Universidade Federal de Uberlândia
Faculdade de Engenharia Mecânica

Graduação em Engenharia Mecatrônica

ESTUDO E IMPLEMENTAÇÃO DE MÉTODOS NUMÉRICOS PARA RESOLUÇÃO DE PROBLEMAS ESTACIONÁRIOS

Daniella Faria Freitas

Uberlândia-MG

2021

Daniella Faria Freitas

ESTUDO E IMPLEMENTAÇÃO DE MÉTODOS NUMÉRICOS PARA RESOLUÇÃO DE PROBLEMAS ESTACIONÁRIOS

Trabalho de conclusão de curso apresentado à Coordenação do Curso de Engenharia Mecatrônica como requisito parcial para obtenção do grau de Engenheiro Mecatrônico.

Orientador: Rafael Alves Figueiredo

Uberlândia-MG

2021

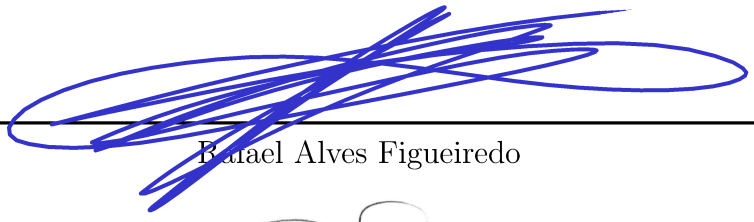
Universidade Federal de Uberlândia
Faculdade de Engenharia Mecânica

Coordenação do Curso de Engenharia Mecatrônica

A banca examinadora, conforme abaixo assinado, certifica a adequação deste trabalho de conclusão de curso para obtenção do grau de Engenheiro Mecatrônico.

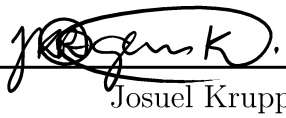
Uberlândia, 22 de outubro de 2021.

BANCA EXAMINADORA



Rafael Alves Figueiredo

Priscila Ferreira Barbosa de Sousa



Josuel Kruppa Rogenski

Uberlândia-MG

2021

Agradecimentos

Agradeço à minha família, minha mãe Eurípedes e meu pai Derly, aos meus amigos mais próximos Renata, Mila e Gustavo, e ao meu namorado Alex, por todo apoio e companheirismo que ofereceram para a minha jornada.

À todos os professores que tive a chance de conhecer ao longo do curso e que contribuíram para minha formação profissional e pessoal.

Agradeço também ao meu orientador Prof. Rafael, por todo apoio, paciência e orientação, que foram de enorme importância para mim.

Resumo

Este trabalho tem como finalidade o estudo e a resolução numérica de problemas estacionários descritos por equações diferenciais parciais (EDPs), que podem ser classificadas como elípticas, parabólicas e hiperbólicas. Tais equações modelam diversos fenômenos físicos de interesse da engenharia sendo que, em especial, problemas físicos em estado estacionário são geralmente descritos por EDPs elípticas.

Assim, o principal objetivo deste trabalho foi resolver EDPs elípticas utilizando métodos de diferenças finitas (MDF) de segunda ordem para discretizar as equações.

Considerando um conjunto de hipóteses, as equações de Navier-Stokes foram reduzidas a um problema estacionário modelado por uma EDP elíptica. Esta equação foi discretizada via MDF de segunda ordem em um domínio computacional, resultando em um sistema linear esparso.

Por fim, o conteúdo abordado foi ilustrado através de aplicações numéricas. A ferramenta utilizada para resolver o sistema linear resultante foi o método dos gradientes conjugados (MGC), para estimar o perfil de velocidade e de tensão de cisalhamento de escoamentos totalmente desenvolvidos em tubos com diferentes seções transversais. A implementação de tal método foi realizada em linguagem de programação C.

Palavras-chave: Equações diferenciais parciais, problemas estacionários, métodos de diferenças finitas, métodos numéricos iterativos.

Abstract

This work studies the numerical solution of steady-state problems described by partial differential equations (PDEs), which can be classified as elliptic, parabolic or hyperbolic. These equations can model several physical phenomena of interest in Engineering, where steady-state problems often are modelled from elliptic PDEs.

Thus, the main objective of this study was to solve elliptic PDE using a second order finite difference method (FDM) to discretize this equation.

Considering a serie of assumptions, Navier-Stokes's equations have been reduced to a steady-state problem which is modelled by the elliptic PDE. This equation was discretized via second order FDM on a computational domain, resulting in a sparse linear system.

At last, the content discussed is then illustrated by numerical applications. The tool used to solve the linear system obtained was the Conjugate Gradient Method (CGM), to estimate the velocity and shear stress profiles in fully developed flows on pipes of different cross-sections. Implementation was performed using C language.

Keywords: Partial differential equations, steady-state problems, finite difference methods, iterative numerical methods.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações e Símbolos

Lista de Abreviações

EDP Equação Diferencial Parcial

ELT Erro Local de Truncamento

MDF Métodos de Diferenças Finitas

MG Método dos Gradientes

MGC Método do Gradiente Conjugado

MGJ Método de Gauss Jacobi

MGS Método de Gauss Seidel

MSOR Método de Sobre-Relaxação Sucessiva

PVC Problema de Valor de Contorno

PVI Problema de Valor Inicial

Lista de Símbolos

Δx espaçamento na direção x

Δy espaçamento na direção y

$\delta\Omega$ fronteira da região Ω

λ autovalor da matriz de coeficientes $\mathbf{A}$

μ viscosidade dinâmica do fluido

∇^2 operador de Laplace

Ω área da seção transversal da geometria

ρ densidade do fluido

τ erro local de truncamento

τ_ω tensão cisalhante

$\mathbf{A}$ matriz de coeficientes da equação discretizada

$\mathbf{b}$ vetor de constantes

$\mathbf{D}$ matriz diagonal

$\mathbf{L}$ matriz triangular inferior

$\mathbf{n}$ vetor normal à fronteira

$\mathbf{r}$ vetor de resíduos

$\mathbf{U}$ matriz triangular superior

Dh diâmetro hidráulico

$Erro_2$ erro na norma 2

$Erro_\infty$ erro na norma infinita

f fator de atrito

g componente gravitacional

$k(\mathbf{A})$ número de condição

Lx comprimento da área da seção transversal da geometria

Ly altura da área da seção transversal da geometria

Nx quantidade de pontos na direção x

Ny quantidade de pontos na direção y

$O(\Delta x)$ ordem de uma aproximação

O_2 ordem de convergência na norma 2

O_∞ ordem de convergência na norma infinita

Pm perímetro molhado

Q' vazão do escoamento

q_{conv} convecção térmica

q_{flux} fluxo térmico

Re número de Reynolds

T temperatura

U velocidade média

u componente de velocidade na direção x

u_{max} velocidade máxima

v componente de velocidade na direção y

w componente de velocidade na direção z

E erro global

u_{aprox} velocidade aproximada

u_{exata} velocidade analítica

1. INTRODUÇÃO

As equações diferenciais parciais (EDPs) descrevem vários fenômenos físicos, de interesse industrial e não industrial, que podem ser resolvidas através da aplicação de algoritmos de resolução numérica. Para obter a solução numérica de uma EDP através dos métodos de diferenças finitas (MDF), é necessário discretizar as equações em um domínio computacional, para obter pontos onde os cálculos serão aplicados.

