] * Soma_2[i]));
    }
}

// DEFINICAO DA FUNÇÃO QUE RESOLVE O SISTEMA DE EQUAÇÕES
GERADO, PELO MÉTODO DE GAUSS-SEIDEL
void resolve_sistema_u(tipo_volume volume[]){
    int k;
    double Soma_Au[N_PONTOS];
    //double var_aux;
    for (k=1; k<=num_rep_u_v; k++){
        for (i=1; i<=num_pts; i++){
            Soma_Au[i] = 0.0;
            for (j=1; j<=volume[i].n_pts; j++){

```

```

                Soma_Au[i] = Soma_Au[i] + (coefic[i].A[j] *
volume[volume[i].viz1[j]].u);
            }
        }

        for (i=1; i<=num_pts; i++){
            volume[i].u = (Soma_Au[i] + coefic[i].bu - (volume[i].vol_diag *
volume[i].Grad_Px))/coefic[i].Ap;
        }
    }
}

```

// DEFINICAO DA FUNÇÃO QUE CALCULA A VELOCIDADES u

```

void calcula_velocidade_u(){
    // Rotina para calculo do gradiente de pressão
    switch (grad_type){
        case 1:{
            calcula_grad_pressao_x_Mal(volume, param);
        }
        break;

        case 2:{
            calcula_grad_pressao_x_Nos(volume, param);
        }
        break;

        case 3:{
            calcula_grad_pressao_x_JeM(volume, param);

```

```

    }
    break;

    case 4:{
        calcula_soma_Tan_pressao();
        calcula_grad_pressao_x_Tan(volume);
    }
    break;
}

// Rotina para resolução do sistema de equações gerado, pelo método de Gauss-
Seidel
    resolve_sistema_u(volume);
}

// DEFINICAO DA FUNÇÃO QUE CALCULA O GRADIENTE DE PRESSÃO NA
DIREÇÃO y
// Método das médias ponderadas dos gradientes existentes nas interfaces
//
// Maliska, pág. 349, eq. 16.46
void calcula_grad_pressao_y_Mal(tipo_volume volume[], tipo_param param[]){
double Soma_Grad_y[N_PONTOS];
    for (i=1; i<=num_pts; i++){
        Soma_Grad_y[i] = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts)
                Soma_Grad_y[i] = Soma_Grad_y[i] + ((volume[volume[i].viz1[j]].p -
volume[i].p) * param[i].ey[j]);
        }
    }
}

```

```

    }

    for (i=1; i<=num_pts; i++){
        volume[i].Grad_Py = Soma_Grad_y[i]/param[i].Soma_L;
        printf ("\nG_py[%d] = %.32lf", i, volume[i].Grad_Py);
    }
}

// Método das médias ponderadas entre os gradientes normais projetados
//
//          Dissertação de mestrado de Fabian Corrêa Cardoso, págs. 20-
//          23
void calcula_grad_pressao_y_Nos(tipo_volume volume[], tipo_param param[]){
double Soma_Grad_y[N_PONTOS];
    for (i=1; i<=num_pts; i++){
        Soma_Grad_y[i] = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts)
                Soma_Grad_y[i] = Soma_Grad_y[i] + ((volume[volume[i].viz1[j]].p -
volume[i].p)
                    * param[i].ey[j]);
        }
    }

    for (i=1; i<=num_pts; i++){
        volume[i].Grad_Py = Soma_Grad_y[i]/param[i].Soma_Ly;
    }
}

// Método

```

```

//                               Jameson e Mavriplis (AIAA Journal, Vol. 24, pág. 611-18,
1986)
void calcula_grad_pressao_y_JeM(tipo_volume volume[], tipo_param param[]){
    for (i=1; i<=num_pts; i++){
        volume[i].Grad_Py = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts)
                volume[i].Grad_Py = volume[i].Grad_Py +
((volume[volume[i].viz1[j]].p
                                * param[i].S[j] * param[i].ey[j] *
0.5)/volume[i].vol_diag);
        }
    }
}

