UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE MATEMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA APLICADA

Geração de Malhas, Condições de Contorno e Discretização de Operadores para Dinâmica de Fluidos Computacional

por

Dagoberto Adriano Rizzotto Justo

Dissertação submetida como requisito parcial
para a obtenção do grau de

Mestre em Matemática Aplicada

Prof. Rudnei Dias da Cunha, D.Phil.
Orientador

Porto Alegre, maio de 2001.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Justo, Dagoberto Adriano Rizzotto

Geração de Malhas, Condições de Contorno e Discretização de Operadores para Dinâmica de Fluidos Computacional / Dagoberto Adriano Rizzotto Justo.—Porto Alegre: PPGMAp da UFRGS, 2001.

117 p.: il.

Dissertação (mestrado) —Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Matemática Aplicada, Porto Alegre, 2001.
Orientador: da Cunha, Rudnei Dias

Dissertação: Matemática Aplicada
Geração de Malhas, Dinâmica de Fluidos Computacional, Coordenadas Generalizadas

Geração de Malhas, Condições de Contorno e Discretização de Operadores para Dinâmica de Fluidos Computacional

por

Dagoberto Adriano Rizzotto Justo

Dissertação submetida ao Programa de Pós-Graduação em Matemática Aplicada do Instituto de Matemática da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do grau de

Mestre em Matemática Aplicada

Linha de Pesquisa: Métodos Analíticos e Numéricos em Dinâmica de Fluidos

Orientador: Prof. Rudnei Dias da Cunha, D.Phil.

Banca examinadora:

José Alberto Cuminato, Ph.D.
ICMC-SC/USP

João Luiz Comba, Ph.D.
PGCC/II/UFRGS

Julio Cesar Ruiz Claeyssen, Ph.D.
PPGMAp/IM/UFRGS

Dissertação apresentada e aprovada em
25 de Maio de 2001.

Prof. Vilmar Trevisan, Ph.D.
Coordenador

À minha mãe

AGRADECIMENTOS

- À minha mãe por ter me ensinado que o mais importante é ter a liberdade de escolher e ao meu mano pelos momentos de lazer.

- À Manu pela ajuda com as correções e as figuras e principalmente por seu carinho...

- Ao Prof. Rudnei, pois desde a graduação tem sido alguém em quem posso me espelhar.

- Aos colegas Ana, Carol, Carlos, Denise, Dulcinéia e Sânzara, pelo companheirismo e discussões matemáticas sobre o nosso e sobre os vossos trabalhos. Em especial à colega Sani que tornou-se uma grande amiga e incentivadora dos meus estudos.

Sumário

Lista de Figuras

Lista de Tabelas

LISTA DE SíMBOLOS

x, y, z	coordenadas no espaço físico.
ξ, η, ζ	coordenadas no espaço lógico.
$U_k, \partial U_k$	espaço lógico de dimensão k e seu contorno.
$\Omega_k^n, \partial\Omega_k^n$	objeto k-dimensional no espaço de dimensão n e seu contorno.
$X_k^n, \partial X_k^n$	transformação do espaço lógico para o espaço físico.
$[\ \ ]$	notação para vetores.
$\mathbf{a}, \mathbf{x}$	notação para vetores.
$\mathcal{J}, \mathcal{I}$	notação para matrizes.
$\mathcal{J}, \mathcal{J}^T$	matriz jacobiana e sua transposta.
$J, \sqrt{g}$	Jacobiano ou determinante da matriz jacobiana.
$\nabla_{\mathbf{x}}\xi, \nabla_{\mathbf{x}}\eta, \nabla_{\mathbf{x}}\zeta$	base de vetores normais contravariantes.
$\mathbf{x}_\xi, \mathbf{x}_\eta, \mathbf{x}_\zeta$	base de vetores tangentes covariantes.
U, V	componentes da velocidade contravariante.
$\mathcal{G}$	tensor métrico.
$g_{i,j}$	elementos do tensor métrico.
$g^{i,j}$	elementos do tensor métrico inverso.
$\nabla^2_{\boldsymbol{\xi}}, \nabla^2_{\mathbf{x}}$	operador Laplace com respeito as variáveis lógicas e físicas.
$\nabla_{\mathbf{x}}$	operador vetorial gradiente com respeito as variáveis físicas.

RESUMO

Neste trabalho desenvolvemos um PSE (Ambiente para Solução de Problemas) para geração de malhas. O programa, chamado MakeGrid, fornece uma interface amigável para a definição da região física a ser discretizada. Esta região pode ser simplesmente ou multiplamente conexa. Vários geradores de malha foram implementados entre métodos algébricos e elípticos, incluindo o gerador TTM não-homogêneo para atração da malha para um ponto ou linha coordenada específica. O programa também oferece ferramentas gráficas para a análise qualitativa da malha gerada.

Utilizando as idéias do método MAC, introduzimos o uso de uma malha de estados para a identificação das células em uma malha generalizada. Assim, cada ponto da malha possui um valor associado informando se este ponto pertence a um corte, uma célula fictícia ou a condição de contorno que deve ser aplicada neste ponto. Pode-se especificar uma condição de contorno diferente para cada ponto da malha para cada variável primitiva utilizada. O programa retorna arquivos contendo a malha de estados e a discretização das condições de contorno requeridas. Os arranjos co-localizado e diferenciado são apresentados como possíveis escolhas para a discretização dos operadores gradiente, divergente e Laplaciano. Estes são implementados para malhas cartesianas e generalizadas.

ABSTRACT

A PSE (Problem-Solving Environment) on Mesh Generation was developed. The program, entitled MakeGrid, provides a friendly interface for the definition of physical region boundaries, which can define simply or multiply-connected domains. Several options for mesh generators among algebraic and elliptical methods were implemented, including the inhomogeneous TTM method for attracting a mesh to a grid point or coordinate line. Besides, the program offers graphical tools for a qualitative analysis of the generated grid.

Based on the ideas of the MAC method, we introduce the use of a state grid for identification of cells in a general mesh. Each grid point receives a associated value which indicates if this point belongs to a cut branch, to a fictitious cells or what boundary condition is required for this point. The used methodology permits the definition of a different boundary condition for each grid cell. The user can define the boundary cells and the conditions to be applied on these cells in a practical way, receiving a state grid and a file containing the discrete form of the specified boundary conditions as result. The co-localized and staggered schemes are presented as possible choices for the gradient, divergent and Laplacian operators discrete form.

1 INTRODUÇÃO

A solução numérica de Equações Diferenciais Parciais (EDPs) requer alguma discretização do domínio da solução em uma coleção de pontos ou volumes elementares. Estas equações diferenciais são então aproximadas por um conjunto de equações algébricas e este é resolvido para produzir um conjunto discreto de valores que aproxima a solução da equação diferencial sobre a região. A geração de malhas surge da necessidade de dividir o domínio (limitado ou não) da solução em pequenos elementos (triângulos, quadriláteros, polígonos, tetraedros ou paralelepípedos) chamados *células*. A *malha*[1] propriamente dita é o conjunto dos pontos formados pelos vértices destes polígonos. Assim, a geração de malhas é um aspecto essencial de todos os métodos numéricos que empregam diferenças finitas, volumes finitos ou elementos finitos para a solução de EDPs.

Este tópico tornou-se um campo de pesquisa por si só, aumentando assim o vasto campo da Dinâmica de Fluidos Computacional (CFD) [32],[46]. Algumas considerações gerais relativas à geração de malhas e suas funções em CFD são as seguintes:

- Quase todo método funciona com uma boa malha, enquanto que um método menos sofisticado somente funciona com uma boa malha;
- Se tivermos uma resolução suficiente (i.e. pontos suficientes), então a qualidade da malha será de menor importância, desde que alguns requisitos básicos sejam satisfeitos; porém se há restrições quanto ao número de pontos, sua qualidade torna-se essencial;
- Uma boa malha pode acelerar a convergência da solução, enquanto que uma malha ruim pode levar à não convergência do método iterativo.

1.1 A transformação

Em muitas aplicações, é possível transformar a *região física* a um quadrado em duas dimensões (ou um cubo em três dimensões), de tal maneira que os contornos do quadrado correspondam aos contornos da região física (veja figura 1.1). Este quadrado é chamado *região lógica*, ou *região computacional*, e as linhas coordenadas uniformes correspondem a linhas na região física.

As variáveis x, y e z são as coordenadas no *espaço físico* e as variáveis ξ, η e ζ são as coordenadas no *espaço lógico*. Esses espaços são subconjuntos do $\mathbf{R}^n$, com $n = 1, 2, 3$. Algumas

[1] *Grid* ou *mesh*.

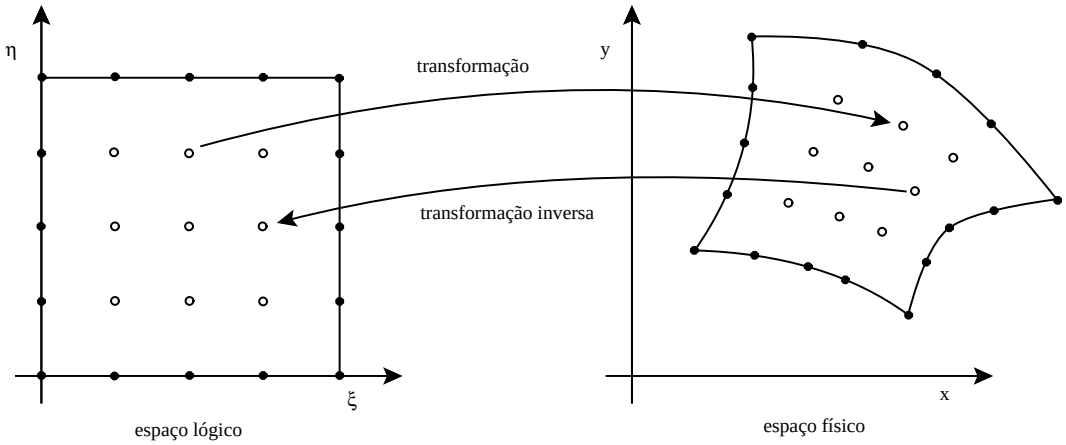


Figura 1.1: *Transformação e pontos da malha.*

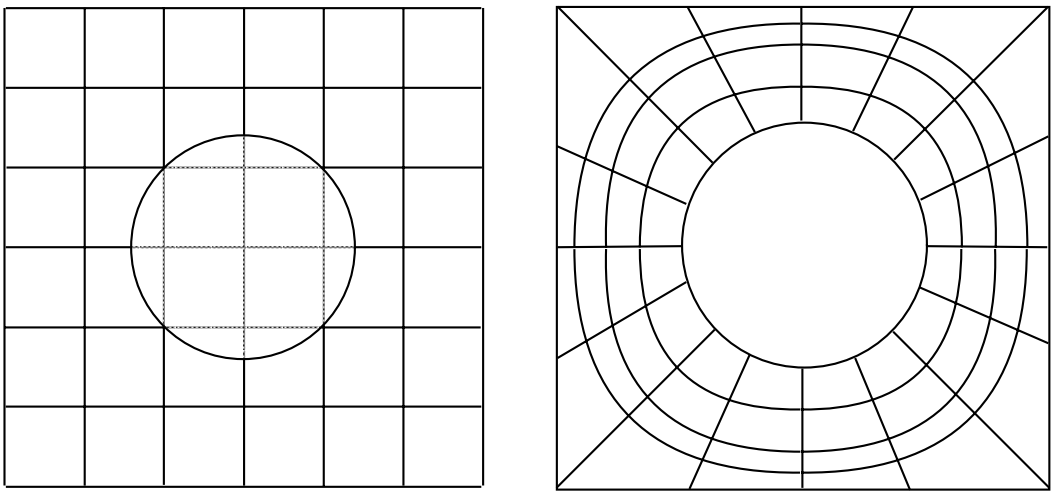
vezes é útil expressar as coordenadas com índices, usando para isto a notação vetorial. Assim, para o espaço lógico temos $\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3)$, enquanto que para o espaço físico temos $\mathbf{x} = (x_1, x_2, x_3)$.

O espaço lógico é escolhido como o intervalo unitário U_1 em $\mathbf{R}$, como o quadrado unitário U_2 em $\mathbf{R}^2$, ou como o cubo unitário U_3 em $\mathbf{R}^3$. Os contornos de ambas as regiões físicas e lógicas tem um papel importante nos cálculos numéricos (veja a tabela 1.1).

n	**Espaço lógico** U_k	**Contorno** ∂U_k
1	$U_1 = \{\xi \in \mathbf{R} : 0 \leq \xi \leq 1\}$	2 pontos
2	$U_2 = \{(\xi, \eta) \in \mathbf{R}^2 : 0 \leq \xi, \eta \leq 1\}$	4 segmentos 4 pontos
3	$U_3 = \{(\xi, \eta, \zeta) \in \mathbf{R}^3 : 0 \leq \xi, \eta, \zeta \leq 1\}$	6 faces 12 segmentos 8 pontos

Tabela 1.1: *Contornos do espaço lógico.*

Queremos definir uma *transformação*, ou *mapeamento*, entre esses espaços que seja inversível. A tarefa de resolver uma EDP numericamente na malha física apresenta uma dificuldade extra, pois a malha nem sempre é uniforme. Entretanto, podemos resolver a EDP transformada para a malha no espaço lógico, neste caso um quadrado com espaçamento uniforme, e então a tarefa de discretizar os operadores se tornará mais fácil. Outra vantagem em usar o espaço transformado está na simplificação da aplicação das condições de contorno, já que podemos fazer com que o contorno da malha no plano físico coincida com o contorno de um objeto (veja figura 1.2). O domínio da transformação é o espaço lógico enquanto que a imagem é o espaço físico. Essa transformação fornece um sistema de coordenadas generalizadas no objeto físico e possui dois parâmetros: a dimensão do objeto, k, e a dimensão do espaço físico, n. Um objeto k-dimensional no espaço físico n-dimensional é definido por Ω_k^n (por exemplo, uma curva no $\mathbf{R}^2$ é dada por Ω_1^2 enquanto que uma



(a) Malha Cartesiana

(b) Malha Generalizada

Figura 1.2: *Malha cartesiana e malha generalizada coincidente com a fronteira.*

região no $\mathbf{R}^3$ é dada por Ω_3^3). É necessário que o mapa leve o contorno ∂U_k do espaço lógico para o contorno $\partial\Omega_k^n$. Isto pode ser feito definindo a transformação dos contornos através do mapa

$$\partial X_k^n : \partial U_k \to \partial\Omega_k^n \tag{1.1}$$

e então estendendo este para o interior da região U_k obtendo a transformação

$$X_k^n : U_k \to \Omega_k^n \tag{1.2}$$

do interior do espaço lógico para o interior do espaço físico. É assumido que $0 < n \leq 3$ e $0 < k \leq n$. Podemos escrever esta transformação em notação vetorial como

$$\mathbf{x} = \mathbf{x}(\boldsymbol{\xi}). \tag{1.3}$$

Duas situações indesejáveis podem surgir: (i) um ponto no espaço lógico é mapeado para fora do objeto físico (isto é denominado como *dobra*[2] ou *sobra*[3] da malha), (ii) dois ou mais pontos no espaço lógico são mapeados para o mesmo ponto no objeto físico. É de importância fundamental realizar a transformação de tal forma que um ponto no espaço lógico seja mapeado para um único ponto no espaço físico (neste caso a transformação é chamada *um-a-um*). Isto é, para cada ponto $\boldsymbol{\xi} \in U_k$, existe um único ponto correspondente $\mathbf{x} \in \Omega_k^n$ e vice-versa.

Em relação a este problema, a quantidade matemática mais útil para o estudo da geração de malhas é a *matriz jacobiana* $\mathcal{J}$. As derivadas parciais $\partial x_i/\partial\xi_j$ estão definidas e con-

[2] *Folding.*
[3] *Spillover.*

seqüentemente os elementos de $\mathcal{J}$

$$\mathcal{J}_{ij} = \frac{\partial x_i}{\partial \xi_j}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, k \tag{1.4}$$

são definidos. O Jacobiano, J, é o determinante da matriz jacobiana e se ele for zero em algum ponto, então a transformação não será inversível e falhará em preservar propriedades físicas essenciais produzindo então malhas *dobradas*[4].

Também é conhecido que os erros de aproximação das equações a serem resolvidas dependem não somente das derivadas das equações e do espaçamento da malha, mas também da taxa de troca do espaçamento da malha e da ortogonalidade [45]. Para um dado espaçamento da malha, uma malha *ortogonal* e *suave* normalmente resulta em erros menores em problemas simples. Assim, outro objetivo principal na geração de malhas é a obtenção de malhas suaves, ou seja, malhas onde o espaçamento entre células adjacentes varie pouco e onde os ângulos entre as linhas da malha não se tornem muito pequenos. Entretanto, nem sempre é possível gerar uma malha ortogonal para uma dada geometria.

1.2 Problemas conceituais

O problema de gerar uma malha é não trivial, ou seja, não existe uma solução única. Dada uma região física, não existe apenas uma maneira de relacionar os pontos de um espaço para com o outro. Todo o processo de geração de malhas inicia com a definição de uma superfície (no caso bidimensional são linhas). Esta definição raramente é uma tarefa fácil. A entrada pode consistir de pontos, linhas, *splines*, curvas, retalhos de superfícies, etc. que especificam um objeto físico de forma *paramétrica*, *implícita* ou *numérica*. Depois que o objeto é dado parametricamente[5], então um conjunto discreto de pontos do objeto pode ser escolhido discretizando o parâmetro. Por exemplo, sendo ξ o parâmetro, podemos escolher $M+1$ pontos $\xi_i = i/M$, para $i = 0, \ldots, M$. Se o objeto é dado implicitamente, então precisamos encontrar a forma paramétrica que o descreve e algumas vezes precisamos reparametrizar[6] uma dada parametrização para o intervalo $[0, 1]$.

Existem alguns problemas conceituais que devem ser pensados quando escolhemos um sistema de geração de malhas para um problema particular:

- Domínio da solução delimitado ou não delimitado;
- Topologia do domínio (C, H, O e combinações delas);

[4] *Folded*.

[5] Um círculo de raio r, centrado na origem, é dado parametricamente através das equações $[x, y] = [r\cos\theta, r\sin\theta]$. Neste caso θ é o parâmetro que varia de 0 a 2π.

[6] Podemos reparametrizar o círculo através de $\xi = \theta/2\pi$ e teremos então o parâmetro ξ variando de 0 a 1.

- Um bloco ou múltiplos blocos;
- Tipo de interface entre blocos (contínua e suave);
- Com sobreposição ou não de malhas (ex. tipo Chimera);
- Malha estruturada ou não-estruturada;
- Geração algébrica ou de uma equação diferencial;
- Fixa ou adaptativa;
- Duas ou três dimensões.

Quando o domínio for delimitado, estamos definindo uma região do espaço, caso contrário, definimos somente um contorno interno da região e o outro contorno pode variar dependendo do método a ser utilizado para gerar a malha.

É importante notar que uma mesma região pode ser mapeada de formas diferentes. Por exemplo, um aerofólio pode ter uma malha do tipo C, do tipo O ou do tipo H. Dependo da maneira que definimos essas malhas podemos notar as suas diferenças e o motivo pelo qual recebem esses nomes (veja figura 1.3). Nem sempre é fácil mapear a região física para um quadrado unitário; precisamos, então, usar *múltiplos blocos* para definir a região física.

Malhas com sobreposição de dois ou mais blocos de malhas são denominadas serem do tipo *Chimera*. Para o tratamento desse tipo de malha, toda uma metodologia especial precisa ser utilizada.

Podemos discretizar o domínio tal que cada volume interno possua sempre o mesmo número de vizinhos e a numeração dos mesmos tenha uma seqüência natural. Neste caso, a malha será denominada *estruturada*. Uma das vantagens está no momento da implementação, pois o ordenamento dos elementos simplificará a generalização das rotinas. Além disso, a matriz resultante dessa discretização será uma matriz com elementos somente nas k primeiras diagonais, que são mais fáceis de se implementar e possuem métodos específicos para sua resolução. De outro modo, podemos discretizar o domínio tal que os elementos possuam um número variável de vizinhos e estejam dispostos sem nenhuma ordem aparente (veja a figura 1.4). Assim, teremos malhas *não-estruturadas* e as matrizes resultantes serão esparsas, dificultando algumas vezes a resolução do sistema associado.

A região a ser discretizada pode ser *convexa*[7] ou *não-convexa*. É relativamente mais fácil gerar malhas para regiões *convexas*, pois estas não apresentam reentrâncias do contorno sobre

[7] Uma região é *convexa* se quaisquer dois pontos nela podem ser ligados por uma *reta* inteiramente contida na região.

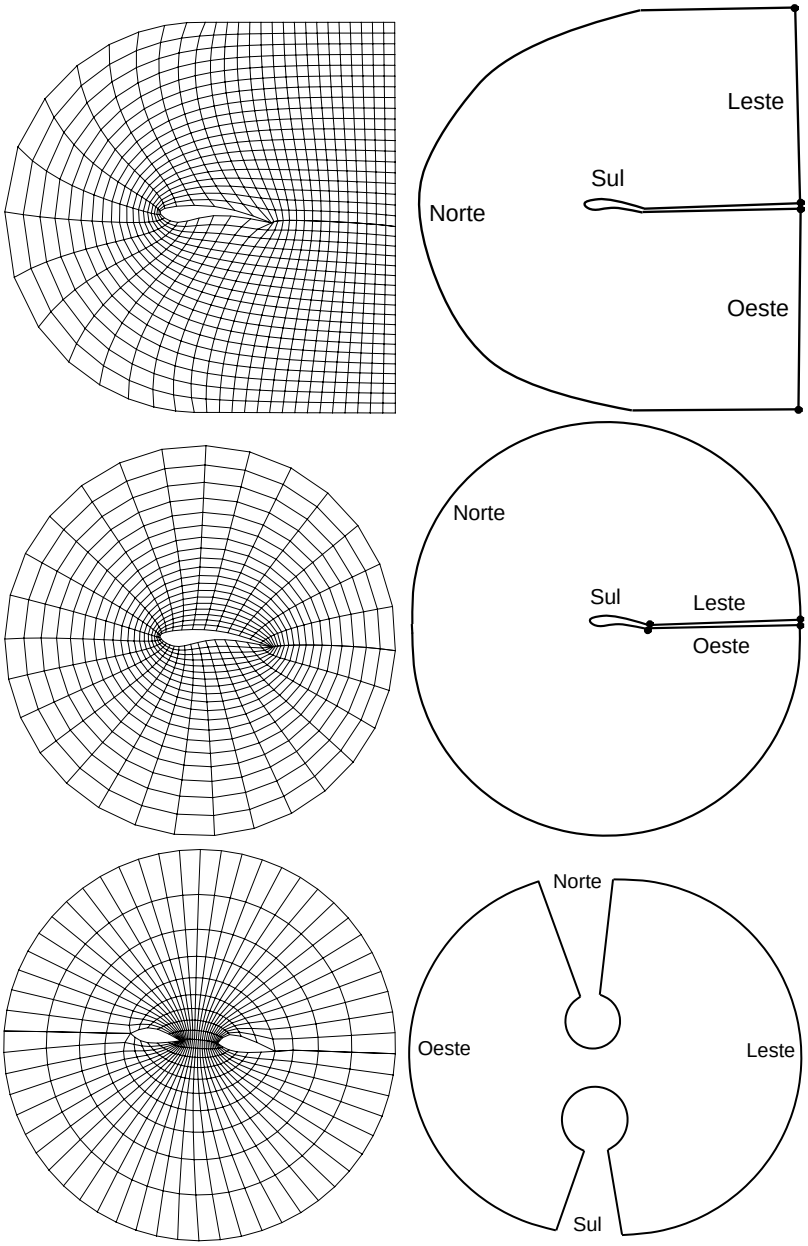


Figura 1.3: *Diversas topologias para malhas sobre aerofólios: (a) malha tipo C, (b) malha tipo O e (c) malha tipo H.*

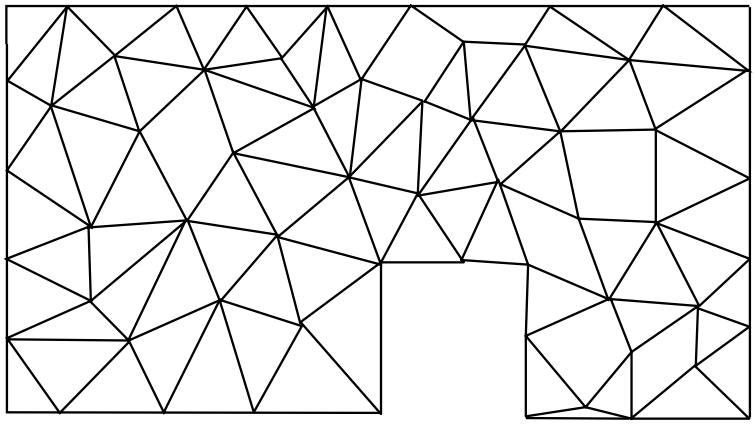


Figura 1.4: *Exemplo de malha não-estruturada. Normalmente este tipo de malha é usado com o método de elementos finitos.*

o interior da região. A região também pode ser classificada como *conexa*[8] ou *não-conexa*. Uma região conexa não possui duas regiões do espaço isoladas uma da outra. Em uma dimensão a única possibilidade é um intervalo. Em duas dimensões, entretanto, uma região conexa pode ser algo mais complicado; por exemplo, a região entre dois círculos, um anel, é conexa. Certas vezes, é necessário dividir a região física através de um *corte*[9], tornando-a uma região *simplesmente conexa*. Este é o caso de um anel e de um aerofólio (a malha tem um ou mais "buracos" no meio). O tratamento quanto às diversas maneiras de mapear uma região física dependerá da forma geométrica, das propriedades matemáticas desejadas para a malha e do tipo de equação a ser resolvida sobre a malha. Uma grande exposição sobre esses aspectos é dada no livro de Thompson et al [46, Cap. 2].

Cuidados especiais devem ser dados aos *cortes*. Fisicamente, esta região não apresenta nenhuma diferença a qualquer outro ponto interior da malha e, matematicamente, poderíamos ter localizado o corte em outra posição. Não podemos diferenciar os pontos que fazem parte do corte com os pontos interiores da malha. Isto significa dizer que, se temos continuidade das linhas coordenadas passando num ponto interior da malha, então desejamos ter continuidade também sobre as linhas coordenadas passando sobre um ponto do corte (veja figura 1.5). As linhas coordenadas precisam ser suaves nos pontos do corte tal qual possuem os outros pontos, caso contrário os erros de truncamento podem ser maiores nessa região do que na outra [46]. Esse tratamento é dado nesse trabalho e esta é a mesma metodologia necessária para a técnica de múltiplos blocos.

[8] Em uma região *conexa* quaisquer dois pontos pertencentes a região podem ser ligados por uma *curva* inteiramente contida na região.

[9] *Cut branch.*

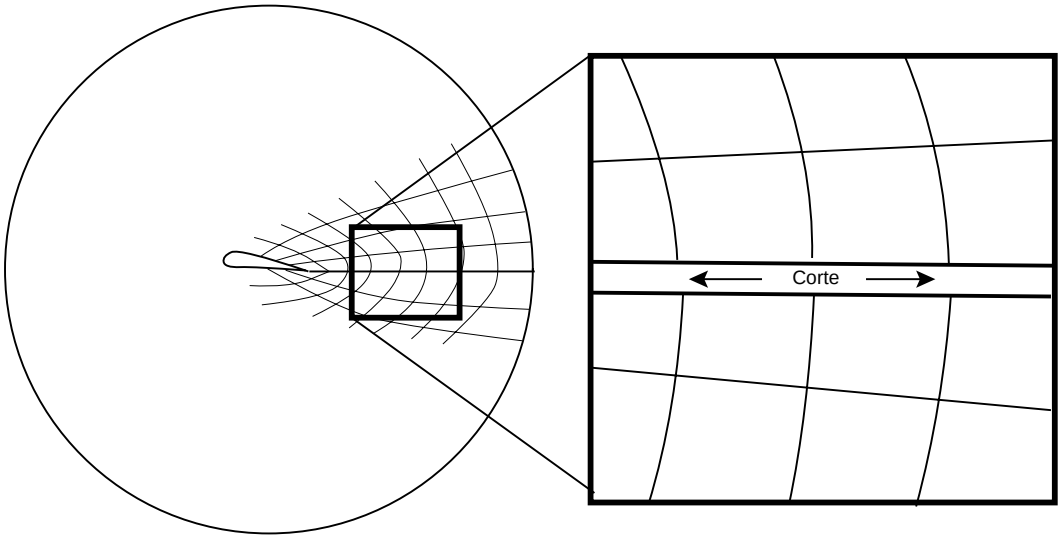


Figura 1.5: *Exemplo de malha sobre um aerofólio com detalhe sobre o* corte.

1.3 Métodos para Geração de Malhas

Abaixo estão descritos alguns dos principais métodos algébricos e diferenciais que fazem um resumo dos métodos disponíveis.

Métodos algébricos

São baseados na transformação de coordenadas através de uma expressão algébrica. Na sua forma mais simples, teremos uma expressão que fornece a transformação para cada ponto, bastando calcular o valor desta expressão. Para isso, podem ser usadas funções de interpolação de Lagrange e Hermite (também chamadas transformações de cisalhamento[10]). Alguns métodos são baseados em esquemas de interpolação em várias dimensões. A interpolação transfinita [24] e transformações multi-superfície produzem malhas boas para domínios fechados. A integração destes métodos com o controle adicional no contorno e dissipação elíptica fornecem eficientes sistemas para geração de malhas. Estes métodos, em sua forma mais desenvolvida, permitem algum controle nos valores das derivadas no contorno.

Métodos elípticos

São baseados na solução de equações parciais elípticas com algumas condições (chamadas termos fontes) para forçar as aproximações em certos pontos. O problema é formulado via

[10] *Shearing.*

um conjunto de equações de Poisson [45] com termos fontes normalmente definidos através de uma função de controle apropriada [39]. Sistemas elípticos produzem malhas suaves (às vezes suaves em excesso) e podem ser usados para suavizar descontinuidades nas métricas produzidas por interpolação transfinita (para esse propósito pode ser suficiente a utilização do operador de Laplace).

Métodos hiperbólicos

São baseados na solução de equações diferenciais hiperbólicas, que são resolvidas por esquemas de solução *marchante* do contorno do domínio. A idéia de usar EDPs hiperbólicas é muito eficiente para fluxos externos onde os contornos do corpo (aerofólio, asa, carro, etc.) são bem definidos, entretanto o contorno longe do corpo é deixado arbitrário. Esta situação também elimina a necessidade de especificar a distribuição de pontos em uma das arestas do domínio de fluxo, facilitando assim a definição do domínio em questão. Na formulação básica, o gerador hiperbólico é baseado na condição de ortogonalidade e na condição de volume de célula.

Métodos adaptativos

Todos os métodos descritos acima fazem uso de algum conhecimento empírico sobre a forma da solução da EDP. Este conhecimento faz com que adicionemos muitos pontos nas regiões de mais alto gradiente. Soluções melhores podem ser obtidas se for possível adaptar uma malha inicial com o esquema numérico marchante no tempo, seguindo exatamente a evolução do campo de gradientes, criando assim uma *malha adaptativa* (um problema particularmente difícil é a posição da onda de choque num fluxo transônico). Métodos que podem ser usados para seguir a solução incluem: funções peso, suavização de Poisson e analogia às equações da eletrodinâmica [32, seção 3.8]. O maior problema dos sistemas adaptativos é que eles devem ser construídos junto com o programa que soluciona numericamente a EDP.

Atualmente, não há um método que permita resolver todos os problemas desse tipo. Na geração de malhas estruturadas, métodos elípticos são os preferidos por permitirem um maior controle da forma e do interior da malha.

Uma vez que a malha tenha sido gerada, devemos visualizá-la verificando se não existem erros na sua configuração (estes podem ser de uma variedade enorme). Os métodos mais recentes tem sido embutidos em ferramentas multi-disciplinares que vem com uma interface amigável para o usuário, técnicas de tratamento de superfícies, ferramentas para visualização, pós-processamento, etc., onde freqüentemente são permitidas a possibilidade da utilização de malhas com múltiplos blocos.

1.4 Ambiente para Solução de Problemas (PSE)

As bibliotecas de softwares são atualmente organizadas em termos dos modelos matemáticos que suportam. Os avanços em *software*, *hardware*, *workstation clustering* e a tecnologia de máquinas paralelas, juntamente com a facilidade de acesso à supercomputação tem dificultado a tarefa de desenvolver novos programas científicos para o estudo de fenômenos da ciência e engenharia. Embora a biblioteca de software forneça alguma forma de abstração e a facilidade para reutilizar partes do software, ela ainda requer um nível de conhecimento além da experiência e habilidades do cientista e engenheiro médio que normalmente está envolvido no projeto de artefatos manufaturados. Este fato tem levado a um novo conceito de reaproveitamento de software chamado *Ambiente para Solução de Problemas*, PSE-*Problem Solving Environment* [1] .

Um PSE é um sistema computadorizado que fornece todas as facilidades computacionais necessárias para resolver uma classe alvo de problemas. Esta característica inclui métodos de solução avançada, seleção automática e semi-automática de métodos de solução e meios para facilmente incorporar novos métodos de solução. Além disso, PSEs usam a linguagem da classe de problemas alvo; assim, o usuário pode solucionar os problemas sem conhecimento especializado do hardware e software usado. Explorando tecnologias modernas tais como gráficos coloridos interativos, processadores poderosos e computação paralela, PSEs podem criar ferramentas para resolver problemas e permitem aos usuários revê-las facilmente. Sobretudo, PSEs são sistemas que fazem todo tipo de tarefa para todo tipo de usuário; PSEs resolvem problemas simples ou complexos, fornecem uma rápida construção de softwares ou análises detalhadas, e ainda podem ser usados em cursos introdutórios ou nas fronteiras da ciência.

Os atuais PSEs consistem de pequenos conjuntos de módulos, normalmente tomados de bibliotecas existentes, integrados (empacotados) para resolver uma classe predefinida de problemas da engenharia e da matemática. Evolução similar tem sido observada nas ferramentas de pré-processamento (CAD, geração de malhas) e pós-processamento (visualização de dados). Estas bibliotecas, interfaces e ferramentas de pré e pós-processamento aumentaram o nível de abstração da definição do problema e permitem a usuários com um mínimo de experiência computacional elaborar programas complexos. PSEs diferem de sistemas monolíticos pelo amplo domínio dos problemas ou aplicações que podem tratar. PSEs possuem extensibilidade, flexibilidade e oferecem facilidades na construção de projetos. A deficiência comum dos atuais PSEs é a perda do conhecimento adequado baseado nos sistemas para aplicabilidade, compatibilidade e performance (i.e. complexidade) de módulos, a seleção do usuário definindo parâmetros e a navegação. Um PSE ideal é aquele que pode tomar muitas decisões para o usuário consultando sua base de conhecimento associada.

1.5 Escopo deste trabalho

Neste trabalho desenvolvemos um programa, que pode ser chamado de um PSE, para a geração de malhas bidimensionais. Como será apresentado no capítulo 8, o programa denominado MakeGrid oferece opções para a definição dos contornos, incluindo o desenho de linhas, splines e arcos na tela com o auxílio do *mouse* e a leitura dos dados através de um arquivo.

Algoritmos práticos são enfatizados e muitos deles podem ser entendidos e implementados sem uma completa compreensão dos resultados matemáticos. Entretanto, o projeto destes algoritmos e a predição da forma da malha que os algoritmos geram depende potencialmente do entendimento matemático, portanto é importante ter em mente algumas informações antes de começar a gerar malhas.

Vamos tratar apenas de malhas estruturadas, visto que sua implementação é mais fácil e que o método usado para resolver a EDP neste caso é o método de diferenças finitas ou volumes finitos, enquanto que malhas não estruturadas são usadas com o método de elementos finitos [27, 2, 8]. Estaremos interessados tanto em domínios convexos como também os não-convexos. A região deve ser simplesmente conexa, o que pode ser alcançado através da utilização de cortes do domínio. A metodologia utilizada é baseada em um único bloco, porém sua extensão para múltiplos blocos é direta. Existe também um cuidado especial para que a interface entre os blocos tenha continuidade, como veremos na seção 3.6.