As EDPs possuem três classes com características diferentes: elípticas, parabólicas ou hiperbólicas [9]. As EDPs elípticas geralmente descrevem problemas em estado de equilíbrio, em que as propriedades físicas não se alteram ao longo do tempo. Um exemplo de equação elíptica é a equação de Laplace, a seguir

$$\nabla^2\phi = \frac{\partial\phi^2}{\partial x^2} + \frac{\partial\phi^2}{\partial y^2} + \frac{\partial\phi^2}{\partial z^2} = 0, \tag{1.1}$$

onde ∇^2 é o operador de Laplace. Outro exemplo de equação elíptica é a equação de Poisson.

Considerando uma função contínua em um intervalo $[a, b]$, com derivadas contínuas até a ordem N, e um ponto x pertencente ao intervalo, têm-se o Teorema de Taylor para duas variáveis, da seguinte forma

$$\begin{aligned} f(x_i \pm \Delta x, y_j \pm \Delta y) = f(x_i, y_j) \pm (\Delta x)\left.\frac{\partial f}{\partial x}\right|_{i,j} \pm (\Delta y)\left.\frac{\partial f}{\partial y}\right|_{i,j} + \\ + \frac{(\Delta x)^2}{2!}\left.\frac{\partial f^2}{\partial x^2}\right|_{i,j} + 2\frac{(\Delta x)(\Delta y)}{2!}\left.\frac{\partial f^2}{\partial x \partial y}\right|_{i,j} + \frac{(\Delta y)^2}{2!}\left.\frac{\partial f^2}{\partial y^2}\right|_{i,j} + R_N, \end{aligned} \tag{1.2}$$

onde R_N é o resto.

Partindo da expansão em série de Taylor e realizando uma série de manipulações matemáticas convenientes, obtêm-se a forma da aproximação de segunda ordem. Uma forma de aproximação de segunda ordem é a fórmula dos 5 pontos da equação de Laplace $\nabla^2 f = 0$, por diferenças centrais de segunda ordem, a seguir

$$\left.\frac{\partial f^2}{\partial x^2}\right|_{i,j} + \left.\frac{\partial f^2}{\partial y^2}\right|_{i,j} = \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{(\Delta x)^2} + \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{(\Delta y)^2} + O[(\Delta x^2), (\Delta y^2)]. \tag{1.3}$$

Considera-se a equação de Laplace discretizada, bidimensional, através da fórmula de 5

pontos em uma malha regular, com $\beta = \dfrac{\Delta x}{\Delta y}$, e simplificada, como

$$f_{i+1,j} + f_{i-1,j} - 2(1+\beta^2)f_{i,j} + \beta^2(f_{i,j+1} + f_{i,j-1}) = 0. \tag{1.4}$$

Essa aproximação é comumente utilizada na resolução da equação de Poisson.

Essa equação 1.4 é válida para todos os pontos da malha computacional, resultando em um sistema linear cujo número de equações depende do número de pontos discretizados. Na forma matricial, o sistema linear resultante é descrito por

$$\mathbf{Ax} = \mathbf{b}, \tag{1.5}$$

onde $\mathbf{A}$ é a matriz dos coeficientes, $\mathbf{x}$ é o vetor da solução aproximada e $\mathbf{b}$ é o vetor de constantes.

Esse sistema linear resultante pode ser resolvido via métodos numéricos. Para as equações elípticas, a matriz $\mathbf{A}$ resultante será uma matriz esparsa.

Os métodos numéricos diretos, quando aplicados a sistemas grandes ou esparsos, podem perder consideravelmente a eficiência devido à quantidade de operações necessárias para chegar a uma solução e à propagação de erros de arredondamento, e não atingirão a solução exata do problema [7].

Já nos métodos iterativos, a solução é encontrada com a sucessão de passos de cálculo que convergem para a real solução do sistema linear. Fornecem bons resultados aproximados e são indicados para problemas grandes ou com matrizes mal-condicionadas como as matrizes esparsas. Assim, para este trabalho, o Método dos Gradientes Conjugados (MGC) será utilizado como uma ferramenta de resolução para o sistema linear obtido pela discretização via MDF.

As equações de Navier Stokes são equações diferenciais que descrevem o fenômeno do escoamento de fluidos, como a equação da quantidade de movimento, dada por

$$\rho D\frac{\vec{v}}{Dt} = \mu\nabla^2\vec{v} - \nabla^2\vec{p} + \rho\vec{g}, \tag{1.6}$$

onde ρ é a densidade do fluido, $\vec{v}$ é a velocidade, $\vec{g}$ a componente gravitacional e $D\dfrac{\vec{v}}{Dt}$ é a derivada material de $\vec{v}$.

Sob a hipótese de um escoamento laminar totalmente desenvolvido para um duto, essas equações podem ser simplificadas à equação de Poisson, uma equação elíptica, e discretizadas via MDF, como será visto no decorrer deste estudo.

2. Fundamentação Teórica

As equações diferenciais parciais modelam diversos fenômenos físicos de interesse, assim, torna-se indispensável entender como as EDPs funcionam.

Para isso, inicialmente é necessário o estudo acerca de suas classificações e características matemáticas. Estas informações serão responsáveis por indicar a categoria à qual o fenômeno físico pertence e, desta forma, sabendo-se as características do fenômeno físico, é possível descrevê-lo utilizando uma EDP hiperbólica, elíplica ou parabólica, e, por fim, traçar a melhor estratégia de resolução [9].

2.1 Classificação das EDPs

Duas características matemáticas importantes das EDPs são a linearidade e a ordem. A ordem refere-se à ordem da maior derivada parcial da equação, enquanto a linearidade é definida através dos termos que acompanham as derivadas parciais.

Para entender isso, considera-se a equação diferencial parcial genérica de 2ª ordem, dada por

$$A\frac{\partial\phi^2}{\partial x^2} + B\frac{\partial\phi^2}{\partial x\partial y} + C\frac{\partial\phi^2}{\partial y^2} + D\frac{\partial\phi}{\partial x} + E\frac{\partial\phi}{\partial y} + F\phi = G, \tag{2.1}$$

onde A, B, C, D, F, E e G são coeficientes constantes ou não.

Se esses coeficientes forem constantes ou funções unicamente de x e y, a equação é uma equação diferencial parcial linear. Em contrapartida, se esses termos forem variáveis, a equação é dita não linear [9].

Dentro do conjunto das EDPs não lineares, têm-se, também, equações ditas quase lineares ou semi lineares. As derivadas de maior ordem das EDPs compõem a parte principal da EDP. Se essa parte principal, em uma equação não linear, for linear, então a EDP é uma equação semi linear.