// Método do mínimo resíduo quadrático
//                               Taniguchi et alli (Computer & Fluids, Vol. 19, N°. 34, 287-95,
1991)
void calcula_grad_pressao_y_Tan(tipo_volume volume[]){
    for (i=1; i<=num_pts; i++){
        volume[i].Grad_Py = ((Soma_1[i] * Soma_5[i]) - (Soma_2[i]
                                * Soma_3[i]))/((Soma_1[i] * Soma_4[i])
                                - (Soma_2[i] * Soma_2[i]));
    }
}

// DEFINICAO DA FUNÇÃO QUE RESOLVE O SISTEMA DE EQUAÇÕES
GERADO, PELO MÉTODO DE GAUSS-SEIDEL
void resolve_sistema_v(tipo_volume volume[], tipo_coefic coefic[]){

```



```

int k;
double Soma_Av[N_PONTOS];
    for (k=1; k<=num_rep_u_v; k++){
        for (i=1; i<=num_pts; i++){
            Soma_Av[i] = 0.0;
            for (j=1; j<=volume[i].n_pts; j++){
                Soma_Av[i] = Soma_Av[i] + (coefic[i].A[j] *
volume[volume[i].viz1[j]].v);
            }
        }

        for (i=1; i<=num_pts; i++){
            volume[i].v = (Soma_Av[i] + coefic[i].bv - (volume[i].vol_diag *
volume[i].Grad_Py))/coefic[i].Ap;
        }
    }

// DEFINICAO DA FUNÇÃO QUE CALCULA A VELOCIDADE v
void calcula_velocidade_v(){
    // Rotina para calculo do gradiente de pressao
    switch (grad_type){
        case 1:{
            calcula_grad_pressao_y_Mal(volume, param);
        }
        break;

        case 2:{

```

```

        calcula_grad_pressao_y_Nos(volume, param);
    }
    break;

    case 3:{
        calcula_grad_pressao_y_JeM(volume, param);
    }
    break;

    case 4:{
        calcula_grad_pressao_y_Tan(volume);
    }
    break;
}

// Rotina para resolução do sistema de equações gerado, pelo método de Gauss-
Seidel
    resolve_sistema_v(volume, coefic);
}

// DEFINICAO DA FUNÇÃO QUE ZERA p LINHA
void zera_p_lin(){
    for (i=1; i<=num_pts; i++)
        volume[i].p_lin = 0.0;
}

// DEFINICAO DA FUNÇÃO QUE CALCULA dwij
void calcula_dw(tipo_volume volume[], tipo_coefic coefic[]){

```

```

for (i=1; i<=num_pts; i++){
    for (j=1; j<=volume[i].n_pts; j++){
        if (volume[i].viz1[j] <= num_pts){
            volume[i].dw[j] = ((volume[i].vol_diag/(coefic[i].Ap-Soma_A[i])) +
(volume[volume[i].viz1[j]].vol_diag
                                                                    /
                                                                    (coefic[volume[i].viz1[j]].Ap-
Soma_A[volume[i].viz1[j]]))) * 0.5;
        }
        else{
//          Para as fronteiras, não ha' esta influencia, pois não e' necessária haver uma
correção de
//          pressao, neste caso.
            volume[i].dw[j] = 0.0;
        }
    }
}
}

```

// DEFINICAO DA FUNÇÃO QUE CALCULA AS INFLUENCIAS DOS VIZINHOS

```

void calcula_a(tipo_volume volume[], tipo_param param[], tipo_coefic coefic[]){
    for (i=1; i<=num_pts; i++)
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts){
                coefic[i].a[j] = (param[i].Rho[j] * param[i].S[j] *
volume[i].dw[j])/param[i].L[j];
            }
        }
}
}

```

// DEFINICAO DA FUNÇÃO QUE CALCULA O COEFICIENTE CENTRAL

```
void calcula_ap(tipo_volume volume[], tipo_coefic coefic[]){
double Soma_a[N_PONTOS];
    for (i=1; i<=num_pts; i++){
        Soma_a[i] = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts){
                Soma_a[i] = Soma_a[i] + coefic[i].a[j];
            }
        }
        coefic[i].ap = Soma_a[i];
    }
}
```

// DEFINICAO DA FUNÇÃO QUE RESOLVE O SISTEMA DE EQUAÇÕES GERADO, PELO MÉTODO DE GAUSS-SEIDEL