No capítulo 2, trataremos das malhas unidimensionais e a metodologia básica para o desenvolvimento de algoritmos para a geração de malhas. É importante salientar que malhas unidimensionais são usadas para definir os contornos de uma região bidimensional. As malhas bidimensionais serão tratadas no capítulo 3, onde serão vistos métodos para a sua geração, tais como a interpolação transfinita, métodos elípticos e hiperbólicos, os quais foram utilizados no programa. Não falaremos a respeito do *mapeamento conforme* que utiliza o plano complexo para fazer as transformações do espaço físico para o espaço lógico [15], devido à dificuldade de sua extensão para o caso tridimensional. Os livros texto de Knupp e Steinberg [32] e Thompson et al. [46] são a base deste trabalho e descrevem quase que por completo a geração de malhas estruturadas e a discretização de operadores. Alguns outros aspectos são mostrados em outras referências tais como Fletcher [20, cap. 12 e 13] e Peyret e Taylor [36, seção 11.4]. Alguns tópicos específicos sobre geração de malhas podem ser consultados em [10, 31, 41, 42, 43].

A transformação dos operadores do espaço físico para o espaço lógico é tratada no capítulo 4. A discretização dos operadores transformados é feita no capítulo 5, juntamente com uma descrição dos diversos tipos de arranjos que podem ser utilizados para as variáveis sobre

a malha. As condições de contorno e suas discretizações são tratados no capítulo 6 mostrando também a utilização das células fictícias.

Apresentamos também um método para resolver numericamente as equações de Navier-Stokes para um fluido incompressível no capítulo 7. Vários livros podem ser citados como referência para a solução das equações de Navier-Stokes tais como Ferziger e Peric [19], Malalasekera e Versteeg [47], Patankar [35], Maliska [34], Fortuna [22] e De Bortoli [5].

No capítulo 8, o programa MakeGrid é apresentado. Com a solução de um problema exemplo, o uso do programa é explicado passo a passo, com alguns detalhes de sua utilização e implementação. Um dos melhores meios de testar a qualidade de um gerador de malhas é testá-lo para um número diferente de problemas. Vários exemplos de malhas geradas com o programa são mostrados na seção 8.5.

2 GERAÇÃO DE MALHAS UNIDIMENSIONAIS

Vamos descrever neste capítulo a teoria básica envolvida na geração de malhas unidimensionais. Ela é necessária mesmo quando estamos no caso bidimensional, pois os contornos da região bidimensional serão tratados como malhas unidimensionais.

Primeiramente apresentaremos a idéia básica para o desenvolvimento de um algoritmo para a geração de malhas, seguindo com a definição da versão unidimensional de dois geradores clássicos para o caso bidimensional: o gerador AH e o gerador TTM. Por último, apresentaremos algumas maneiras de resolver numericamente as equações encontradas para a obtenção da malha.

2.1 A idéia básica

Uma malha no intervalo $[a, b]$ será dada por x_i, $i = 0, \ldots, M$, onde $x_0 = a$ e $x_M = b$. Queremos definir uma transformação $x_i = x(\xi_i)$ que relacione os pontos no espaço lógico ξ_i com os pontos x_i do espaço físico (ver figura 2.1). O espaço lógico será por definição igualmente espaçado de tal modo que $\xi_i = i/M$, $i = 0, \ldots, M$. Queremos também uma maneira de controlar o espaçamento entre dois pontos consecutivos que deve ser proporcional a uma função peso $\phi(\xi)$, para ξ no intervalo $[0, 1]$.

Como $\phi(\xi)$ é uma proporção, claramente $\phi(\xi) > 0$. O problema é gerar a malha tal que o comprimento dos intervalos $[x_i, x_{i+1}]$ seja proporcional à função ϕ no ponto médio do intervalo, ou seja, achar x_i tal que

$$x_{i+1} - x_i = K\phi\left(\frac{\xi_{i+1} + \xi_i}{2}\right), \quad 0 \leq i \leq M - 1, \tag{2.1}$$

onde K é uma constante positiva a ser encontrada.

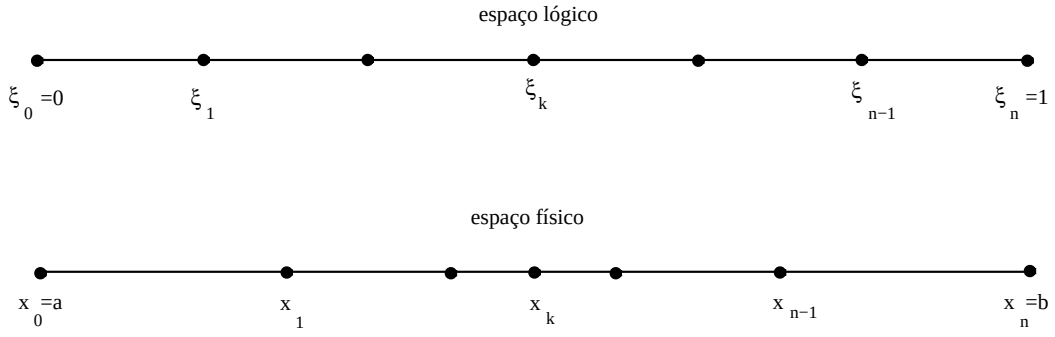


Figura 2.1: *Exemplo de malha unidimensional no espaço lógico e no espaço físico.*

Observe que se ϕ é contínua, então quando o comprimento do intervalo tende a zero, $\Delta\xi \to 0$, o lado esquerdo fica igual a zero enquanto que o lado direito não. Para contornar esse problema, façamos $K = C\Delta\xi$, onde C é uma constante arbitrária. Substituindo em (2.1) obtemos

$$\frac{x(\xi_{i+1}) - x(\xi_i)}{\Delta\xi} = C\phi\left(\frac{\xi_{i+1} + \xi_i}{2}\right). \tag{2.2}$$

Para um ponto qualquer $\xi \in [0, 1]$ temos

$$\frac{x(\xi + \Delta\xi) - x(\xi)}{\Delta\xi} = C\phi\left(\xi + \frac{\Delta\xi}{2}\right). \tag{2.3}$$

Como x é contínuo, então o limite da equação (2.3) quando $\Delta\xi \to 0$ fornece a equação diferencial ordinária

$$x_\xi(\xi) = C\phi(\xi). \tag{2.4}$$

Dividindo a equação (2.4) por ϕ, pois assumimos $\phi > 0$, e diferenciando com respeito a ξ temos

$$\left(\frac{x_\xi}{\phi}\right)_\xi = 0. \tag{2.5}$$

Note que quando diferenciamos, removemos a constante C do problema. Se ϕ é diferenciável e x é duas vezes diferenciável, usando a regra do quociente para derivar obtemos

$$x_{\xi\xi} - \frac{\phi_\xi}{\phi}x_\xi = 0. \tag{2.6}$$

A transformação deve satisfazer às condições de contorno $x(0) = a$ e $x(1) = b$. Esta equação diferencial *linear*, juntamente com as condições de contorno, determinam unicamente a transformação $x(\xi)$. O Jacobiano do mapa é $J = x_\xi$ e a transformação tem a propriedade que

$$x_\xi = J(\xi) = C\phi(\xi), \tag{2.7}$$

que é o análogo contínuo de (2.2).

A equação diferencial em (2.5) é a equação de Laplace a coeficientes variáveis na *forma conservativa*. Ela não requer que ϕ seja suave, portanto leva vantagem sobre a forma não-conservativa (2.6). Qualquer uma destas pode ser discretizada para obter um algoritmo numérico para a geração de uma malha unidimensional. A constante C não aparece em nenhuma das duas formas, assim não é necessária para a resolução numérica, mas pode ser útil conhecer o seu valor para solução analítica. Pode se verificar facilmente que C satisfaz

$$\frac{1}{C} = \frac{1}{b-a}\int_0^1 \phi(\xi)d\xi. \tag{2.8}$$

O objetivo da função peso é modificar uma malha uniforme tal que os pontos se aproximem ou se afastem de uma localização no plano físico. Nem sempre é fácil determinar uma função peso adequada para que a malha se concentre em torno de um ponto x_p escolhido. Para

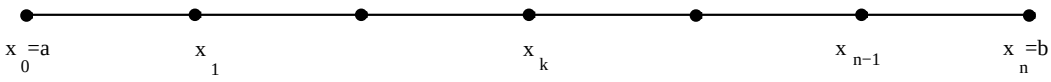


Figura 2.2: *Exemplo de malha com espaçamento uniforme.*

isso, devemos utilizar uma função peso dependente da variável x no espaço físico. Seguindo o mesmo raciocínio a partir de (2.1) e substituindo $\phi(\xi)$ por $w(x(\xi))$, podemos chegar a

$$\left(\frac{x_\xi(\xi)}{w(x(\xi))}\right)_\xi = 0 \tag{2.9}$$

que é o análogo da equação (2.5). Novamente usando a regra do quociente, diferenciamos e obtemos

$$x_{\xi\xi} - \frac{w_x}{w}x_\xi^2 = 0. \tag{2.10}$$

Usando a regra da cadeia obtemos

$$x_{\xi\xi} - \frac{w_\xi}{w}x_\xi = 0, \tag{2.11}$$

a qual, com as condições de contorno $x(0) = a$ e $x(1) = b$, é a equação a ser resolvida para obter a malha desejada. Observe que a equação (2.6) é linear, enquanto que a equação (2.11) é não-linear, o que pode dificultar a sua solução dependendo da escolha de $w(x)$.

2.1.1 Exemplos

Deve ser observado que para uma função $\phi(\xi)$ dada, teremos sempre a mesma solução, bastando resolver a equação diferencial apenas uma vez e utilizar a expressão encontrada para gerar a malha unidimensional. Desta forma estaremos utilizando um método algébrico para gerar a malha. Alguns exemplos são dados a seguir.

Malha com espaçamento uniforme

Para gerar uma malha com espaçamento uniforme temos que resolver a equação (2.5) com uma função peso constante, $\phi(\xi) = C$, ou seja, temos que resolver a equação diferencial

$$\left(\frac{x_\xi}{C}\right)_\xi = 0 \tag{2.12}$$

com $x(0) = a$ e $x(1) = b$. Integrando duas vezes em ξ e aplicando as condições de contorno obtemos

$$x(\xi) = (b - a)\xi + a. \tag{2.13}$$

Desta forma, discretizando o parâmetro ξ como $\xi_i = \frac{i}{M}$ e fazendo i, inteiro, variar de 0 até M, obtemos os $M + 1$ valores de x_i que definem a malha (veja figura 2.2).

Podemos utilizar essa metodologia para gerar uma malha unidimensional em $\mathbf{R}^2$, ou seja, gerar um conjunto de pontos em $\mathbf{R}^2$ ligando os pontos $\mathbf{x} = \mathbf{a}$ e $\mathbf{x} = \mathbf{b}$ conforme a figura 2.3.

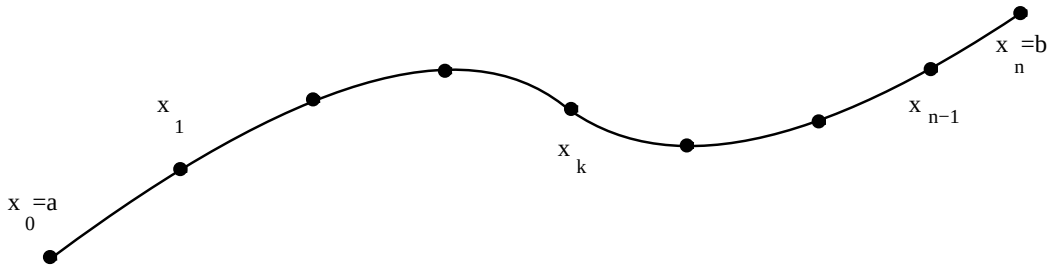


Figura 2.3: *Malha unidimensional no* $\mathbf{R}^2$ *com espaçamento uniforme.*

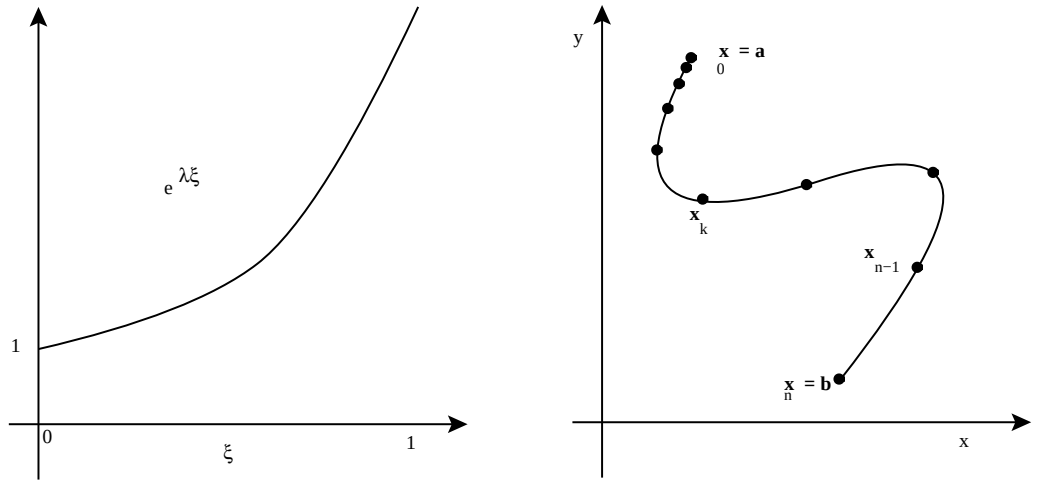


Figura 2.4: *Função peso* $\phi(\xi) = e^{\lambda\xi}$ *e malha aproximada na extremidade* $x = a$.

Concentração numa extremidade

Outra opção que podemos querer utilizar é atrair os pontos para um extremo a, a fim de termos uma malha mais refinada próximo desse extremo. Uma possível escolha para que os pontos ao se afastarem de a tenham o espaçamento entre eles aumentado é tornar a função peso igual a uma função exponencial, $\phi(\xi) = e^{\lambda\xi}$. Substituindo na equação (2.5) temos que resolver

$$\left(\frac{x_\xi}{e^{\lambda\xi}}\right)_\xi = 0 \tag{2.14}$$

com $x(0) = a$ e $x(1) = b$. Novamente, integrando duas vezes em ξ obtemos

$$x(\xi) = \frac{(b-a)e^{\lambda\xi} + ae^{\lambda} - b}{e^{\lambda} - 1}. \tag{2.15}$$

Na figura 2.4 temos a função peso $\phi(\xi) = e^{\lambda\xi}$ e a malha gerada aproximando os pontos para uma das extremidades.

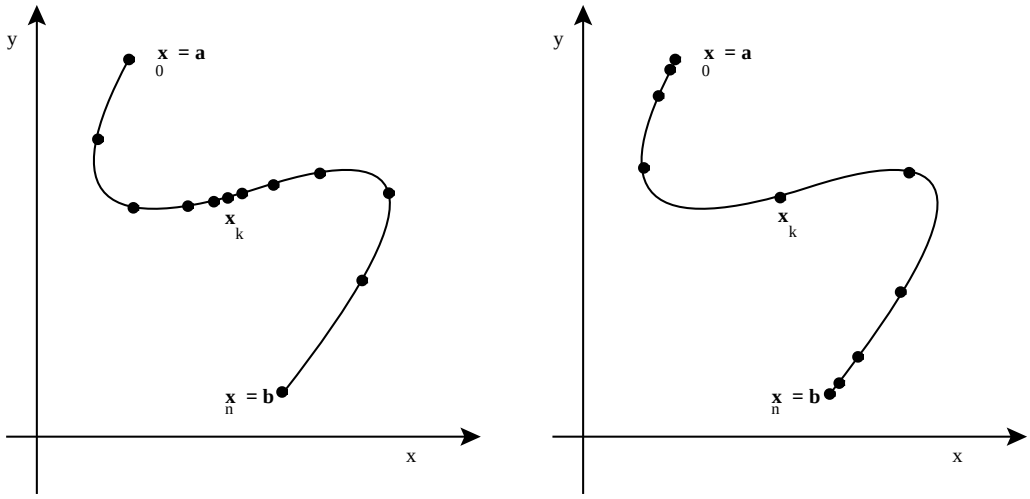


Figura 2.5: *Malhas com pontos aproximados (a) no ponto médio da curva e (b) em ambas as extremidades.*

Concentração para o centro do intervalo

Para aproximar para o ponto médio do intervalo $[a, b]$, usamos como função peso a composição de duas funções exponenciais tal que

$$\phi(\xi) = \begin{cases} e^{\lambda(1/2-\xi)} & \text{se } 0 \leq \xi \leq 1/2, \\ e^{\lambda(\xi-1/2)} & \text{se } 1/2 < \xi \leq 1. \end{cases} \tag{2.16}$$

e para aproximar para os pontos extremos do intervalo usamos

$$\phi(\xi) = \begin{cases} e^{\lambda\xi} & \text{se } 0 \leq \xi \leq 1/2, \\ e^{\lambda(1-\xi)} & \text{se } 1/2 < \xi \leq 1. \end{cases} \tag{2.17}$$

Seguindo a mesma metodologia, estas duas últimas funções peso podem fornecer as aproximações para as malhas geradas na figura 2.5.

2.2 Geradores através da Equação de Poisson

Nesta seção apresentaremos dois geradores de malhas unidimensionais baseados na equação de Poisson. A motivação para o uso da equação de Poisson é dada no livro de Maliska [34, pp. 254-260]. Estes geradores são a versão unidimensional dos geradores bidimensionais AH (Amsden e Hirt [3]) e TTM (Thompson, Thames e Mastin [44]). A equação de Poisson é utilizada pois as condições de contorno para o problema são os contornos da malha e as soluções que ela gera são suaves, uma propriedade desejada.

A versão unidimensional do gerador AH assume que a transformação satisfaz a equação de Poisson

$$x_{\xi\xi} = P(\xi), \quad x(0) = a, \quad x(1) = b, \tag{2.18}$$

onde P é uma função contínua dada que serve como uma função de peso no espaço lógico. Este é um problema de contorno linear e tem solução geral dada por

$$x(\xi) = (b-a)\xi + a + \xi \int_0^1 \eta P(\eta) d\eta - \xi \int_\xi^1 P(\eta) d\eta - \int_0^\xi \eta P(\eta) d\eta, \tag{2.19}$$

como pode ser facilmente checado.

O Jacobiano de (2.19) é

$$x_\xi = b - a - \int_\xi^1 P(\eta) d\eta + \int_0^1 \eta P(\eta) d\eta, \tag{2.20}$$

o que mostra que a principal dificuldade deste gerador é a relação entre o Jacobiano (neste caso o comprimento da célula) e a função peso é não trivial. Ainda que $P(\xi) > 0$, para $0 \leq \xi \leq 1$, nada garante que o Jacobiano seja sempre maior que zero. Entretanto, o gerador AH é o mesmo que os geradores (2.6) e (2.10) se

$$P = \frac{\phi_\xi}{\phi} x_\xi, \quad P = \frac{w_\xi}{w} x_\xi^2. \tag{2.21}$$

Em duas dimensões, o gerador AH produz malhas dobradas ainda para regiões simples. A generalização do gerador TTM é consideravelmente mais robusta. A idéia básica do gerador TTM é considerar a transformação inversa e assumir que esta função satisfaz o problema de valor de contorno

$$\xi_{xx} = P(\xi), \quad \xi(a) = 0, \quad \xi(b) = 1, \tag{2.22}$$

onde $P(\xi)$ é uma função contínua dada.

Para comparar o gerador TTM com os outros geradores, devemos transformar a equação para o espaço lógico. Lembrando que $x(\xi)$ e $\xi(x)$ são inversos um do outro e que $x(\xi(x)) = x$, diferenciando esta igualdade em relação a x obtemos $x_\xi \xi_x = 1$ ou $\xi_x = 1/x_\xi$. Aplicando duas vezes temos

$$\xi_{xx} = \frac{1}{x_\xi} \left(\frac{1}{x_\xi} \right)_\xi = -\frac{x_{\xi\xi}}{x_\xi^3}. \tag{2.23}$$

Aplicando em (2.22) obtemos o problema de contorno não-linear

$$x_{\xi\xi} + P(\xi) x_\xi^3 = 0, \quad x(0) = a, \quad x(1) = b. \tag{2.24}$$

2.3 Implementação Numérica

Os geradores de malhas apresentados nesta seção podem ser aproximados usando diferenças finitas. Várias aproximações diferentes vão ser usadas agora para os geradores derivados.

O problema é, novamente, calcular $M+1$ pontos x_i, $0 \leq i \leq M$, satisfazendo $x_0 = a$ e $x_M = b$. A aproximação em diferenças finitas para cada uma das equações diferenciais da seção anterior resultarão sempre num sistema de equações algébricas da forma

$$L_i x_{i-1} + C_i x_i + R_i x_{i+1} = G_i, \quad 1 \leq i \leq M-1, \tag{2.25}$$

onde

$$C_i = -(L_i + R_i). \tag{2.26}$$

Assim, somente R_i, L_i e G_i necessitam ser especificados para o problema.

Vamos discretizar agora as derivadas primeira e segunda de uma função. Esta discretização será usada para transformar a equação diferencial num sistema algébrico. Queremos que a discretização seja de segunda ordem e que o estêncil[1] seja simétrico.

Se $f(\xi)$ é uma função suave, então sua derivada no centro do intervalo $\Delta\xi$ é aproximada por

$$f_\xi(\xi) \approx \frac{f(\xi + \frac{\Delta\xi}{2}) - f(\xi - \frac{\Delta\xi}{2})}{\Delta\xi}, \tag{2.27}$$

enquanto que o valor de f no centro do intervalo pode ser aproximado pela média

$$f(\xi) \approx \frac{f(\xi + \frac{\Delta\xi}{2}) + f(\xi - \frac{\Delta\xi}{2})}{2}. \tag{2.28}$$

Como o espaço lógico que foi construído é uniforme e tem $M+1$ pontos então

$$\Delta\xi = \frac{1}{M}, \quad \xi_i = i\Delta\xi, \quad 0 \leq i \leq M. \tag{2.29}$$

Os pontos médios dos intervalos

$$\xi_{i+\frac{1}{2}} = \left(i + \frac{1}{2}\right)\Delta\xi, \quad 0 \leq i \leq M-1 \tag{2.30}$$

também são necessários.

Discretizando a equação (2.5) para o ponto x_i teremos

$$\left(\left(\frac{x_\xi(\xi)}{\phi(\xi)}\right)_\xi\right)_i \approx \frac{(\frac{x_\xi}{\phi})_{i+\frac{1}{2}} - (\frac{x_\xi}{\phi})_{i-\frac{1}{2}}}{\Delta\xi}, \tag{2.31}$$

onde

$$\left(\frac{x_\xi}{\phi}\right)_{i+\frac{1}{2}} \approx \frac{1}{\phi_{i+\frac{1}{2}}}\frac{x_{i+1} - x_i}{\Delta\xi}. \tag{2.32}$$

Os coeficientes desta equação serão

$$R_i = \frac{1}{\phi_{i+\frac{1}{2}}\Delta\xi^2}, \quad L_i = R_{i-1}, \quad G_i = 0. \tag{2.33}$$

[1] Entende-se por *estêncil* o conjunto de pontos que são levados em conta no momento da discretização de um operador sobre um determinado ponto.

O sistema (2.33) é um sistema tridiagonal simétrico que pode ser resolvido através do *algoritmo de Thomas*[2] [35]. Um tratamento especial deve ser dados aos pontos de contorno; assim, para $i = 1$, a equação pode ser expressa como

$$C_1 x_1 + R_1 x_2 = -L_1 a. \tag{2.34}$$

Para automatizar a operação podemos modificar os coeficientes de (2.25) tal que

$$L_1 = 0, \quad G_1 = -R_0 a. \tag{2.35}$$

Uma modificação similar será necessária em $i = M - 1$.

A equação (2.6) é também uma equação diferencial ordinária simétrica homogênea linear, entretanto ela não é escrita na forma simétrica. Diferenças centrais fornecem

$$R_i = \frac{4\phi_i - (\phi_{i+1} - \phi_{i-1})}{\Delta\xi^2}, \quad L_i = \frac{4\phi_i + (\phi_{i+1} - \phi_{i-1})}{\Delta\xi^2}, \quad G_i = 0. \tag{2.36}$$

Esta aproximação é não simétrica, assim (2.33) é preferida, embora o sistema (2.36) seja ainda tridiagonal e possa ser resolvido por um algoritmo para matrizes tridiagonais.

Quando estamos usando a função peso $w(x)$ que depende do espaço físico, a equação ordinária derivada é não-linear, mas ainda assim tem a mesma forma que as duas equações anteriores. A equação diferencial (2.9) permite

$$R_i = \frac{1}{w(x_{i+\frac{1}{2}})\Delta\xi^2}, \quad L_i = R_{i-1}, \quad G_i = 0. \tag{2.37}$$

Este é um sistema não-linear que pode ser resolvido usando um algoritmo de iteração não-linear.

Qualquer problema onde R_i, C_i, L_i ou G_i depende de x é não-linear. Tais problemas são resolvidos usando um algoritmo de iteração não-linear. Um parâmetro de tolerância pequeno, *tol*, que determina a precisão da solução deve ser dado. Para iniciar a iterar, uma solução inicial, neste caso uma malha x_i^{old}, deve ser gerada, podendo ser por exemplo uma interpolação linear. Então o laço descrito abaixo é executado até que a tolerância mínima seja alcançada, que pode ser medida através da norma da solução anterior menos a solução atual.

Algoritmo: Iteração não-linear

1. Enquanto ($\delta \leq tol$)
2. Calcule L_i, R_i, C_i e G_i usando x_i^{old}.
3. Resolva o sistema linear resultante para x_i.
4. Faça $x_i^{new} = x_i$.
5. Calcule $\delta = \max_i \|x_i^{new} - x_i^{old}\|$.

[2] Também chamado *TDMA-TriDiagonal Matrix Algorithm.*

O passo 3 pode ser resolvido através do algoritmo de Thomas ou por outro sistema iterativo, tal como Jacobi, Gauss-Seidel e SOR.

Uma abordagem similar pode ser usada para resolver a equação não-linear (2.36). Ao invés disso, usaremos um algoritmos diferente. Os termos de ordem baixa podem ser calculados na malha anterior para determinar o lado direito do estêncil e os termos de segunda ordem podem ser usados para calcular os coeficiente do lado esquerdo, ou seja,

$$R_i = L_i = 1, \quad G_i = \frac{(w_{i+1} - w_{i-1})(x_{i+1} - x_{i-1})}{4w_i \Delta\xi^2}. \tag{2.38}$$

Como L_i, R_i e C_i são constantes, um algoritmo particular e eficiente pode ser usado para resolver este sistema.

3 GERAÇÃO DE MALHAS BIDIMENSIONAIS

Este capítulo apresenta alguns métodos para a geração de malhas bidimensionais, começando pela descrição do problema básico de geração de malhas, seguindo com o método de *Interpolação Transfinita*. A solução gerada por este método servirá por vezes como condição inicial para os métodos iterativos apresentados a seguir, tais como os geradores elípticos, *Length* e *Winslow*. O capítulo termina com uma breve discussão de métodos parabólicos e sua utilização para a geração de células fictícias.

Apresentaremos mais detalhadamente os métodos elípticos pois foram esses os implementados no programa MAKEGRID e por estes obterem melhores resultados para geometrias diversas.

3.1 O problema de geração de malhas no plano

O problema básico para geração de malhas bidimensionais pode ser abordado do seguinte modo: dada uma região simplesmente conexa $\Omega \subset \mathbf{R}^2$ no espaço físico, queremos achar uma transformação $\mathbf{x} = \mathbf{x}(\xi, \eta)$ do quadrado unitário U_2 no espaço lógico para a região Ω. A região física é especificada fornecendo seu contorno; isto pode ser feito definindo-se um conjunto de quatro funções paramétricas (ver figura 3.1),

$$\mathbf{x}_N(s), \quad \mathbf{x}_S(s), \quad 0 \leq s \leq 1, \tag{3.1}$$

$$\mathbf{x}_E(s), \quad \mathbf{x}_W(s), \quad 0 \leq s \leq 1. \tag{3.2}$$

Os índices em $\mathbf{x}$ referem-se aos contornos *norte*, *sul*, *leste* e *oeste* no domínio lógico[1]. Note que $\mathbf{x}$ é um vetor e portanto $\mathbf{x}_N(s) = [x_N(s), y_N(s)]$ e a solução do problema será achar $\mathbf{x}(\xi, \eta) = [x(\xi, \eta), y(\xi, \eta)]$. É importante checar as quatro condições de consistência para os contornos dadas por

$$\mathbf{x}_S(0) = \mathbf{x}_W(0), \quad \mathbf{x}_S(1) = \mathbf{x}_E(0), \tag{3.3}$$

$$\mathbf{x}_N(0) = \mathbf{x}_W(1), \quad \mathbf{x}_N(1) = \mathbf{x}_E(1). \tag{3.4}$$

O problema básico possui outras variações. Por exemplo, o contorno da região pode ser reparametrizado antes que a extensão para o interior seja realizada. Então, se o parâmetro s for reparametrizado em função de r, teremos $s = s(r)$ com $0 \leq r \leq 1$. Assim, a função composta

[1] Usamos os subscritos indicando as direções cardeais, com as iniciais tomadas de seus nomes na língua inglesa, por consistência com a notação comumente usada na literatura

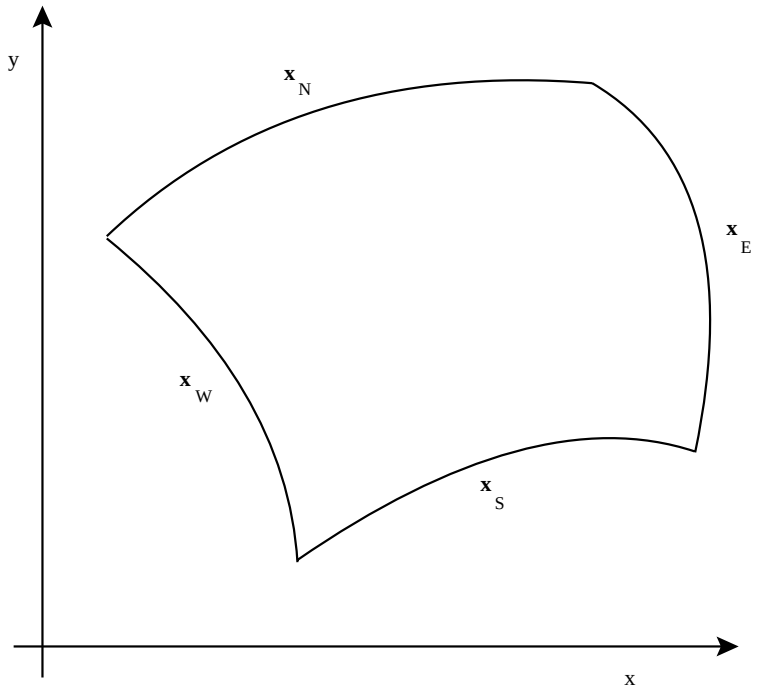


Figura 3.1: *Quatro funções paramétricas definem o contorno bidimensional.*

$\mathbf{x}_N = \mathbf{x}_N(s(r)) = \hat{\mathbf{x}}_N(r)$ é uma reparametrização do contorno norte de Ω. Esta é uma técnica importante já que o contorno original é freqüentemente inconveniente.

Outra modificação ao problema é reduzir o número de contornos ao que se deseja fixar. Por exemplo, somente três contornos são necessários em alguns geradores ortogonais [18] e somente um contorno é necessário em geradores hiperbólicos. Claramente, os outros contornos estão livres para variar e estão fora do controle do usuário. A necessidade de que a região seja simplesmente conexa também pode ser relaxada introduzindo cortes no domínio físico, como por exemplo no caso de cilindros e aerofólios onde um corte pode ser adicionado, ligando o contorno interno ao contorno externo da região, de tal forma a tornar a região simplesmente conexa. Regiões com alto grau de conectividade[2] podem ser tratadas adicionando-se múltiplos cortes (veja por exemplo a figura 3.2).

As transformações são calculadas usando aproximações discretas. Uma malha no espaço lógico é definida como um conjunto de pontos (ξ_i, η_i) tais que

$$\Delta\xi = \frac{1}{M}, \quad \xi_i = i\Delta\xi, \quad 0 \leq i \leq M, \tag{3.5}$$

$$\Delta\eta = \frac{1}{N}, \quad \eta_j = j\Delta\eta, \quad 0 \leq j \leq N. \tag{3.6}$$

[2] Por exemplo, a região sobre dois cilindros é multiplamente conexa.

Os valores inteiros de i, j são usados para os vértices (ou nós) das células. Algumas vezes, para certas discretizações, são necessários os pontos médios dos intervalos, os quais possuem significados geométricos diferentes, de acordo com a tabela 3.1 e a figura 3.3.

3.2 Interpolação Transfinita (TFI)

Métodos algébricos utilizando interpolações apresentam duas vantagens principais: o cálculo rápido da malha e o controle direto sobre a localização dos pontos na malha. A principal desvantagem é o fato que nem sempre esses métodos geram malhas suaves e podem também gerar malhas *dobradas*. Além disso, quando o contorno não é suave, estas não-suavidades são propagadas para o interior da malha.

O método padrão é conhecido como interpolação transfinita[3] (TFI) [24]. No caso unidimensional, a interpolação transfinita é a mesma que a interpolação linear. Os polinômios de Lagrange de primeiro grau $1-\xi$, ξ, $1-\eta$ e η são usados como funções de *mistura*[4] na fórmula de interpolação transfinita, que é

$$\begin{aligned}\mathbf{x}(\xi,\eta) = (1-\eta)\mathbf{x}_S(\xi) + \eta\mathbf{x}_N(\xi) + (1-\xi)\mathbf{x}_W(\eta) + \xi\mathbf{x}_E(\eta) \\ - \{\xi\eta\mathbf{x}_N(1) + \xi(1-\eta)\mathbf{x}_S(1) + \eta(1-\xi)\mathbf{x}_S(0) + (1-\xi)(1-\eta)\mathbf{x}_S(0)\}. \quad (3.7)\end{aligned}$$

Outras funções de interpolação são possíveis, como splines, polinômios de Hermite e de Bernstein-Bezier, visando a suavização das linhas da malha.

3.3 Geradores elípticos de malhas

Os geradores elípticos são baseados na resolução de sistemas de equações diferenciais parciais elípticas e têm como maior vantagem a suavidade no interior da malha. A suavidade da malha é necessária a fim de se obter erros de truncamento pequenos na solução da equação desejada através de métodos de diferenças finitas. Outra vantagem na abordagem elíptica é que podemos especificar os quatro contornos do domínio, satisfazendo assim a idéia básica de um problema de geração de malhas. O interior da malha é praticamente insensível a parametrização dos contornos, diferentemente dos métodos algébricos e hiperbólicos. Existem soluções para uma ampla variedade de domínios e mudam gradualmente quando os dados do contorno são trocados. O maior problema na aproximação elíptica é a perda de velocidade relativa aos outros métodos mencionados. Outro problema é a dificuldade na determinação da função de controle no interior da malha.

[3] *Transfinite Interpolation.*
[4] *Blending.*

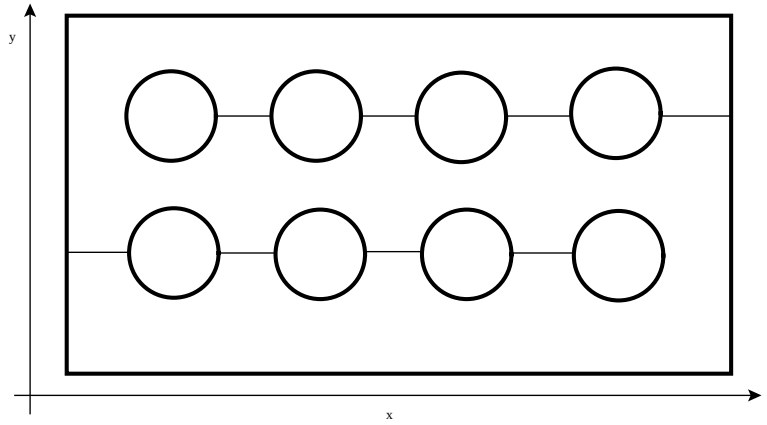


Figura 3.2: *Região multiplamente conexa transformada em uma região simplesmente conexa através de cortes.*

Local	pontos
vértices das células	$(i\Delta\xi, j\Delta\eta)$
aresta horizontal	$((i+\frac{1}{2})\Delta\xi, j\Delta\eta)$
aresta vertical	$(i\Delta\xi, (j+\frac{1}{2})\Delta\eta)$
centro da célula	$((i+\frac{1}{2})\Delta\xi, (j+\frac{1}{2})\Delta\eta)$

Tabela 3.1: *Posição física na malha e coordenadas* (i, j).