Outro ponto importante para entender o comportamento de uma EDP é sua classificação quanto a três famílias distintas: elípticas, parabólicas ou hiperbólicas. Um conceito importante para essa classificação é o conceito de características. Este conceito é necessário pois muitas EDPs podem apresentar comportamento misto, mostrando tanto comportamento elíptico em um termo, quanto comportamento parabólico em outro, por exemplo.

O processo para obter essa classificação pode ser melhor acompanhado em Fortuna [9], e consiste em encontrar uma curva característica, dentre infinitas, para a equação em questão.

Para isso, considera-se a parte principal da equação genérica (2.1), como a seguir

$$A\frac{\partial\phi^2}{\partial x^2} + B\frac{\partial\phi^2}{\partial x\partial y} + C\frac{\partial\phi^2}{\partial y^2} - G = 0. \tag{2.2}$$

Considera-se pequenas variações com relação a x e y, da forma

$$dp = \frac{\partial p}{\partial x}dx + \frac{\partial p}{\partial y}dy = \frac{\partial\phi^2}{\partial x^2}dx + \frac{\partial\phi^2}{\partial x\partial y}dy, \tag{2.3}$$

onde p $= \dfrac{\partial\phi}{\partial x}$, e

$$dq = \frac{\partial q}{\partial x}dx + \frac{\partial q}{\partial y}dy = \frac{\partial\phi^2}{\partial y^2}dy + \frac{\partial\phi^2}{\partial x\partial y}dx, \tag{2.4}$$

onde q $= \dfrac{\partial\phi}{\partial y}$. E isola-se $\dfrac{\partial\phi^2}{\partial x}$ em (2.3) e $\dfrac{\partial\phi^2}{\partial y}$ em (2.4), obtendo, respectivamente

$$\frac{\partial\phi^2}{\partial x^2} = \frac{\partial p}{\partial x} + \frac{\partial p}{\partial y}\frac{dy}{dx} - \frac{\partial\phi^2}{\partial x\partial y}\frac{dy}{dx}, \tag{2.5}$$

$$\frac{\partial\phi^2}{\partial y^2} = \frac{\partial q}{\partial x}\frac{dx}{dy} + \frac{\partial q}{\partial y} - \frac{\partial\phi^2}{\partial x\partial y}\frac{dx}{dy}. \tag{2.6}$$

Com isso, substitui-se em (2.2) e multiplica-se toda a equação por $\dfrac{dy}{dx}$, obtendo

$$\left[A\left(\frac{dy}{dx}\right)^2 - B\left(\frac{dy}{dx}\right) + C\right]\frac{\partial\phi^2}{\partial x\partial y} - \left[A\frac{dpdy}{dxdx} + C\frac{dq}{dx} - G\frac{dy}{dx}\right] = 0, \tag{2.7}$$

onde $\dfrac{\partial\phi^2}{\partial x\partial y}$ é determinado ao longo da curva característica. Assim, têm-se

$$A\left(\frac{dy}{dx}\right)^2 - B\left(\frac{dy}{dx}\right) + C = 0, \tag{2.8}$$

uma equação simples de 2º grau de resolução já conhecida, que tem por solução

$$\frac{dy}{dx} = \frac{B \pm \sqrt{B^2 - 4AC}}{2A}, \tag{2.9}$$

Assim, considerando os coeficientes da parte principal da equação (2.1), A, B e C, pode-se classificar as EDPs em três famílias: elípticas, parabólicas e hiperbólicas, da seguinte forma:

- elíptica, se $B^2 - 4AC < 0$,
- parabólica, se $B^2 - 4AC = 0$, e
- hiperbólica, se $B^2 - 4AC > 0$.

Cada uma dessas classificações transmite informação sobre as possíveis soluções da equação.

A equação classificada como elíptica não possui características no plano real, enquanto a equação parabólica apresenta apenas uma característica no plano real, e as equações hiperbólicas possuem duas características no plano real.

2.1.1 EQUAÇÕES ELÍPTICAS

Equações elípticas são dadas pelo modelo da equação de Laplace, bidimensional, e possuem a forma

$$\nabla^2\phi = \frac{\partial\phi^2}{\partial x^2} + \frac{\partial\phi^2}{\partial y^2} = 0, \tag{2.10}$$

onde ∇^2 é o Laplaciano ou operador de Laplace e x e y são coordenadas espaciais. Um exemplo de equação elíptica é a equação de Poisson.

Neste modelo, é possível notar que a propriedade de interesse independe do tempo. Isto é, a variável de estudo não sofre alteração com a variação do tempo. Essa é, necessariamente, uma característica de problemas estacionários, também conhecidos por problemas de equilíbrio. Estes problemas de equilíbrio, regidos então por equações elípticas, serão objeto de estudo principal ao longo deste trabalho.

Neste caso, para tratar um problema regido por uma equação elíptica, é necessário impor condições sobre a fronteira da região de estudo. Problemas assim são chamados Problemas de Valor de Contorno (PVC)[2]. Nestes problemas, as perturbações em determinados pontos da geometria são propagados de forma radial, resultando em uma influência nos demais pontos da região, sendo essa influência maior nos pontos próximos à origem da perturbação e menor à medida que distancia-se dessa origem.

2.1.2 EQUAÇÕES PARABÓLICAS

Equações parabólicas, por sua vez, possuem um termo dissipativo no modelo, da forma

$$\frac{\partial\phi}{\partial t} = \alpha\frac{\partial\phi^2}{\partial x^2}, \tag{2.11}$$

onde α é a difusividade térmica do material, sendo um exemplo para este caso a equação transiente de difusão de calor.

Nota-se, neste modelo, o fenômeno da dissipação de energia, podendo ocorrer através da dissipação de calor ou dissipação viscosa, por exemplo.

Outro exemplo de equação parabólica é a equação de Burgers viscosa, sendo sua versão unidimensional dada por

$$\frac{\partial v}{\partial t} = -v\frac{\partial v}{\partial x} + \Gamma\frac{\partial v^2}{\partial x^2}, \tag{2.12}$$

onde v é a velocidade de transporte e Γ é o coeficiente de difusão. No lado direito da equação (2.12), nota-se a influência do termo difusivo, o segundo termo, em conjunto com o termo convectivo, primeiro termo, caracterizando o processo como convecção-difusão.

Nestes problemas, é necessária a especificação de uma condição inicial, sendo chamado, portanto, de Problema de Valor Inicial (PVI). Este valor inicial é dado à região de análise no instante de tempo inicial do processo transiente e, nesse caso, as perturbações na região se propagam para os instantes de tempo subsequentes.

2.1.3 EQUAÇÕES HIPERBÓLICAS

A equação hiperbólica, por fim, possui um termo convectivo ou inercial, e é dada pelo modelo

$$\frac{\partial \phi}{\partial t} = -v\frac{\partial \phi}{\partial x}, \tag{2.13}$$

onde v é a velocidade do transporte. Este termo convectivo foi observado anteriormente, compondo uma parcela da equação parabólica (2.12).

Neste modelo, a dissipação é mínima e pode ser desprezada. Geralmente, são relacionados a problemas de vibração ou convecção [9]. Um exemplo clássico para a EDP hiperbólica é a equação da onda de segunda ordem

$$\frac{\partial \phi^2}{\partial t^2} = c^2\frac{\partial \phi^2}{\partial x^2}, \tag{2.14}$$

onde c é a velocidade da luz no vácuo.