```
void resolve_sistema_p_lin(tipo_volume volume[], tipo_coefic coefic[]){
int k;
double Soma_ap[N_PONTOS];
double Soma_RWS[N_PONTOS];
double pref_lin=0;
    for (i=1; i<=num_pts; i++){
        Soma_RWS[i] = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            Soma_RWS[i] = Soma_RWS[i] + (param[i].Rho[j] * param[i].W[j] *
param[i].S[j]);
        }
    }
}
```

```

    }

    for (k=1; k<=num_rep_p; k++){
        for (i=1; i<=num_pts; i++){
            Soma_ap[i] = 0.0;
            for (j=1; j<=volume[i].n_pts; j++){
                if (volume[i].viz1[j] <= num_pts){
                    Soma_ap[i] = Soma_ap[i] + (coefic[i].a[j] *
volume[volume[i].viz1[j]].p_lin);
                }
            }
        }

        for (i=1; i<=num_pts; i++){
            volume[i].p_lin = ((Soma_ap[i] - Soma_RWS[i])/coefic[i].ap) *
(ALFA_p_lin) + volume[i].p_lin * (1-ALFA_p_lin);
        }
    }
}

```

// DEFINICAO DA FUNÇÃO QUE CALCULA A PRESSAO p

```
void calcula_p_lin(){
```

```
    // Rotina para calculo do
```

```
    calcula_dw(volume, coefic);
```

```
    // Rotina para calculo das influencias dos vizinhos
```

```
    calcula_a(volume, param, coefic);
```

```

// Rotina para calculo do gradiente de pressao
calcula_ap(volume, coefic);

// Rotina para resolução do sistema de equações gerado, pelo método de Gauss-
Seidel
resolve_sistema_p_lin(volume, coefic);
}

// DEFINICAO DA FUNÇÃO QUE CALCULA A PRESSAO JÁ CORRIGIDA
void calcula_p_corrigido(tipo_volume volume[]){
double pref=0;
for (i=1; i<=num_pts; i++){
    volume[i].p = volume[i].p + (ALFA_p * volume[i].p_lin);
}

pref = volume[1].p;
for (i=1; i<=num_pts; i++){
    volume[i].p = volume[i].p - pref;
}
}

// DEFINICAO DA FUNÇÃO QUE CALCULA A VELOCIDADE CORRIGIDA NAS
FACES DOS VOLUMES DE CONTROLE
void calcula_W_corrigido(tipo_volume volume[], tipo_param param[]){
double Grad_PW_lin[N_PONTOS][MAX_VIZ];
double W_lin[N_PONTOS][MAX_VIZ];
for (i=1; i<=num_pts; i++)
    for (j=1; j<=volume[i].n_pts; j++){

```

```

        if (volume[i].viz1[j] <= num_pts){
            Grad_PW_lin[i][j] = (volume[volume[i].viz1[j]].p_lin -
volume[i].p_lin)/param[i].L[j];
            W_lin[i][j] = -1 * volume[i].dw[j] * Grad_PW_lin[i][j];
            param[i].W[j] = param[i].W[j] + W_lin[i][j];
        }
        else{
            param[i].W[j] = param[i].W[j] + 0.0;
        }
    }
}

```

// DEFINICAO DA FUNÇÃO QUE CALCULA OS SOMATÓRIOS PARA
 CALCULO DO GRADIENTE NA
 // DIREÇÃO x E y PELO MÉTODO DE TANIGUCHI, LEVANDO EM CONTA A
 PRESSAO LINHA

```

void calcula_soma_Tan_pressao_lin(){
    for (i=1; i<=num_pts; i++){
        Soma_3[i] = 0.0;
        Soma_5[i] = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts){
                Soma_3[i] = Soma_3[i] + (g[i][j] * param[i].ex[j]
                    * (volume[volume[i].viz1[j]].p_lin
                    - volume[i].p_lin)/param[i].L[j]);
                Soma_5[i] = Soma_5[i] + (g[i][j] * param[i].ey[j]
                    * (volume[volume[i].viz1[j]].p_lin
                    - volume[i].p_lin)/param[i].L[j]);
            }
        }
    }
}

```