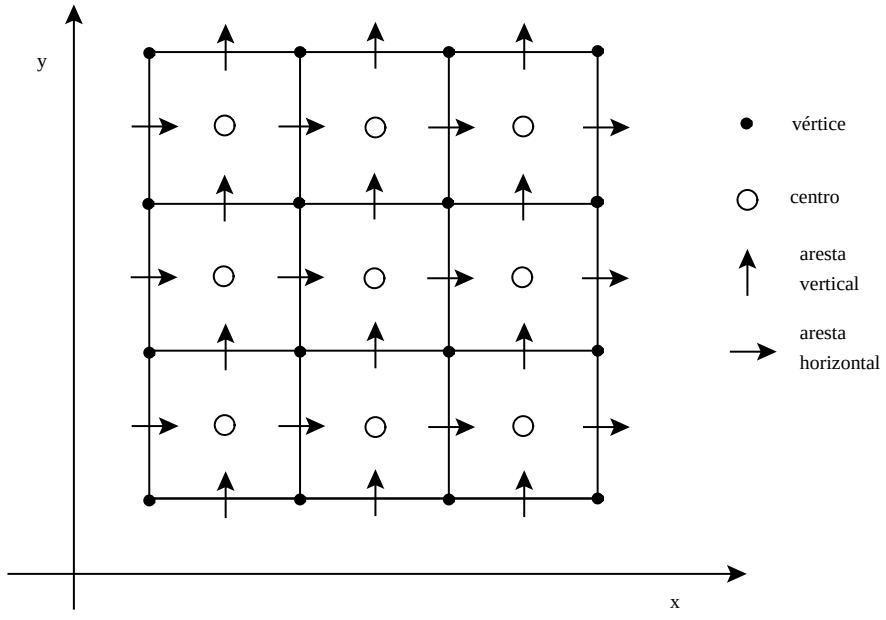


Figura 3.3: *Diferentes posições na malha.*

3.3.1 O gerador *Length*

O gerador elíptico mais simples, chamado gerador AH (Amsden-Hirt [3]) ou gerador *Length*, é a generalização do gerador AH unidimensional dado pela equação (2.18) no capítulo 2. Este gerador requer que cada componente do mapa satisfaça a equação de Laplace

$$\nabla^2 x = \nabla^2_{\boldsymbol{\xi}} x = x_{\xi\xi} + x_{\eta\eta} = 0, \quad \nabla^2 y = \nabla^2_{\boldsymbol{\xi}} y = y_{\xi\xi} + y_{\eta\eta} = 0. \tag{3.8}$$

A motivação para o uso desta equação é dada no livro de Maliska [34, pp. 254-260]. Os quatro contornos do mapa dados em (3.1),

$$x(\xi, 0) = x_N(\xi), \quad x(\xi, 1) = x_S(\xi), \tag{3.9}$$

$$x(0, \eta) = x_W(\eta), \quad x(1, \eta) = x_E(\eta), \tag{3.10}$$

fornecem as condições de contorno para a equação diferencial em x. Condições similares para y são definidas da mesma forma.

As equações para determinar x e y são *desacopladas*, *lineares* e formuladas no domínio lógico. A equação diferencial parcial é a equação de Laplace e as condições de contorno são chamadas de *condições de Dirichlet*, portanto este problema é chamado problema de contorno com condições de Dirichlet. Pode se mostrar (ver [4, 21, 23]) que tais problemas tem uma solução única e que a solução é infinitamente diferenciável (em $\mathcal{C}^\infty$) no interior de U_2, dado que o contorno do mapa é contínuo. A única questão em aberto é se a transformação é um-a-um, isto é, se o Jacobiano do mapa é diferente de zero; a resposta é negativa, pois as malhas tipicamente *dobram-se* para regiões não-convexas. Veja um exemplo de malha dobrada na figura 3.4.

Mesmo sem a garantia que a malha não dobrará, o gerador AH é ainda atrativo em alguns casos. O caso mais importante é o de domínios convexos tendo um ou mais pontos onde o contorno possua derivada descontínua. Neste caso, geradores algébricos falham em produzir malhas suaves, enquanto que o gerador AH produz uma malha suave e não dobrada. O cálculo da malha AH é relativamente rápido comparado a outros geradores elípticos não-lineares, pois as equações deste gerador são lineares.

A solução numérica da equação AH é uma tarefa fácil em métodos numéricos. A derivada segunda da equação de Laplace é discretizada usando diferenças centrais tal que

$$\frac{x_{i-1,j} - 2x_{i,j} + x_{i+1,j}}{\Delta\xi^2} + \frac{x_{i,j-1} - 2x_{i,j} + x_{i,j+1}}{\Delta\eta^2} = 0, \tag{3.11}$$

para $1 \leq i \leq M-1$, $1 \leq j \leq N-1$. As condições de contorno são dadas por (3.9) e (3.10). Existem várias maneiras de resolver este conjunto de equações; dentre estas, escolhemos o seguinte algoritmo.

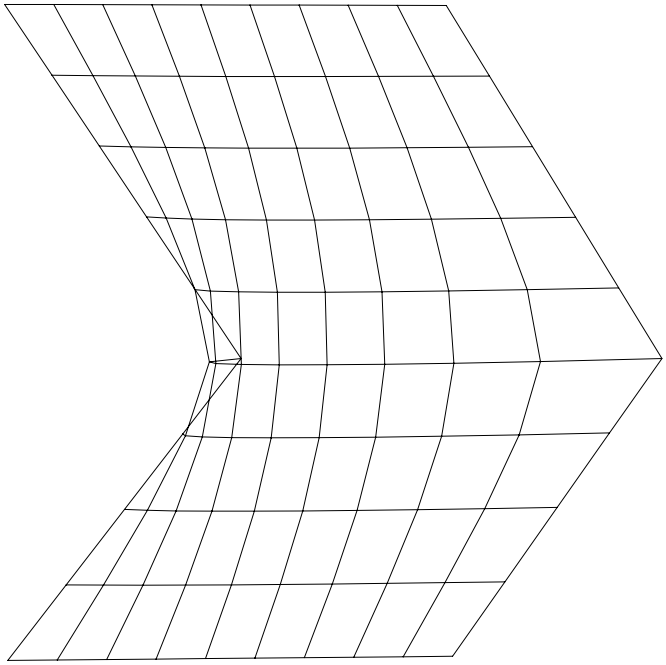


Figura 3.4: *Malha dobrada para o gerador* Length.

Algoritmo: Resolução da equação de Laplace.

1. Defina as condições de contorno $\mathbf{x}_{i,0}$, $\mathbf{x}_{i,M}$, $\mathbf{x}_{0,j}$ e $\mathbf{x}_{M,j}$.
2. Enquanto $\delta < tol$
3. Para $i = 1$ até $M-1$, $j = 1$ até $N-1$
4. $x_{i,j}^n = \frac{\Delta\xi^2\Delta\eta^2}{2\Delta\xi^2+2\Delta\eta^2}(\frac{x_{i-1,j}+x_{i+1,j}}{\Delta\xi^2} + \frac{x_{i,j-1}+x_{i,j+1}}{\Delta\eta^2})$
   $y_{i,j}^n = \frac{\Delta\xi^2\Delta\eta^2}{2\Delta\xi^2+2\Delta\eta^2}(\frac{y_{i-1,j}+y_{i+1,j}}{\Delta\xi^2} + \frac{y_{i,j-1}+y_{i,j+1}}{\Delta\eta^2})$
5. $\delta = \max_{i,j} \left\| x_{i,j}^n - x_{i,j} \right\|$

3.3.2 O gerador *Winslow*

O gerador de malhas mais amplamente usado dentre os geradores elípticos é o gerador *Winslow* ou *TTM homogêneo (Thompson-Thames-Mastin)*[5]. Este gerador é a generalização bidimensional da equação (2.22) que foi mostrada no capítulo 2, com $P = 0$. O método surge da necessidade de um gerador elíptico que produza malhas não dobradas, ou seja, com Jacobiano diferente de zero. Ele também requer que as componentes da transformação inversa $\xi = \xi(x, y)$ e $\eta = \eta(x, y)$ sejam funções harmônicas (ao invés do próprio mapa), ou seja, que satisfaçam a

[5] O método é também referenciado como gerador com *suavidade* (*smoothness*).

equação de Laplace

$$\nabla^2_{\mathbf{x}}\xi = \nabla^2_{\mathbf{x}}\xi = \xi_{xx} + \xi_{yy} = 0, \quad \nabla^2_{\mathbf{x}}\eta = \nabla^2_{\mathbf{x}}\eta = \eta_{xx} + \eta_{yy} = 0, \tag{3.12}$$

no domínio físico. Desta forma é possível mostrar que a transformação é um-a-um [33]. Então, a solução contínua da equação de Winslow resulta em uma transformação com Jacobiano diferente de zero no interior do domínio físico.

Assim como o gerador elíptico AH, se o contorno de Ω é suave, então o problema de Winslow possui uma solução única infinitamente diferenciável no interior de Ω.

É difícil resolver de forma numérica a equação de Winslow no domínio físico diretamente[32], pois, para isso, deveríamos especificar as condições de contorno no espaço físico. Para desenvolver um algoritmo numérico conveniente para calcular o mapa de Winslow, as equações (3.12) são transformadas para o espaço lógico; desta forma temos que determinar as condições de contorno para uma linha coordenada, o qual é uma reta. Supomos o Jacobiano $J \neq 0$ para que a transformação seja inversível. O resultado da inversão é [32, p. 154]

$$\mathcal{Q}_w x = g_{22}x_{\xi\xi} - 2g_{12}x_{\xi\eta} + g_{11}x_{\eta\eta} = 0, \quad \mathcal{Q}_w y = g_{22}y_{\xi\xi} - 2g_{12}y_{\xi\eta} + g_{11}y_{\eta\eta} = 0, \tag{3.13}$$

onde

$$g_{11} = x_\xi^2 + y_\xi^2, \tag{3.14}$$

$$g_{12} = x_\xi x_\eta + y_\xi y_\eta, \tag{3.15}$$

$$g_{22} = x_\eta^2 + y_\eta^2. \tag{3.16}$$

e $\mathcal{Q}_w$ é chamado o operador quasi-linear de Winslow. A métrica g_{11} é o quadrado do comprimento do vetor tangente à linha coordenada ξ, g_{22} é o quadrado do comprimento do vetor tangente à linha coordenada η e g_{12} é o produto interno dos dois vetores tangentes. Isto fornece uma interpretação geométrica útil dos coeficientes da equação transformada, que comumente chamamos de *métricas*.

Em contraste com (3.8), as equações (3.13) são *acopladas* através dos coeficientes da matriz das métricas, que dependem de x e y, e são quasilineares.

A equação TTM não homogênea discretizada em diferenças finitas segue diretamente, de forma análoga, à equação (3.11), adicionando a esta a discretização dos termos das métricas g_{11}, g_{12} e g_{22}. Entretanto, mais trabalho é requerido em resolver numericamente o sistema de equações de Winslow do que o sistema AH, devido ao seu acoplamento e a não-linearidade.

Uma maneira de implementar a resolução destas equações é usar o algoritmo de iteração de *Picard* (também chamado de *substituições sucessivas*), o qual é a versão bidimensional do algoritmo de iteração não-linear do capítulo 2.

Algoritmo: Substituições sucessivas de Picard

1. Defina uma malha inicial $(\mathbf{x}_{i,j}^{old}, \mathbf{y}_{i,j}^{old})$

   $it = 0$

2. Enquanto $(\delta \leq tol)$ e $(it < it_{final})$

3. Calcule os coeficientes g_{11}, g_{12}, g_{22} usando $\mathbf{x}_{i,j}^{old}$.

4. Resolva o sistema linear resultante para $\mathbf{x}_{i,j}$
   usando SOR.

5. Calcule $\delta = \max_{i,j} \|\mathbf{x}_{i,j} - \mathbf{x}_{i,j}^{old}\|$.

6. Faça $\mathbf{x}_{i,j}^{old} = \mathbf{x}_{i,j}$.
   it=it+1

O algoritmo funciona do seguinte modo. Uma malha inicial $(x_{i,j}^{old}, y_{i,j}^{old})$ é gerada na região física, tipicamente usando interpolação transfinita (TFI). O passo 3 no laço externo é usar os valores antigos $(x_{i,j}^{old}, y_{i,j}^{old})$ para calcular os valores das métricas g_{11}, g_{12} e g_{22}, que permanecem fixas durante a solução da equação linear resultante, que pode ser resolvida através de um método de relaxação SOR ou outro método iterativo. Com isso obteremos os valores $(x_{i,j}, y_{i,j})$, que, depois de serem usados para estimar o erro, δ, são copiados para $(x_{i,j}^{old}, y_{i,j}^{old})$. Se a tolerância *tol* não é satisfeita, então voltamos ao passo 2 para calcular as métricas novamente; caso contrário, o laço encerra. Note que ambas as equações para x e y possuem os mesmos coeficientes, portanto devemos utilizar esse fato para reduzir a armazenagem e aumentar a eficiência nos algoritmos numéricos para geração da malha.

Vários testes foram realizados visando determinar qual a melhor maneira para aplicar o algoritmo de Picard. O algoritmo consiste na iteração do laço externo, com parâmetros it_{final} e *tol*, e a iteração de um laço interno no passo 4 dado pela iteração SOR, que possui parâmetros it_{SOR} e tol_{SOR}. Testamos duas maneiras diferentes de gerenciar o número de iterações it e o valor da tolerância *tol*. O primeiro modo consiste em escolher $it_{final} = it_{SOR}$ e $tol = tol_{SOR}$, ou seja, em cada iteração calculamos os coeficientes e solucionamos numericamente o sistema através do método SOR com precisão máxima. Neste processo surge uma questão: se na primeira iteração externa os valores estão longe de serem corretos, por que exigir tamanha precisão no laço interno? Desta forma, o segundo modo consiste em realizar algumas poucas iterações no laço interno e mesmo que a tolerância não seja alcançada, pular para o laço externo e calcular os novos valores dos coeficientes.

Na figura 3.5 temos os resultados dos testes para os dois modos para uma malha sobre um aerofólio. Este gráfico comporta-se qualitativamente da mesma maneira para todos os tipos de malhas testadas. Podemos observar nesta figura que a convergência do laço interno para o primeiro modo é rápida, porém o cálculo dos coeficientes prejudica a convergência do laço externo.

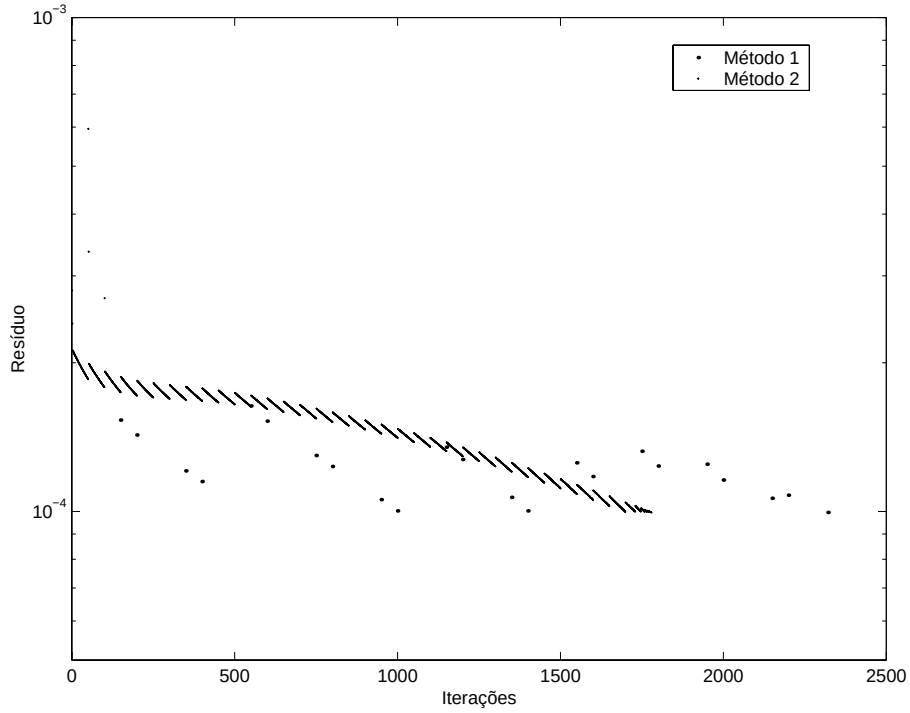


Figura 3.5: *Gráfico do resíduo pelo número de iterações do método* 1 *(com* $it_{SOR} = 1000$*) e do método* 2 *(com apenas* 50 *iterações para o laço interno). Em ambos* $tol = 10^{-4}$.

Isto ocorre porque os coeficientes linearizados estão distantes dos coeficientes reais quando o laço interno alcança a tolerância exigida, o que gera um salto no resíduo. Como o segundo modo não exige a convergência do laço interno, o salto gerado pelo cálculo dos coeficientes é menor e este alcança a convergência do laço externo mais rapidamente. Com base nesta informação, optamos por utilizar o segundo modo de iteração.

Com esse sistema de geração de malhas as linhas coordenadas tenderão a ser igualmente espaçadas na ausência de curvaturas no contorno, pois possui um forte efeito suavizante da equação de Laplace. Sobre um contorno convexo as linhas serão atraídas, enquanto que num contorno côncavo as linhas serão afastadas naturalmente (veja figura 3.6).

É possível substituir o algoritmo de iteração de Picard pelo de Newton, que tipicamente converge mais rápido, entretanto sua implementação é mais trabalhosa. Alternativas mais rápidas, porém mais complexas, são oferecidas pelos métodos *multigrid* [6, 7, 9, 28].

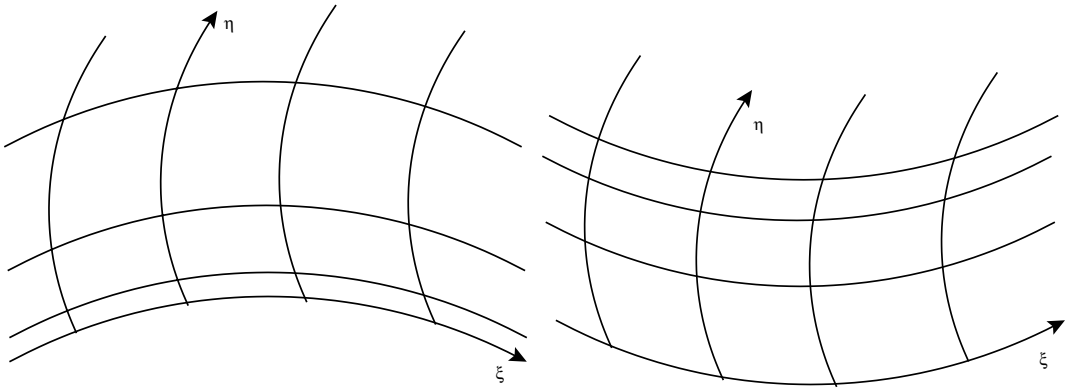


Figura 3.6: *Aproximação devido à concavidade do contorno da região: (a) aproximação sobre um contorno convexo e (b) repulsão sobre um contorno côncavo.*

3.4 O gerador TTM não-homogêneo

O propósito desta seção é apresentar a versão planar do gerador *não homogêneo de Thompson-Thames-Mastin* e mostrar como é usado para controlar a malha no interior do domínio. O gerador pode ser usado para atrair ou repelir os nós da malha para pontos específicos no domínio lógico e mover as linhas coordenadas de uma maneira similar. A abordagem original é apresentada em Thompson et al [44], que adicionou termos fontes à equação diferencial parcial homogênea. Existem duas formas básicas do gerador de malhas não homogêneo; em ambas é necessário especificar funções peso a serem usadas nos termos não homogêneos. A abordagem original é

$$\nabla^2_{\mathbf{x}}\xi = P, \quad \nabla^2_{\mathbf{x}}\eta = Q, \tag{3.17}$$

enquanto que a abordagem reformulada é [48]

$$\nabla^2_{\mathbf{x}}\xi = \frac{g_{22}}{g}P, \quad \nabla^2_{\mathbf{x}}\eta = \frac{g_{11}}{g}Q. \tag{3.18}$$

Esta última é normalmente a preferida, pois as funções de controle P e Q são de ordem de magnitude menor do que a forma original. O teorema de Rado [33] fornece uma base teórica para o gerador Winslow, pois ele implica que a malha gerada possuirá Jacobiano diferente de zero quando $\Delta\xi \to 0$ e $\Delta\eta \to 0$, ou seja, a malha não dobrará. Entretanto, isso não implica que a malha, possuindo $\Delta\xi$ e $\Delta\eta$ maiores que zero, não fique dobrada. Quando os termos não homogêneos são adicionados às equações, o teorema de Rado não mais se aplica, isto é, não existe nenhuma garantia contra o *dobramento* da malha gerada.

Além disso, mesmo que o gerador seja elíptico, ao usarmos equações não homogêneas, perdemos a garantia de gerar uma malha suave. Por exemplo, se uma malha não suave é dada (por exemplo, por um gerador algébrico) então é possível calcular as métricas da malha, tangentes,

etc. e, portanto, P e Q usando a equação não homogênea. As funções calculadas P e Q gerariam então uma malha não suave se usadas para resolver a equação TTM não homogênea. Restrições em P e Q são necessárias se deseja-se obter uma malha não dobrada e suave.

Invertendo a forma original (3.17) da equação TTM obtemos

$$\mathcal{Q}_w \mathbf{x} = -g(P\mathbf{x}_\xi + Q\mathbf{x}_\eta), \tag{3.19}$$

enquanto que invertendo a equação (3.18) obtemos

$$\mathcal{Q}_w \mathbf{x} = -g_{22}P\mathbf{x}_\xi - g_{11}Q\mathbf{x}_\eta. \tag{3.20}$$

O controle local da malha é obtido escolhendo P e Q tendo formas exponenciais [44]

$$\begin{aligned} P(\xi,\eta) = &- \sum_{m=1}^{M} a_m \mathrm{sgn}(\xi-\xi_m) e^{-c_m|\xi-\xi_m|} \\ &- \sum_{i=1}^{I} b_i \mathrm{sgn}(\xi-\xi_i) e^{-d_i\sqrt{(\xi-\xi_i)^2+(\eta-\eta_i)^2}} \end{aligned} \tag{3.21}$$

$$\begin{aligned} Q(\xi,\eta) = &- \sum_{m=1}^{M} a_m \mathrm{sgn}(\eta-\eta_m) e^{-c_m|\eta-\eta_m|} \\ &- \sum_{i=1}^{I} b_i \mathrm{sgn}(\eta-\eta_i) e^{-d_i\sqrt{(\xi-\xi_i)^2+(\eta-\eta_i)^2}} \end{aligned} \tag{3.22}$$

onde M é o número de linhas a serem aproximadas e I é o número de pontos da malha para onde serão atraídos; a_m, b_i, c_m, d_i, ξ_i e η_i são parâmetros e sgn retorna -1, 0 ou 1 dependendo do sinal do operando. Note que P e Q são funções peso no espaço lógico.

Esta abordagem que utiliza P e Q como funções peso para controlar o interior da malha é razoavelmente eficiente mas perde em automação, ou seja, o usuário necessita intervir no processo. A determinação dos parâmetros nos termos fontes requer experiência e habilidade; o controle da malha é impreciso e não automático e depende das variáveis no espaço lógico. A implementação numérica do gerador não homogêneo requer somente pequenas modificações a partir do gerador homogêneo, a fim de implementar os termos fontes (veja exemplos de malhas na figura 3.7).

3.5 Geração de Malhas através de EDPs Parabólicas e Hiperbólicas

Este método segue do trabalho de Steger e Chausee [40], que sugere o seguinte sistema não-linear de primeira ordem para ser usado na geração de malhas sobre aerofólios em duas dimensões:

$$x_\xi x_\eta + y_\xi y_\eta = 0, \quad x_\xi y_\eta - x_\eta y_\xi = V, \tag{3.23}$$

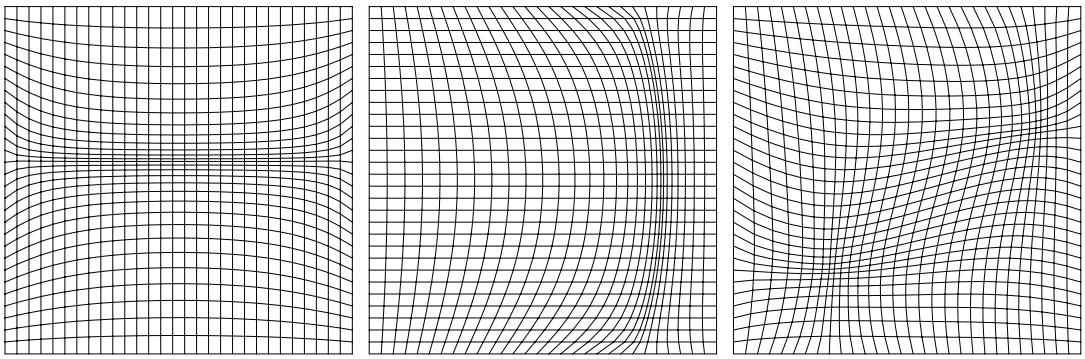


Figura 3.7: *Utilização do gerador TTM não-homogêneo para aproximação (a) para uma linha ξ, (b) para uma linha η e (c) para dois pontos específicos.*

ou, em notação vetorial,

$$\mathbf{x}_\xi \cdot \mathbf{x}_\eta = 0, \quad J = V, \tag{3.24}$$

onde $V = V(\xi, \eta)$ é uma função de controle definida pelo usuário. O que podemos observar é que a primeira equação força a ortogonalidade e a segunda controla a área local da transformação através de V, ou seja, o sistema é equivalente a $g_{12} = 0$ e $J = V$. Se a transformação é não-singular, $J \neq 0$, então o sistema pode ser expresso vetorialmente como

$$\mathbf{x}_\eta = \frac{V}{g_{11}} \mathbf{x}_\xi^\perp. \tag{3.25}$$

Devido à não-linearidade esta equação é mista, não podendo definir sua natureza hiperbólica, parabólica ou elíptica. Entretanto, se a equação é linearizada, o sistema resultante tem a forma

$$\mathcal{A}\mathbf{x}_\xi + \mathcal{B}\mathbf{x}_\eta = \mathbf{v} \tag{3.26}$$

e é hiperbólico.

Os sistemas (3.25) e (3.26), podem ser vistos como problemas de valor inicial e ser resolvidos por esquemas marchantes (no qual os dados do contorno são fornecidos no contorno interno de um domínio ilimitado; a malha solução é então obtida fazendo o método percorrer as linhas, de dentro para fora, bastando para isso percorrer apenas uma vez toda a malha). Entretanto, sabe-se que os esquemas marchantes para equações elípticas são instáveis a menos que certas restrições sejam postas sobre a razão de aspecto[6] do espaço lógico [37]. Por outro lado, existem esquemas marchantes para sistemas hiperbólicos (sem nenhuma restrição na razão de aspecto da célula) como em [40]. O algoritmo básico para um esquema marchante muito simples, como pode ser visto a seguir. Basicamente, o método deve percorrer uma vez a malha toda, podendo variar a direção com que isto é feito.

[6] A razão de aspecto de uma célula da malha é dada pelo quociente entre a largura e a altura desta célula. Uma descrição mais detalhada pode ser encontrada na seção 8.4.

Algoritmo: Esquema marchante para resolver EDP hiperbólica

1. Defina as condições iniciais $\mathbf{x}_{i,0}$, $\mathbf{x}_{0,j}$ e $\mathbf{x}_{M,j}$.
2. Para $i = 1, \ldots, M-1$, $j = 1, \ldots, N-1$.
3. $$x_{i,j+1} = x_{i,j} - \frac{\Delta\eta}{2\Delta\xi}(\frac{V}{g_{11}})_{i,j}(y_{i+1,j} - y_{i-1,j}).$$
$$y_{i,j+1} = y_{i,j} - \frac{\Delta\eta}{2\Delta\xi}(\frac{V}{g_{11}})_{i,j}(x_{i+1,j} - x_{i-1,j}).$$

A geração de malhas através de esquemas marchantes é muito rápida, se comparada aos métodos que resolvem sistemas elípticos de segunda ordem. Entretanto, existem limitações inerentes na geração de malhas hiperbólicas: por exemplo, o método se aplica somente a domínios não-limitados, isto é, um dos contornos não pode ser especificado devido ao caráter hiperbólico do sistema. Comportamentos similares a choques podem aparecer na solução, produzindo malhas não suaves, principalmente se a condição de contorno não possuir suavidade. Desde que a função V deve ser especificada, o método não é totalmente automático e a experimentação é necessária para encontrar um V efetivo.

3.6 Cortes e Células Fictícias

Como citado na introdução, devemos ter alguns cuidados especiais nos cortes. Fisicamente, esta região não apresenta nenhuma diferença a qualquer outro ponto interior da malha e, matematicamente, poderíamos ter localizado o corte em outra posição. Não podemos diferenciar os pontos que fazem parte do corte dos pontos interiores da malha. Isto significa dizer que se temos continuidade das linhas coordenadas passando num ponto interior da malha, então desejamos ter continuidade também sobre as linhas coordenadas passando sobre um ponto do corte.

A maneira mais fácil para lidar com esses pontos é tratá-los do mesmo modo que os pontos interiores da malha utilizando o conceito de *células fictícias*. Uma célula fictícia é uma célula a mais colocada na região lógica do lado externo ao contorno, de tal forma que ao calcularmos a discretização de um operador tenhamos os pontos vizinhos de todos os lados da célula - veja a figura 3.8. Desta forma, podemos gerar dois tipos de células fictícias: (i) células fictícias que são exteriores ao contorno e (ii) células fictícias que pertencem ao interior do contorno (aquelas que pertencem ao corte).

As células fictícias exteriores ao contorno são usadas em volumes finitos, como no método MAC[7] [26], para estender o domínio e permitir o cálculo dos operadores na fronteira (veja figura 3.8b). Podemos, também, utilizar essas células numa região que necessita um corte. Neste

[7] *Marker And Cell.*

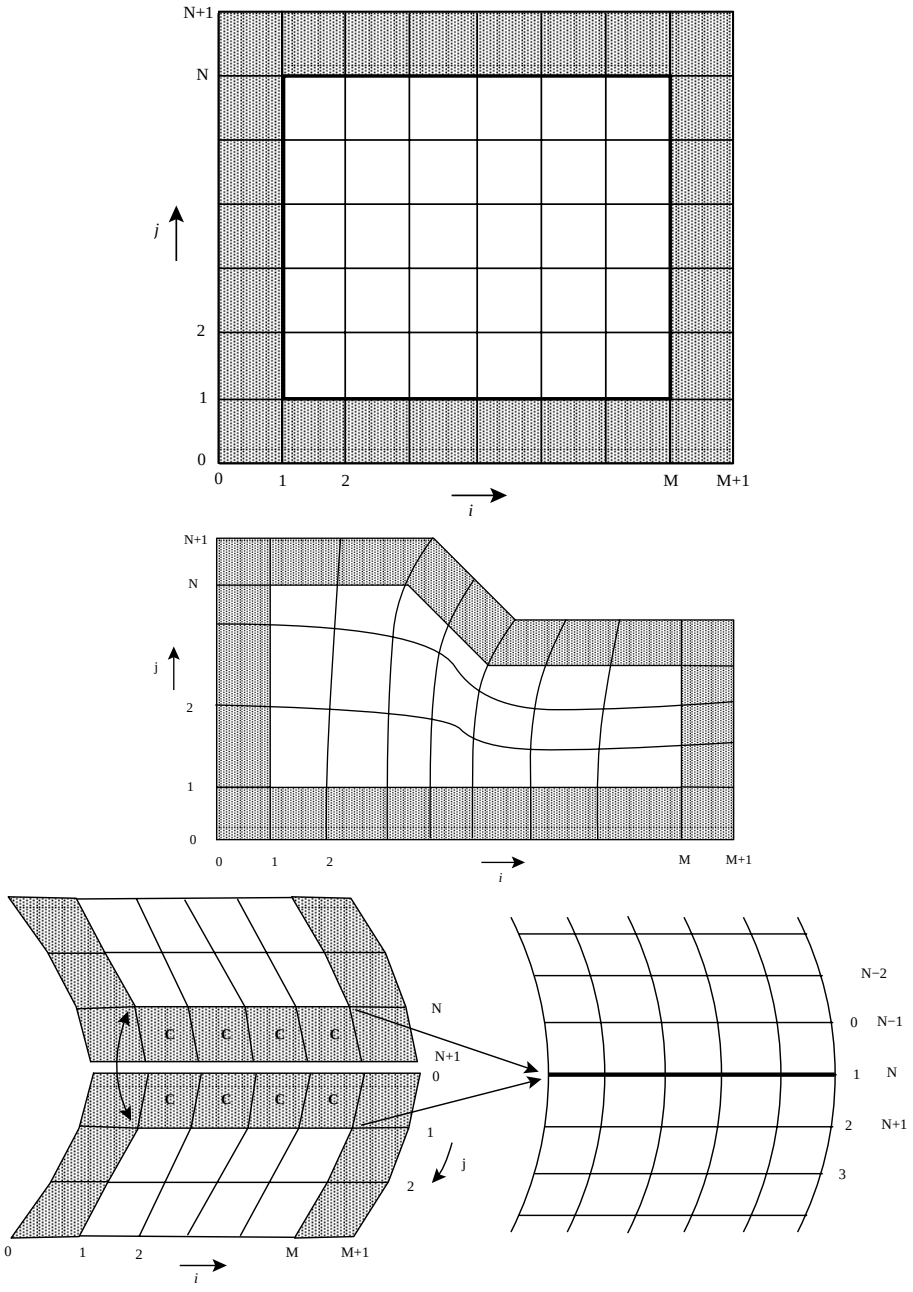


Figura 3.8: *Células fictícias (a) no domínio lógico, (b) células fictícias exteriores ao contorno e (c) células fictícias sobre um corte. As células fictícias exteriores ao contorno são hachuradas e as células fictícias que pertencem ao corte são marcadas com a letra* ***C****.*

caso, a célula fictícia é uma célula exterior ao domínio lógico, mas sobrepõem-se sobre uma célula do domínio físico. Pode-se observar na figura 3.8c que o nó 1 ocupa a mesma posição no espaço físico que o nó N, enquanto que o nó 2 corresponde ao nó $N+1$ e o nó 0 corresponde ao nó $N-1$.

Para calcular a posição dos pontos do corte utilizamos a mesma equação que usamos para gerar os outros pontos da malha. Ou seja, se estamos usando a equação de Winslow, então para todos os pontos pertencentes ao corte $x_{i,1}$, $1 \leq i \leq M$, utilizamos também a equação de Winslow. Claramente não precisamos calcular os pontos $x_{i,N}$, pois esses são iguais aos $x_{i,1}$, para $1 \leq i \leq M$. Depois de calcularmos todos os pontos da malha, inclusive os pontos do corte, precisamos igualar também os pontos da linha $N+1$ ao pontos da linha 2 e os pontos da linha 0 aos pontos da linha $N-1$. Desta forma, o corte não permanecerá fixo no espaço físico, podendo "flutuar" para uma posição mais adequada de forma a satisfazer o sistema de equações que gera a malha.

Note que esse procedimento é o que deveria ser usado para calcular as equações do fluxo que estão sendo resolvidas, mas nem sempre isso é feito na prática. O que se faz é aplicar alguma condição de contorno do tipo Neumann ou extrapolação no corte, mas, como já foi dito, não existe nenhuma razão matemática ou até mesmo física para que esses pontos sejam tratados de uma maneira diferente. Usando essa metodologia, estamos garantindo, acima de tudo, a correta implementação do modelo matemático do problema físico, sem criar falsas condições de contorno.

Um cuidado em especial deve ser dado quanto à variação dos índices. Para cortes localizados em lados opostos, como por exemplo numa malha em O sobre um cilindro, a região é uma simples continuação através do corte. Neste caso, quando o índice ξ aumenta de um lado da malha, do outro lado ξ também aumenta, o mesmo valendo para η. Entretanto, para uma malha do tipo C, quando de um lado da malha ξ ou η aumenta, do outro lado ξ ou η diminui [46, pp. 70-76] (veja a figura 3.9).