Outro ponto interessante sobre as equações hiperbólicas é a permissão de soluções descontínuas. Isso se deve à ausência do termo dissipativo na equação. Com este termo exercendo influência na equação, como visto anteriormente através da equação de Burgers viscosa, a equação adquire caráter parabólico.

2.2 CONDIÇÕES AUXILIARES

Como apresentado anteriormente, a resolução de problemas de equações diferenciais parciais envolve a imposição de condições ao problema, sejam estas condições de contorno, aplicadas às fronteiras da geometria, ou condições iniciais, aplicadas ao tempo inicial do processo transiente.

Ao conjunto de condições, condições de contorno e condições iniciais, dá-se o nome de condições auxiliares. Estas condições auxiliares possibilitam a resolução do problema e devem ser especificadas corretamente para que a solução exista, seja única e dependa continuamente das condições impostas.

A seguir é apresentado sobre a condição inicial e três condições de contorno importantes e comumente utilizadas.

2.2.1 CONDIÇÃO INICIAL

Os problemas de valor inicial necessitam que seja imposto o valor no instante inicial para todas as grandezas físicas, em toda a geometria de estudo, juntamente com as condições de fronteira.

Desta forma, considerando um problema comum de transferência de calor, ao se esquentar uma placa plana a partir de distribuições iniciais diferentes de temperatura, a distribuição de temperatura obtida ao longo da placa, para cada passo de tempo, será diferente entre os dois casos. Isto é, torna-se possível obter uma família de soluções para o problema, caso não se especifique a condição inicial, mas para obter um único resultado esperado ao longo do tempo, é necessária a indicação correta desta condição inicial.

2.2.2 CONDIÇÕES DE CONTORNO DIRICHLET, NEUMANN E ROBIN

A condição de contorno do tipo Dirichlet consiste em especificar um valor para ϕ na fronteira da região analisada [9]. Como exemplo de uma condição de fronteira do tipo Dirichlet, têm-se a temperatura imposta às paredes de uma placa plana, em um problema de transferência de calor, ou a velocidade imposta igual a zero nas paredes, impermeáveis com característica de não deslizamento, de um canal pelo qual passa um escoamento.

Já a condição de contorno do tipo Neumann possui a forma

$$\frac{\partial \phi}{\partial n} = f$$

ou

$$\frac{\partial \phi}{\partial s} = g,$$

ou seja, o gradiente normal ou tangencial é especificado para uma determinada fronteira [9]. Por exemplo, o gradiente de velocidade normal indicado a uma fronteira que representa a saída livre do escoamento em um canal.

Além disso, caso a função $f = 0$ têm-se um caso particular da condição de Neumann, chamado de condição natural ou homogênea.

Por fim, têm-se a condição de contorno do tipo Robin, a qual trata-se de uma combinação linear das condições anteriores [9], como

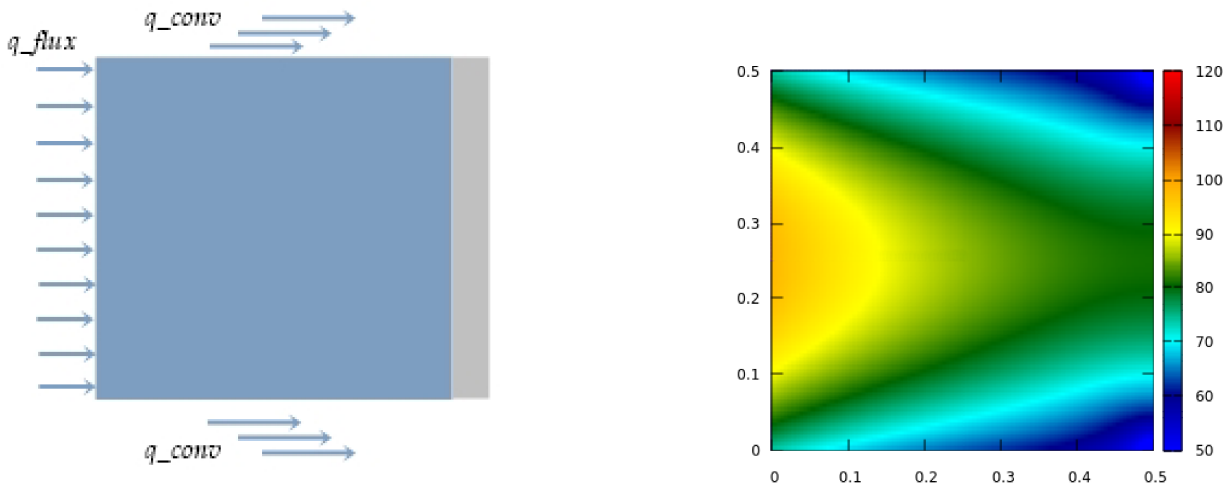
$$\alpha \frac{\partial \phi}{\partial n} + \beta \phi = f$$

Um exemplo de condição de contorno do tipo Robin é a radiação, expressa por

$$I = e\sigma T^4, \tag{2.15}$$

onde I é a intensidade da radiação térmica, e é a emissividade do corpo, grandeza adimensional com valores entre 0 e 1 e σ é a constante de Boltzmann, com valor fixo de $\sigma = 5,67 \times 10^{-}8 \frac{W}{m^2 K^4}$.

Cabe ressaltar, também, que as condições de contorno podem aparecer e geralmente aparecem de forma mista, isto é, em forma de combinação entre os três tipos apresentados. Para exemplificar as condições de contorno apresentadas, olharemos novamente para um problema comum de transferência de calor, como na figura 2.1.



(a) Geometria do problema

(b) Solução numérica

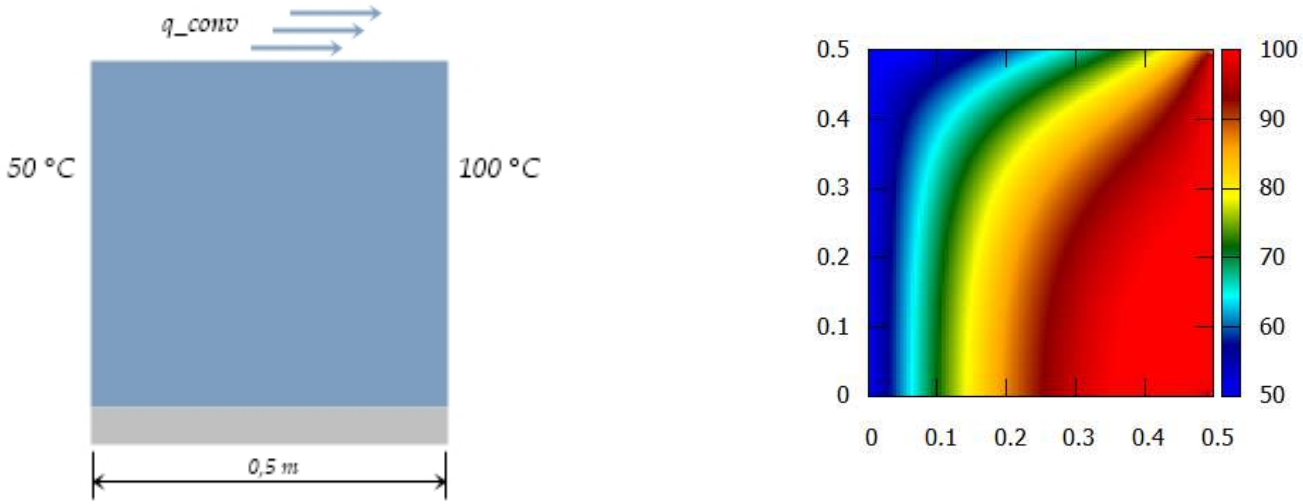
Figura 2.1: Exemplo de problema de transferência de calor com condições de fronteira de Neumann.
[fonte: autora]