```

        }
    }
}

// DEFINICAO DA FUNÇÃO QUE CALCULA O GRADIENTE DE PRESSÃO
// LINHA NA DIREÇÃO x
// Método das médias ponderadas dos gradientes existentes nas interfaces
// Maliska, pág. 349, eq. 16.46
void calcula_grad_px_lin_Mal(tipo_volume volume[], tipo_param param[]){
double Soma_Grad_x[N_PONTOS];
    for (i=1; i<=num_pts; i++){
        Soma_Grad_x[i] = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts)
                Soma_Grad_x[i] = Soma_Grad_x[i] +
                ((volume[volume[i].viz1[j]].p_lin - volume[i].p_lin) * param[i].ex[j]);
        }
    }

    for (i=1; i<=num_pts; i++){
        volume[i].Grad_px_lin = Soma_Grad_x[i]/param[i].Soma_L;
    }
}

// Método das médias ponderadas entre os gradientes normais projetados
// Dissertação de mestrado de Fabian Corrêa Cardoso, págs. 20-
23
void calcula_grad_px_lin_Nos(tipo_volume volume[], tipo_param param[]){

```



```

double Soma_Grad_x[N_PONTOS];
    for (i=1; i<=num_pts; i++){
        Soma_Grad_x[i] = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts)
                Soma_Grad_x[i] = Soma_Grad_x[i] +
                ((volume[volume[i].viz1[j]].p_lin - volume[i].p_lin) * param[i].ex[j]);
        }
    }

    for (i=1; i<=num_pts; i++){
        volume[i].Grad_px_lin = Soma_Grad_x[i]/param[i].Soma_Lx;
    }
}

// Método
//
//                               Jameson e Mavriplis (AIAA Journal, Vol. 24, pág. 611-18,
//                               1986)
void calcula_grad_px_lin_JeM(tipo_volume volume[], tipo_param param[]){
    for (i=1; i<=num_pts; i++){
        volume[i].Grad_px_lin = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts)
                volume[i].Grad_px_lin = volume[i].Grad_px_lin +
                (((volume[volume[i].viz1[j]].p_lin
                    * param[i].S[j] * param[i].ey[j])
                    / volume[i].vol_diag) * 0.5);
        }
    }
}

```

```

}
// Método do mínimo resíduo quadrático
//                               Taniguchi et alli (Computer & Fluids, Vol. 19, Nº. 34, 287-95,
1991)
void calcula_grad_px_lin_Tan(tipo_volume volume[]){
    for (i=1; i<=num_pts; i++){
        volume[i].Grad_px_lin = ((Soma_3[i] * Soma_4[i]) - (Soma_2[i]
            * Soma_5[i]))/((Soma_1[i] * Soma_4[i])
            - (Soma_2[i] * Soma_2[i]));
    }
}

// DEFINICAO DA FUNÇÃO QUE CALCULA A VELOCIDADE U CORRIGIDA
void calcula_u_corrigido(){
double u_lin[N_PONTOS];
    switch (grad_type){
        case 1:{
            calcula_grad_px_lin_Mal(volume, param);
        }
        break;

        case 2:{
            calcula_grad_px_lin_Nos(volume, param);
        }
        break;

        case 3:{
            calcula_grad_px_lin_JeM(volume, param);

```

```

    }
    break;

    case 4:{
        calcula_soma_Tan_pressao_lin();
        calcula_grad_px_lin_Tan(volume);
    }
    break;
}

for (i=1; i<=num_pts; i++){
    u_lin[i] = (-1 * volume[i].vol_diag * volume[i].Grad_px_lin)/coefic[i].Ap;
}

for (i=1; i<=num_pts; i++)
    volume[i].u = volume[i].u + u_lin[i];
}

```

// DEFINICAO DA FUNÇÃO QUE CALCULA O GRADIENTE DE PRESSÃO LINHA NA DIREÇÃO y

// Método das médias ponderadas dos gradientes existentes nas interfaces

// Maliska, pág. 349, eq. 16.46

void calcula_grad_py_lin_Mal(tipo_volume volume[], tipo_param param[]){

double Soma_Grad_y[N_PONTOS];

for (i=1; i<=num_pts; i++){

Soma_Grad_y[i] = 0.0;

for (j=1; j<=volume[i].n_pts; j++){

if (volume[i].viz1[j] <= num_pts)