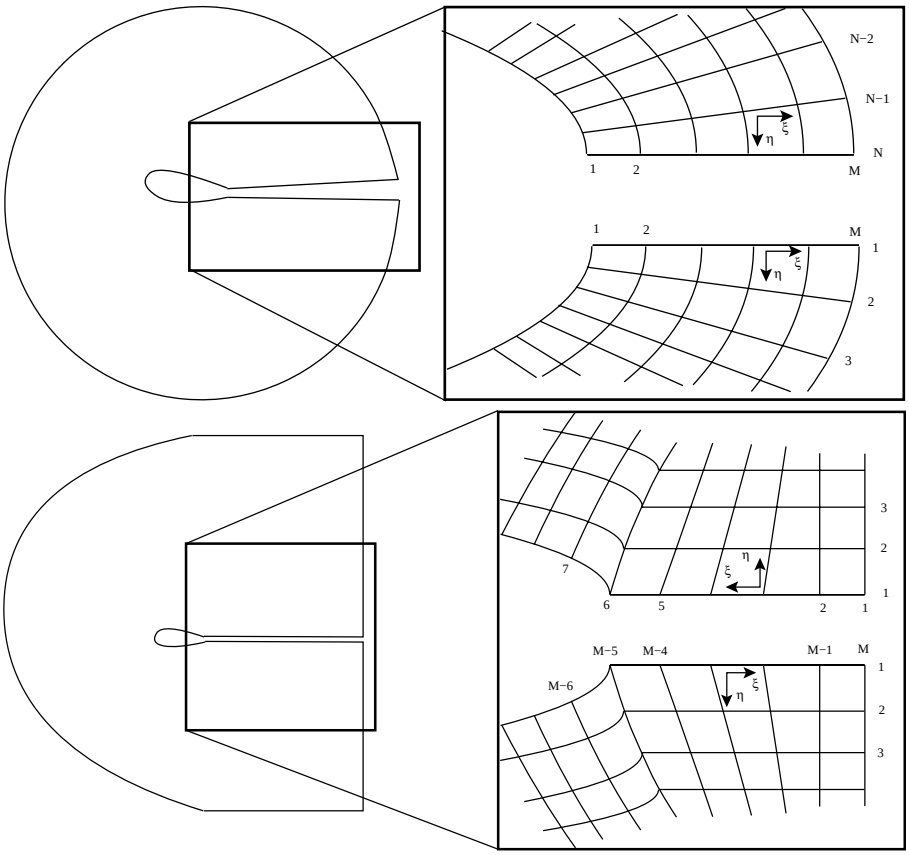


Figura 3.9: *(a) Zoom sobre uma malha* O*, onde sobre o corte, as linhas coordenadas aumentam na mesma direção. (b) Zoom sobre uma malha* C*, onde sobre o corte, as linhas coordenadas estão com sentido inverso.*

4 TRANSFORMAÇÃO DOS OPERADORES

Neste capítulo vamos obter as transformações dos operadores diferenciais do plano físico para o plano lógico, aplicando o Teorema da Divergência para um volume finito limitado por superfícies coordenadas. A partir da expressão para o operador gradiente, obtemos as transformações para os operadores divergente, rotacional e Laplaciano. Obtemos também a relação entre a base de vetores contravariantes e covariantes.

Pelo Teorema da Divergência [46],

$$\iiint_V \nabla_{\mathbf{x}} \cdot \mathbf{A} dV = \oint_S \mathbf{A} \cdot \mathbf{n} dS \tag{4.1}$$

para qualquer tensor ou vetor $\mathbf{A}$, onde $\mathbf{n}$ é o vetor normal unitário apontando para fora da superfície S que delimita o volume V. Para um elemento de superfície diferencial pertencendo à superfície coordenada ζ, temos

$$\mathbf{n} dS^{\zeta} = \pm \mathbf{x}_{\xi} \times \mathbf{x}_{\eta} d\xi d\eta \tag{4.2}$$

e, de forma análoga, para as superfícies coordenadas ξ e η, onde o sinal depende da localização da superfície com relação ao volume. Considerando, então, um elemento de volume diferencial, ∂V, limitado por seis faces que são as superfícies coordenadas, como mostrado na figura 4.1, e usando o fato que o elemento de volume pode ser expresso em função do Jacobiano como $dV = \sqrt{g} d\xi d\eta d\zeta$, juntamente com a equação (4.2), obtemos

$$\iiint_{\partial V} (\nabla_{\mathbf{x}} \cdot \mathbf{A}) \sqrt{g} d\xi d\eta d\zeta = \sum_{i=1}^{3} \iint_{\partial S_+^i} \mathbf{A} \cdot (\mathbf{x}_{\xi_j} \times \mathbf{x}_{\xi_k}) d\xi_j d\xi_k - \iint_{\partial S_-^i} \mathbf{A} \cdot (\mathbf{x}_{\xi_j} \times \mathbf{x}_{\xi_k}) d\xi_j d\xi_k \tag{4.3}$$

onde ∂S_+^i e ∂S_-^i indicam os elementos nos dois lados do volume de controle no qual ξ_i permanece constante. Nesta equação, os índices (i, j, k) são cíclicos[1] e $(\xi_1, \xi_2, \xi_3) \equiv (\xi, \eta, \zeta)$.

4.1 O Divergente

Quando o elemento de volume na equação (4.3) tende a zero, temos então uma expressão para o divergente dada por

$$\nabla_{\mathbf{x}} \cdot \mathbf{A} = \frac{1}{\sqrt{g}} \sum_{i=1}^{3} [(\mathbf{x}_{\xi_j} \times \mathbf{x}_{\xi_k}) \cdot \mathbf{A}]_{\xi_i}, \tag{4.4}$$

onde o subscrito ξ_i indica diferenciação parcial.

[1] Se (i, j, k) são índices cíclicos, então $(i, j, k) = (1, 2, 3)$ ou $(i, j, k) = (2, 3, 1)$ ou $(i, j, k) = (3, 1, 2)$, conforme o valor de i.

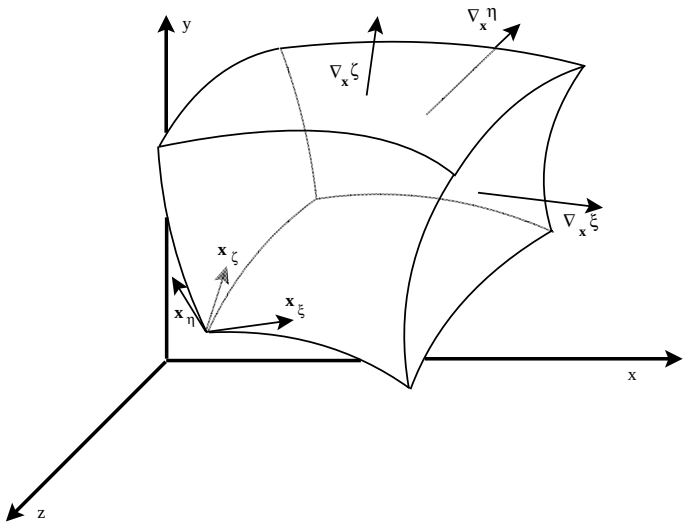


Figura 4.1: *Representação dos tensores métricos covariante* $\mathbf{x}_\xi$ *e contravariante* $\nabla_{\mathbf{x}}\xi$.

Ao expandir esta derivada, obtemos uma identidade métrica fundamental que é dada por[2]

$$\sum_{i=1}^{3} (\mathbf{x}_{\xi_j} \times \mathbf{x}_{\xi_k})_{\xi_i} = \mathbf{0}. \tag{4.5}$$

Assim, o divergente pode ser escrito como

$$\nabla_{\mathbf{x}} \cdot \mathbf{A} = \frac{1}{\sqrt{g}} \sum_{i=1}^{3} (\mathbf{x}_{\xi_j} \times \mathbf{x}_{\xi_k}) \cdot \mathbf{A}_{\xi_i}. \tag{4.6}$$

Embora as equações (4.4) e (4.6) sejam equivalentes, a sua representação numérica não o é. A equação (4.4) é chamada de forma *conservativa* e a equação (4.6), onde o produto vetorial foi expandido, é chamada de *não-conservativa*. A diferença básica é que a área utilizada para calcular o fluxo na forma conservativa é a área da face do volume de controle; ao passo que, na forma não-conservativa, a área utilizada é a de uma seção transversal passando pelo centro do volume de controle. A vantagem da forma conservativa é que, quando somamos as equações sobre todo o domínio da solução, os termos nas faces interiores dos volumes de controle se anulam como numa *soma telescópica*[3], sobrando apenas os termos relativos as faces externas do domínio da solução. Podemos ver aqui claramente o resultado do teorema da divergência onde uma integral de

[2] Da uniformidade de $\mathbf{A}$ segue que

$$\sum_{i=1}^{3} (\mathbf{x}_{\xi_j} \times \mathbf{x}_{\xi_k})_{\xi_i} = \sum_{i=1}^{3} (\mathbf{x}_{\xi_j \xi_i} \times \mathbf{x}_{\xi_k}) + \sum_{i=1}^{3} (\mathbf{x}_{\xi_j} \times \mathbf{x}_{\xi_k \xi_i}) = \mathbf{0}.$$

[3] Uma soma telescópica pode ser escrita na forma $\sum_{i=1}^{N-1} a_i - a_{i+1} = a_1 - a_2 + a_2 - a_3 + \ldots + a_{N-1} - a_N = a_1 - a_N$

volume é substituída por uma integral de superfície. Isto favorece o uso da representação numérica da forma conservativa dos operadores.

É importante notar que, como a forma conservativa do divergente - assim como a do gradiente, do rotacional e do Laplaciano - é obtida diretamente da integral de superfície fechada no Teorema de Divergência, o uso da forma conservativa para os operadores é equivalente a usar a forma em diferenças para a integral de superfície fechada. Isto significa dizer que a implementação numérica das formas diferencial e integral da equação do movimento são equivalentes.

4.2 O Gradiente

Desde que a equação (4.1) é válida para $\mathbf{A}$ substituído por um escalar com o produto escalar trocado por uma simples aplicação do operador $\nabla_{\mathbf{x}}$, podemos obter diretamente das equações (4.4) e (4.6) as expressões para o gradiente na forma conservativa como

$$\nabla_{\mathbf{x}} A = \frac{1}{\sqrt{g}} \sum_{i=1}^{3} \left[(\mathbf{x}_{\xi_j} \times \mathbf{x}_{\xi_k}) A \right]_{\xi_i} \tag{4.7}$$

e na forma não-conservativa como

$$\nabla_{\mathbf{x}} A = \frac{1}{\sqrt{g}} \sum_{i=1}^{3} (\mathbf{x}_{\xi_j} \times \mathbf{x}_{\xi_k}) A_{\xi_i}. \tag{4.8}$$

4.3 O Rotacional

Do mesmo modo que o gradiente, o rotacional também pode ser obtido da equação (4.1) substituindo o produto escalar pelo produto vetorial. Assim, a forma conservativa será dada por

$$\nabla_{\mathbf{x}} \times \mathbf{A} = \frac{1}{\sqrt{g}} \sum_{i=1}^{3} \left[(\mathbf{x}_{\xi_j} \times \mathbf{x}_{\xi_k}) \times \mathbf{A} \right]_{\xi_i}, \tag{4.9}$$

enquanto que a forma não conservativa é dada por

$$\nabla_{\mathbf{x}} \times \mathbf{A} = \frac{1}{\sqrt{g}} \sum_{i=1}^{3} (\mathbf{x}_{\xi_j} \times \mathbf{x}_{\xi_k}) \times \mathbf{A}_{\xi_i}. \tag{4.10}$$

Expandindo o produto vetorial[4] obtemos também as seguintes expressões para a forma conservativa

$$\nabla_{\mathbf{x}} \times \mathbf{A} = \frac{1}{\sqrt{g}} \sum_{i=1}^{3} \left[(\mathbf{x}_{\xi_j} \cdot \mathbf{A}) \mathbf{x}_{\xi_k} - (\mathbf{x}_{\xi_k} \cdot \mathbf{A}) \mathbf{x}_{\xi_j} \right]_{\xi_i} \tag{4.11}$$

[4] Usando a identidade vetorial $\mathbf{A} \times (\mathbf{B} \times \mathbf{C}) = (\mathbf{A} \cdot \mathbf{C})\mathbf{B} - (\mathbf{A} \cdot \mathbf{B})\mathbf{C}$.

e para a forma não-conservativa

$$\nabla_{\mathbf{x}} \times \mathbf{A} = \frac{1}{\sqrt{g}} \sum_{i=1}^{3} (\mathbf{x}_{\xi_j} \cdot \mathbf{A}_{\xi_i}) \mathbf{x}_{\xi_k} - (\mathbf{x}_{\xi_k} \cdot \mathbf{A}_{\xi_i}) \mathbf{x}_{\xi_j}. \tag{4.12}$$

4.4 O Laplaciano

Procuramos implementar neste trabalho a idéia dos operadores miméticos apresentados numa série de trabalhos de Steinberg et al. [11, 12, 13]. Esses operadores procuram imitar a forma conservativa e simétrica dos operadores diferenciais na sua forma discreta. Desta forma, o operador divergente será o negativo do adjunto do operador gradiente, $DIV^* = -GRAD$, e conseqüentemente o Laplaciano será um operador simétrico expresso como $LAP = DIV\,GRAD$.

As expressões para o Laplaciano seguem também diretamente das equações (4.4) e (4.6), com $\mathbf{A}$ sendo substituído por $\nabla_{\mathbf{x}} A$ da equação (4.7) ou (4.8). Assim, a forma conservativa para o Laplaciano é

$$\nabla^2_{\mathbf{x}} A = \nabla_{\mathbf{x}} \cdot (\nabla_{\mathbf{x}} A) = \sum_{i=1}^{3} \sum_{l=1}^{3} \left\{ \frac{1}{\sqrt{g}} (\mathbf{x}_{\xi_j} \times \mathbf{x}_{\xi_k}) \cdot [(\mathbf{x}_{\xi_m} \times \mathbf{x}_{\xi_n}) A]_{\xi_l} \right\}_{\xi_i} \tag{4.13}$$

para (i, j, k) e (l, m, n) cíclicos. A forma não conservativa é dada por

$$\nabla^2_{\mathbf{x}} A = \sum_{i=1}^{3} \sum_{l=1}^{3} (\mathbf{x}_{\xi_j} \times \mathbf{x}_{\xi_k}) \cdot \left[\frac{1}{\sqrt{g}} (\mathbf{x}_{\xi_m} \times \mathbf{x}_{\xi_n}) A_{\xi_l} \right]_{\xi_i}. \tag{4.14}$$

4.5 Relações entre a base de vetores covariantes e contravariantes

A equação (4.8) para o gradiente permite que a base de vetores contravariantes seja expressa em termos da base de vetores covariantes como segue. Com $A = \xi_m$ em (4.8), teremos

$$\nabla_{\mathbf{x}} \xi_m = \frac{1}{\sqrt{g}} \sum_{i=1}^{3} (\mathbf{x}_{\xi_j} \times \mathbf{x}_{\xi_k}) \delta_i^m = \frac{1}{\sqrt{g}} \mathbf{x}_{\xi_j} \times \mathbf{x}_{\xi_k}, \tag{4.15}$$

visto que as três coordenadas curvilíneas são independentes uma da outra. Isto fornece a relação entre as derivadas das coordenadas curvilíneas $\nabla_{x_j} \xi_i$ e as derivadas do espaço físico $(x_i)_{\xi_j}$. Assim, multiplicando escalarmente por $\mathbf{x}_\xi$ a equação (4.15), obtemos

$$\mathbf{x}_{\xi_i} \cdot \nabla_{\mathbf{x}} \xi_j = \frac{1}{\sqrt{g}} \mathbf{x}_{\xi_i} \cdot (\mathbf{x}_{\xi_k} \times \mathbf{x}_{\xi_l}), \tag{4.16}$$

onde (j, k, l) são cíclicos. Se $i \neq j$, então o produto escalar do lado direito é igual a zero, pois serão perpendiculares. Mas se $i = j$, então, como $\mathbf{x}_{\xi_k} \times \mathbf{x}_{\xi_l} = \mathbf{x}_{\xi_j}$ e multiplicando por $\mathbf{x}_{\xi_j}$, teremos a unidade. Portanto, em geral, teremos

$$\mathbf{x}_{\xi_i} \cdot \nabla_{\mathbf{x}} \xi_j = \delta_i^j. \tag{4.17}$$

Devido a esta relação, podemos expressar as componentes contravariantes do vetor $\mathbf{A}$ como $A^i = \nabla_{\mathbf{x}} \xi_i \cdot \mathbf{A}$ e as componentes covariantes como $A_i = \mathbf{x}_{\xi_i} \cdot \mathbf{A}$.

4.6 Operadores na forma conservativa

Utilizando a equação (4.15), o produto vetorial dos vetores da base covariante nas expressões dadas para o gradiente, divergente, Laplaciano e rotacional pode ser substituída diretamente pelos vetores da base contravariante (multiplicados pelo Jacobiano). Com isso, o produto vetorial será eliminado destas expressões. Apresentaremos agora apenas a versão conservativa dos operadores visto que essas expressões são mais consistentes numericamente, sendo estas as implementadas neste trabalho.

As formas conservativas são:

$$\nabla_{\mathbf{x}} A = \frac{1}{\sqrt{g}} \sum_{i=1}^{3} (\sqrt{g} \nabla_{\mathbf{x}} \xi_i A)_{\xi_i}, \tag{4.18}$$

$$\nabla_{\mathbf{x}} \cdot \mathbf{A} = \frac{1}{\sqrt{g}} \sum_{i=1}^{3} (\sqrt{g} \nabla_{\mathbf{x}} \xi_i \cdot \mathbf{A})_{\xi_i}, \tag{4.19}$$

$$\nabla_{\mathbf{x}} \times \mathbf{A} = \frac{1}{\sqrt{g}} \sum_{i=1}^{3} (\sqrt{g} \nabla_{\mathbf{x}} \xi_i \times \mathbf{A})_{\xi_i}, \tag{4.20}$$

$$\nabla^2_{\mathbf{x}} A = \frac{1}{\sqrt{g}} \sum_{i=1}^{3} \sum_{j=1}^{3} [\nabla_{\mathbf{x}} \xi_i \cdot (\sqrt{g} \nabla_{\mathbf{x}} \xi_i A)_{\xi_j}]_{\xi_i}. \tag{4.21}$$

Expandindo as derivadas internas, o Laplaciano pode ser expresso como

$$\nabla^2_{\mathbf{x}} A = \frac{1}{\sqrt{g}} \sum_{i=1}^{3} \sum_{j=1}^{3} [\sqrt{g} g^{ij} A_{\xi_j}]_{\xi_i}, \tag{4.22}$$

onde g^{ij} é o tensor métrico contravariante. Note que, para obter a expressão para a derivada em relação a x, $\frac{\partial}{\partial x}$, basta utilizar a primeira componente do vetor $\nabla_{\mathbf{x}} A$.

4.7 Derivada Tangencial e Normal

Expressões para as derivadas normais e tangenciais às linhas coordenadas são necessárias para as condições de contorno de diversos problemas e são obtidas das bases vetoriais como segue.

4.7.1 Derivada tangencial à linha coordenada

Desde que os vetores da base covariante são tangentes às linhas coordenadas, a derivada tangencial numa linha coordenada onde ξ_i varia é dada por

$$(A)^i_\tau = \frac{\mathbf{x}_{\xi_i}}{|\mathbf{x}_{\xi_i}|} \cdot \nabla_{\mathbf{x}} A = \frac{1}{|\mathbf{x}_{\xi_i}|} \sum_{j=1}^{3} (\mathbf{x}_{\xi_i} \cdot \nabla_{\mathbf{x}} \xi_j) A_{\xi_j} \tag{4.23}$$

que, a partir da equação (4.17), se reduz a

$$(A)_\tau^i = \frac{A_{\xi_i}}{\sqrt{g_{ii}}}. \tag{4.24}$$

4.7.2 Derivada normal à superfície coordenada

Também, desde que os vetores da base contravariante são normais à superfície coordenada, a derivada normal a superfície coordenada no qual ξ_i é constante é dada por

$$(A)_n^i = \frac{\nabla_{\mathbf{x}}\xi_i}{|\nabla_{\mathbf{x}}\xi_i|} \cdot \nabla_{\mathbf{x}} A = \frac{1}{|\nabla_{\mathbf{x}}\xi_i|} \sum_{j=1}^{3} (\nabla_{\mathbf{x}}\xi_i \cdot \nabla_{\mathbf{x}}\xi_j) A_{\xi_j}, \tag{4.25}$$

que por sua vez é igual a

$$(A)_n^i = \frac{1}{\sqrt{g_{ii}}} \sum_{j=1}^{3} g^{ij} A_{\xi_j}. \tag{4.26}$$

4.8 Forma bidimensional

Nas seções anteriores apresentamos a transformação dos operadores para o caso tridimensional, onde os operadores são expressos como somatórios. Para facilitar a consulta, vamos apresentar agora a transformação dos operadores para o caso bidimensional. Em duas dimensões, a direção z não varia e pode ser considerada $z = 0$, assim $\mathbf{x}_\zeta = \nabla_{\mathbf{x}}\zeta = \hat{\mathbf{k}}$ e a base vetorial covariante é

$$\mathbf{x}_\xi = [x_\xi, y_\xi], \tag{4.27}$$

$$\mathbf{x}_\eta = [x_\eta, y_\eta]. \tag{4.28}$$

As componentes do tensor métrico são

$$\begin{bmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{bmatrix} = \begin{bmatrix} x_\xi^2 + y_\xi^2 & x_\xi x_\eta + y_\xi y_\eta & 0 \\ x_\xi x_\eta + y_\xi y_\eta & x_\eta^2 + y_\eta^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.29}$$

e o Jacobiano é dado por

$$\sqrt{g} = \sqrt{g_{11}g_{22} - g_{12}^2} = x_\xi y_\eta - x_\eta y_\xi. \tag{4.30}$$

A base contravariante, de acordo com a equação (4.15), é

$$\nabla_{\mathbf{x}}\xi = \left[\frac{y_\eta}{\sqrt{g}}, \frac{-x_\eta}{\sqrt{g}}\right] \tag{4.31}$$

$$\nabla_{\mathbf{x}}\eta = \left[\frac{-y_\xi}{\sqrt{g}}, \frac{x_\xi}{\sqrt{g}}\right] \tag{4.32}$$

e o tensor das métricas contravariantes é

$$\begin{bmatrix} g^{11} & g^{12} & g^{13} \\ g^{21} & g^{22} & g^{23} \\ g^{31} & g^{32} & g^{33} \end{bmatrix} = \begin{bmatrix} \frac{g_{22}}{g} & -\frac{g_{12}}{g} & 0 \\ -\frac{g_{12}}{g} & \frac{g_{11}}{g} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.33}$$

Os operadores em sua forma conservativa podem ser expressos como segue:

- divergente (da eq. (4.4)),

$$\nabla_{\mathbf{x}} \cdot \mathbf{A} = \frac{1}{\sqrt{g}} \left[(y_\eta A_1 - x_\eta A_2)_\xi + (-y_\xi A_1 + x_\xi A_2)_\eta \right]; \tag{4.34}$$

- gradiente (da eq. (4.7)),

$$\nabla_{\mathbf{x}} f = \begin{bmatrix} f_x \\ f_y \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{g}}[(y_\eta f)_\xi - (y_\xi f)_\eta] \\ \frac{1}{\sqrt{g}}[-(x_\eta f)_\xi + (x_\xi f)_\eta] \end{bmatrix}; \tag{4.35}$$

- rotacional (da eq. (4.9)),

$$\nabla_{\mathbf{x}} \times \mathbf{A} = \frac{\mathbf{k}}{\sqrt{g}}[(y_\eta A_2 + x_\eta A_1)_\xi - (y_\xi A_2 + x_\xi A_1)_\eta]; \tag{4.36}$$

- Laplaciano (da eq. (4.13)),

$$\nabla^2_{\mathbf{x}} f = \frac{1}{\sqrt{g}} \left\{ \left[\frac{1}{\sqrt{g}} (g_{22} f_\xi - g_{12} f_\eta) \right]_\xi + \left[\frac{1}{\sqrt{g}} (g_{11} f_\eta - g_{12} f_\xi) \right]_\eta \right\}. \tag{4.37}$$

As derivadas em relação à normal na forma conservativa são dadas por

$$f_n(\xi) = \frac{1}{\sqrt{g\, g_{22}}} \{ y_\eta [(y_\eta f)_\xi - (y_\xi f)_\eta] - x_\eta [-(x_\eta f)_\xi + (x_\xi f)_\eta] \}, \tag{4.38}$$

$$f_n(\eta) = \frac{1}{\sqrt{g\, g_{11}}} \{ -y_\xi [(y_\eta f)_\xi - (y_\xi f)_\eta] + x_\xi [-(x_\eta f)_\xi + (x_\xi f)_\eta] \}. \tag{4.39}$$

e a derivada tangencial por

$$f_\tau(\xi) = \frac{1}{\sqrt{g\, g_{22}}} \{ x_\eta [(y_\eta f)_\xi - (y_\xi f)_\eta] - y_\eta [(x_\eta f)_\xi - (x_\xi f)_\eta] \}, \tag{4.40}$$

$$f_\tau(\eta) = \frac{1}{\sqrt{g\, g_{11}}} \{ x_\xi [(y_\eta f)_\xi - (y_\xi f)_\eta] - y_\xi [(x_\eta f)_\xi - (x_\xi f)_\eta] \}. \tag{4.41}$$

A forma não-conservativa é apresentada também por ser de mais fácil implementação. Assim, a derivada não-conservativa em relação à normal é dada por

$$f_n(\xi) = \frac{1}{\sqrt{g\, g_{22}}} \left(g_{22} f_\xi - g_{12} f_\eta \right), \tag{4.42}$$

$$f_n(\eta) = \frac{1}{\sqrt{g\, g_{11}}} \left(-g_{12} f_\xi + g_{11} f_\eta \right), \tag{4.43}$$

e a derivada tangencial na forma não-conservativa como

$$f_\tau(\xi) = \frac{1}{\sqrt{g_{22}}} f_\eta, \tag{4.44}$$

$$f_\tau(\eta) = \frac{1}{\sqrt{g_{11}}} f_\xi. \tag{4.45}$$

5 DISCRETIZAÇÃO DOS OPERADORES

No capítulo anterior apresentamos a transformação dos operadores do espaço físico para o espaço lógico. Estas transformações visam facilitar a implementação das derivadas em domínios complexos, principalmente a implementação das condições de contorno. Quando transformamos as equações do espaço físico para o espaço lógico, as equações resultantes são do mesmo tipo que a original, mas são mais complicadas, no sentido de que possuem mais termos e coeficientes variáveis. Por outro lado, o domínio é largamente simplificado desde que ele é transformado para uma região retangular, facilitando dessa forma a imposição de condições de contorno. Esta é uma das características que faz a geração de malhas generalizadas uma ferramenta tão valiosa e importante na solução numérica de equações diferenciais parciais em domínios arbitrários.

Uma vez que o problema é discretizado, a solução numérica do problema transformado pode ser obtida usando técnicas padrões, ou seja, usando técnicas desenvolvidas baseadas em malhas ortogonais. Desde que o domínio seja estacionário e retangular e que os incrementos das coordenadas curvilíneas, $\Delta\xi$ e $\Delta\eta$, sejam arbitrários, o cálculo pode ser sempre feito em uma malha quadrada e uniforme. As derivadas podem, portanto, ser representadas por diferenças finitas convencionais ou por volumes finitos. De fato, o problema transformado tem a aparência de um problema numa malha cartesiana uniforme e pode ser tratado como tal, tanto na formação das equações, quanto na sua solução.

A forma específica das equações transformadas vai depender de quais relações foram usadas do capítulo anterior, conservativa ou não-conservativa. Vamos dar mais ênfase à forma conservativa, por esta apresentar propriedades de implementação numérica melhores, como citado anteriormente.

Existe uma decisão a ser tomada no momento da discretização dos operadores. Devemos escolher entre os vários arranjos possíveis para a localização das *variáveis primitivas*[1] que podem estar combinados com o tipo de malha a ser resolvido. Os arranjos se dividem em dois grupos principais: *arranjos co-localizados* e *arranjos diferenciados*.

[1] Variáveis primitivas são as incógnitas do problema, como por exemplo, a velocidade (u, v, w), a pressão p ou a temperatura T.

5.1 Arranjo Co-localizado

Os valores aproximados das derivadas espaciais de uma função que aparece nas equações transformadas podem ser achadas, em um dado ponto da malha, em termos dos valores da função naquele ponto e em seus vizinhos.

No arranjo co-localizado, todas as variáveis primitivas são armazenadas no mesmo ponto da malha, como mostra a figura 5.1. Podemos, também, ter dois tipos de localizações para armazenar as variáveis. Elas podem estar armazenadas todas sobre um *nó* da célula ou todas no *centro* da célula. Aqui, usaremos a primeira forma, por facilitar o cálculo das métricas, já que teremos os valores da malha também no nó. Caso contrário, precisaríamos interpolar os valores nos vértices de uma determinada célula para achar a posição do centro da célula.

Em duas dimensões as derivadas parciais primeira, segunda e mista com respeito às variáveis no espaço lógico ξ e η, são representadas como derivadas ordinárias num ponto interior (i, j) por diferenças finitas. Normalmente, esse arranjo está relacionado com diferenças finitas.

Esta "molécula computacional" é preferida, pois as representações das derivadas preservam a simetria e são de segunda ordem de precisão.

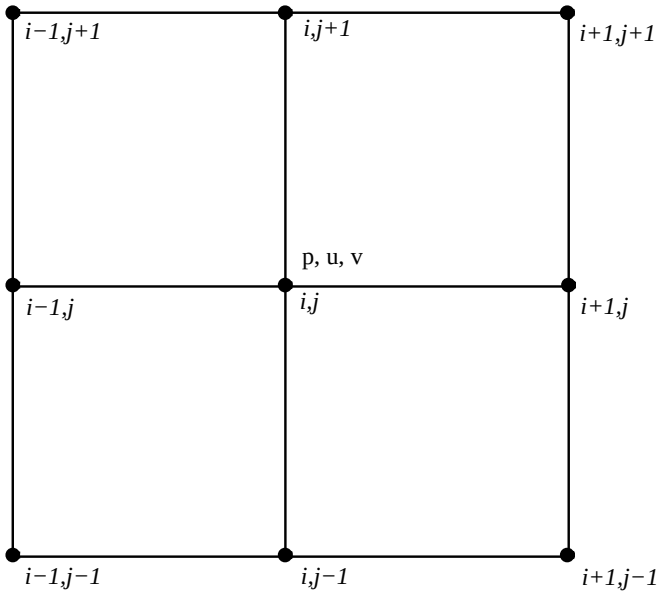


Figura 5.1: *Estêncil com* 9 *pontos para o arranjo co-localizado.*

Uma maneira de discretizar as derivadas de primeira e segunda ordem é a utilização de diferenças centrais, que podem ser dadas da seguinte forma:

$$(f_\xi)_{ij} = \frac{f_{i+1,j} - f_{i-1,j}}{2d\xi}, \tag{5.1}$$

$$(f_\eta)_{ij} = \frac{f_{i,j+1} - f_{i,j-1}}{2d\eta}, \tag{5.2}$$

$$(f_{\xi\xi})_{ij} = \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{d\xi^2}, \tag{5.3}$$

$$(f_{\eta\eta})_{ij} = \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{d\eta^2}, \tag{5.4}$$

$$(f_{\xi\eta})_{ij} = \frac{f_{i+1,j+1} - f_{i-1,j+1} - f_{i+1,j-1} + f_{i-1,j-1}}{4d\xi d\eta}. \tag{5.5}$$

Para obter a discretização dos operadores no arranjo co-localizado, devemos utilizar as transformações dos operadores do capítulo anterior diretamente nas equações acima. Por exemplo, para o gradiente em relação a x usamos a equação (4.35),

$$f_x = \frac{1}{\sqrt{g}}[(y_\eta f)_\xi - (y_\xi f)_\eta]. \tag{5.6}$$

Para discretizar este operador usamos as equações (5.1) e (5.2) para as derivadas em relação a ξ e a η. Da mesma forma, utilizamos esta mesma metodologia para o cálculo das métricas e para implementar os demais operadores.

5.2 Arranjo Diferenciado

No arranjo diferenciado, as variáveis primitivas são armazenadas em diferentes localizações sobre a malha. A maneira geralmente utilizada é armazenar as variáveis escalares (tais como a pressão p e a temperatura T) no centro da célula e as variáveis escalares (tal como a velocidade $\mathbf{u} = (u, v)$) nas faces da célula da seguinte maneira: a primeira componente, u, é armazenada nas faces leste e oeste, e a segunda componente, v, nas faces superior e inferior, como mostra a figura 5.2.

Para uma malha bidimensional, obtemos

$$(f_\xi)_{ij} = \frac{f_{i+\frac{1}{2},j} - f_{i-\frac{1}{2},j}}{d\xi}, \tag{5.7}$$

$$(f_\eta)_{ij} = \frac{f_{i,j+\frac{1}{2}} - f_{i,j-\frac{1}{2}}}{d\eta}, \tag{5.8}$$

com os valores nas faces aproximados como uma média dos valores dos centros de duas células dividindo a mesma face, ou seja,

$$f_{i+\frac{1}{2},j} = \frac{f_{i+1,j} + f_{i,j}}{2}. \tag{5.9}$$

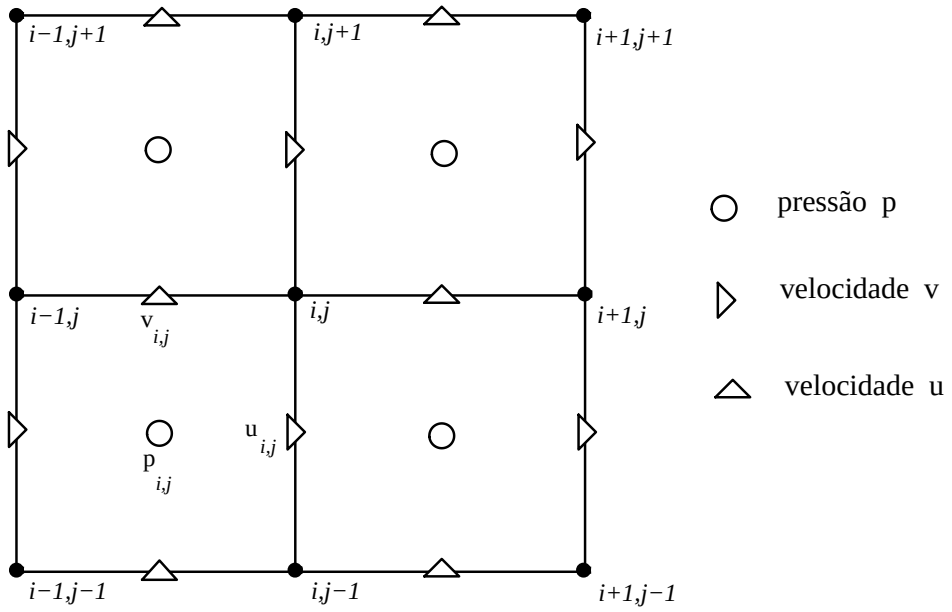


Figura 5.2: *Estêncil sobre o ponto i, j para o arranjo diferenciado.*

Para as derivadas de segunda ordem obtemos

$$(f_{\xi\xi})_{ij} = \frac{(f_\xi)_{i+\frac{1}{2},j} - (f_\xi)_{i-\frac{1}{2},j}}{d\xi}, \tag{5.10}$$

$$(f_{\xi\eta})_{ij} = \frac{(f_\xi)_{i,j+\frac{1}{2}} - (f_\xi)_{i,j-\frac{1}{2}}}{d\eta}, \tag{5.11}$$

$$(f_{\eta\xi})_{ij} = \frac{(f_\eta)_{i+\frac{1}{2},j} - (f_\eta)_{i-\frac{1}{2},j}}{d\xi}, \tag{5.12}$$

$$(f_{\eta\eta})_{ij} = \frac{(f_\eta)_{i,j+\frac{1}{2}} - (f_\eta)_{i,j-\frac{1}{2}}}{d\eta}. \tag{5.13}$$

Entretanto, não podemos usar a mesma metodologia utilizada na equação (5.9). Ao invés disso, uma representação de segunda ordem pode ser obtida na molécula de nove pontos usando as expressões (5.7) e, assim, obtemos as mesmas equações de segunda ordem como no caso de diferenças finitas.