Para a resolução deste problema via métodos numéricos é necessário, além de conhecer as grandezas físicas do problema, conhecer as condições de contorno nas paredes da placa.

Na figura 2.1, têm-se uma placa quadrada de dimensão $0.5m$, com fluxo imposto q_{flux} na parede esquerda, condição de parede adiabática, sem troca de calor, na parede direita e convecção, q_{conv}, nas paredes superior e inferior.

Neste exemplo é possível ver as condições de Neumann impostas às fronteiras da região. A condição de Neumann é dada pela convecção e, também, pela imposição do fluxo de calor q_{flux} à parede à esquerda e, como um caso especial da condição de Neumann, a imposição do fluxo de calor igual a zero na parede direita adiabática.

Por fim, um exemplo com condições de contorno combinadas é mostrado na figura 2.2.



(a) Geometria do problema

(b) Solução numérica

Figura 2.2: Exemplo de problema de transferência de calor com condições de fronteira combinadas.
[fonte: autora]

Neste exemplo, a parede superior se manteve como no exemplo anterior, com condição de convecção q_{conv}, mas as paredes esquerda e direita receberam valores de temperatura imposta de $50°C$ e $100°C$, respectivamente. A parede inferior, por sua vez, é adiabática. Neste caso, têm-se a condição de Dirichlet aplicada nas paredes laterais, porque, como visto, a temperatura recebeu diretamente um valor.

Estes exemplos foram formulados e resolvidos com o intuito de exemplificar diferentes condições de contorno e o resultado gráfico que pode ser obtido para a temperatura em estado de equilíbrio.

Partindo da equação de difusão do calor

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) + \dot{q} = \rho c\frac{\partial T}{\partial t}, \tag{2.16}$$

onde ρ é a massa específica, c é o calor específico, k é a condutividade térmica e $\dot{q}$ é a taxa de geração de energia térmica. Para problemas estacionários, $\frac{\partial T}{\partial t} = 0$.

Para a resolução numérica, foram utilizadas as seguintes informações: massa específica $\rho = 7900\ [kg/m^3]$, calor específico $c = 477\ [J/kgK]$, condutividade térmica $k = 15\ [W/mK]$, coeficiente de transferência de calor $h = 50\ [W/mK]$, $T_\infty = 10\ [°C]$, $q'' = 10\ [W/m^2]$ e $\dot{q} = 10^4\ [W/m^3]$.

Estes perfis de temperatura e a aplicação do método são conhecidos das disciplinas de transferência de calor e cálculo numérico, cursadas no curso de engenharia mecatrônica.

2.3 EQUAÇÕES DE DIFERENÇAS FINITAS

Para obter a solução de uma EDP é necessário o processo de discretização do domínio de cálculo. Esse processo consiste em escolher um conjunto de pontos nos quais os cálculos serão aplicados. Esse conjunto de pontos pode possuir pontos com espaçamento igual entre si ou pontos com espaçamentos diferentes entre si, e recebe o nome de malha. À malha cujos pontos possuem espaçamento Δx igual, dá-se o nome de malha com pontos uniformemente espaçados ou malha regular. Mas nem sempre a malha regular é aplicada em toda a região de interesse.

Em alguns casos cuja geometria de interesse é complexa, utiliza-se a malha de forma mista. De modo geral, em regiões onde o gradiente será elevado utiliza-se uma malha mais refinada, com Δx menor, enquanto para regiões com o gradiente menor, uma malha com espaçamento maior se faz suficiente. Esta estratégia é capaz de reduzir os erros referentes ao processo de expansão das equações em série de Taylor, o que será apresentado a seguir. Entretanto, para este trabalho será utilizada a malha com espaçamento uniforme por toda a geometria de estudo, como mostrado na Figura 2.3.

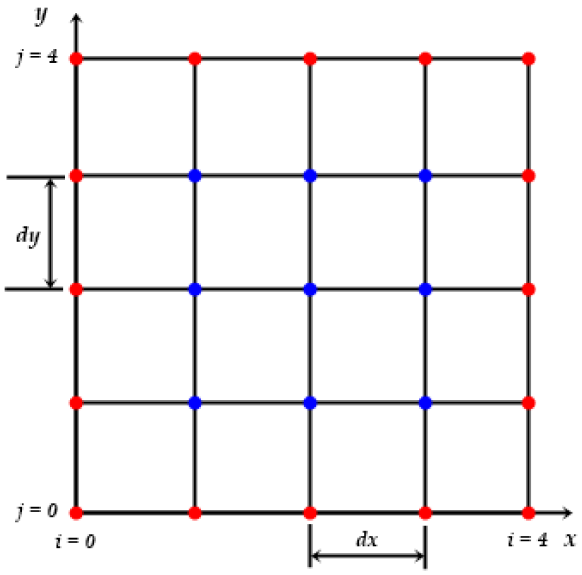


Figura 2.3: Malha com espaçamento uniforme a ser utilizada.
[fonte: autora]

2.3.1 EXPANSÃO EM SÉRIE DE TAYLOR

O raciocínio a seguir pode ser acompanhado através das referências [9] e [10]. Considerando $f = f(x)$ uma função contínua em um intervalo [a,b], com derivadas contínuas até a ordem N, e um ponto x pertencente ao intervalo, têm-se o Teorema de Taylor para este ponto, da seguinte forma

$$f(x) = f(x_0) + (\Delta x)\left.\frac{df}{dx}\right|_{x_0} + \frac{(\Delta x)^2}{2!}\left.\frac{d^2f}{dx^2}\right|_{x_0} + \frac{(\Delta x)^3}{3!}\left.\frac{d^3f}{dx^3}\right|_{x_0} + ... + R_N, \tag{2.17}$$

onde $\Delta\ x = x$ - x_0 e R_N é o resto

$$R_N = \frac{(\Delta x)^N}{N!} \left.\frac{d^N f}{dx^N}\right|_\zeta, \zeta \in [a, b].$$