```

                Soma_Grad_y[i] = Soma_Grad_y[i] +
((volume[volume[i].viz1[j]].p_lin - volume[i].p_lin) * param[i].ey[j]);
            }
        }

        for (i=1; i<=num_pts; i++){
            volume[i].Grad_py_lin = Soma_Grad_y[i]/param[i].Soma_L;
        }
    }

// Método das médias ponderadas entre os gradientes normais projetados
//
//          Dissertação de mestrado de Fabian Corrêa Cardoso, págs. 20-
//          23
void calcula_grad_py_lin_Nos(tipo_volume volume[], tipo_param param[]){
double Soma_Grad_y[N_PONTOS];
    for (i=1; i<=num_pts; i++){
        Soma_Grad_y[i] = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts)
                Soma_Grad_y[i] = Soma_Grad_y[i] +
((volume[volume[i].viz1[j]].p_lin - volume[i].p_lin) * param[i].ey[j]);
        }
    }

    for (i=1; i<=num_pts; i++){
        volume[i].Grad_py_lin = Soma_Grad_y[i]/param[i].Soma_Ly;
    }
}

// Método

```

```

//                                     Jameson e Mavriplis (AIAA Journal, Vol. 24, pág. 611-18,
1986)
void calcula_grad_py_lin_JeM(tipo_volume volume[], tipo_param param[]){
    for (i=1; i<=num_pts; i++){
        volume[i].Grad_py_lin = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            if (volume[i].viz1[j] <= num_pts)
                volume[i].Grad_py_lin = volume[i].Grad_py_lin +
((volume[volume[i].viz1[j]].p_lin
                * 0.5 * param[i].S[j] * param[i].ey[j])
                / volume[i].vol_diag);
        }
    }
//    printf ("\nGrad_py'[%d] = %lf", i, volume[i].Grad_py_lin);
//
}
// Método do mínimo resíduo quadrático
//                                     Taniguchi et alli (Computer & Fluids, Vol. 19, Nº. 34, 287-95,
1991)
void calcula_grad_py_lin_Tan(tipo_volume volume[]){
    for (i=1; i<=num_pts; i++){
        volume[i].Grad_py_lin = ((Soma_1[i] * Soma_5[i]) - (Soma_2[i]
                * Soma_3[i]))/((Soma_1[i] * Soma_4[i])
                - (Soma_2[i] * Soma_2[i]));
    }
}

// DEFINICAO DA FUNÇÃO QUE CALCULA A VELOCIDADE U CORRIGIDA
void calcula_v_corrigido(){

```

```
double v_lin[N_PONTOS];
switch (grad_type){
    case 1:{
        calcula_grad_py_lin_Mal(volume, param);
    }
    break;

    case 2:{
        calcula_grad_py_lin_Nos(volume, param);
    }
    break;

    case 3:{
        calcula_grad_py_lin_JeM(volume, param);
    }
    break;

    case 4:{
        calcula_grad_py_lin_Tan(volume);
    }
    break;
}

for (i=1; i<=num_pts; i++)
    v_lin[i] = (-1 * volume[i].vol_diag * volume[i].Grad_py_lin)/coefic[i].Ap;

for (i=1; i<=num_pts; i++){
    volume[i].v = volume[i].v + v_lin[i];
}
```

```

    }
}

// DEFINICAO DA FUNÇÃO QUE ATUALIZA AS CONDIÇÕES DE CONTORNO
void atualiza_contorno(){
    printf ("\nAtualizando o contorno...");
}

// DEFINICAO DA FUNÇÃO QUE CALCULA O CRITÉRIO DE PARADA
double calcula_criterio_parada(){
double erro[N_PONTOS];
double Erro = 0.0;
    calcula_F(volume, param, coefic);
    for (i=1; i<=num_pts; i++){
        erro[i] = 0.0;
        for (j=1; j<=volume[i].n_pts; j++){
            erro[i] = erro[i] + coefic[i].F[j];
        }
        if (fabs(erro[i]) > Erro)
            Erro = fabs(erro[i]);
    }
    return Erro;
}