Normalmente, o método de volumes-finitos está associado com este arranjo. Para obter a discretização dos operadores diferenciais sobre o arranjo diferenciado, devemos integrar as equações para os operadores diretamente sobre o volume de controle. Por exemplo, para obter a discretização do operador gradiente de f em relação a x sobre o centro da célula i, j, usamos

novamente a equação (5.6), e integramos sobre o volume de controle de tal forma a obter

$$\begin{aligned}
\int_S^N \int_W^E \sqrt{g} f_x d\xi d\eta &= \int_S^N \int_W^E \frac{1}{\sqrt{g}}[(y_\eta f)_\xi - (y_\xi f)_\eta] d\xi d\eta && (5.14)\\
&= \int_S^N \int_W^E (y_\eta f)_\xi d\xi d\eta - \int_S^N \int_W^E (y_\xi f)_\eta d\xi d\eta && (5.15)\\
&= \int_S^N (y_\eta f)|_W^E d\eta - \int_E^W (y_\xi f)|_S^N d\xi && (5.16)\\
&= (y_\eta f)|_W^E d\eta - (y_\xi f)|_S^N d\xi && (5.17)\\
&= ((y_\eta f)_E - (y_\eta f)_W) d\eta - ((y_\xi f)_N - (y_\xi f)_S) d\xi. && (5.18)
\end{aligned}$$

Dividindo a última equação por $d\xi d\eta$ e calculando nas faces N, S, E e W obtemos

$$\frac{(y_\eta f)_{i+\frac{1}{2},j} - (y_\eta f)_{i-\frac{1}{2},j}}{d\xi} - \frac{(y_\xi f)_{i,j+\frac{1}{2}} - (y_\xi f)_{i,j-\frac{1}{2}}}{d\eta}, \tag{5.19}$$

que é a expressão para o gradiente em relação a x. Os índices fracionários representam quantidades calculadas nas faces da célula como dado pela tabela 3.1 e pela figura 3.3.

No caso do uso do arranjo diferenciado em malhas generalizadas, uma pequena modificação deve ser feita. Devemos armazenar nas faces não as velocidades, mas sim as componentes contravariantes do fluxo, pois nem sempre as componentes u e v do vetor velocidade são normais às faces dos volumes de controle neste tipo de malha [19, 50]. As componentes contravariantes da velocidade U e V são dadas por

$$U = \sqrt{g}\nabla_{\mathbf{x}}\xi \cdot \mathbf{u}, \quad V = \sqrt{g}\nabla_{\mathbf{x}}\eta \cdot \mathbf{u}. \tag{5.20}$$

5.3 O esquema upwind

Quando usamos diferenças centrais em problemas de difusão-convecção, onde o termo convectivo seja dominante, podemos encontrar problemas de estabilidade se a malha escolhida for pouco refinada [25, 36]. Isto pode resultar em oscilações não-realísticas.

Para evitar esse problema, podemos aproximar as derivadas através de diferenças *upwind*. Se a velocidade u for positiva usamos diferenças para frente[2] e caso contrário usamos diferenças para trás[3]. Assim, a primeira derivada é discretizada como

$$\frac{d(uf)}{d\xi} = \frac{u_r f_r - u_l f_l}{d\xi}, \tag{5.21}$$

onde u_r e u_l são as velocidades calculadas nas faces do volume de controle

$$u_r = u_{i+\frac{1}{2},j} = \frac{u_{i+1,j} + u_{i,j}}{2} \qquad u_l = u_{i-\frac{1}{2},j} = \frac{u_{i-1,j} + u_{i,j}}{2}. \tag{5.22}$$

[2] *Forward differences.*
[3] *Backward differences.*

As quantidades f_r e f_l são calculadas de acordo com as velocidades nas faces da seguinte forma

$$f_r = \begin{cases} f_{i,j}, & \text{se } u_r > 0, \\ f_{i+1,j}, & \text{se } u_r < 0, \end{cases} \qquad f_l = \begin{cases} f_{i-1,j}, & \text{se } u_l > 0, \\ f_{i,j}, & \text{se } u_l < 0. \end{cases} \tag{5.23}$$

Entre as desvantagens do esquema *upwind* está sua baixa ordem de precisão e o aparecimento de uma falsa difusão. Para contornar este problema temos o esquema híbrido [25], que utiliza uma média ponderada entre os esquemas *upwind* e diferenças centrais, obtendo

$$\frac{d(uf)}{d\xi} = \gamma \frac{d(uf)}{d\xi}^{\text{upwind}} + (1-\gamma)\frac{d(uf)}{d\xi}^{\text{central}}, \quad 0 < \gamma < 1. \tag{5.24}$$

5.4 Identidades Métricas

Quando transformamos as equações para o domínio lógico usando a forma conservativa, é possível que as métricas introduzam termos indesejáveis nas equações. Isto acontece pois as métricas foram incluídas no operador diferencial e a sua diferenciação não satisfaz a identidade

$$\sum_{i=1}^{3} (\sqrt{g}\nabla_{\mathbf{x}}\xi_i)_{\xi_i} = (\sqrt{g}\nabla_{\mathbf{x}}\xi)_{\xi} + (\sqrt{g}\nabla_{\mathbf{x}}\eta)_{\eta} + (\sqrt{g}\nabla_{\mathbf{x}}\zeta)_{\zeta} = \mathbf{0}. \tag{5.25}$$

Esta identidade surge quando utilizamos uma quantidade escalar A constante na forma conservativa do gradiente dada pela equação (4.18). Esta equação é uma identidade métrica que deve ser satisfeita numericamente para que as expressões conservativas para o gradiente, divergente, rotacional e Laplaciano tornem-se nulas quando a variável física é constante. Esta consideração não é necessária com a forma não conservativa, uma vez que a quantidade A é diferenciável diretamente nestas expressões.

Considere o exemplo a seguir. A transformação para a primeira derivada é

$$f_x = \frac{1}{\sqrt{g}}[(fy_\eta)_\xi - (fy_\xi)_\eta] \tag{5.26}$$

que, para f uniforme, reduz-se a

$$0 = y_{\eta\xi} - y_{\xi\eta}. \tag{5.27}$$

É fácil observar que a relação acima é a identidade métrica (5.25) para duas dimensões. O esquema requer então $y_{\xi\eta} = y_{\eta\xi}$ para qualquer ponto no domínio físico. Considerando um ponto localizado no centro da célula, $(i+\frac{1}{2}, j+\frac{1}{2})$, obtemos

$$(y_{\eta\xi})_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{(y_\eta)_{i+1,j+\frac{1}{2}} - (y_\eta)_{i,j+\frac{1}{2}}}{d\xi}. \tag{5.28}$$

Se usarmos $(y_\eta)_{i+1,j+\frac{1}{2}} = \frac{y_{i+1,j+1} - y_{i+1,j}}{d\eta}$ obtemos a identidade

$$(y_{\eta\xi})_{i+\frac{1}{2},j+\frac{1}{2}} = (y_{\xi\eta})_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{y_{i+1,j+1} - y_{i+1,j} - y_{i,j+1} + y_{i,j}}{d\xi d\eta}. \tag{5.29}$$

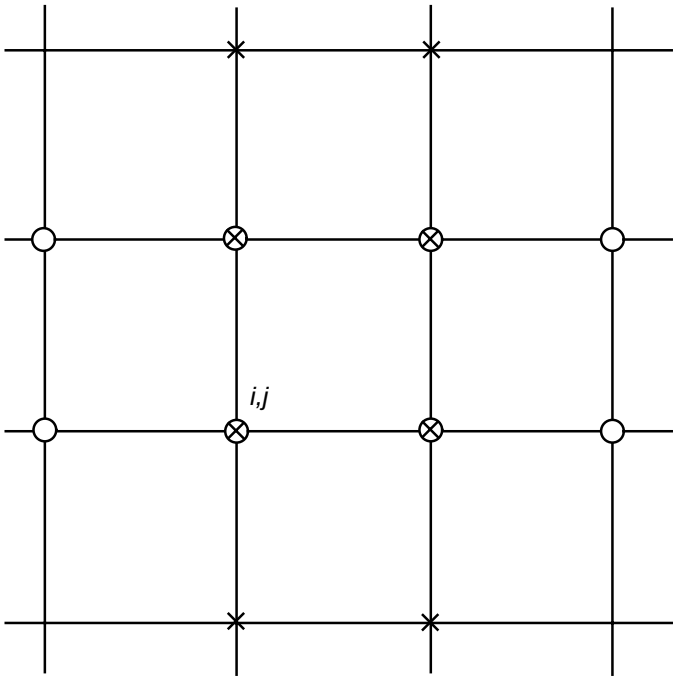


Figura 5.3: *Diferença entre duas discretizações para (∘) $y_{\xi\eta}$ e (×) $y_{\eta\xi}$.*

Uma alternativa seria calcular a métrica y_η como uma média, ou seja,

$$(y_\eta)_{i+1,j+\frac{1}{2}} = \frac{(y_\eta)_{i+1,j+1} + (y_\eta)_{i+1,j}}{2} \tag{5.30}$$

que fornece

$$(y_{\eta\xi})_{i+\frac{1}{2},j+\frac{1}{2}} = \frac{y_{i+1,j+2} - y_{i+1,j} + y_{i+1,j+1} - y_{i+1,j-1}}{2d\eta d\xi} - \frac{y_{i,j+2} - y_{i,j} + y_{i,j+1} - y_{i,j-1}}{2d\eta d\xi}. \tag{5.31}$$

É fácil observar que este esquema não satisfaz a equação (5.27). Como pode ser visto na figura 5.3, a expressão (5.31) usa os pontos marcados com × enquanto que a expressão análoga para $y_{\xi\eta}$ utiliza os pontos marcados com ∘.

Deve-se observar que ambas as representações são consistentes e de mesma ordem de precisão. Se os coeficientes das métricas nos pontos da malha foram calculados e armazenados, seria natural seguir a segunda abordagem, usando médias das métricas nos pontos intermediários. Isto, entretanto, não é aceitável, pois esta não satisfaz a identidade métrica envolvida (5.25), introduzindo falsos gradientes não nulos num campo uniforme.

Este exemplo sugere uma regra básica que deveria ser seguida: *nunca faça as médias das métricas. Ao invés disso, faça a média das coordenadas e depois calcule as métricas* [46, p. 162].

5.5 A derivada temporal

Visto que neste trabalho não consideramos o modelo com uma malha adaptativa (onde a malha muda com o passar do tempo), as derivadas temporais $\frac{\partial u}{\partial t}$, $\frac{\partial v}{\partial t}$ e $\frac{\partial \phi}{\partial t}$ são representadas todas da mesma maneira. Para discretizar o termo temporal usamos o *método de Euler*, o qual emprega diferenças de primeira ordem do tipo

$$\left[\frac{\partial u}{\partial t}\right]_{i,j}^{n+1} = \frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\partial t}. \tag{5.32}$$

6 CONDIÇÕES DE CONTORNO

A escolha das condições de contorno é uma das tarefas mais importantes na resolução de um problema de dinâmica de fluidos computacional. O número de condições de contorno e os seu tipos dependem principalmente do problema que se está resolvendo e do tipo de equação: parabólica, hiperbólica ou elíptica [30].

Neste capítulo, trataremos dos diversos tipos de condições de contorno a serem usadas. Sem perda de generalidade, expressaremos as fórmulas para o contorno *oeste*, que é dado por $x_{i,1}$, para $i = 1, \ldots, N$, mas as equações obtidas podem ser modificadas facilmente para os outros contornos.

6.1 Arranjo diferenciado

Para o arranjo diferenciado, precisamos determinar qual o tipo de condição de contorno vamos usar e, para sua implementação, usaremos as células fictícias já apresentadas na seção 3.6.

Existe uma pequena diferença na implementação para uma variável escalar, localizada no centro da célula, e para uma variável vetorial armazenada nas faces da célula. Como pode ser visto nas figuras 6.1 e 6.2, vamos usar aqui o sub-índice 0 para indicar uma variável localizada na célula fictícia, 1 para indicar uma variável no interior do domínio e c para indicar a variável exatamente sobre o contorno. Para os outros contornos basta usar $N+1$ para a célula fictícia e N para a célula do interior.

6.1.1 Condição de Dirichlet

A variável escalar está armazenada no centro da célula e o contorno passa pela face da célula como mostram as figuras 6.1 e 6.2. Se o valor da variável ϕ é igual ϕ_c no contorno então

$$\phi_c = \frac{\phi_0 + \phi_1}{2} \quad \Rightarrow \quad \phi_0 = 2\phi_c - \phi_1. \tag{6.1}$$

Para uma variável vetorial, por exemplo (u, v), esta é a mesma condição que usaremos para a variável u nos lados leste e oeste da malha e para v nos lados norte e sul. Como u está localizada diretamente nos lados norte e sul, assim como v está localizada nos lados leste e oeste, para estes casos usamos

$$u_0 = u_c, \tag{6.2}$$

onde u_0 é a variável no contorno e u_c é o valor desta variável.

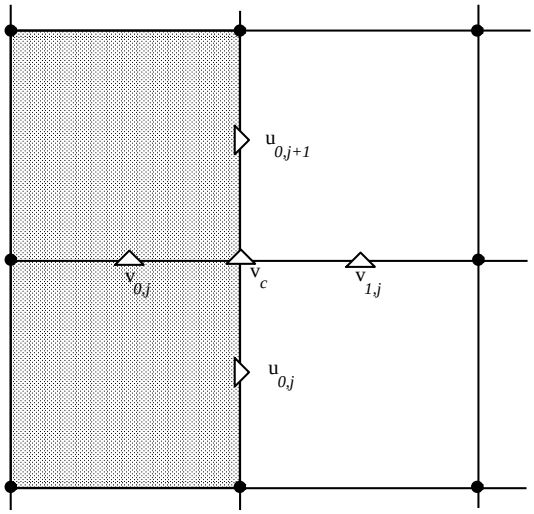


Figura 6.1: *Contorno oeste com células fictícias hachuradas. Como a componente v da velocidade não está localizada no contorno, obtemos a velocidade no contorno como* $v_c = \frac{v0+v1}{2}$.

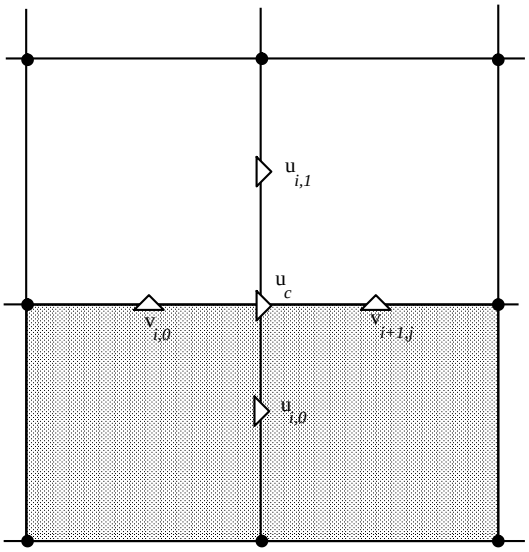


Figura 6.2: *Contorno sul com células fictícias hachuradas. Como a componente u da velocidade não está localizada no contorno, obtemos a velocidade no contorno como* $u_c = \frac{u0+u1}{2}$.

6.1.2 Condição de Neumann

Quando estamos usando esta condição estamos fazendo com que a derivada em relação à normal da variável seja igual a uma constante, na maioria das vezes igual a zero. Então, teremos

$$\frac{\partial \phi}{\partial n_c} = \frac{\phi_1 - \phi_0}{\partial n} = 0 \quad \Rightarrow \quad \phi_0 = \phi_1. \tag{6.3}$$

Para uma variável vetorial, esta mesma condição é usada para u ou v, ou seja, $u_0 = u_1$. A condição de *deslizamento*[1] é um caso particular onde temos a derivada da velocidade tangencial em relação a normal igual a zero e a velocidade normal a face igual a zero.

Podemos também aplicar outras condições de contorno especiais que serão a combinação destas duas anteriores. Quando estamos usando a velocidade igual a zero no contorno, esta condição também é chamada *condição de não-deslizamento*[2]. Neste caso, as variáveis diretamente no contorno serão expressas por $u_1 = 0$, enquanto as que não estão diretamente no contorno serão dadas por

$$u_c = \frac{u_0 + u_1}{2} = 0 \quad \Rightarrow \quad u_0 = -u_1. \tag{6.4}$$

Uma condição normalmente usada na saída do fluxo, denominada *condição parabólica* ou *outflow*, é usar a condição de Neumann para ambas as componentes da velocidade, o que significa que a velocidade não muda na direção normal ao contorno. Para a entrada do fluxo, chamada de *inflow*, as velocidades são dadas explicitamente com a condição de Dirichlet.

Uma condição de contorno periódica também pode ser dada no contorno. Para isto, basta igualarmos as condições dos dois lados, por exemplo, fazendo $u_0 = u_{M-1}$ e $u_M = u_1$, onde condições similares podem ser achadas para v e ϕ nos outros contornos da malha.

6.2 Arranjo co-localizado

Neste caso, também precisamos determinar qual o tipo de condição de contorno vamos usar. Neste arranjo, todas as variáveis estão armazenadas nos *vértices* das células. Assim, todas as discretizações assumem a mesma forma, conforme mostrado a seguir.

[1] Condição *free-slip*.
[2] Condição *no-slip*.

6.2.1 Condição de Dirichlet

Para implementar esta discretização, basta fixarmos o valor desejado no contorno, ou seja, caso tenhamos o valor a para a velocidade u então a condição será

$$u_1 = a. \tag{6.5}$$

6.2.2 Condição de Neumman

Para implementar a condição de Neumann, que significa termos a derivada em relação à normal igual a zero, há duas possibilidades. Do primeiro modo, podemos usar a discretização

$$\frac{\partial u}{\partial x} = \frac{u_2 - u_1}{\partial x} = 0 \quad \Rightarrow u_1 = u_2. \tag{6.6}$$

onde u_1 é a velocidade no contorno e u_2 é a velocidade no interior da malha. Esta discretização é de primeira ordem e apresenta uma desvantagem pois, na verdade, estamos representando a derivada para o ponto intermediário, isto é, o ponto $u_{1+\frac{1}{2}}$.

A segunda possibilidade utiliza as células fictícias, discretizando a mesma derivada através de

$$\frac{\partial u}{\partial x} = \frac{u_2 - u_0}{2\partial x} = 0 \quad \Rightarrow u_0 = u_2. \tag{6.7}$$

onde u_0 é o valor no vértice da célula fictícia. Desta forma a derivada está sendo expressa exatamente para o ponto u_1 e, além disso, temos uma discretização de segunda ordem. O problema é que acrescentamos mais uma incógnita no problema. Devemos então adicionar mais uma equação para o ponto u_1, que é a própria equação que estamos usando para resolver o fluxo no interior da malha [38].

Para a condição de *não deslizamento* basta fixarmos as velocidades iguais a zero; para a condição de *deslizamento*, fixamos a velocidade normal à face igual a zero e aplicamos a condição de Neumann para a outra variável.

Para a condição de entrada, *inflow*, fixamos as velocidades nos valores desejados com a condição de Dirichlet e, para a condição de saída, *outflow*, aplicamos a condição *parabólica* para a velocidade.

7 A EQUAÇÃO DE NAVIER STOKES

Neste capítulo vamos introduzir o modelo matemático que usamos para simular alguns problemas de dinâmica de fluidos. As equações de Navier-Stokes são apresentadas diretamente. Uma dedução detalhada de cada um dos termos desta equação pode ser consultada no livro de Wendt et al. [49]. Vamos apresentar, também, uma metodologia para a resolução destas equações que segue o método PRIME. Outros métodos para resolver a equação acoplada pressão-velocidade, assim como o PRIME, podem ser encontrados em [25, 35, 47]. Os resultados obtidos serão apresentados no próximo capítulo.

7.1 O modelo matemático

Para o caso de simulação de fluidos, a equação empregada é a de *Navier-Stokes*. Vamos considerar aqui apenas fluxos *laminares*[1] de fluidos *viscosos*[2] e *incompressíveis*[3].

Neste caso, restringimos nossos problemas aos descritos pelas equações do *momento* e da *continuidade* que são respectivamente na forma vetorial

$$\frac{\partial}{\partial t}\mathbf{u} + (\mathbf{u}\cdot \mathrm{grad})\mathbf{u} = \frac{1}{Re}\nabla^2_{\mathbf{x}}\mathbf{u} - \mathrm{grad}p + \mathbf{g} \tag{7.1}$$

$$\nabla_{\mathbf{x}}\cdot\mathbf{u} = 0, \tag{7.2}$$

onde Re é o número de *Reynolds* e $\mathbf{g}$ são as forças de corpo, como por exemplo a gravidade. Este sistema de equações é denominado de *equações de Navier-Stokes*. O número de Reynolds é um número adimensional que relaciona as forças viscosas com as forças de inércia, sendo dado por

$$Re = \frac{\rho_\infty u_\infty L}{\mu}, \tag{7.3}$$

onde ρ_∞ e u_∞ são a densidade e a velocidade característica do escoamento (normalmente dadas longe do corpo ou na corrente livre[4]), L é o comprimento característico (por exemplo o comprimento de um aerofólio) e μ é a viscosidade dinâmica.

[1] Um fluxo que não é turbulento.
[2] Consideramos os termos que apresentam as segundas derivadas.
[3] A densidade é considerada constante.
[4] A corrente livre e a região de um fluxo externo que não é afetada pela presença do corpo.

Podemos expressar as equações de Navier-Stokes utilizando apenas as derivadas, o que fornece

$$\frac{\partial u}{\partial t}+\frac{\partial (uu)}{\partial x}+\frac{\partial (uv)}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)-\frac{\partial p}{\partial x}+g_x \tag{7.4}$$

$$\frac{\partial v}{\partial t}+\frac{\partial (vu)}{\partial x}+\frac{\partial (vv)}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)-\frac{\partial p}{\partial y}+g_y \tag{7.5}$$

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y} = 0 \tag{7.6}$$

onde o termo $(\mathbf{u}\cdot \text{grad})\mathbf{u}$ foi reescrito usando a equação da continuidade (7.2).

Devemos notar que estas equações apresentam basicamente três operadores: o gradiente, o divergente e o Laplaciano, e cada um deles é discretizado de uma forma diferente. O divergente na equação do momento é responsável pela convecção do problema e o Laplaciano é responsável pela difusão. Um número de Reynolds pequeno dará maior importância aos efeitos difusivos enquanto que quanto maior o número de Reynolds, maior influência terão os termos convectivos.

7.2 O algoritmo numérico

O algoritmo que utilizamos para a solução das equações de Navier-Stokes é o algoritmo **PRIME**-(**PR**essão **I**mplícita, **M**omento **E**xplícito) dado em [25]. Iniciamos no tempo $t=0$ com valores iniciais para u e v, pré-especificados. O tempo é incrementado por ∂t em cada passo do laço externo até que um tempo final $t^{(f)}$ seja alcançado. Num dado tempo $t^{(n)}$, os valores de todas as variáveis são conhecidas e desejamos então, encontrar, os valores para o tempo $t^{(n+1)}$.

Primeiramente, discretizamos a derivada temporal nas equações do momento através de (5.32) obtendo

$$u^{(n+1)} = u^{(n)}+\partial t\left[\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)-\frac{\partial (uu)}{\partial x}-\frac{\partial (uv)}{\partial y}-\frac{\partial p}{\partial x}+g_x\right] \tag{7.7}$$

$$v^{(n+1)} = v^{(n)}+\partial t\left[\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)-\frac{\partial (vu)}{\partial x}-\frac{\partial (vv)}{\partial y}-\frac{\partial p}{\partial y}+g_y\right] \tag{7.8}$$

e usaremos as quantidades

$$\hat{u}^{(n)} = u^{(n)}+\partial t\left[\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)-\frac{\partial (uu)}{\partial x}-\frac{\partial (uv)}{\partial y}+g_x\right] \tag{7.9}$$

$$\hat{v}^{(n)} = v^{(n)}+\partial t\left[\frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)-\frac{\partial (vu)}{\partial x}-\frac{\partial (vv)}{\partial y}+g_y\right] \tag{7.10}$$

para obter a forma

$$u^{(n+1)} = \hat{u}^{(n)}-\partial t\frac{\partial p}{\partial x}^{(n+1)} \tag{7.11}$$

$$v^{(n+1)} = \hat{v}^{(n)}-\partial t\frac{\partial p}{\partial y}^{(n+1)} \tag{7.12}$$

onde $\hat{u}$ e $\hat{v}$ são velocidades intermediárias sem considerar a pressão. Vemos, então, que a pressão é dada implicitamente, já que aparece no tempo $t^{(n+1)}$, e a velocidade é calculada explicitamente, necessitando apenas ser corrigida através do gradiente da pressão.

Por último, precisamos uma equação para a pressão. Substituindo as equações (7.11) na equação da continuidade obtemos

$$0 = \frac{\partial u}{\partial x}^{(n+1)} + \frac{\partial v}{\partial y}^{(n+1)} = \frac{\partial \hat{u}}{\partial x}^{(n)} - \partial t \frac{\partial^2 p}{\partial x^2}^{(n+1)} + \frac{\partial \hat{v}}{\partial y}^{(n)} - \partial t \frac{\partial^2 p}{\partial y^2}^{(n+1)}. \tag{7.13}$$

Arranjando os termos, temos a *equação de Poisson para a pressão* $p^{(n+1)}$

$$\frac{\partial^2 p}{\partial x^2}^{(n+1)} + \frac{\partial^2 p}{\partial y^2}^{(n+1)} = \frac{1}{\partial t}\left(\frac{\partial \hat{u}}{\partial x}^{(n)} + \frac{\partial \hat{v}}{\partial y}^{(n)} \right). \tag{7.14}$$

Resumidamente, temos os seguintes passos do algoritmo PRIME.

Algoritmo: PRIME

1. $t = 0$, $n = 0$
2. Defina as condições iniciais para u, v e p.
3. Enquanto $t < t_{(f)}$
4. Escolha Δt de acordo com o critério de convergência.
5. Use as condições de contorno para u e v.
6. Calcule $\hat{u}$ e $\hat{v}$.
7. Calcule o termo fonte $\nabla_{\mathbf{x}} \cdot \hat{\mathbf{u}}$ para a pressão.
8. $it = 0$
9. Enquanto $it < itmax$ e $\|r^{it}\| > \epsilon$.
10. Realize um ciclo SOR.
11. Calcule a norma residual $\|r^{it}\|$ da pressão.
12. it=it+1
13. Calcule $u^{(n+1)}$ e $v^{(n+1)}$
14. $t = t + \Delta t$
15. $n = n + 1$

No passo 4, temos que escolher o valor de Δt de tal modo que garanta a estabilidade e evite gerar oscilações na solução. Usamos três condições para estimar o valor de Δt, que são

$$\frac{2\Delta t}{Re} = \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right)^{-1}, \quad |u_{max}|\Delta t < \Delta x, \quad |v_{max}|\Delta t < \Delta y, \tag{7.15}$$

onde $|u_{max}|$ e $|v_{max}|$ são os valores máximos absolutos da velocidade na malha [25, p. 39]. As duas últimas condições são as *condições de Courant-Friedrichs-Lewy (CFL)*, que dizem que uma

partícula não pode se mover uma distância maior do que o espaçamento da malha em um intervalo de tempo Δt. Uma discussão detalhada sobre análise de estabilidade pode ser achada em [36]. Assim, o valor de Δt é escolhido satisfazendo as três condições de tal modo que

$$\Delta t = \tau \cdot \min\left(\frac{Re}{2}\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}, \frac{\Delta x}{|u_{max}|}, \frac{\Delta y}{|v_{max}|}\right), \tag{7.16}$$

onde $0 < \tau \leq 1$ é um fator de segurança. O algoritmo pode calcular o valor de Δt em cada passo do laço externo ou podemos também escolher um Δt fixo no começo das iterações.

No passo 10, quando estamos realizando um passo do ciclo SOR, devemos aplicar também as condições de contorno para a pressão. Segundo o *método da projeção* desenvolvido por Chorin [14], temos que a condição de contorno para a pressão é a condição de Neumann, ou seja, a derivada em relação à normal é $\frac{\partial p}{\partial n} = 0$.

8 O PROGRAMA *MAKEGRID*

Neste capítulo vamos apresentar o software MAKEGRID, desenvolvido durante este trabalho. O objetivo principal do programa é auxiliar o usuário em duas tarefas iniciais básicas na resolução de um sistema de equações para Dinâmica de Fluidos Computacional (CFD), que são a geração da malha e a especificação das condições de contorno.

O programa foi feito na linguagem FORTRAN 90 sobre a plataforma Windows 95. Apresentaremos o uso do programa passo-a-passo afim de esclarecer todas as opções oferecidas. Detalhes técnicos sobre a implementação são oferecidos ao fim do capítulo.

8.1 Passo a passo

Para demonstrar as principais características do programa vamos definir um problema de CFD e mostrar como o programa pode ser usado para resolvê-lo.

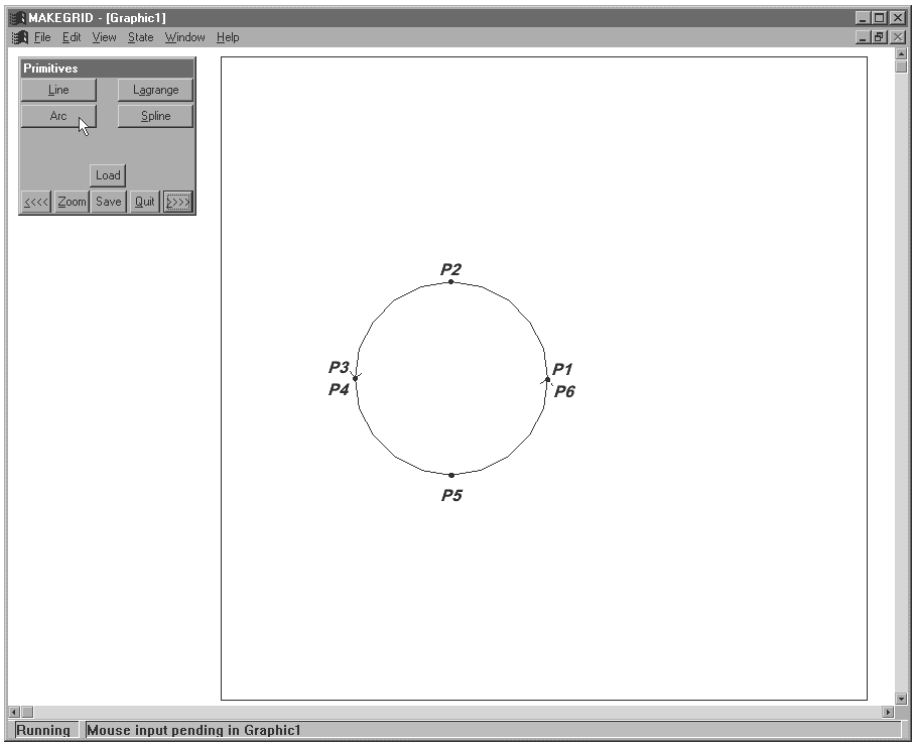
Suponha que queremos simular o escoamento sobre um cilindro. Inicialmente devemos delimitar a região de escoamento. Para isso, precisamos definir o contorno do cilindro e o contorno externo que delimita a região da solução. A primeira etapa é o desenho ou a leitura dos dados das primitivas que compõem a região em questão. Através de um menu de opções, podemos escolher dentre várias primitivas a ideal para a delimitação da região. Neste trabalho, uma *primitiva* é um objeto gráfico básico, tal como uma reta, utilizado para definir curvas na tela. As primitivas básicas disponíveis no programa são retas, splines, polinômios de Lagrange e arcos de círculos.

Passo 1: As primitivas

Para o exemplo do cilindro definimos primeiramente o cilindro interno através de dois arcos. Como mostra a figura 8.1a, para delimitarmos um arco, marcamos os pontos $P1$, $P2$ e $P3$, como sendo o ponto inicial, o ponto final e um ponto intermediário do arco. Depois, devemos marcar o segundo arco, dado pelos pontos $P4$, $P5$ e $P6$. Note que os arcos devem ter os pontos $P1 = P6$ e $P3 = P4$ para que os arcos tenham pontos inicial e final coincidentes.

Para a definição do contorno externo ao cilindro, que delimita a região do problema, é necessário o uso de mais dois arcos, conforme a figura 8.1b.

Dada que a região definida por essas duas curvas é multiplamente conexa, devemos definir um corte para tornar a região simplesmente conexa. Usamos então uma reta ligando o ponto $P1$ do contorno externo com o ponto $P2$ do contorno interno (veja figura 8.2a).



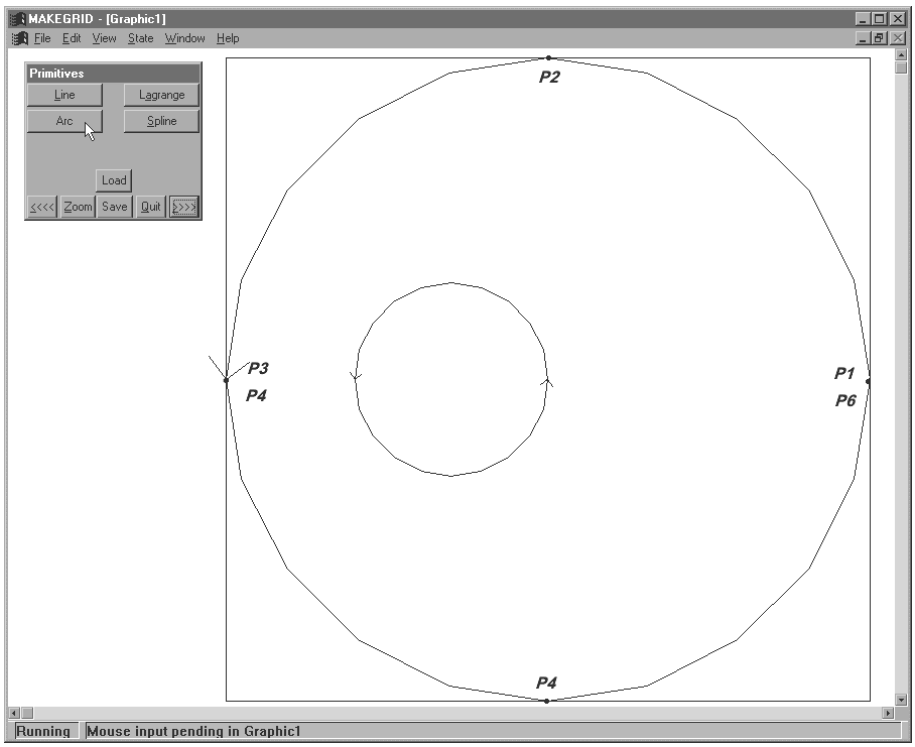


Figura 8.1: *(a) Definição dos arcos internos e (b) externos.*

Passo 2: Os contornos

O passo seguinte é informar ao programa como estas primitivas estão conectadas para definir os quatro contornos do quadrado unitário no domínio lógico. Devemos então selecionar no menu qual dos contornos Norte, Sul, Leste ou Oeste queremos definir. Escolhido um contorno, devemos selecionar na tela quais as primitivas que compõem este contorno. Na figura 8.2b estamos definindo o contorno Sul, neste caso composto pelos dois arcos internos, que são selecionados um a um. O contorno Norte é dado pelos dois arcos externos. Este contorno deve possuir a mesma orientação do contorno Sul. Podemos utilizar a opção $\leftrightarrow$ para mudar o sentido das primitivas, caso seja necessário, e após defini-las como componentes do contorno Norte. Por fim, temos os contorno Leste e Oeste que, no nosso exemplo, são a reta que define o corte.

Antes de passar para o próximo passo, o programa irá verificar se os quatro contornos foram corretamente definidos. Se alguma primitiva não estiver ligada a outra, como é o caso da figura 8.3a, elas podem ser conectadas através da opção JOIN.