Considerando uma malha de pontos uniformemente espaçados, com espaçamento de Δx entre pontos consecutivos e expandindo a série de Taylor em $f(x_i + \Delta x)$ em torno do ponto x_i, obtêm-se

$$f(x_i + \Delta x) = f(x_i) + \Delta x \left.\frac{df}{dx}\right|_i + \frac{(\Delta x)^2}{2!} \left.\frac{d^2 f}{dx^2}\right|_i + \frac{(\Delta x)^3}{3!} \left.\frac{d^3 f}{dx^3}\right|_i + ... + R_N, \tag{2.18}$$

Isolando a primeira derivada da equação (2.18) e dividindo toda a equação por Δx, têm-se

$$\left.\frac{df}{dx}\right|_i = \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x} + \left[-\frac{(\Delta x)}{2!} \left.\frac{d^2 f}{dx^2}\right|_i - \frac{(\Delta x)^2}{3!} \left.\frac{d^3 f}{dx^3}\right|_i - ... - R_N\right], \tag{2.19}$$

Segundo Fortuna [9], para obter a aproximação de $\frac{df}{dx}$ por diferenças finitas pode-se considerar o processo inverso para a obtenção do limite de uma função. Então, considerando o conceito da derivada de uma função, dado por

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}, \tag{2.20}$$

a aproximação é obtida pelo lado direito da equação (2.20), sem a aplicação do limite.

Então, na equação (2.19), a primeira derivada é dada pelo termo

$$\frac{f(x_i + \Delta x) - f(x_i)}{\Delta x},$$

enquanto o restante da equação representará o chamado erro local de truncamento (ELT)

$$ELT = \left[-\frac{(\Delta x)}{2!} \left.\frac{d^2 f}{dx^2}\right|_i - \frac{(\Delta x)^2}{3!} \left.\frac{d^3 f}{dx^3}\right|_i - ... - R_N\right]. \tag{2.21}$$

A equação (2.21) mostra que esse erro varia com o valor do espaçamento Δx e, desta forma, pode ser reduzido com a redução desse espaçamento.

2.3.2 DIFERENÇAS PROGRESSIVAS

A aproximação de primeira ordem para a primeira derivada de f é de diferenças progressivas quando o ponto utilizado é $f(x_i + \Delta x)$, ou seja, o ponto adiante de x_i, dando forma à equação

$$\left.\frac{\partial f}{\partial x}\right|_i = \frac{f_{i+1} - f_i}{\Delta x} + O(\Delta x). \tag{2.22}$$

Ao olharmos novamente para a (2.19), agora, pode-se ver a aproximação por diferenças progressivas, uma vez em que o termo $O(\Delta x)$, ordem Δx, passa a representar o ELT da equação.

Nota-se que este termo é de primeira ordem, e, mais a seguir, será apresentado sobre a influência da ordem na redução do erro.

2.3.3 DIFERENÇAS ATRASADAS

Outra aproximação de primeira ordem é obtida expandindo em Série de Taylor $f(x_i - \Delta x)$, em torno do ponto x_i, obtendo

$$\left.\frac{\partial f}{\partial x}\right|_i = \frac{f_i - f_{i-1}}{\Delta x} + O(\Delta x). \tag{2.23}$$

Nota-se que a equação 2.23 é análoga à equação (2.22), a diferença é que utiliza o valor da função f avaliada em um ponto anterior ao ponto x_i.

2.3.4 DIFERENÇAS CENTRAIS

Para obter uma aproximação de segunda ordem, $O(\Delta x^2)$, utiliza-se pontos adiante e anterior ao ponto x_i. Considere a expansão em série de Taylor das seguintes funções:

$$f(x_i + \Delta x) = f(x_i) + \Delta x \left.\frac{df}{dx}\right|_i + \frac{(\Delta x)^2}{2!} \left.\frac{d^2 f}{dx^2}\right|_i + O(\Delta x^3) \tag{2.24}$$

e

$$f(x_i - \Delta x) = f(x_i) - \Delta x \left.\frac{df}{dx}\right|_i + \frac{(\Delta x)^2}{2!} \left.\frac{d^2 f}{dx^2}\right|_i + O(\Delta x^3). \tag{2.25}$$

Subtraindo a segunda equação da primeira, elimina-se a segunda derivada de f, obtendo

$$f(x_i + \Delta x) - f(x_i - \Delta x) = 2(\Delta x) \left.\frac{df}{dx}\right|_i + O(\Delta x^3),$$

e rearranjando a equação de forma conveniente, segue que

$$\left.\frac{\partial f}{\partial x}\right|_i = \frac{f_{i+1} - f_{i-1}}{2\Delta x} + O(\Delta x^2). \tag{2.26}$$

Nota-se que, com a equação (2.26), ocorre a redução do erro da aproximação de forma quadrádica, a partir do aumento da ordem de $O(\Delta x)$ pela redução do espaçamento Δx.

Agora deseja-se obter uma aproximação para derivadas de segunda ordem. Para isso, as derivadas de segunda ordem, anteriormente eliminadas pela manipulação das equações, devem ser mantidas. Isso será feito somando as equações (2.24) e (2.25), obtendo

$$f(x_i + \Delta x) + f(x_i - \Delta x) = 2(x_i) + (\Delta x)^2 \left.\frac{\partial f^2}{x^2}\right|_i + O(\Delta x^4),$$

e, rearranjando de forma conveniente, têm-se

$$\left.\frac{\partial f^2}{\partial x^2}\right|_i = \frac{f_{i+1} - 2f_i + f_{i-1}}{(\Delta x)^2} + O(\Delta x^2), \tag{2.27}$$

a aproximação para derivadas de segunda ordem, com um erro de ordem 2.

Além disso, pode-se obter a aproximação para duas dimensões, isto é, para o incremento de Δx, na direção horizontal, i, e para o incremento de Δy, na direção vertical, j, de forma simultânea. Estes espaçamentos podem ser iguais entre si ou diferentes.

Para isso, supõe-se novamente uma função f contínua com derivadas parciais contínuas até a ordem N na região analisada e parte-se da expansão em série de Taylor para duas variáveis, dada por

$$\begin{aligned} f(x_i + \Delta x, y_j + \Delta y) = f(x_i, y_j) + (\Delta x) \left.\frac{\partial f}{\partial x}\right|_{i,j} + (\Delta y) \left.\frac{\partial f}{\partial y}\right|_{i,j} + \\ + \frac{(\Delta x)^2}{2!} \left.\frac{\partial f^2}{\partial x^2}\right|_{i,j} + 2\frac{(\Delta x)(\Delta y)}{2!} \left.\frac{\partial f^2}{\partial x \partial y}\right|_{i,j} + \frac{(\Delta y)^2}{2!} \left.\frac{\partial f^2}{\partial y^2}\right|_{i,j} + R_N, \end{aligned} \tag{2.28}$$

onde R_N é o resto.

A partir disso, através da manipulação das equações de maneira análoga à feita até aqui e com a combinação de $f(x_i \pm \Delta x, y_j - \Delta y)$ e $f(x_i \pm \Delta x, y_j + \Delta y)$, obtêm-se a equação procurada

$$\left.\frac{\partial^2 f}{\partial x \partial y}\right|_{i,j} = \frac{f_{i+1,j+1} - f_{i+1,j-1} - f_{i-1,j+1} + f_{i-1,j-1}}{4(\Delta x)(\Delta y)} + O[(\Delta x)^2, (\Delta y)^2], \tag{2.29}$$

Mais detalhes sobre o caminho traçado para a obtenção da equação (2.29) podem ser encontrados em [9].