// DEFINICAO DA FUNÇÃO QUE IMPRIME OS RESULTADOS FINAIS
void imprime_result(tipo_volume volume[]){
    printf ("\nImprimindo os resultados finais...");
    printf ("\ntipo do gradiente: %d", grad_type);
}

```

```
for (i=1; i<=num_pts; i++){
    printf ("\n%i\t%lf\t%lf\t%lf", i, volume[i].u, volume[i].v, volume[i].p);
}
}

// DEFINICAO DA FUNÇÃO PRINCIPAL
void main(){
int tempo;
double Erro;
    // Rotinas para ler a malha
    le_malha();

    // Rotina para atribuir os parâmetros iniciais
    atribui_param_iniciais(volume, param);

    // Rotina para calculo da norma do vetor normal de cada face dos volumes de
controle
    calcula_norma(volume, param);

    // Rotina para calculo dos parâmetros geométricos
    calcula_param_geom(volume, param);

    // Rotina para calculo do Rhoij
    calcula_Rho(volume, param);

    // Rotina para calculo do Gamaj
    calcula_Gama(volume, param);
```



```

// Rotina para calculo do A0i
calcula_A0(volume, coefic, param);

// Rotina para calculo do Dij para Gamaij constante
calcula_D(volume, param, coefic);

// Rotina para calculo do Wij
calcula_W(volume, param);

if (grad_type == 4)
    // Rotina para calculo do Somatório ...
    calcula_soma_Tan();

//struct timeval time1, time2;
/* -----*/

// A linha abaixo, serve para pegar o valor em "Segundos" do Início dos Cálculos de
sua rotina.

gettimeofday(&time1, NULL);

for (tempo=1; tempo<=1; tempo++){
    for (i=1; i<=num_pts; i++){
        volume[i].u_old = volume[i].u;
        volume[i].v_old = volume[i].v;
        volume[i].p_old = volume[i].p;
    }
}

// for (num_iter=1; num_iter<=N_ITERACOES; num_iter++){

```

```
do{  
    num_iter++;  
    printf ("\n%i", num_iter);  
  
    // Rotinas para calculo dos coeficientes  
    calcula_coeficientes();  
  
    // Rotina para calculo da velocidade u  
    calcula_velocidade_u();  
  
    // Rotina para calculo da velocidade v  
    calcula_velocidade_v();  
  
    // Rotina para calculo do Wij  
    calcula_W(volume, param);  
  
    // Rotina para zerar p linha  
    zera_p_lin();  
  
    // Rotina para calculo da correção de pressao p linha  
    calcula_p_lin();  
  
    // Rotina para calculo da pressao corrigida  
    calcula_p_corrigido(volume);  
  
    // Rotina para calculo do critério de parada  
    Erro = calcula_criterio_parada();
```

```

// Rotina para correção da velocidade nas faces dos volumes de controle
para teste

// da conservação da massa
calcula_W_corrigido(volume, param);

// Rotina para corrigir a velocidade u
calcula_u_corrigido();

// Rotina para corrigir a velocidade v
calcula_v_corrigido();

// Rotina para atualizar as condições de contorno
atualiza_contorno();

if ((double) num_iter/100 > (double) 4.0){ // começa imprimir depois da
iteração 400)

    // Malha simuladamente estruturada com 100 pontos
/*
double u_medio = (volume[41].u+volume[51].u)/2;
printf ("\nu[y=5] = %.5lf", u_medio);
u_medio = (u_medio + (volume[42].u+volume[52].u)/2)/2;
printf ("\nu[y=10] = %.5lf", u_medio);
u_medio = (volume[45].u+volume[55].u)/2;
printf ("\nu[y=45] = %.5lf", u_medio);
u_medio = ((volume[45].u+volume[55].u)/2 +
(volume[46].u+volume[56].u)/2)/2;
printf ("\nu[y=50] = %.5lf", u_medio);
u_medio = (volume[49].u+volume[59].u)/2;
printf ("\nu[y=85] = %.5lf", u_medio);

```