Passo 3: A discretização

Agora iremos definir quantos pontos a malha terá. Através de duas caixas de edição escolhemos a quantidade de pontos no contorno norte e sul, M, e os pontos na dimensão leste e oeste, N. O número total de pontos de um contorno é distribuído uniformemente entre as primitivas que o compõem. A opção UNIFORM discretiza cada primitiva com pontos igualmente espaçados. Pode-se discretizar os pontos de uma primitiva de modo não-uniforme, atraindo os pontos para as extremidades ou para o centro da primitiva. Para isto, basta escolher o modo de atração desejado e clicar sobre a primitiva. No caso do cilindro, queremos que os pontos sejam atraídos em direção ao cilindro. Selecionamos então a opção de atração para o ponto inicial de uma primitiva e clicamos sobre a reta que define o corte (veja a figura 8.3b). Pode-se também escolher o grau de atração, alterando o valor de a.

Passo 4: A Geração da Malha

Nos passos 1 e 2 delimitamos a região do problema, definindo as primitivas e a forma como estas se encaixam. No passo 3, discretizamos os pontos do contorno. Neste passo, vamos gerar os pontos internos da malha. Terminado o passo 3, será gerada uma malha através do método TFI automaticamente. Se esta não for a malha desejada pelo usuário, esta malha será a condição inicial para os outros geradores. O usuário tem à sua disposição o método *Length*, o *Winslow* e o *TTM* não homogêneo, além do próprio *TFI*. Cada um desses métodos pode ser aplicado a uma

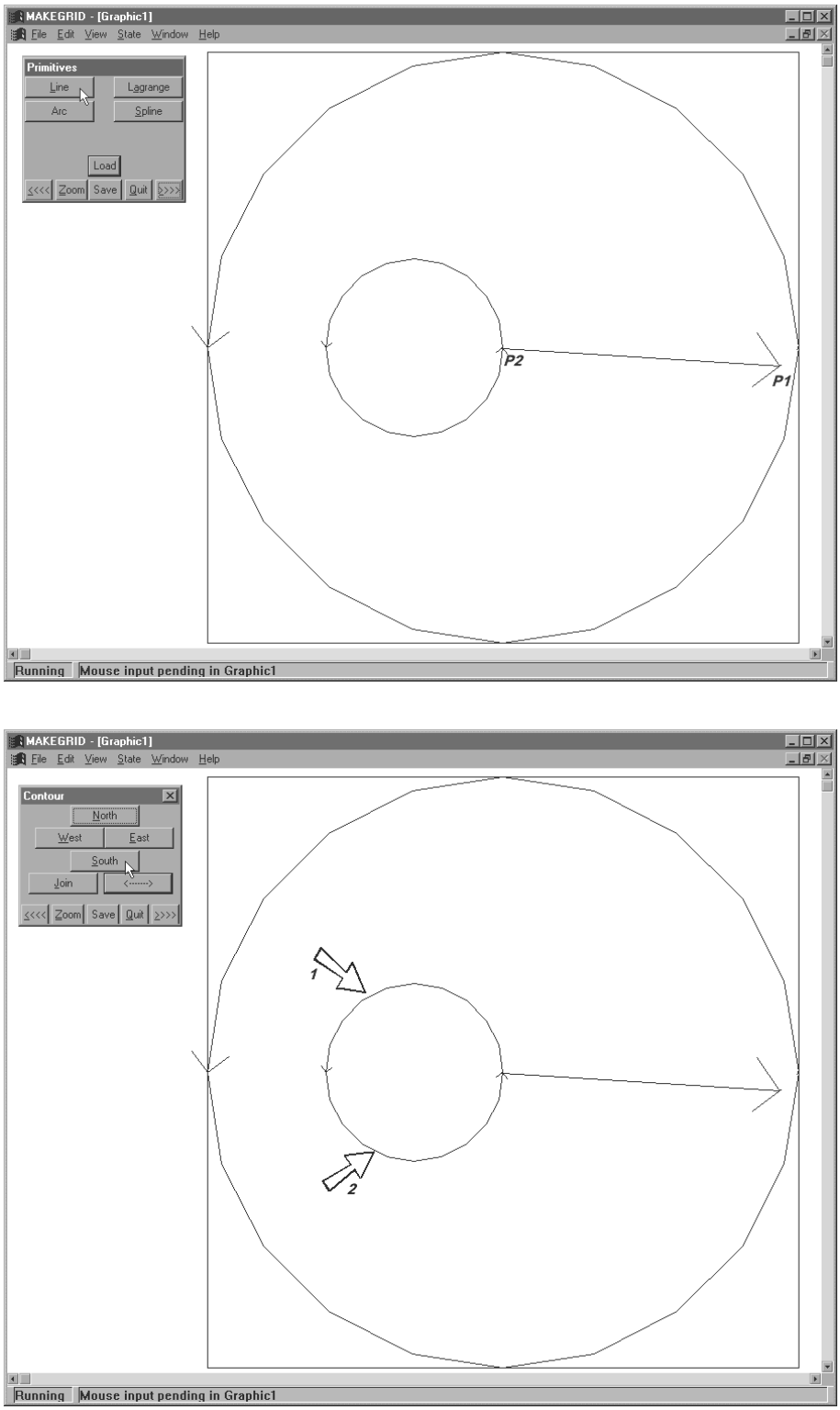


Figura 8.2: *(a) Definição do corte e (b) dos contornos.*

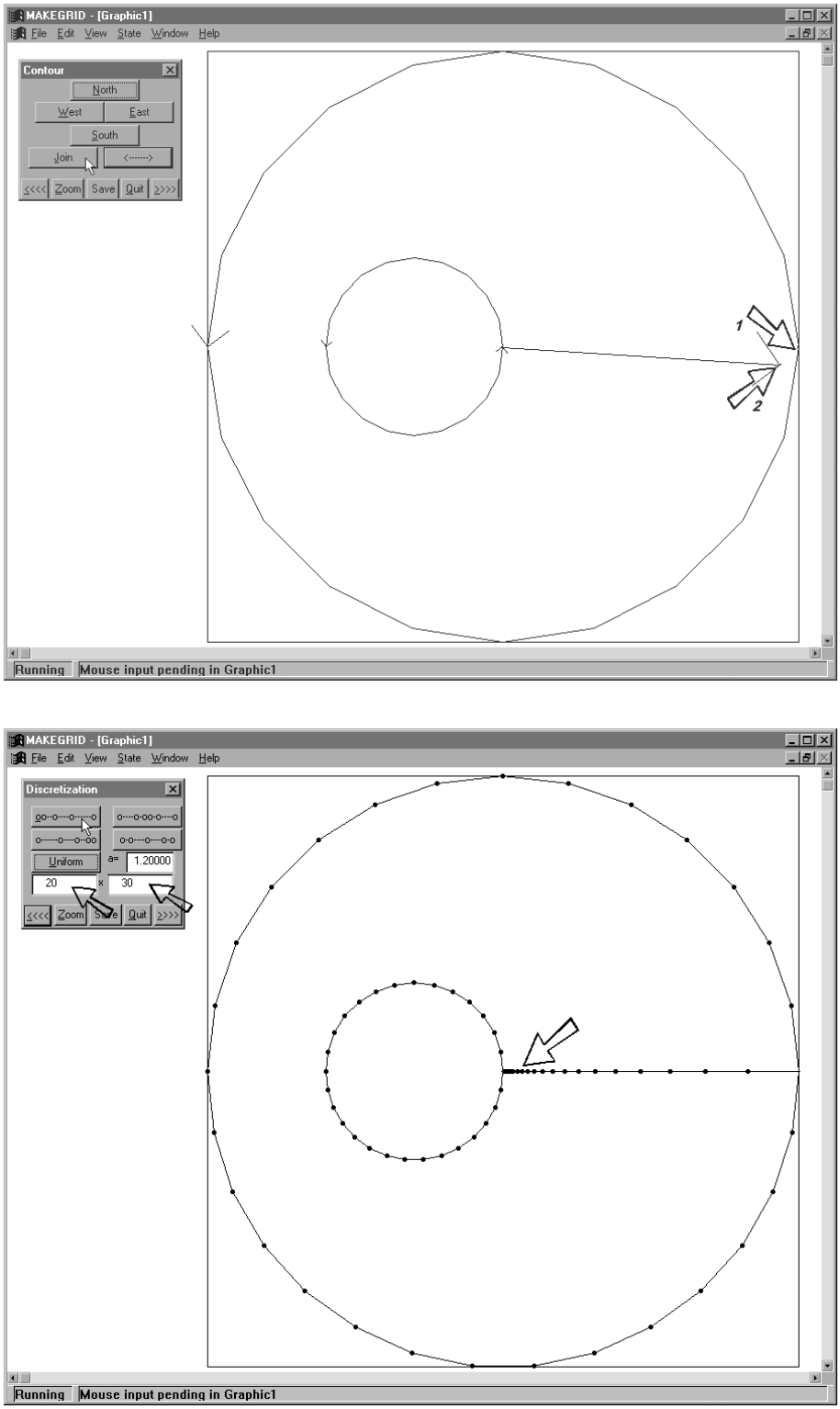


Figura 8.3: *(a) Conexão entre dois pontos distintos. (b) A malha terá* 20×30 *pontos e será atraída em direção ao cilindro.*

região específica da malha, que pode ser a malha toda. O usuário pode combinar e reaplicar os métodos disponíveis como desejar.

No nosso exemplo, vamos aplicar o método de Winslow sobre toda a malha, como mostra a figura 8.4a. Para isto, devemos escolher WINSLOW no menu e clicar duas vezes sobre um ponto fora da malha, para indicar ao programa que queremos que o gerador seja aplicado a toda a malha. Para selecionar uma região específica, devemos selecionar dois pontos na malha, que serão vértices opostos de um quadrado no espaço lógico, que definirá a região física desejada. Como ilustrado na figura 8.4b, esta região física será definida de acordo com a intersecção das linhas coordenadas que cruzam os pontos selecionados.

Alguns parâmetros específicos do algoritmo para o gerador de malhas podem ser alterados, tais como o parâmetro de relaxação w, o número de iterações it do laço interno e do laço externo do gerador e a tolerância tol.

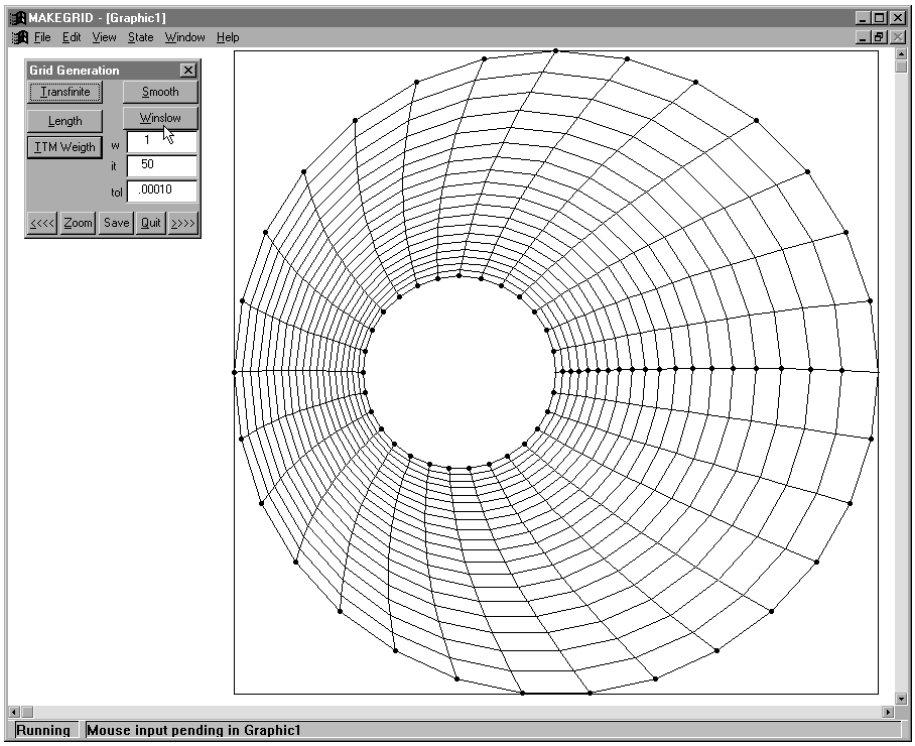
Passo 5: Análise da Malha

Após a geração da malha, algumas opções gráficas úteis para a análise de qualidade da malha são disponibilizadas. Entre estas, temos o desenho da própria malha e suas células fictícias e mapas de cores para a razão de aspecto das células e suavidade das linhas coordenadas. Outra possibilidade é a geração de gráficos para as métricas envolvidas na transformação da malha, g_{11}, g_{12} e g_{22}, onde g_{11} e g_{22} fornecem a informação sobre a concentração das linhas na direção ξ e η e a métrica g_{12} informa sobre a ortogonalidade da malha. O cálculo dessas quantidades é explicado na sessão 8.4.

Neste ponto, o usuário já possui uma malha disponível que pode ser salva em um arquivo e utilizada para gerar a solução através de seus próprios programas e métodos (o formato do arquivo de dados da malha é dado no apêndice A). Entretanto, o usuário pode utilizar o processo de definição de condições de contorno oferecido por este programa. Para isto, são necessários dois passos explicados a seguir.

Passo 6: As Variáveis Primitivas

O primeiro passo na definição das condições de contorno refere-se à definição das variáveis primitivas do problema, que são definidas pelo usuário através de uma lista. Na caixa de diálogo é informado somente um nome com o qual será referenciado a quantidade em questão. Qualquer *string* de caracteres pode ser utilizada. Por exemplo, podem ser definidas as variáveis U, V e Phi, que representam as componentes da velocidade e a pressão. As variáveis primitivas são



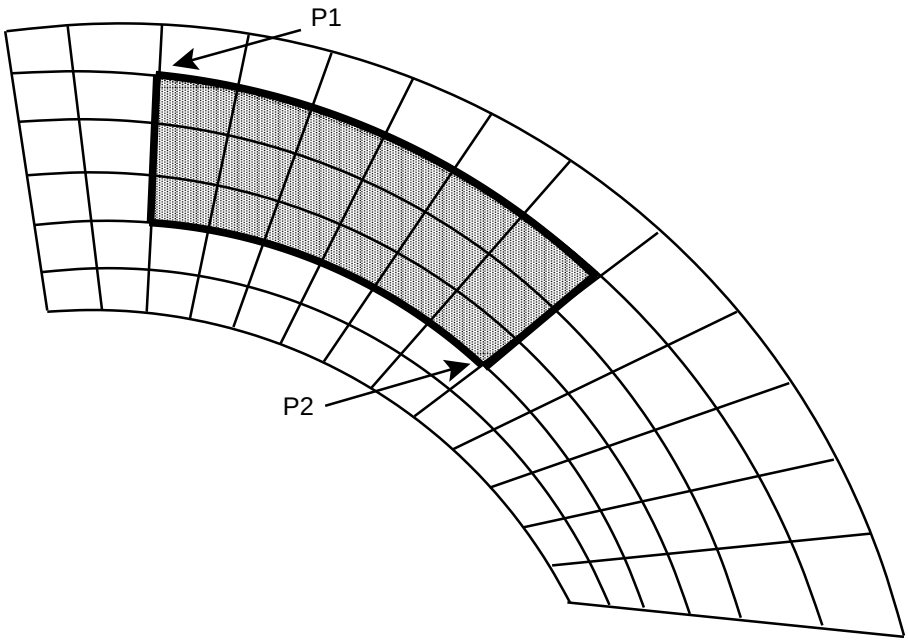


Figura 8.4: *(a) Geração da malha utilizando a equação de Winslow. (b) Definição de uma região da malha através dos pontos $P1$ e $P2$.*

incluídas através da opção INCLUDE e podem ser excluídas através da opção EXCLUDE (figura 8.5b).

Passo 7: As Condições de Contorno

A partir das variáveis primitivas escolhidas no passo anterior, podemos agora definir quais as condições de contorno para o problema que estamos resolvendo. Pode-se definir uma condição de contorno diferente para cada ponto do contorno. Em nosso exemplo, vamos usar as seguintes condições de contorno para a velocidade:

- no contorno interno usamos a condição de não-deslizamento, ou seja, a velocidade é zero, no contorno do corpo.
- na metade do contorno externo do lado direito usamos a condição de Neumann para as componentes u e v da velocidade.
- na metade esquerda usamos uma condição de Dirichlet, ou seja, fixamos as componentes da velocidade como $u = 1$ e $v = 0$; desta forma, estamos simulando uma condição de entrada para o fluxo.

A condição de contorno para a pressão será a condição de Neumann, de acordo com o método PRIME descrito no capítulo anterior.

Primeiramente, devemos digitar na caixa de diálogo um nome pelo qual nos referenciaremos a um determinado conjunto de condições de contorno no programa. Este conjunto é formado pelas condições de contorno para cada variável primitiva definida. Começaremos pelo contorno interno, que receberá o nome de *corpo*. Definimos então no *corpo* a velocidade $[u, v] = [0, 0]$, escolhendo a condição de Dirichlet e o valor 0 na caixa de edição para ambas as componentes da velocidade. A condição para a pressão é a condição de Neumann em relação a normal igual a zero, $\frac{\partial p}{\partial n} = 0$. Depois de definidas essas condições, escolhemos sobre quais pontos da malha queremos que este conjunto de condições de contorno seja aplicado. Para isso, marcamos na tela um ponto inicial e um ponto final de forma a delimitar uma região da malha sobre a qual estas condições serão aplicadas. Esses dois pontos podem definir um segmento de uma linha coordenada, uma região retangular da malha ou um contorno inteiro da malha.

Portanto, para a condição no *corpo* selecionamos duas vezes um mesmo ponto do cilindro interno para que todo o cilindro interno seja selecionado (veja figura 8.6a).

Depois, definimos a condição de *saída*, que está na metade direita do círculo externo. Definimos então a condição de Neumann em relação à normal para as componentes da velocidade

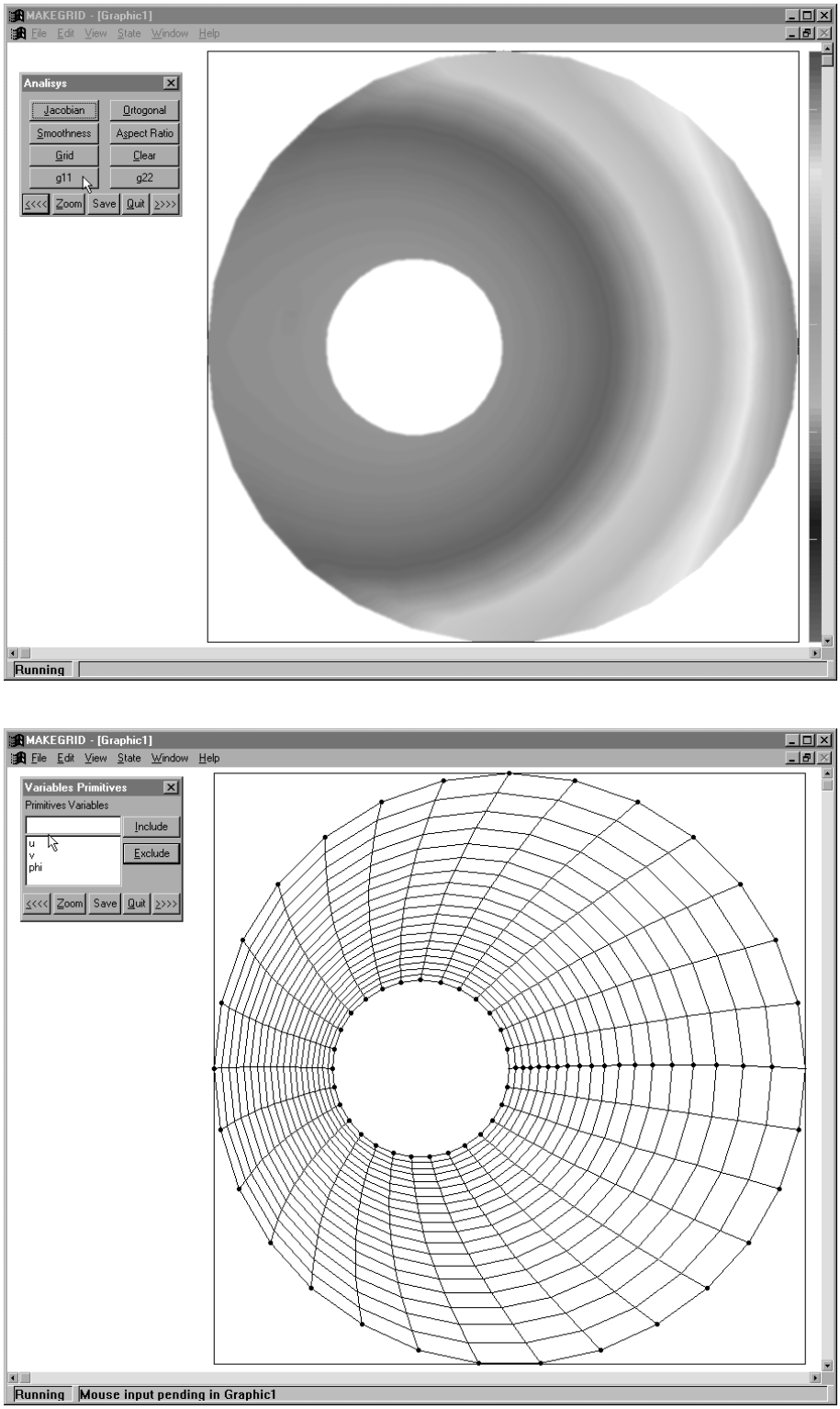


Figura 8.5: *(a) Análise da malha através do gráfico da métrica g_{11} e (b) definição das variáveis primitivas u, v e Phi.*

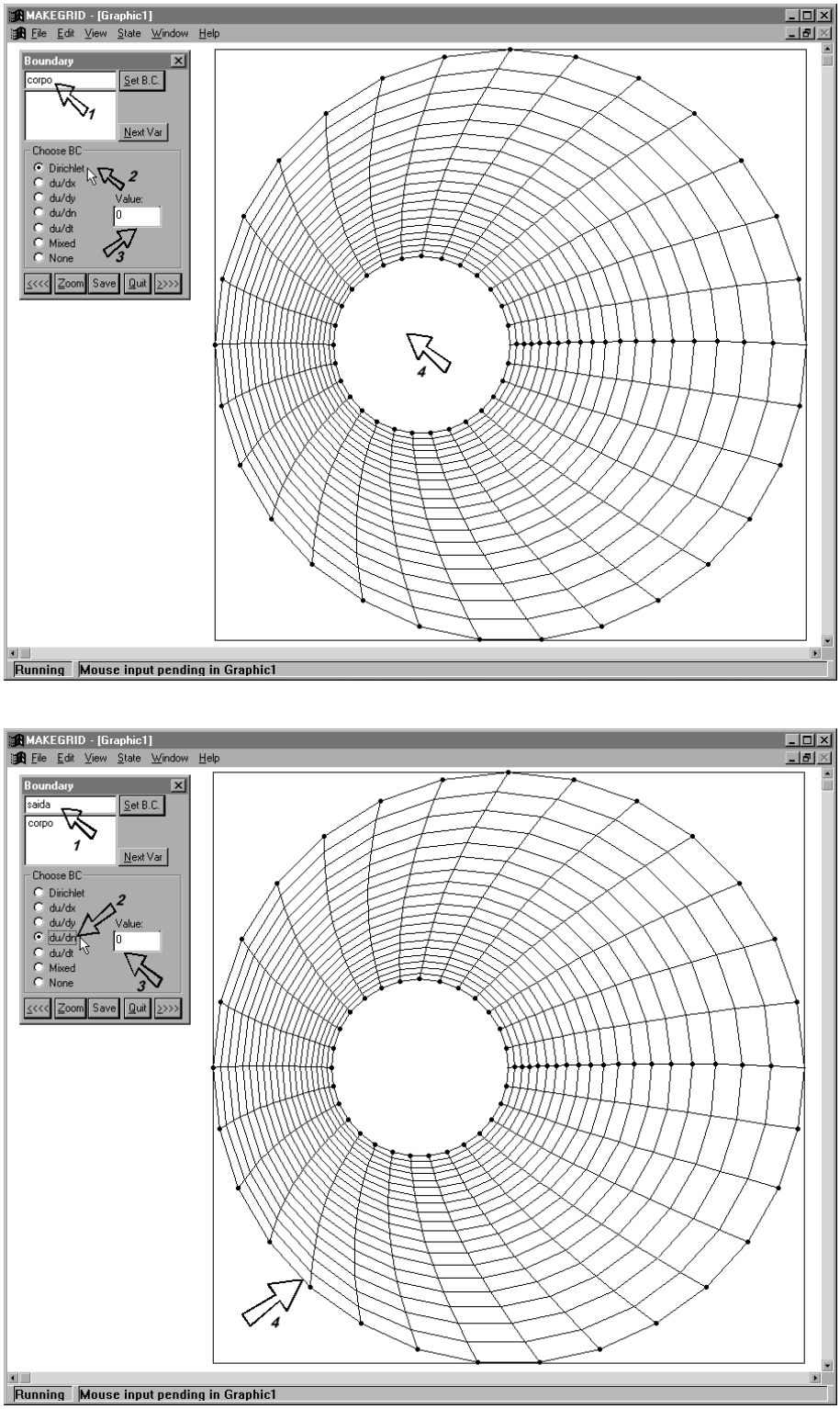


Figura 8.6: *Definição do contorno interno selecionando duas vezes o mesmo ponto do cilindro interno e (b) definição do contorno externo selecionando duas vezes um ponto do contorno norte.*

e para a pressão. Selecionamos duas vezes um mesmo ponto externo de forma a escolher todo o contorno externo (veja figura 8.6b). É claro que somente metade do contorno externo será a *saída*, mas corrigiremos isso no momento da definição da outra condição que será a *entrada*. Na *entrada* temos as componentes $u = 1$ e $v = 0$, ambas condições de Dirichlet. No momento da seleção do contorno, escolhemos os pontos inicial e final da metade esquerda do contorno externo (veja figura 8.7a). Deve-se observar a orientação do contorno ao escolhermos os pontos inicial e final, para que não seja escolhida a metade direita do contorno.

Para o corte, não é necessário definir condições de contorno, pois deve ser tratado como uma linha coordenada interna qualquer da malha. Ao final deste passo, o usuário pode salvar as condições de contorno para que possam ser utilizadas posteriormente. O formato do arquivo onde as condições de contorno são salvas está disponível no apêndice C.

Passo 8: Solução das Equações

O programa oferece a opção para a solução numérica da equação de Laplace, que foi utilizada para os testes iniciais do programa. Esta é utilizada para a condução do calor e também pode ser usada para a solução da função corrente no caso de escoamentos em baixa velocidade. Esta equação é solucionada numericamente através do método de Gauss-Seidel [38] utilizando as condições de contorno definidas pelo usuário nos passos anteriores. Além desta opção, implementamos o método PRIME, que pode ser usado para resolver escoamentos incompressíveis para diversas configurações. Foram implementados os arranjos co-localizado e diferenciado para malhas generalizada e cartesiana, totalizando quatro diferentes opções de solução.

Para solucionarmos o escoamento incompressível sobre o cilindro, usamos as condições de contorno escolhidas nos passos anteriores e calculamos it iterações utilizando o método PRIME. Durante o procedimento de solução, é apresentado na tela um gráfico do resíduo em função do número de iterações para que o usuário acompanhe o processo (veja figura 8.7b. Alguns exemplos de soluções numéricas são apresentados ao final do capítulo.

8.2 Descrição da Metodologia

O programa foi feito na linguagem Fortran 90 sobre a plataforma Windows 95, utilizando-se para isso o compilador Fortran Power Station 4.0 da Microsoft e vários recursos avançados como, por exemplo, orientação a objetos. Como pode ser observado através do exemplo da seção anterior, o programa é totalmente modular, ou seja, o programa é dividido em várias etapas que estão interligadas entre si.

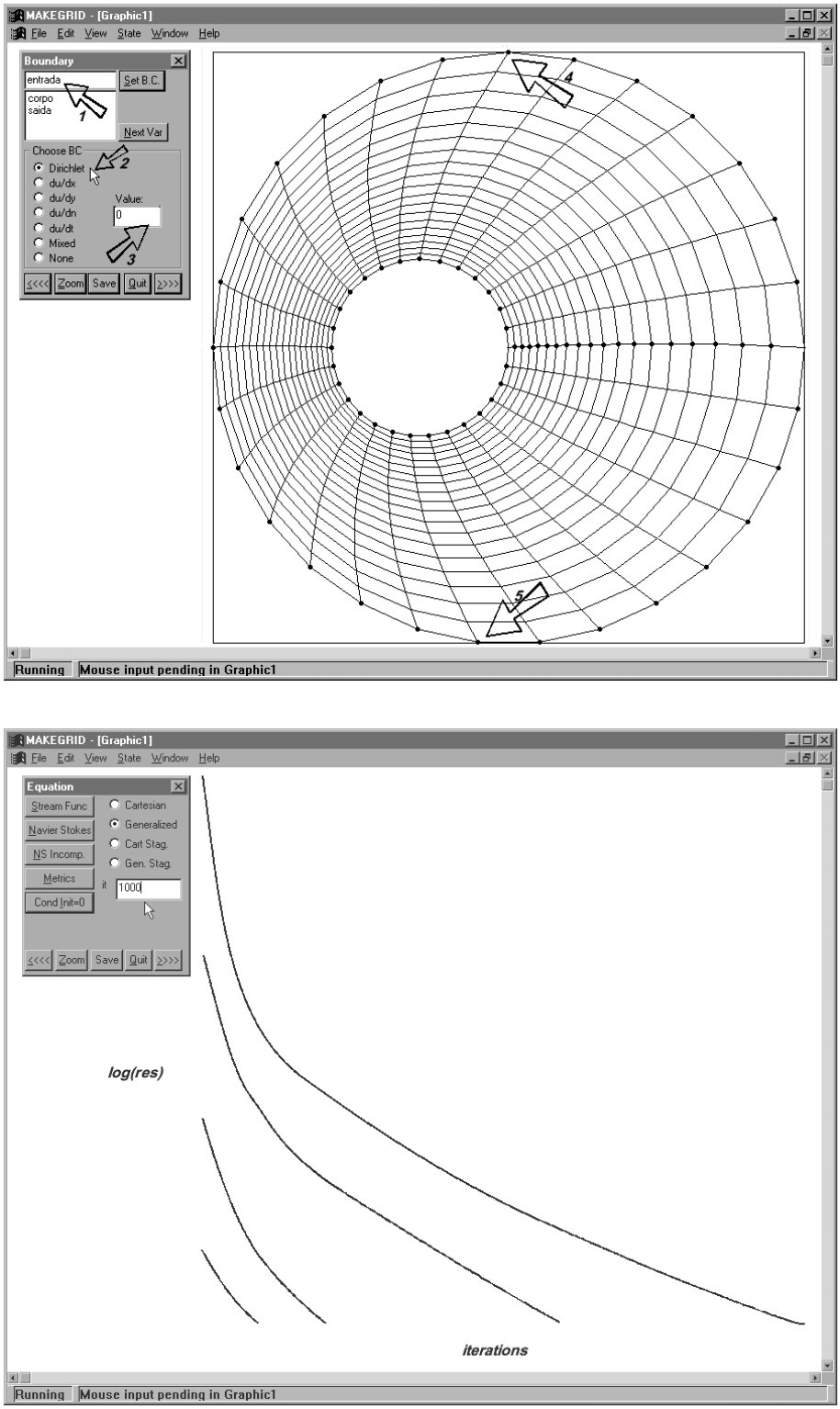


Figura 8.7: *(a) Definição da parte do contorno externo que delimita a entrada do escoamento. (b) Solução das equações: gráfico do resíduo × número de iterações.*

O programa possui mais de 7000 linhas de código, sem levar em conta a utilização de uma biblioteca gráfica que possui aproximadamente 3000 linhas. Este está dividido em 19 módulos contendo mais de 200 subrotinas e funções. Claramente, faz-se necessária a utilização de uma metodologia que divida as diversas tarefas do programa de forma a facilitar sua codificação, manutenção e extensões futuras. De uma forma geral, o programa pode ser apresentado esquematicamente de acordo com a figura 8.8.

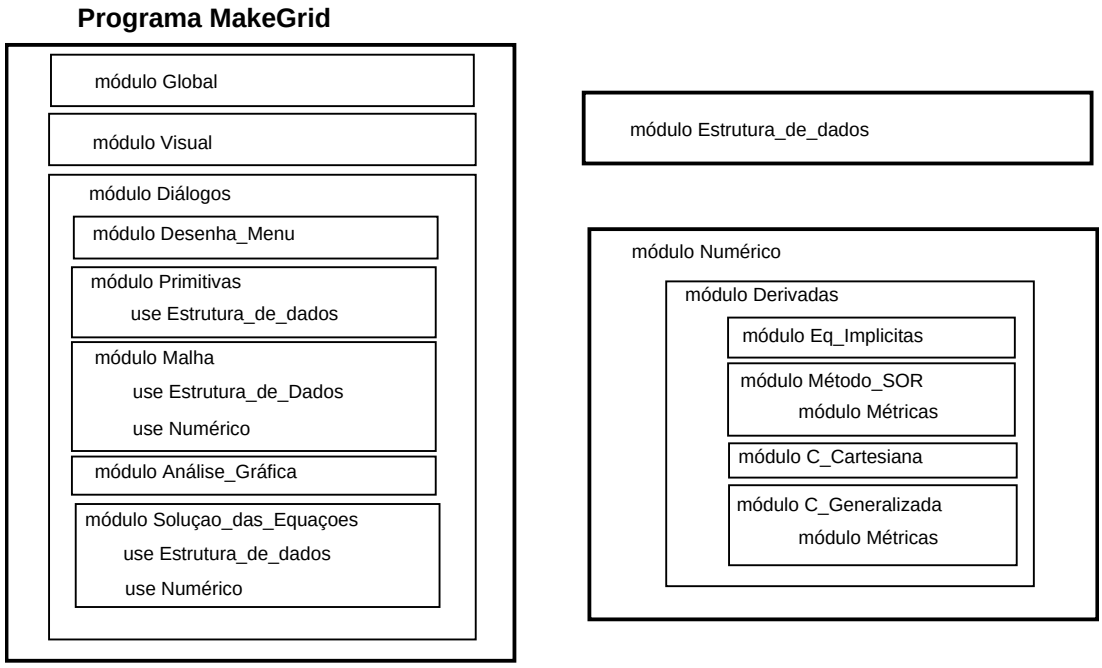


Figura 8.8: *Descrição modular do programa.*

O programa principal chama-se MakeGrid. Na verdade, este é responsável somente pela chamada dos diferentes módulos e pela concatenação das variáveis necessárias entres estes. O programa principal também é responsável por ler e salvar os arquivos de dados do programa, que são explicados no apêndice B. O módulo Global possui a definição de todas as variáveis que são globais ao programa, ou seja, variáveis que fazem parte de quase todas as subrotinas. Desta forma estas não precisam ser passadas como parâmetros na chamada das subrotinas.

O módulo Visual é responsável pela ligação entre o programa e a biblioteca gráfica VISUAL. Esta biblioteca é um conjunto de subrotinas que fazem parte do programa VISUAL[29], reformuladas no início do projeto a fim de otimizá-las e fornecer um encapsulamento de forma a facilitar sua utilização em qualquer código. Por exemplo, para desenhar uma malha, um programa que use a biblioteca VISUAL pode utilizar um comando simples como

```
call V2Grid (X(1:M,1:N),Y(1:M,1:N))
```

onde as matrizes X e Y contém os pontos da malha de dimensão $M \times N$. A biblioteca se encarregará das funções de abrir a janela gráfica e plotar os dados na tela. Desta forma, o trabalho gráfico envolvido no programa fica a cargo desta biblioteca, facilitando desta forma o desenvolvimento do programa em si.

O módulo Diálogos, juntamente com o programa principal, é responsável pelo gerenciamento dos dados entre as diferentes etapas do programa, que são as seguintes:

- entrada das primitivas;
- definição dos contornos;
- discretização dos contornos;
- geração da malha;
- análise da malha;
- variáveis primitivas;
- definição das condições de contorno;
- solução das equações desejadas.

O módulo Desenha_Menus trata da chamada das funções da biblioteca *Dialogm*, a qual faz parte do Microsoft Fortran, e gerencia os menus e caixas de diálogos.

Durante o processo da solução de um determinado problema, o usuário realiza etapa por etapa, seguindo uma ordem lógica para a resolução do problema e, se necessário, o usuário pode retornar algumas etapas para reconfigurar o problema.