2.4 Discretização de domínios através de malhas computacionais

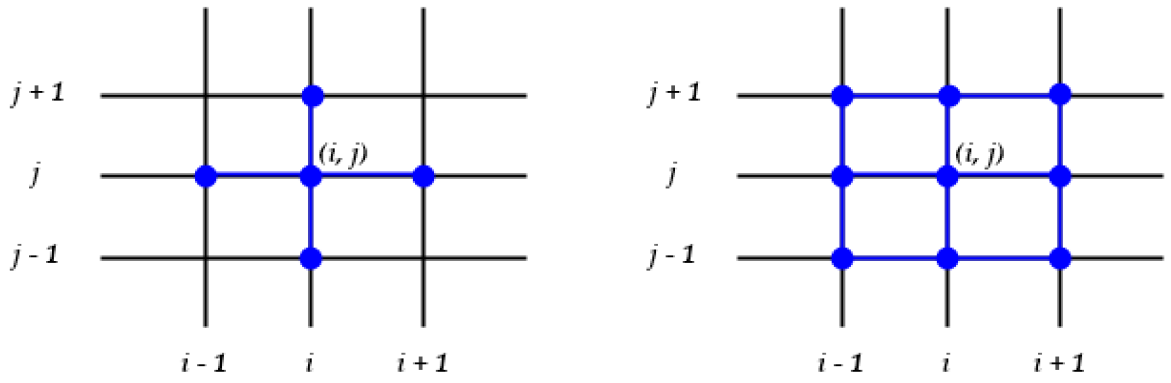
Como visto na seção 2.3, a discretização do domínio de cálculo gera uma malha composta de pontos, nos quais é possível realizar os cálculos. Esses pontos são referenciados a partir do ponto de análise, x_i, e, portanto, apresentam dependência entre si [8].

Serão apresentadas duas formas no decorrer desta seção, o estêncil de 5 e 9 pontos, e o desenvolvimento das matrizes de coeficientes para cada uma delas. O raciocínio desta seção pode ser encontrado com maiores detalhes nas literaturas [8], [9] e [4]. Na Figura 2.4, mostra-se graficamente essa dependência entre os pontos existentes em uma equação e a posição relativa entre eles.

Considerando um ponto de coordenadas (i, j), onde i indica sua posição na linha horizontal e j sua posição na linha vertical, é apresentado a seguir a aproximação conhecida por fórmula dos 5 pontos da equação de Laplace $\nabla^2 f = 0$, por diferenças centrais de segunda ordem.

$$\left.\frac{\partial f^2}{\partial x^2}\right|_{i,j} + \left.\frac{\partial f^2}{\partial y^2}\right|_{i,j} = \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{(\Delta x)^2} + \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{(\Delta y)^2} + O[(\Delta x^2), (\Delta y^2)]. \tag{2.30}$$

Essa aproximação é, também, comumente utilizada na resolução da equação de Poisson.



(a) (b)

Figura 2.4: Ilustração do (a) estêncil de 5 pontos e (b) estêncil de 9 pontos. [fonte: autora]

Conhecendo a forma da matriz dos coeficientes **A**, podemos olhar para os problemas que serão foco de estudo, as equações estacionárias. Para isso, considera-se a equação de Laplace discretizada, bidimensional, através da fórmula de 5 pontos em uma malha regular

$$\left.\frac{\partial f^2}{\partial x^2}\right|_{i,j} + \left.\frac{\partial f^2}{\partial y^2}\right|_{i,j} \approx \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{(\Delta x)^2} + \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{(\Delta y)^2} = 0.$$

Simplifica-se a equação multiplicando por $(\Delta x)^2$ e obtêm-se

$$f_{i+1,j} - 2f_{i,j} + f_{i-1,j} + \frac{(\Delta x)^2}{(\Delta y)^2}(f_{i,j+1} - 2f_{i,j} + f_{i,j-1}) = 0,$$

fazendo $\dfrac{\Delta x}{\Delta y} = \beta$ e organizando os termos de modo conveniente

$$f_{i+1,j} + f_{i-1,j} - 2(1+\beta^2)f_{i,j} + \beta^2(f_{i,j+1} + f_{i,j-1}) = 0. \tag{2.31}$$

Considere uma placa de dimensões conhecidas e com temperaturas mantidas constantes nas fronteiras, apresentada na Figura 2.5. Considere uma malha regular nas duas direções x e y, com valores de temperatura $f_{(i,j)}$, onde $i, j = 0, 1, ..., 4$.

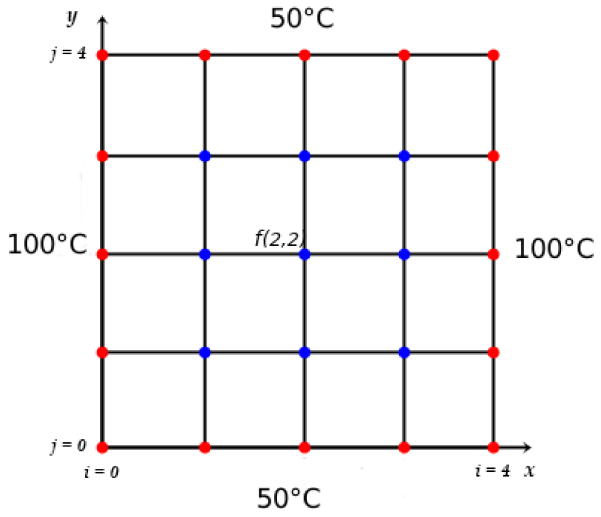


Figura 2.5: Exemplo para formulação da equação.
[fonte: autora]

A equação 2.31 é válida para todos os pontos da malha, nos permitindo escrever um sistema linear.

Assim, monta-se um sistema matricial $\mathbf{Ax} = \mathbf{b}$. A matriz dos coeficientes da equação é $\mathbf{A}$, $\mathbf{x}$ é o vetor dos valores das temperaturas $f_{(i,j)}$ no interior da placa e $\mathbf{b}$ é o vetor de constantes. No exemplo, as temperaturas constantes no contorno da região caracterizam a condição de fronteira de Dirichlet e são valores armazenados no vetor $\mathbf{b}$, do lado direito da equação. Assim, o sistema matricial, para este caso, fica