```

u_medio = (volume[50].u+volume[60].u)/2;
printf ("\nu[y=95] = %.5lf", u_medio);
double v_medio = (volume[15].v + volume[16].v)/2;
printf ("\nv[x=15] = %.5lf", v_medio);
    v_medio = ((volume[45].v + volume[46].v)/2 + (volume[55].v +
volume[56].v)/2)/2;
    printf ("\nv[x=50] = %.5lf", v_medio);
    v_medio = (volume[85].v + volume[86].v)/2;
    printf ("\nv[x=85] = %.5lf", v_medio);
    v_medio = (v_medio + (volume[95].v + volume[96].v)/2)/2;
    printf ("\nv[x=90] = %.5lf", v_medio);
    v_medio = (volume[95].v + volume[96].v)/2;
    printf ("\nv[x=95] = %.5lf", v_medio);    /*
// Malha hexagonal com 95 pontos
// malha para v
/*
printf ("\nv[x=15] = %.5lf", volume[15].v);
    double v_medio = ((volume[43].v+volume[44].v)/2 + volume[53].v)/2;
    printf ("\nv[x=50] = %.5lf", v_medio);
    v_medio = (volume[82].v + volume[81].v)/2;
    printf ("\nv[x=85] = %.5lf", v_medio);
    v_medio = (v_medio + volume[91].v)/2;
    printf ("\nv[x=90] = %.5lf", v_medio);
    printf ("\nv[x=95] = %.5lf", volume[91].v); */
// malha para u
/*
double u_medio = (volume[5].u + volume[6].u)/2;
    printf ("\nu[y=5] = %.5lf", u_medio);
    u_medio = (u_medio + volume[15].u)/2;
    printf ("\nu[y=10] = %.5lf", u_medio);

```

```

u_medio = (volume[43].u + volume[44].u)/2;
printf ("\nu[y=45] = %.5lf", u_medio);
u_medio = (u_medio + volume[53].u)/2;
printf ("\nu[y=50] = %.5lf", u_medio);
u_medio = (volume[81].u + volume[82].u)/2;
printf ("\nu[y=85] = %.5lf", u_medio);
printf ("\nu[y=95] = %.5lf", volume[91].u); */
// Malha hexagonal com 613 pontos
printf ("\nv[x=06] = %.5lf", volume[296].v);
printf ("\nv[x=09] = %.5lf", volume[297].v);
printf ("\nv[x=15] = %.5lf", volume[298].v);
printf ("\nv[x=22] = %.5lf", volume[300].v);
printf ("\nv[x=50] = %.5lf", volume[307].v);
double v_medio = (volume[314].v + volume[315].v)/2;
printf ("\nv[x=80] = %.5lf", v_medio);
printf ("\nv[x=86] = %.5lf", volume[316].v);
printf ("\nv[x=90] = %.5lf", volume[317].v);
printf ("\nv[x=94] = %.5lf", volume[318].v);
printf ("\nv[x=98] = %.5lf", volume[319].v);
double u_medio = (volume[37].u + volume[38].u)/2;
printf ("\nu[y=06] = %.5lf", u_medio);
printf ("\nu[y=10] = %.5lf", volume[62].u);
printf ("\nu[y=18] = %.5lf", volume[111].u);
printf ("\nu[y=26] = %.5lf", volume[160].u);
u_medio = (volume[282].u + volume[283].u)/2;
printf ("\nu[y=46] = %.5lf", v_medio);
printf ("\nu[y=50] = %.5lf", volume[307].u);
u_medio = (volume[380].u + volume[381].u)/2;

```