8.2.1 A estrutura de dados

O módulo Primitivas é responsável pelas três primeiras etapas da definição de uma malha, descritas na lista acima. Este módulo trata das definições das primitivas, que são retas, splines, arcos e polinômios de Lagrange. Como o programa é modular, facilmente podem ser implementadas novas primitivas desde que estas possam ser definidas de uma forma paramétrica. Faz-se essa necessidade, pois no momento da discretização do contorno, o parâmetro s que gera a primitiva é transformado para o intervalo $0 \leq r \leq 1$ e então r é discretizado de acordo com a opção escolhida, isto é, utilizamos o parâmetro para obter a aproximação para um determinado ponto da malha.

As primitivas e os contornos possuem uma estrutura especial, que é gerenciada pelo módulo `Estrutura_de_dados` de forma a ligá-los uns aos outros para poderem definir a região da solução.

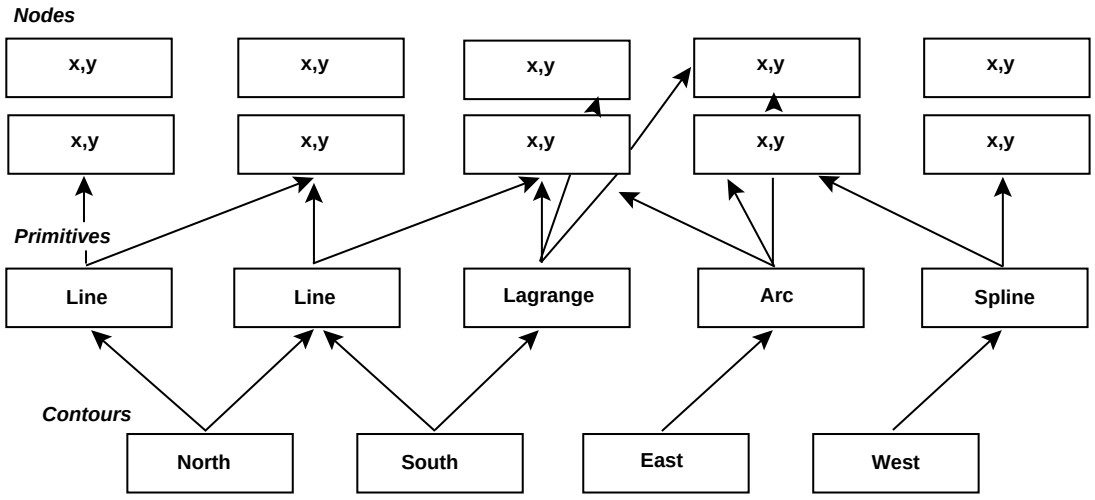


Figura 8.9: *Estrutura de dados.*

No momento que desenhamos uma primitiva ou a lemos de um arquivo de dados, esta é armazenada numa lista de primitivas, que por sua vez é ligada com uma lista de nós. Para uma linha reta, necessitamos dois nós que serão o ponto inicial e o ponto final. Para definirmos um arco precisamos do ponto inicial, de um ponto intermediário e do ponto final do arco. Uma spline e um polinômio de Lagrange são determinados através de n pontos que pertençam à curva. A estrutura dos dados é apresentada esquematicamente na figura 8.9. Cada uma dessas primitivas pertence à lista de primitivas e possui ponteiros para uma lista de nós. Algumas vezes um nó pertence a duas primitivas ao mesmo tempo, indicando que as duas primitivas estão ligadas.

Na etapa da definição dos contornos, são definidas quatro listas, norte, sul, leste e oeste, uma para cada contorno e são definidas na tela, ou através do próprio arquivo de dados, quais primitivas pertencem a cada contorno. Novamente, uma primitiva pode pertencer a dois contornos diferentes, como no caso de um corte. Além disso, nos dois contornos a primitiva poderá ter sentidos diferentes. Para passar para a próxima etapa, são verificados os vínculos de ligação entre os contornos da malha que são dados pelas equações (3.3) e (3.4).

8.2.2 A geração da malha

Na etapa da discretização dos contornos, utilizamos as equações do capítulo 2, pois estes são tratados como malhas unidimensionais no plano. Isto é feito no módulo `Malha`, que

também contém as rotinas necessárias para a geração das malhas bidimensionais que são a interpolação transfinita, os geradores *Lenght*, *Winslow* e TTM não homogêneo. Existe também uma opção chamada SMOOTH que serve para suavizar uma região da malha utilizando-se para isso o operador Laplaciano. Para definirmos onde aplicaremos cada um desses geradores devemos selecionar uma região na malha. Fazemos isso definindo dois pontos que definem uma região quadrada no espaço lógico (veja figura 8.4).

Um dos pontos fortes do programa é a facilidade do uso do gerador TTM não-homogêneo para a concentração da malha em linhas coordenadas e pontos específicos. Quando esta opção é escolhida, o usuário poderá determinar graficamente para onde ele quer que a malha seja atraída. Para isto, basta indicar com o *mouse* para qual linha ξ a malha deve se concentrar. Depois, se houver, deve-se selecionar para qual linha η a malha deve ser atraída e por último, quais os pontos de atração da malha. Note que qualquer uma das três possibilidades (linha ξ, linha η ou ponto P_{ij}) pode ser escolhida em conjunto com as demais. Pode-se também variar o número de linhas ξ ou η e o número de pontos para onde a malha será atraída. Por exemplo, se um determinado ponto for escolhido duas ou mais vezes, então o fator de atração para esse ponto será o dobro ou mais, de acordo com a escolha, o mesmo funcionando para linhas selecionadas duas vezes. Entretanto, se forem escolhido muitos pontos, o gerador TTM não-homogêneo terá uma função peso com valores muito altos e o método iterativo pode não convergir. Apesar desta restrição, inerente ao gerador TTM, esta possibilidade gráfica elimina a árdua tarefa da determinação dos coeficientes das funções peso dadas por (3.21) e (3.22). Alguns exemplos de atração da malha para um ponto e para uma linha coordenada podem ser vistos na figura 3.7.

A análise das malhas é feita de modo gráfico pelo usuário através de várias informações que podem ser medidas sobre a malha e através do intervalo de variação e da localização dos valores sobre a malha. Isto será visto na seção 8.4.

8.2.3 Condições de contorno

Para definirmos as condições de contorno que serão usadas no programa, devemos primeiramente informar ao programa quais serão as variáveis primitivas do problema a ser resolvido. Através da caixa de diálogo devemos informar um rótulo pelo qual identificaremos cada variável primitiva. Por exemplo, poderão ser `u`, `v`, `phi`, `pressao`, etc. A cada um desses rótulos estará associado um valor inteiro $k = 1, \ldots, n$, onde n é a quantidade de variáveis primitivas do problema.

No momento da geração da malha, é criada uma *malha de estados* que associa a cada ponto da malha uma variável que indica que tipo de nó é aquele ponto. Ele pode ser um ponto de contorno, um ponto interior da malha, um ponto pertencente ao corte ou um ponto pertencente

a célula fictícia. Esses quatro tipos de estados são definidos automaticamente quando a malha é gerada. Veja na tabela abaixo um exemplo de como a malha de estados associada a cada malha é representada. Note que estamos representando duas malhas diferentes que estão unidas por um corte. Dependendo da posição de cada célula, esta receberá um código diferente de forma a identificar o tipo de célula em questão.

G	**B**	i	i	i	**B**	G	
G	**E**	c	c	c	**E**	G	$i = K$
F	F	F	F	F	F	F	$i = K + 1$

F	F	F	F	F	F	F	$i = 0$
G	**E**	c	c	c	**E**	G	$i = 1$
G	**B**	i	i	i	**B**	G	$i = 2$
G	**B**	i	i	i	**B**	G	
G	**B**	i	i	i	**B**	G	
G	**B**	i	i	i	**B**	G	
G	**B**	**B**	**B**	**B**	**B**	G	$i = N$
G	G	G	G	G	G	G	$i = N + 1$

Tabela 8.1: *Representação da malha de estados no corte que une duas malhas diferentes. As células podem ser de diferentes tipos: (**B**) célula do contorno, (**E**) célula especial entre o corte e o contorno, (**c**) célula do corte, (**i**) célula do interior, (**G**) célula fictícia externa e (**F**) célula fictícia do corte.*

Cada um desses quatro rótulos terá um código associado, dado pela tabela 8.2.3, juntamente com os códigos das condições de contorno definidas pelo usuário. Os códigos do usuário vão de 1 até *ncc*, onde *ncc* é o número de condições de contorno definido pelo usuário.

Estado	Código
nenhum	0
Condição 1	1
Condição 2	2
⋮	⋮
Condição *nc*	*nc*
Interior	32
Corte	64
Especial	128
Contorno	256

Tabela 8.2: *Códigos para cada tipo de célula e para cada uma das condições de contorno definida pelo usuário.*

Uma condição de contorno será definida através de um conjunto de condições de contorno, ou seja, para cada variável primitiva associamos uma das diversas condições de contorno possíveis. A esse conjunto de condições de contorno daremos um rótulo, e a cada um desses rótulos será dado um código. Por exemplo, para condição de não-deslizamento poderemos dar o nome *no-slip* e associarmos as condições de Dirichlet para as componentes da velocidade iguais a zero. Para

a pressão, se estivermos usando o método PRIME, usamos a condição de Neumann igual a zero. As condições de contorno e os códigos associados a cada condição estão na tabela 8.2.3. Podemos também não associar nenhuma condição de contorno a um ponto ou associar uma condição mista que será definida a cargo do usuário.

Condição de Contorno	Código
nenhuma	0
Dirichlet	1
$\frac{\partial \phi}{\partial x} = 0$	2
$\frac{\partial \phi}{\partial y} = 0$	3
$\frac{\partial \phi}{\partial n} = 0$	4
$\frac{\partial \phi}{\partial \tau} = 0$	5
Mista	6

Tabela 8.3: *Códigos para cada tipo de célula e para cada uma das condições de contorno definida pelo usuário.*

O usuário pode salvar esta malha de estados (veja formato do arquivo no apêndice C) para utilizar em seus programas. Assim, ao percorrer a malha, basta verificar o código que o ponto possui e aplicar o procedimento indicado pelo rótulo deste código.

8.3 O Módulo Numérico

O módulo numérico agrega as principais rotinas matemáticas do programa, possuindo as expressões para as derivadas, para o cálculo das métricas e para a solução de sistemas lineares.

Durante o desenvolvimento do programa foram utilizadas técnicas de orientação a objetos [16, 17] visando não somente a eficiência do programa, mas principalmente sua manutenção e futura extensão. As discretizações dos operadores (conforme visto no capítulo 5) foram implementadas de quatro maneiras diferentes:

- malha cartesiana e arranjo diferenciado;
- malha cartesiana e arranjo co-localizado;
- malha generalizada e arranjo diferenciado;
- malha generalizada e arranjo co-localizado.

Para cada arranjo, cada operador foi implementado na forma de função. Assim, cada módulo possui as discretizações para o Laplaciano, gradiente e divergente. Por exemplo, o Laplaciano de F no ponto i, j é dado pela função

```
function LAPLACIANO(F,i,j)
```

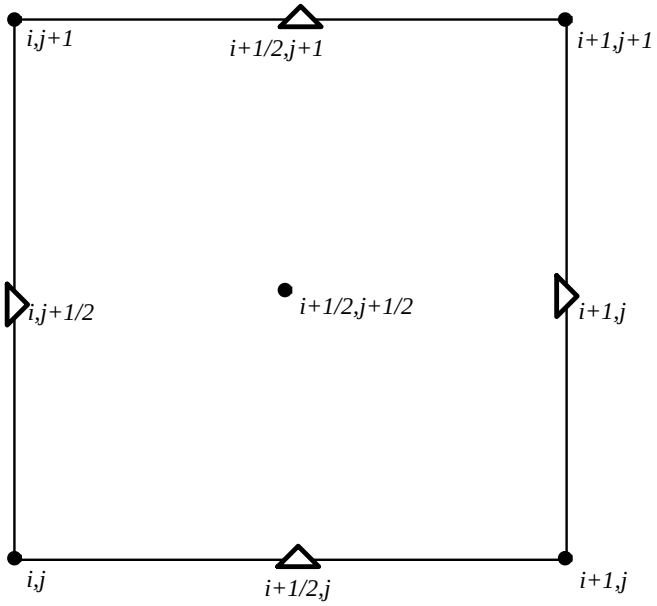


Figura 8.10: *Molécula com as posições i, j inteiras sobre os nós e as fracionárias sobre as faces e o centro da célula.*

onde F é uma matriz e i, j são valores inteiros ou não. Quando os valores não são inteiros, estamos calculando esta quantidade num ponto que não se intercepta com a malha. Isto é útil no caso do arranjo diferenciado.

O módulo `Metricas` contém as funções para o cálculo das componentes do tensor das métricas g_{11}, g_{12} e g_{22}, do Jacobiano e das derivadas em relação a ξ e η. Estas funções permitem o cálculo dessas quantidades em qualquer ponto i, j, aceitando valores não inteiros para i e j, sendo este o caso de pontos em uma face ou no centro de uma célula (veja figura 8.10).

Os operadores foram implementados com segunda ordem de precisão, exceto a derivada *upwind*, que é de primeira ordem. Como estas implementações estão na forma de módulos e funções, podemos sem muitas alterações usar o mesmo código que calcula uma determinada equação para diferentes arranjos de malhas. Da mesma forma, isso facilitará futuras implementações de métodos de mais alta ordem.

O algoritmo SOR para os diferentes arranjos e malhas foi implementado no módulo `Metodo_SOR` e compõe um módulo isolado do programa. Desta forma futuras otimizações podem ser feitas sem alterar o restante do código, inclusive uma possível paralelização do método SOR.

8.4 Análise Gráfica da Malha

Através de gráficos de várias quantidades, tais como suavidade, ortogonalidade e Jacobiano, podemos analisar a qualidade da malha e verificar se esta possui as características desejadas. As métricas, por exemplo, tem um papel importante na discretização dos operadores e influenciam diretamente na convergência do método utilizado. A maioria dos métodos necessita que essas quantidades físicas sejam suaves, ou seja, elas não podem apresentar uma variação brusca entre células vizinhas. Caso uma malha não esteja de acordo com as expectativas do usuário, ele pode retornar à etapas anteriores e mudar alguma definição dada no contorno ou refinar mais a malha para obter uma de melhor qualidade.

Suavidade

A suavidade S (*smoothness*) das linhas coordenadas da malha é estimada por

$$S = \frac{\mathbf{\Delta x}_e \cdot \mathbf{\Delta x}_w}{|\mathbf{\Delta x}_e \cdot \mathbf{\Delta x}_w|} + \frac{\mathbf{\Delta y}_n \cdot \mathbf{\Delta y}_s}{|\mathbf{\Delta y}_n \cdot \mathbf{\Delta y}_s|}. \tag{8.1}$$

O primeiro termo desta equação é o produto interno normalizado de dois vetores, que indicam o segmento à direita e o segmento à esquerda do ponto i, j sobre a linha coordenada ξ. Caso estes dois vetores estejam alinhados, o produto interno entre estes dois vetores resultará em 1, implicando que a suavidade da linha coordenada nesta direção é máxima. O segundo termo, da mesma forma, indicará a suavidade na direção η. Somando estes dois vetores temos que o valor ideal de S será igual a 2. Valores distantes deste valor ideal podem gerar descontinuidades indesejadas na solução, inexistentes no problema físico.

Razão de Aspecto

A razão de aspecto fornece a relação entre a base da célula e a altura da célula, sendo dada por

$$R = \frac{\sqrt{\mathbf{\Delta x} \cdot \mathbf{\Delta x}}}{\sqrt{\mathbf{\Delta y} \cdot \mathbf{\Delta y}}}. \tag{8.2}$$

Para um quadrado teremos que $R = 1$, enquanto que para um retângulo com base maior do que altura teremos $R < 1$. A razão de aspecto pode influenciar nas discretizações e devemos ter uma variação pequena entre o R máximo e mínimo.

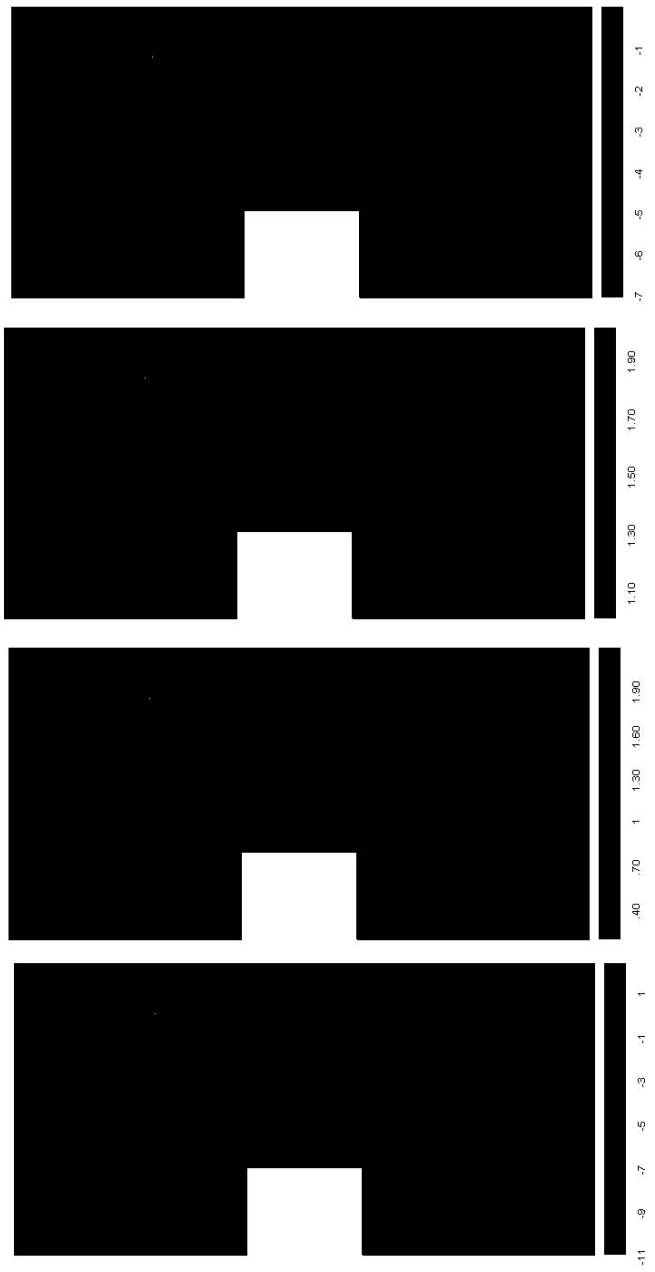


Figura 8.11: *Gráficos para análise da malha sobre um duto com restrição. Note que a metade esquerda é gerada com o gerador TFI e a direita com o Winslow. (a) Jacobiano, (b) suavidade, (c) razão de aspecto e (d) métrica* g_{12}.

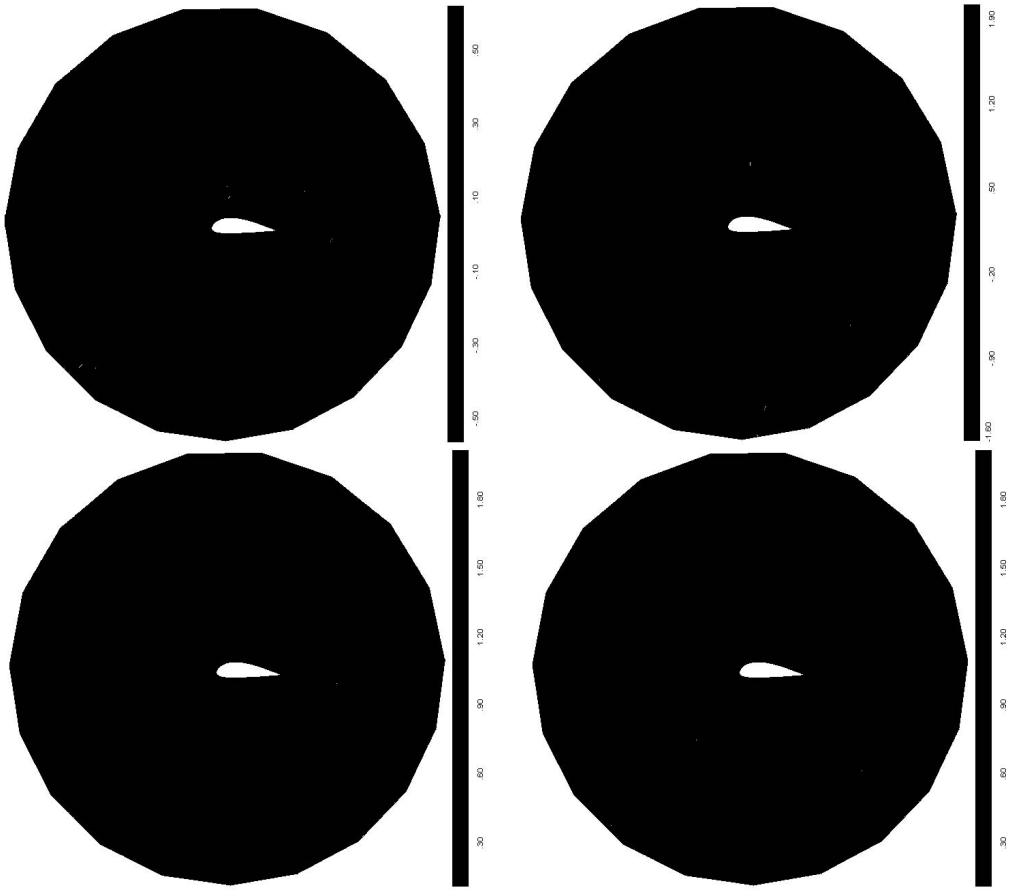


Figura 8.12: *Gráficos para análise da malha sobre uma aerofólio: (a) métrica g_{12} numa malha TFI, (b) métrica g_{12} numa malha Winslow, (c) suavidade numa malha TFI e (d) suavidade numa malha Winslow.*

Jacobiano

Na geração de malhas, umas das quantidades mais importantes é o Jacobiano, dado por

$$J = x_\xi y_\eta - y_\xi x_\eta. \tag{8.3}$$

O Jacobiano possui uma interpretação geométrica importante. Pode-se verificar facilmente que o Jacobiano fornece a área da célula. Como foi mencionado várias vezes nos capítulos iniciais, o Jacobiano não pode ser igual a zero, pois nesse caso a transformação não é inversível. A conseqüência física imediata é que no ponto onde o Jacobiano é zero, a malha dobra, pois o Jacobiano está trocando de sinal na vizinhança do ponto. Dependendo como as linhas coordenadas são dadas, o Jacobiano deverá ser sempre positivo ou sempre negativo, dependendo da orientação da malha. Não é aconselhável ter uma malha onde o gradiente do Jacobiano seja muito alto, isto é, onde o Jacobiano varie abruptamente numa pequena região da malha, pois nesse caso podem aparecer descontinuidades inexistentes na solução real do problema.

Métricas

O programa pode graficar também as métricas g_{ij}, dadas por

$$g_{11} = \mathbf{x}_\xi \cdot \mathbf{x}_\xi, \tag{8.4}$$

$$g_{12} = \mathbf{x}_\xi \cdot \mathbf{x}_\eta, \tag{8.5}$$

$$g_{22} = \mathbf{x}_\eta \cdot \mathbf{x}_\eta. \tag{8.6}$$

As métricas g_{11} e g_{22} fornecem os comprimentos da célula na direção ξ e η, respectivamente, e sua variação indica a concentração das linhas nesta direção. Este valores devem ser pequenos onde quisermos que a solução seja mais acurada, ou seja, onde quisermos captar de uma melhor forma fenômenos físicos, tais como vórtices localizados.

Para termos uma malha ortogonal, o valor de g_{12} deve ser igual a zero. Isso não é possível para quase todas as regiões que queremos discretizar, mas podemos fazer com que este valor torne-se tão pequeno quanto possível. Por esse motivo, as malhas generalizadas são geralmente chamadas de malhas não ortogonais. Existem geradores específicos [32, Seção 5.3] para minimizar o valor de g_{12} que não foram implementados neste programa, pois nem sempre conseguimos obter uma malha não dobrada com esses geradores, tornando sua utilização muito restrita a certas regiões. Na prática, o valor de g_{12} deve ser pelo menos uma ordem abaixo dos valores de g_{11} e g_{22}.

8.5 Resultados

A seguir, apresentamos uma série de exemplos de malhas geradas com o método Winslow obtidos através do programa MakeGrid. Estes exemplos foram escolhidos de modo a utilizar as opções oferecidas pelo programa e ilustrar a facilidade na geração de malhas sobre geometrias complexas. Ao final desta seção, são apresentados alguns resultados obtidos em algumas simulações numéricas, utilizando as rotinas descritas neste trabalho.

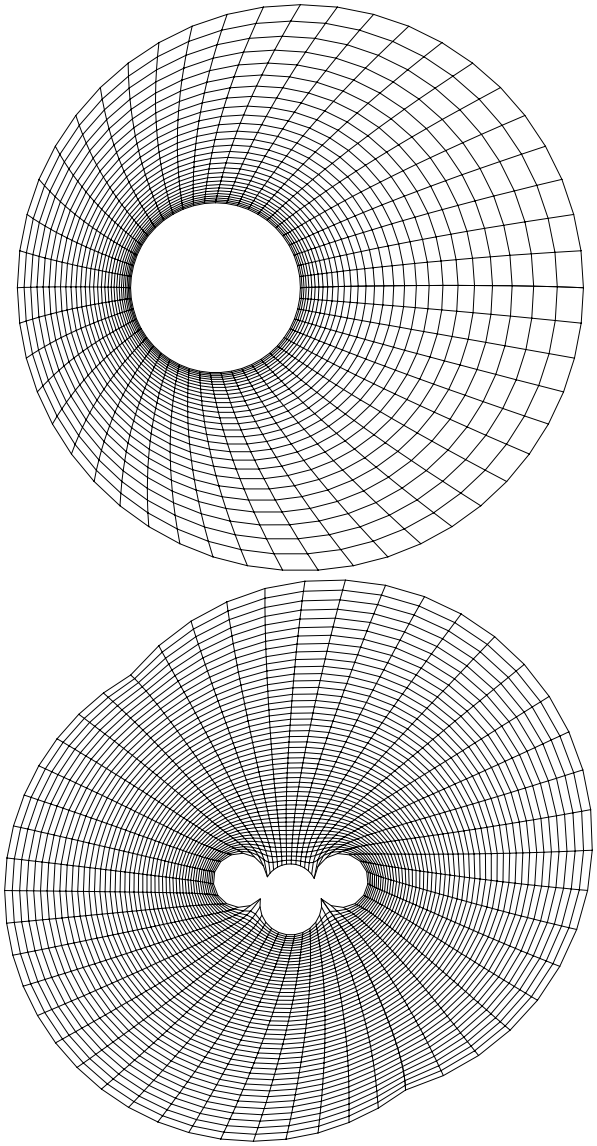


Figura 8.13: *Malhas sobre (a) um cilindro com eixo deslocado (*50×30 *pontos) e (b) três cilindros conjugados (*50×50 *pontos).*

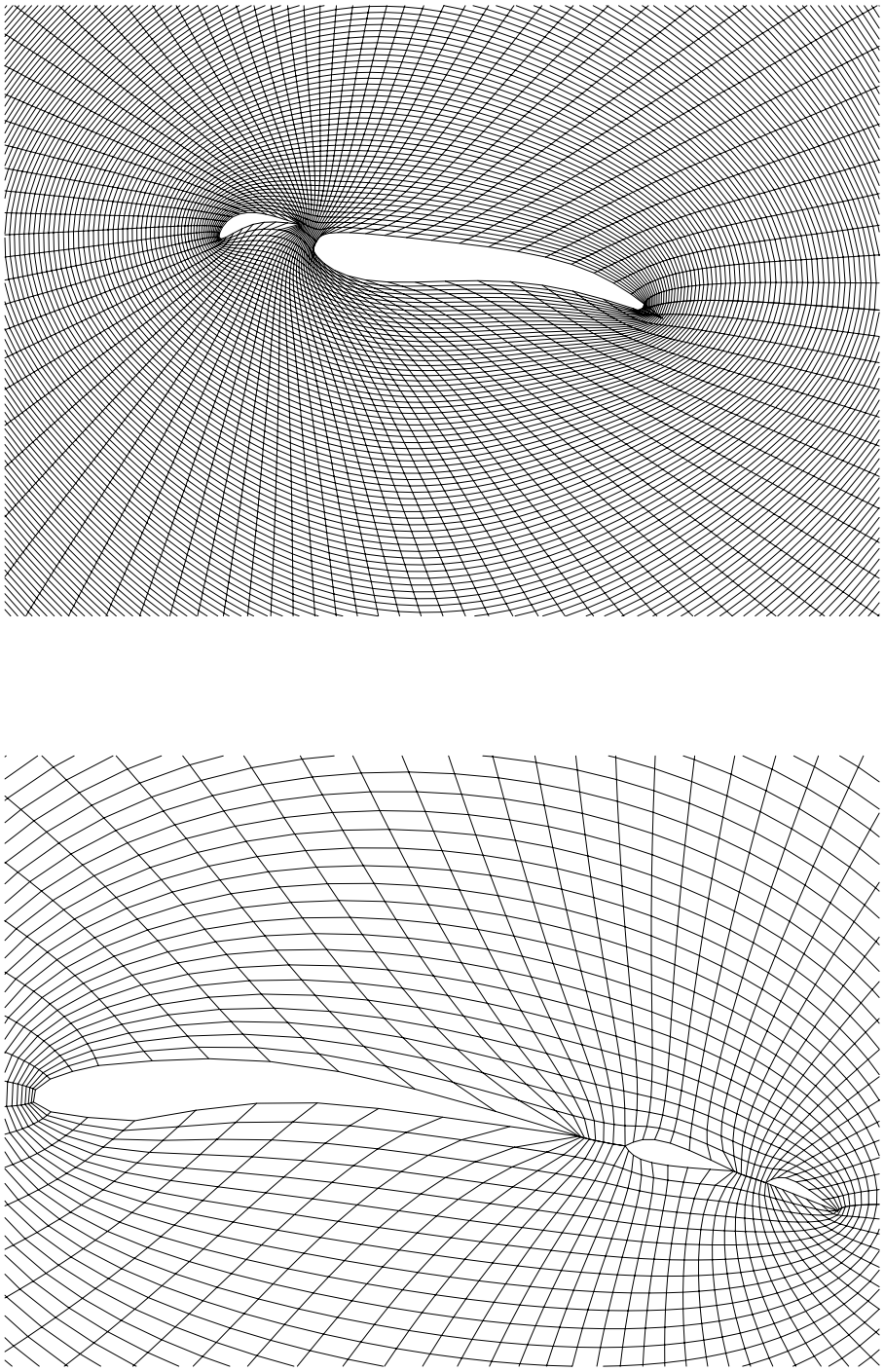


Figura 8.14: *Malhas sobre asas com (a) um* slat *(*100×100 *pontos) e (b) dois* flaps *(*80×80 *pontos). Os pontos foram aproximados em direção aos aerofólios em ambas configurações.*

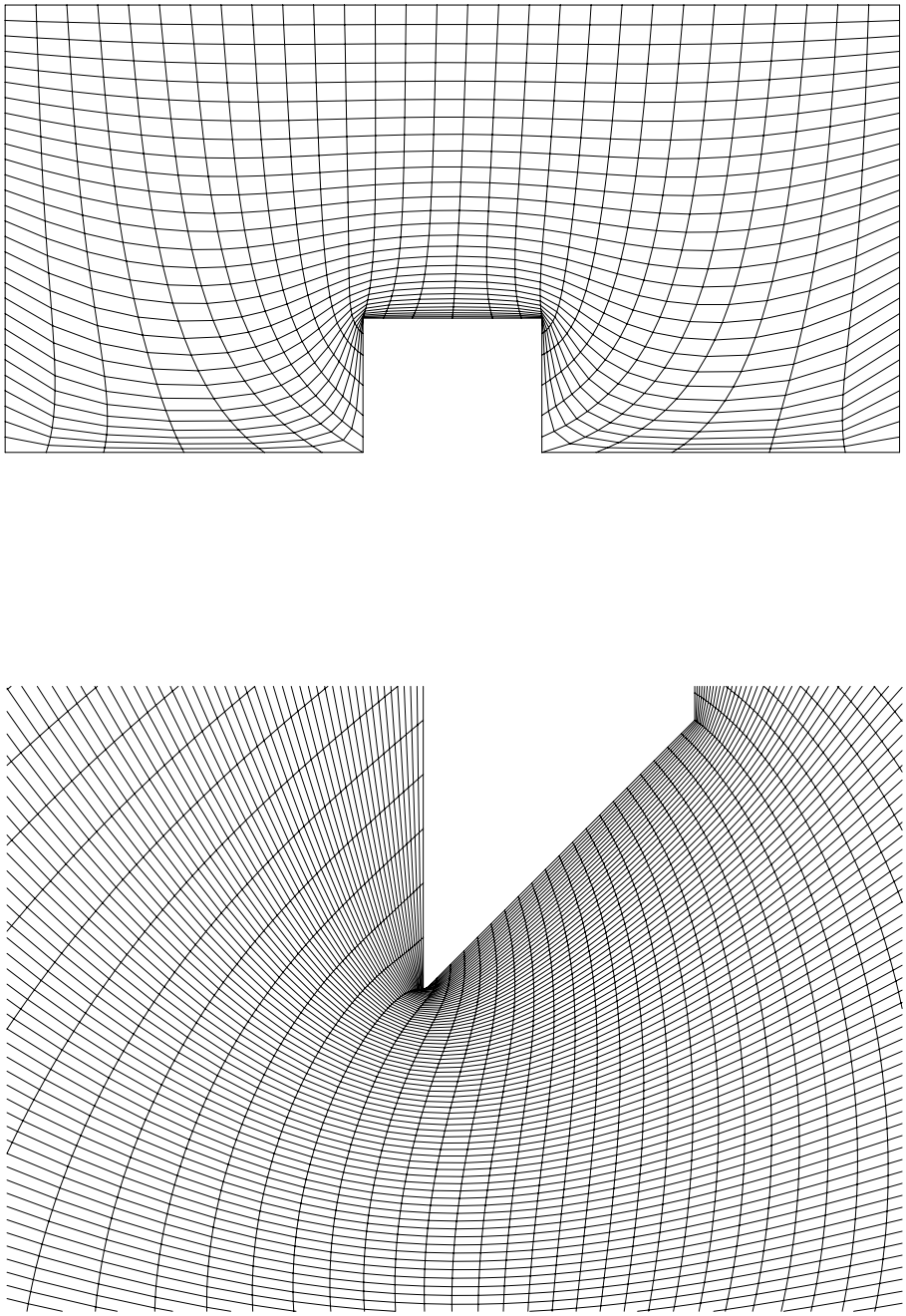


Figura 8.15: *Malhas para escoamentos internos em dutos com restrição. Os pontos foram aproximados nos cantos das restrições. (a)(*30×30 *pontos) (b)(*100×100 *pontos)*