```

printf ("\nu[y=62] = %.5lf", u_medio);
printf ("\nu[y=74] = %.5lf", volume[454].u);
u_medio = (volume[527].u + volume[528].u)/2;
printf ("\nu[y=86] = %.5lf", u_medio);
u_medio = (volume[576].u + volume[577].u)/2;
printf ("\nu[y=94] = %.5lf", u_medio);
u_medio = (u_medio + volume[601].u)/2;
printf ("\nu[y=96] = %.5lf", u_medio);
printf ("\nu[y=98] = %.5lf", volume[601].u);
// Malha de pontos aleatórios, com 100 pts
/*
printf ("\nu[y=05] = %.5lf", volume[48].u);
printf ("\nu[y=10] = %.5lf", volume[49].u);
printf ("\nu[y=45] = %.5lf", volume[61].u);
printf ("\nu[y=50] = %.5lf", volume[1].u);
printf ("\nu[y=73] = %.5lf", volume[36].u);
printf ("\nu[y=85] = %.5lf", volume[16].u);
printf ("\nu[y=95] = %.5lf", volume[73].u);
printf ("\nu[y=97] = %.5lf", volume[26].u);
printf ("\nv[x=06] = %.5lf", volume[69].v);
printf ("\nv[x=09] = %.5lf", volume[76].v);
printf ("\nv[x=15] = %.5lf", volume[20].v);
printf ("\nv[x=23] = %.5lf", volume[90].v);
printf ("\nv[x=50] = %.5lf", volume[1].v);
printf ("\nv[x=80] = %.5lf", volume[47].v);
printf ("\nv[x=85] = %.5lf", volume[15].v);
printf ("\nv[x=90] = %.5lf", volume[66].v);
printf ("\nv[x=95] = %.5lf", volume[32].v); */
}

```

```
        printf ("\nErro = %.21lf", Erro);
    }
    while (Erro >= ERRO);
}

// gettimeofday(&time2,NULL);
//          printf("\nTempo   Inicial:           Seg=%ld   Micro-
Segundos=%ld",time1.tv_sec,time1.tv_usec);
//          printf("\nTempo   Final           :           Seg=%ld   Micro-
Segundos=%ld",time2.tv_sec,time2.tv_usec);

// Imprime os resultados finais
imprime_result(volume);

printf ("\n");
}
```

```
// Programa da Dissertação de Mestrado: voronoi.h
// Ultima alteração 16-12-96
// Ultima correção 30-10-96
// Autor: Fabian Corrêa Cardoso
// Cuidado!!! Programa para malha com 613 pontos nodais, ter atenção nas condições de
contorno.

#ifndef __VORONOI_H

#define __VORONOI_H

#define N_ITERACOES 150 // Número de iterações do algoritmo
#define N_PONTOS 742 // Número de pontos da malha + 41
#define MAX_VIZ 8 // Número máximo de vizinhos + 1

#define TOP_WALL_SLIP 740 // Ponto na parede deslizante, na parte de cima, para
quem olha de frente
#define RIGHT_WALL 730 // Ponto na parede isolada, no lado direito, para quem
olha de frente
#define BOTTOM_WALL 720 // Ponto na parede isolada, na parte de baixo, para
quem olha de frente
#define LEFT_WALL 710 // Ponto na parede isolada, no lado esquerdo, para
quem olha de frente

typedef struct{
int n_pts; // Número de vizinhos (e de vértices)
int viz1[MAX_VIZ]; // Pontos vizinhos
double x[MAX_VIZ], // Coordenadas dos vértices (pontos de Voronoi)
```



```
    y[MAX_VIZ];  
    double cx, cy;    // Coordenadas do ponto nodal  
    double u;  
    double u_old;  
    double v;  
    double v_old;  
    double p;  
    double p_old;  
    double p_lin;  
    double Grad_Px;  
    double Grad_Py;  
    double Grad_Pw[MAX_VIZ];  
    double Grad_px_lin;  
    double Grad_py_lin;  
    double vol_diag;  
    double dw[MAX_VIZ];  
    } tipo_volume;  
  
typedef struct{  
    double Rho[MAX_VIZ];  
    double rho;  
    double Gama[MAX_VIZ];  
    double gama;  
    double L[MAX_VIZ];  
    double Soma_L;  
    double Soma_Lx;  
    double Soma_Ly;  
    double S[MAX_VIZ];
```

```
double ex[MAX_VIZ];  
double ey[MAX_VIZ];  
double W[MAX_VIZ];  
} tipo_param;
```

```
typedef struct{  
double F[MAX_VIZ];  
double D[MAX_VIZ];  
double A[MAX_VIZ];  
double a[MAX_VIZ];  
double A0;  
double Ap;  
double ap;  
double bu;  
double bv;  
} tipo_coefic;
```

```
#endif
```