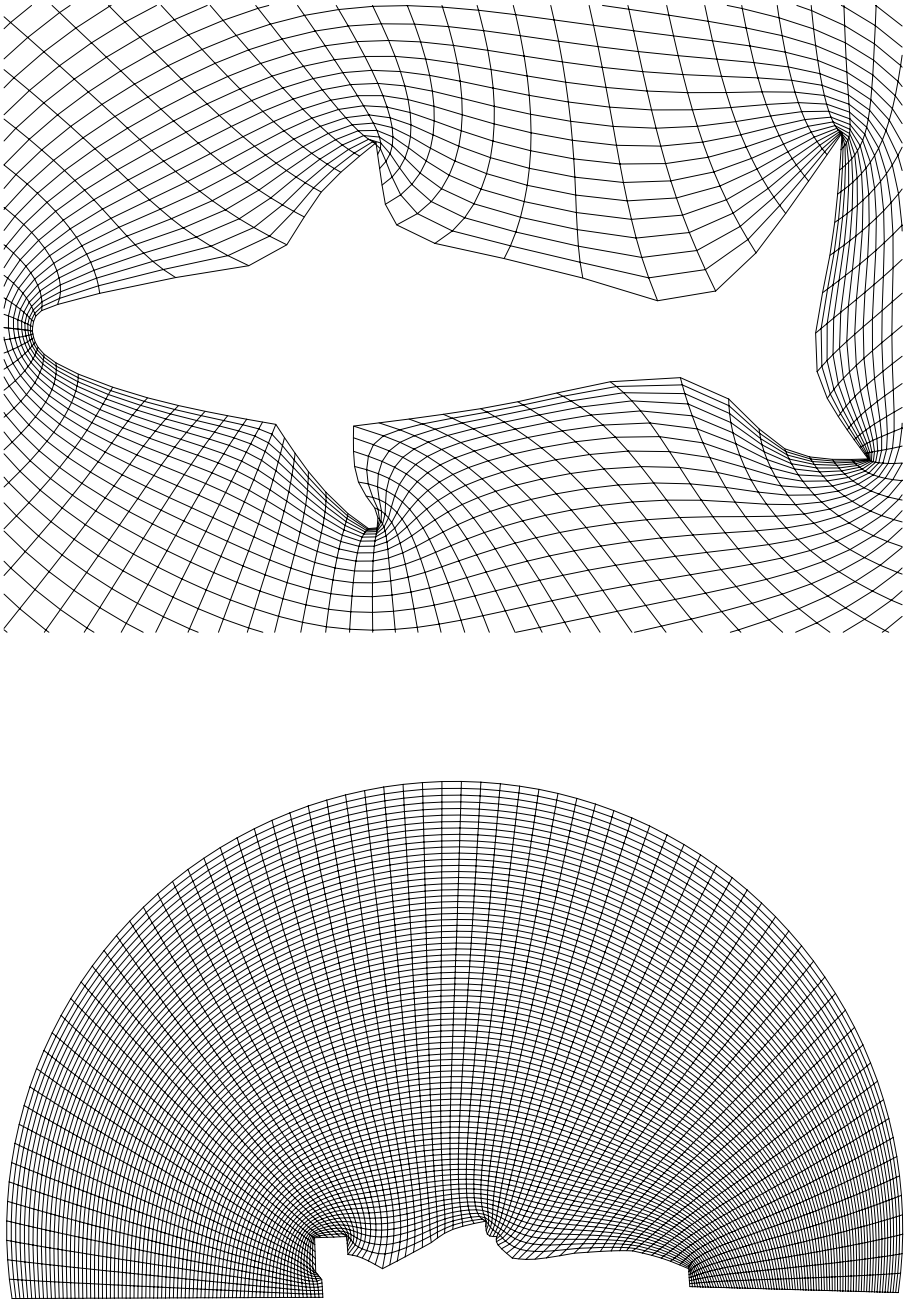


Figura 8.16: *Malhas para escoamento externo sobre (a) um tubarão (*100×100 *células)e (b) um carro de fórmula 1 (*80×80 *células).*

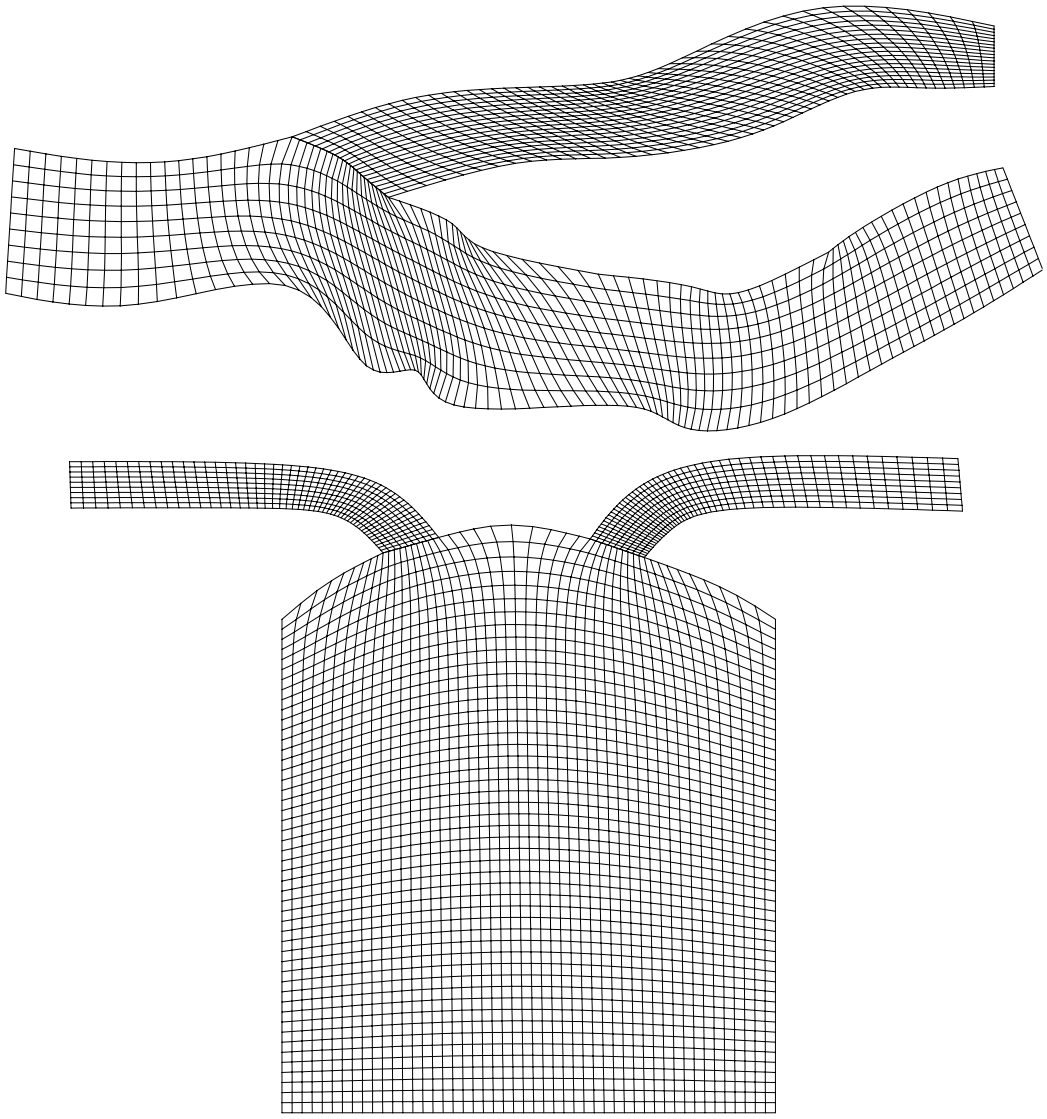


Figura 8.17: *Malhas com a utilização da técnica de multiblocos. (a) Na representação da carótida foram usados* 10×100 *pontos para o bloco na parte superior e* 20×50 *para o outro bloco. (b) Para a malha de uma câmara de combustão foram usados* 3 *blocos: a câmara de combustão (*50×50 *pontos), o duto injetor (*10×50 *pontos) e o duto de saída (*10×50 *pontos).*

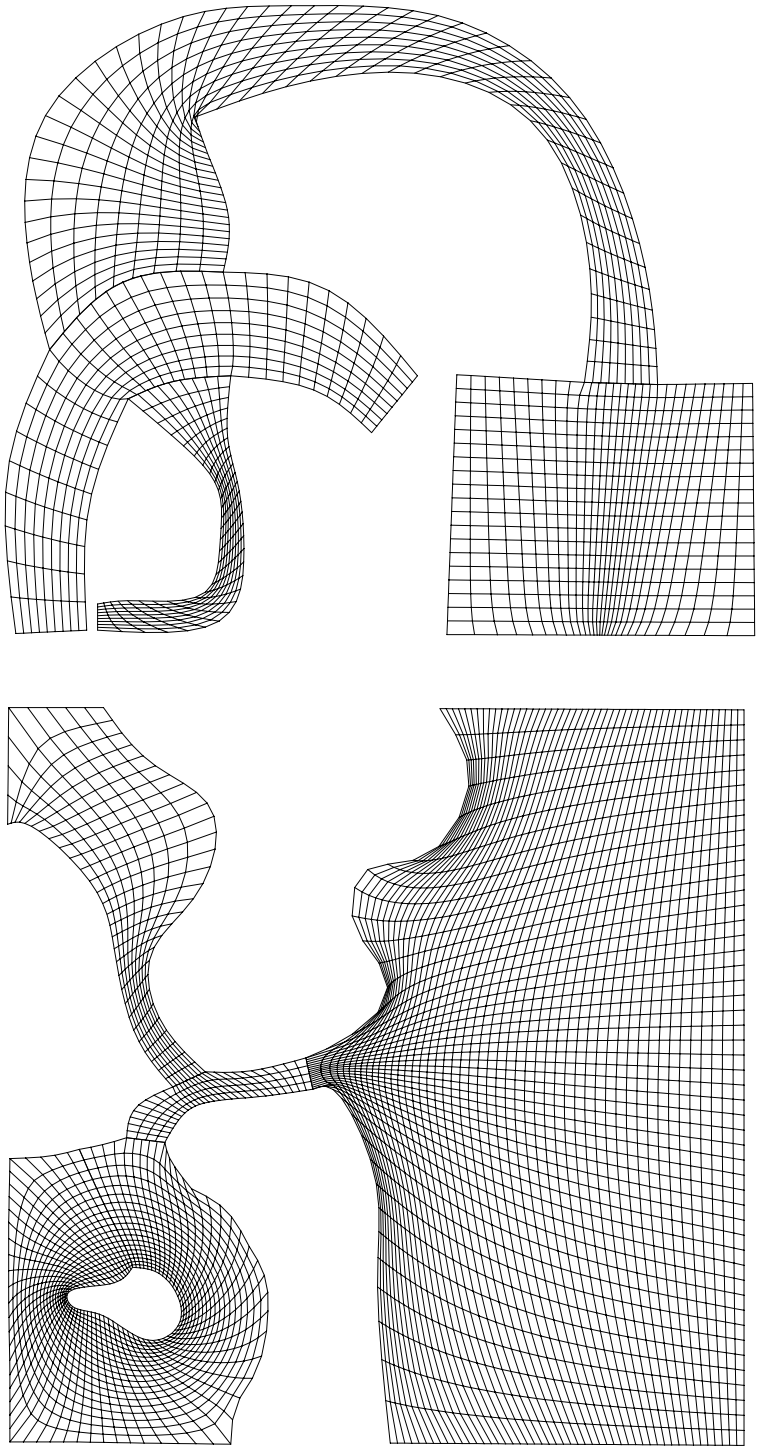


Figura 8.18: *Malhas com a utilização da técnica de multiblocos. (a) Uma malha formada por* 4 *blocos que poderia ser uma peça mecânica e (b) uma malha formada por* 4 *blocos para a simulação do escoamento de um rio e sua foz.*

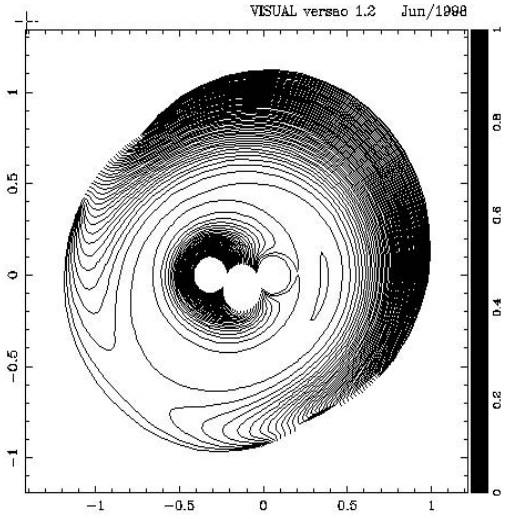


Figura 8.19: *Simulação de condução de calor sobre* 3 *cilindros conjugados. O cilindro interno à direita é resfriado e o à esquerda é mantido aquecido, assim como as paredes do contorno externo são aquecida à direita e resfriada à esquerda.*

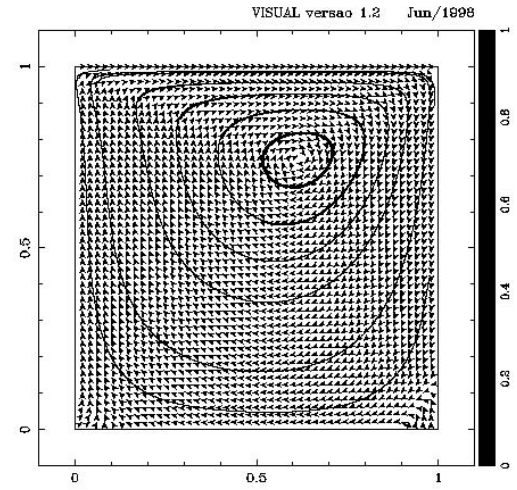


Figura 8.20: *Solução permanente da equação do escoamento para uma cavidade com* $Re = 100$. *Gráfico dos vetores velocidade e da função corrente. Foi utilizada uma malha com arranjo diferenciado.*

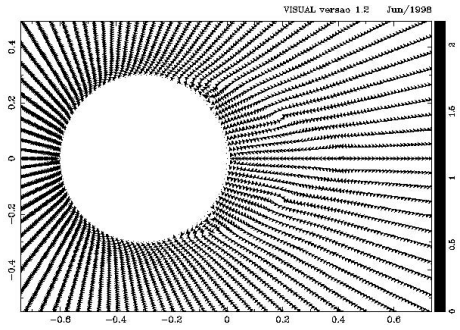


Figura 8.21: *Gráfico dos vetores velocidade sobre um cilindro para a solução da equação de Navier-Stokes com* $Re = 50$. *Neste caso foi utilizado o arranjo co-localizado.*

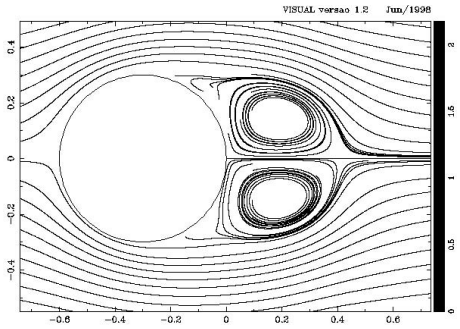


Figura 8.22: *Gráfico das linhas de corrente sobre um cilindro para a solução da equação de Navier-Stokes com* $Re = 50$.

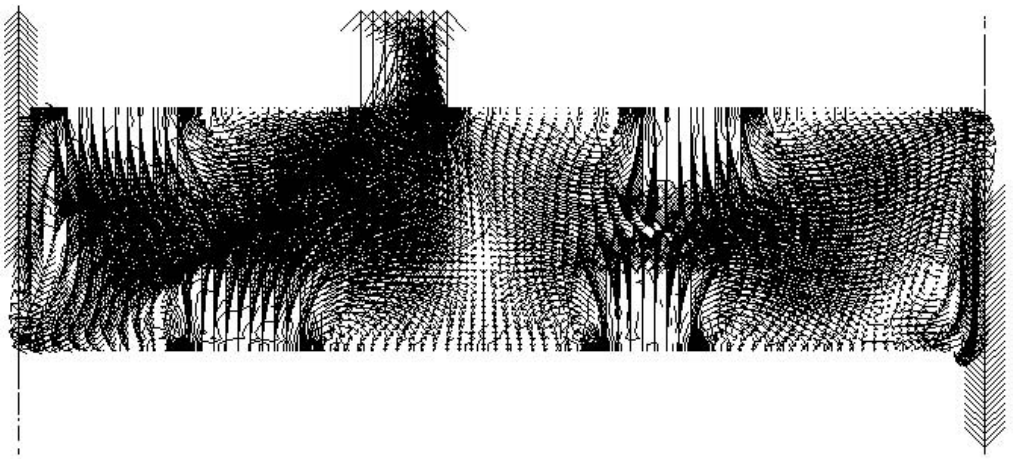


Figura 8.23: *Solução do escoamento em um tubo com condições de contorno variadas.*

9 CONCLUSÕES

A solução numérica de problemas de escoamento é tema de estudo de diversos pesquisadores, tanto neste Instituto como em outros deste país e do mundo. O público alvo do programa MAKEGRID são aqueles pesquisadores que desejam estudar o método de solução numérica de um problema físico e que, para isso, precisam de uma boa malha, sem se preocupar com o cálculo de métricas e quantidades afins. O programa desenvolvido oferece a malha que este usuário necessita e facilita a implementação das condições de contorno, fornecendo uma malha de estados que identifica as propriedades de cada ponto da malha.

Malhas sobre geometrias variadas podem ser geradas seguindo uma ordem clara, o que torna o programa uma ferramenta didática que pode ser utilizada em cursos de transferência de calor e dinâmica dos fluidos computacional. O programa funciona como uma ferramenta de experimentação, pois, através de poucas escolhas, várias configurações podem ser definidas e testadas. Caso em alguma etapa seja necessário voltar atrás, isso é possível, fazendo com que a tarefa de obter uma solução em CFD torne-se mais clara.

Certamente, este não é o primeiro programa gerador de malhas, mas traz vantagens incomuns à maioria dos programas (gratuitos) existentes: possibilidade de gerar malhas sobre regiões multiplamente conexas e não-convexas; facilidade na aproximação de pontos através do método TTM não-homogêneo; identificação automática de células de corte e geração de células fictícias. Aliam-se a estas vantagens a facilidade na definição das condições de contorno e no tratamento de pontos especiais da malha.

Além disso, o programa permite a geração de malhas multiblocos. Na versão atual, esta tarefa requer um pouco de astúcia, pois necessita a especificação de uma condição de contorno especial para os pontos onde as malhas se unem. Este problema, assim como eventuais falhas, podem ser facilmente corrigidos, graças à estrutura modular do programa. Tal estrutura possibilitará futuras extensões do programa; entre estas, pretendemos utilizar idéias avançadas em PSE para a escolha automática da topologia da malha a ser utilizada e geração de malhas adaptativas. Acreditamos que a mesma metodologia de estudo pode ser aplicada diretamente para o caso tridimensional.

Referências Bibliográficas

[1] Akers, R. L., Baffes, P., Kant, E., Randall, C. J., Steinberg, S., and Young, R. L. Automatic synthesis of numerical codes for solving partial differential equations, in the special issue non-standard applications of computer algebra. *ed. Roanes-Lozano, S.Steinberg and H.Hong. Mathematics and Computers in Simulation 45* (1998), 3–22.

[2] Akin, J. E. *Application and implementation of finite element methods.* Academic Press, London, 1982.

[3] Amsden, A. A., and Hirt, C. W. A simple scheme for generating general curvilinear grids. *J. Comp. Phys. 11* (1973), 348–359.

[4] Birkhoff, G., and Lynch, R. E. *Numerical Solution of Elliptic Problems.* SIAM, Philadelphia, 1984.

[5] Bortoli, Á. L. D. *Introdução à Dinâmica de Fluidos Computacional.* Editora da UFRGS, Porto Alegre, 2000.

[6] Brandt, A., and Yavneh, I. *Accelerated Multigrid Convergence and High-Reynolds Recirculating Flows.* Rehovot, Israel, March 1991.

[7] Brandt, A., and Yavneh, I. Inadequacy of first-order upwind difference schemes for some recirculating flows. *J. Comp. Phys. 93*, 1 (March 1991), 128–143.

[8] Brauer, J. R. *What every engineer should know about Finite Element Analisys.* What every engineer should know about series. Marcel Dekker, New York, 1988.

[9] Briggs, W. L. *A multigrid tutorial.* SIAM, Pennsylvanya, 1987.

[10] Castillo, J. E. An adaptive direct variational grid generation method. *Computers Math. Applic. 4*, 1 (1990), 1–9.

[11] Castillo, J. E., Hyman, J. M., Shashkov, M., and Steinberg, S. The sensitivity and accuracy of fourth order finite-difference schemes on nonuniform grids in one dimension. *J. Computers Mathematics with Applications 30*, 8 (September 1995), 41–55.

[12] Castillo, J. E., Hyman, J. M., Shashkov, M., and Steinberg, S. High-order mimetic finite-difference schemes on non-uniform grids. *Houston Journal of Mathematics ICOSAHOM 95* (1996).

[13] Castillo, J. E., Hyman, J. M., Shashkov, M., and Steinberg, S. Fourth and sixth-order conservative finite difference approximations of the divergence and gradient. Tech. rep., Los Alamos National Laboratory, Los Alamos, 2000.

[14] Chorin, A. J. Numerical solution of the navier-stokes equations. *Math. Comp. 22* (1968), 745–762.

[15] Churchill, R. V. *Complex variables and applications*, second ed. Mcgraw-Hill, New York, 1960.

[16] Decyk, V. K., Norton, C. D., and Szymansky, B. K. Introduction to object-oriented concepts using fortran90. Report PPG-1560, University of California, Los Angeles, July 1996. http://www.cs.rpi.edu/ ∼ szymansk/oof90.html.

[17] Decyk, V. K., Norton, C. D., and Szymansky, B. K. How to express c++ concepts in fortran 90. Report, University of California, Los Angeles, 1998. to be published in Scientific Programming.

[18] Eiseman, P. R. *Ortogonal grid generation.* Numerical Grid Generation. North Holland, ed. J. F. Thompson, New York, 1982, pp. 193–226.

[19] Ferziger, J. H., and Peric, M. *Computational Methods for Fluid Dynamics*, 2. ed. Springer-Verlag, Heidelberg, 1999.

[20] Fletcher, C. A. J. *Computational Techniques for Fluid Dynamics*, vol. 1,2 of *Springer Series in Computational Physics.* Springer Verlag, New York, 1988.

[21] Forsythe, G. E., and Wasow, W. R. *Finite Difference Methods for Partial Differential Equations.* Wiley, New York, 1960.

[22] Fortuna, A. O. *Técnicas Computacionais para Dinâmica dos Fluidos: Conceitos Básicos e Aplicações.* Edusp, São Paulo, 2000.

[23] Golub, G. H., and Ortega, J. M. *Scientific Computing and Differential Equations.* Academic Press, Boston, 1991.

[24] Gordon, W. J., and Thiel, L. C. *Transfinite mappings and their applications to grid generarion.* Numerical Grid Generation. North Holland, ed. J. F. Thompson, New York, 1982, pp. 171–192.

[25] Griebel, M., Dornseifer, T., and Neunhoeffer, T. *Numerical simulation in fluid dynamics: a practical introduction.* SIAM Monographs on Mathematical Modeling and Computation. SIAM, Philadelphia, 1997. Codes in ftp.lrz-muenchen.de/pub/science/fluiddynamics/cfd/NaSt2D.

[26] Harlow, F. H., and Welch, J. E. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The Physics of Fluids 8*, 12 (1965), 2182–2189.

[27] HUEBNER, K. H. *The Finite Element Method for Engineers.* John Wiley & Sons, New York, 1975.

[28] JUSTO, D. A. R. Introdução ao método multigrid. Relatório interno, UFRGS, Porto Alegre, RS, dezembro 1999.

[29] JUSTO, D. A. R. Visual: Uma família de softwares para a visualização de fenômenos em mecânica dos fluidos. Manual do visual, UFRGS, Porto Alegre, RS, março 1999. Orientador: Prof. Rudnei Dias da Cunha, D.Phil.

[30] KLEINSTREUER, C. *An interdisciplinary Systems Approach.* Cambridge University Press, Nova York, 1997.

[31] KNUPP, P. M. The direct variational grid generation method extended to curves. *Appl. Math. Comp. 43* (1991), 65–78.

[32] KNUPP, P. M., AND STEINBERG, S. *The Fundamentals of Grid Generation.* CRC Press, Boca Raton, 1993.

[33] LIAO, G. *On harmonic maps.* Mathematical Aspects of Numerical Grid Generation. ed. J. E. Castillo, SIAM, Philadelphia, 1991.

[34] MALISKA, C. R. *Transferência de calor e mecânica dos fluidos computacional.* Livros Técnicos Científicos, Rio de Janeiro, 1995.

[35] PATANKAR, S. V. *Numerical Heat Transfer and Fluid Flow.* Series in Computational Methods in Mechanics and Thermal Sciences. Hemisphere Publishing Corporation, New York, 1980.

[36] PEYRET, R., AND TAYLOR, T. D. *COmputational methods for fluid flow.* Springer Series in Computational Physics. Springer Verlag, New York, 1983.

[37] ROACHE, P. J. Marching methods for elliptic problems, part 1. *Num. Heat Transfer 1* (1978), 1–25.

[38] SMITH, G. D. *Numerical solution of partial differential equations: finite difference methods*, third ed. Oxford applied mathematics and computing science series. Oxford University Press, New York, 1990.

[39] SORENSON, R. L., AND STEGER, J. L. *Numerical generation of two dimensional grids by the use of Poisson equations with grid control.* No. NASA CP 2166 in Numerical Grid Generation Techniques. ed. R. E. Smith, NASA Langley Research Center, Hampton, 1980, pp. 449–461.

[40] Steger, J. L., and Chausee, D. S. Generation of body-fitted coordinates using hyperbolic partial differential equations. *SIAM J. Sci. Stat. Comput. 1*, 4 (1980), 431–437.

[41] Steinberg, S., and Roache, P. J. Variational curve and surface grid generation. *J. Comp. Phys. 100* (1992), 163–178.

[42] Steinberg, S., Roache, P. J., and Salari, K. Hybrid adaptive poisson grid generation and grid smoothness. *Communications in Applied Numerical Methods 7* (1991), 345–354.

[43] Steinberg, S., and Shashkov, M. Supportoperators finitedifference algorithms for general elliptic problems. *J. Comp. Phys. 118* (1995), 131151.

[44] Thompson, J. F., Thames, F. C., and Mastin, C. W. Automatic numerical generation of body-fitted curvilinear coordinates system for field containing any number of arbitrary two-dimensional bodies. *J. Comp. Phys. 15* (1974), 299–319.

[45] Thompson, J. F., Thames, F. C., and Mastin, C. W. Tomcat - a code for numerical generation of boundary-fitted curvilinear coordinates systems on fields containing any number of arbitrary two-dimensional bodies. *J. Comp. Phys. 24* (1977), 274–302.

[46] Thompson, J. F., Warsi, Z. U. A., and Mastin, C. W. *Numerical Grid Generation.* Elsevier Science Publishing, New York, 1985.

[47] Versteeg, H. K., and Malalasekera, W. *An introduction to Computatinal Fluid Dynamics: the finite volume method.* Longman, England, 1995.

[48] Warsi, Z. U. A. *Basic differential models for coordinate generation.* Numerical Grid Generation. North-Holland, ed. J. F. Thompson,, New York, 1982, pp. 41–78.

[49] Wendt, J. F. *Computational Fluid Dynamics: an Introduction*, second ed. Springer Verlag, Germany, 1996.

[50] Zang, Y., Street, R. L., and Koseff, J. R. A non-staggered grid, fractional step method for time-dependent incompressible navier-stokes equations in curvilinear coordinates. *J. Comp. Physics 114* (1994), 18–33.

Apêndice A ARQUIVO DA MALHA

O arquivo de dados da malha é o mesmo do programa Visual [29]. Um malha sobre um cilindro com 20×30 pontos poderá ser dada no arquivo de dados por

```
Malha sobre um cilindro
30
20
x(1,1)   y(1,1)
x(1,2)   y(1,2)
 .         .
 .         .
x(1,20)  y(1,20)
x(2,1)   y(2,1)
 .         .
 .         .
x(30,20) y(30,20)
```

Note que a primeira linha conterá um título, 30 é o número de linhas e 20 é o número de colunas da malha. Os valores posteriores são os vértices dos pontos da malha que totalizam $20 \times 30 = 600$ pontos. É listada toda a linha 1, seguida pela linha 2, e assim por diante até a linha 30.

Esta malha pode ser visualizada através do programa VISUAL e pode ser usada no programa do usuário através do pequeno algoritmo em Fortran abaixo:

```
character*50            ::  Titulo
real,dimension(30,20)   ::  X,Y
integer                 ::  m,n

open(unit=1,file='malha.dat',status='old')
read(1,*) Titulo
read(1,*) m
read(1,*) n

do i=1,m
    do j=1,n
        read(1,*) x(i,j), y(i,j)
    end do
end do
```

Apêndice B ARQUIVO DAS PRIMITIVAS

O usuário pode salvar os dados da região que foi definida na tela para posterior utilização. O mesmo formato de arquivo pode também ser usado para a entrada de dados.

O arquivo de dados deve conter as primitivas a serem utilizadas e poderá também informar os contornos da região. Para o cilindro usado como exemplo no capítulo 8, o arquivo de dados foi o seguinte:

```
Makegrid v1.0
Primitives
          5 = Total of primitives
Arc
          1
          2
          3
          1 = discretization
Arc
          3
          4
          1
          1 = discretization
Arc
          5
          6
          7
          1 = discretization
Arc
          7
          8
          5
          1 = discretization
Line
          1
          5
          1 = discretization
          8 = Total of Nodes
     1       0
     0       1
    -1       0
     0      -1
   0.3       0
     0     0.3
  -0.3       0
     0    -0.3
Contour
North
          2 = Primitives in Contour
          1
          2
South
          2 = Primitives in Contour
          3
          4
East
```

```
          1 = Primitives in Contour
          5
West
          1 = Primitives in Contour
          5
```

Este arquivo de dados refere-se à seguinte região física dada na figura B.1:

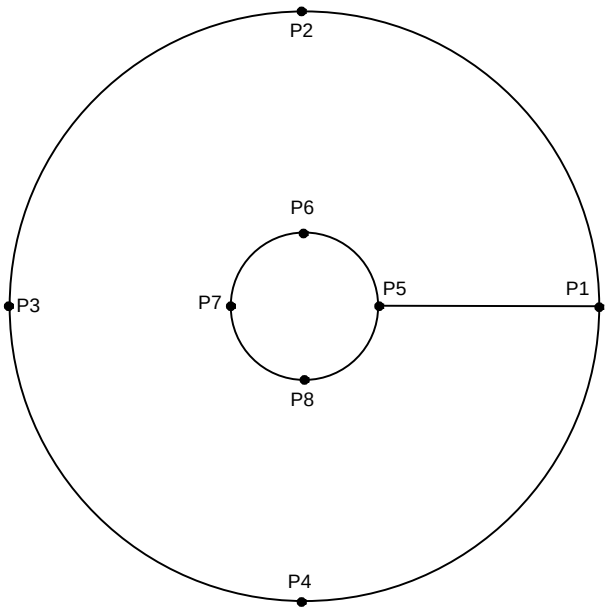


Figura B.1: *Pontos utilizados na definição da malha sobre o cilindro.*

Na primeira linha está o cabeçalho do arquivo seguido pelo número de primitivas, neste caso igual a 5. Após, aparecem listadas cada uma das primitivas, seguidas pelos seus parâmetros. Um `Arc` é seguido pelos três pontos que o compõem, enquanto que uma `Line` é seguida de seus pontos inicial e final. Indicamos apenas o número do ponto que aparecerá numa lista de nós logo a seguir. Na última linha da definição de cada primitiva consta o tipo de discretização, ou seja, um número (seguido do comentário `=discretization`) indica como será a concentração dos pontos de acordo com a tabela B.1.

Tipo de concentração	Código
uniforme	1
no início	2
no final	3
no meio	4
nos extremos	5

Tabela B.1: *Códigos para concentração de pontos nas primitivas.*

As primitivas spline e polinômio de Lagrange são definidas da mesma forma usando os rótulos `Spline` e `Lagrange` respectivamente, seguidas pelo número de nós da primitiva e pela

indicação de quais nós pertencem a esta primitiva. Por exemplo, o círculo externo do cilindro acima poderia ser descrito através de uma única Spline através das seguintes linhas:

```
Spline
          5  nodes
          1
          2
          3
          4
          1
          1 = discretization
```

Logo após as primitivas, aparece o número total de nós utilizados para gerar as primitivas seguido por `TN = Total of nodes`. Abaixo estão as TN coordenadas x e y dos nós mencionados. Cada primitiva irá se referenciar aos nós de acordo com a ordem como estes nós aparecem nesta lista.

Este é o arquivo básico para as primitivas. A relação das primitivas que pertencem a cada contorno também pode ser acrescentada na segunda parte do arquivo. A palavra chave `Contourn` começa esta seção, seguida pelos contornos denominados `North`, `South`, `East` e `West`. Após cada contorno estão o número de primitivas pertencentes a este contorno, seguido por um número que identifica cada primitiva. Este número é dado a cada primitiva de acordo com a ordem que elas são listadas no início do arquivo.

Uma possível maneira de representar esta estrutura de dados para o caso do cilindro é dada na figura B.2.

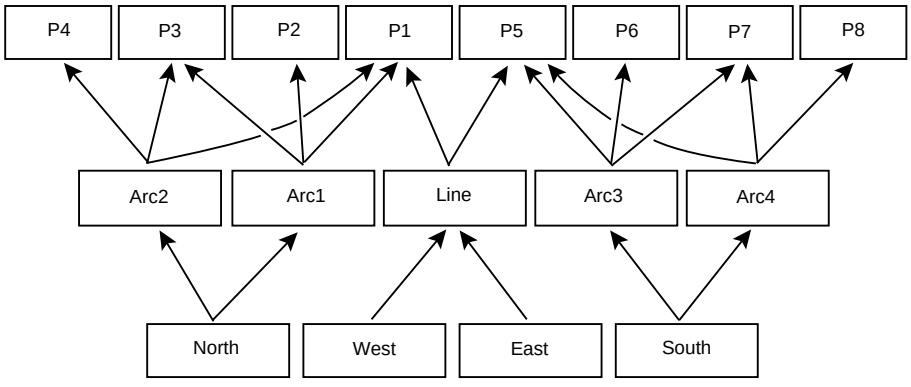


Figura B.2: *Representação da estrutura de dados utilizada para armazenar as primitivas do cilindro.*

Apêndice C ARQUIVO DA MALHA DE ESTADOS

O modo como a malha de estados é definida foi apresentada na seção 8.2.3. O arquivo da malha de estados tem a seguinte estrutura: na primeira linha temos o título `State Grid`, seguido pela quantidade de variáveis primitivas, 3. Estas são dadas a seguir através dos rótulos dados no programa, que neste exemplo são `u`, `v` e `phi`. Logo após são definidos os rótulos para as condições de contorno, neste caso também 3, seguido das condições de contorno para cada uma das variáveis primitivas. O primeiro número é o tipo de condição de contorno dado pela tabela C e o segundo número é o valor da condição de contorno, que é dado na tabela pela variável a. Abaixo, temos o arquivo para o exemplo do capítulo 8:

```
State Grid
3  =  Variables
u
v
phi
3  =  Boundary Conditions
no-slip
1   0   =  u
1   0   =  v
4   0   =  phi
inflow
1   1   =  u
1   0   =  v
4   0   =  phi
outflow
4   0   =  u
4   0   =  v
4   0   =  phi
10
 5
0  0  0  0  0  0  0  0  0  0  0  0  0
0  1  1  1  1  1  1  1  1  1  1  1  0
0 64 32 32 32 32 32 32 32 32 32 64  0
0 64 32 32 32 32 32 32 32 32 32 64  0
0 64 32 32 32 32 32 32 32 32 32 64  0
0  3  3  3  2  2  2  2  2  3  3  3  0
0  0  0  0  0  0  0  0  0  0  0  0  0
```

Condição de Contorno	Código
nenhuma	0
$\phi = a$	1
$\frac{\partial \phi}{\partial x} = a$	2
$\frac{\partial \phi}{\partial y} = a$	3
$\frac{\partial \phi}{\partial n} = a$	4
$\frac{\partial \phi}{\partial \tau} = a$	5
Mista	6

Tabela C.1: *Códigos para definição das condições de contorno*

Além do arquivo com o formato descrito acima, há a opção de um arquivo contendo somente a malha de estados, listada com uma única coluna. Um modo para a utilização deste arquivo é o seguinte:

```
open(1,file='state.dat',status='unknown') ! arquivo contendo a malha de estados

NOSLIP=1
INFLOW=2
OUTFLOW=3
INTERNAL=32
CUT_BRANCH=64

do i=0,M+1
    do j=0,N+1
       read(1,*)state(i,j)               ! le a malha de estados
    end do
end do

do i=1,M
    do j=1,N
      Select Case(state(i,j))
        Case(NOSLIP)
            call noslip_bc(i,j)         ! rotina que aplica a condicao no-slip
        Case(INFLOW)
            call inflow_bc(i,j)         ! rotina que aplica a condicao inflow
        Case(OUTFLOW)
            call outflow_bc(i,j)        ! rotina que aplica a condicao outflow
        Case(INTERNAL.OR.CUT_BRANCH)
            call solve_equations(i,j)   ! rotina que soluciona as equacoes no ponto i,j
    end do
end do
```