$$\begin{bmatrix} \gamma & 1 & 0 & \beta^2 & 0 & 0 & 0 & 0 & 0 \\ 1 & \gamma & 1 & 0 & \beta^2 & 0 & 0 & 0 & 0 \\ 0 & 1 & \gamma & 1 & 0 & \beta^2 & 0 & 0 & 0 \\ \beta^2 & 0 & 0 & \gamma & 1 & 0 & \beta^2 & 0 & 0 \\ 0 & \beta^2 & 0 & 1 & \gamma & 1 & 0 & \beta^2 & 0 \\ 0 & 0 & \beta^2 & 0 & 1 & \gamma & 0 & 0 & \beta^2 \\ 0 & 0 & 0 & \beta^2 & 0 & 0 & \gamma & 1 & 0 \\ 0 & 0 & 0 & 0 & \beta^2 & 0 & 1 & \gamma & 1 \\ 0 & 0 & 0 & 0 & 0 & \beta^2 & 0 & 1 & \gamma \end{bmatrix} \begin{bmatrix} f_{1,1} \\ f_{2,1} \\ f_{3,1} \\ f_{1,2} \\ f_{2,2} \\ f_{3,2} \\ f_{1,3} \\ f_{2,3} \\ f_{3,3} \end{bmatrix} = \begin{bmatrix} -f_{0,1} - \beta^2 f_{1,0} \\ -\beta^2 f_{2,0} \\ -f_{4,1} - \beta^2 f_{3,0} \\ -f_{0,2} \\ 0 \\ -f_{4,2} \\ -f_{0,3} - \beta^2 f_{1,4} \\ -\beta^2 f_{2,4} \\ -f_{4,3} - \beta^2 f_{3,4} \end{bmatrix}$$

onde $\gamma = -2(1 + \beta^2)$.

Nota-se que é importante a imposição da condição de contorno de Dirichlet à região de análise. Com pelo menos uma condição de contorno desse tipo, informada à fronteira, têm-se valores conhecidos para o vetor $\mathbf{b}$ e o sistema matricial apresenta solução única.

Para este trabalho será considerada a malha regular para todo o domínio de cálculo. Com isso, tendo $\Delta x = \Delta y$, segue que

$$\beta = \frac{\Delta x}{\Delta y} = 1$$

e

$$\gamma = -2(1+\beta^2) = -4,$$

e a equação (2.31) torna-se

$$f_{i+1,j} + f_{i-1,j} - 4f_{i,j} + f_{i,j+1} + f_{i,j-1} = 0. \tag{2.32}$$

Com isso, aplicando os dados conhecidos do exemplo 2.5, o sistema matricial $\mathbf{A}\ \mathbf{x} = \mathbf{b}$, pode ser escrito como

$$\begin{bmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{bmatrix} \begin{bmatrix} f_{1,1} \\ f_{2,1} \\ f_{3,1} \\ f_{1,2} \\ f_{2,2} \\ f_{3,2} \\ f_{1,3} \\ f_{2,3} \\ f_{3,3} \end{bmatrix} = \begin{bmatrix} -50 \\ -50 \\ -150 \\ -50 \\ 0 \\ -100 \\ -50 \\ -50 \\ -150 \end{bmatrix}$$

onde $\mathbf{A}$ é uma matriz esparsa, com grande quantidade de zeros.

Outra aproximação a ser considerada é o Laplaciano de 9 pontos. Essa aproximação apresenta quarta ordem e, portanto, maior precisão nos resultados e maior custo computacional para a obtenção deste resultado. É uma aproximação que, diferentemente da aproximação de 5 pontos vista anteriormente, utiliza-se dos pontos com incremento simultâneo nas direções i e em j, isto é, para o ponto de interesse $x_{(i,j)}$, pontos da forma $x_{(i+1,j+1)}$, $x_{(i-1,j-j)}$, $x_{(i+1,j-1)}$ e $x_{(i-1,j+1)}$ são considerados. Desta forma, totaliza-se 9 pontos com influência direta ao ponto de interesse, apesar de apresentarem pesos distintos sobre o resultado obtido para o ponto. A aproximação em questão é dada por

$$\begin{aligned} \left.\frac{\partial f^2}{\partial x^2}\right|_{i,j} + \left.\frac{\partial f^2}{\partial y^2}\right|_{i,j} + \frac{(\Delta x)^2}{12} \times \left(\left.\frac{\partial f^4}{\partial x^4}\right|_{i,j} + 2 \left.\frac{\partial f^2}{\partial x \partial y}\right|_{i,j} + \left.\frac{\partial f^4}{\partial y^4}\right|_{i,j} \right) + O(\Delta x^4) \approx \\ \approx \frac{1}{6(\Delta x)^2} \times [4f_{i-1,j} + 4f_{i+1,j} + 4f_{i,j-1} + 4f_{i,j+1} + f_{i-1,j-1} + \\ + f_{i-1,j+1} + f_{i+1,j-1} + f_{i+1,j+1} - 20f_{i,j}] = 0, \end{aligned} \tag{2.33}$$

Para atingir estes pontos, retorna-se à seção 2.3.4 e utiliza-se a equação (2.28), que permite justamente o incremento simultâneo desejado, e a equação (2.29), também de interesse, relembrada a seguir

$$\left.\frac{\partial f^2}{\partial x \partial y}\right|_{i,j} = \frac{f_{i+1,j+1} - f_{i+1,j-1} - f_{i-1,j+1} + f_{i-1,j-1}}{4(\Delta x)(\Delta y)} + O[(\Delta x)^2, (\Delta y)^2].$$

Expandindo em série de Taylor em torno dos pontos $x_{(i+1,j)}$, $x_{(i-1,j)}$, $x_{(i,j+1)}$ e $x_{(i,j-1)}$, considerando $\Delta x = \Delta y$ e visando a obtenção de uma aproximação de quarta ordem, pode-se obter de forma análoga à vista na seção 2.3.4 para a aproximação de segunda ordem, as seguintes equações

$$\left.\frac{\partial f^4}{\partial x^4}\right|_{i,j} = 12 \times \frac{2f_{(i+1,j)} + 2f_{(i-1,j)} - 4f_{(i,j)}}{(\Delta x)^4} \tag{2.34}$$

e

$$\left.\frac{\partial f^4}{\partial y^4}\right|_{i,j} = 12 \times \frac{2f_{(i,j+1)} + 2f_{(i,j-1)} - 4f_{(i,j)}}{(\Delta y)^4}. \tag{2.35}$$

Então, com as equações (2.34) e (2.35) e as demais aproximações de segunda ordem vistas, pode-se obter a equação (2.33). Retoma-se o exemplo 2.5, para a construção da forma matricial $\mathbf{Ax} = \mathbf{b}$, à partir da aproximação por Laplaciano de 9 pontos, a seguir

$$\begin{bmatrix} -20 & 4 & 0 & 4 & 1 & 0 & 0 & 0 & 0 \\ 4 & -20 & 4 & 1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 4 & -20 & 0 & 1 & 4 & 0 & 0 & 0 \\ 4 & 1 & 0 & -20 & 4 & 0 & 4 & 1 & 0 \\ 1 & 4 & 1 & 4 & -20 & 4 & 1 & 4 & 1 \\ 0 & 1 & 4 & 0 & 4 & -20 & 0 & 1 & 4 \\ 0 & 0 & 0 & 4 & 1 & 0 & -20 & 4 & 0 \\ 0 & 0 & 0 & 1 & 4 & 1 & 4 & -20 & 4 \\ 0 & 0 & 0 & 0 & 1 & 4 & 0 & 4 & -20 \end{bmatrix} \begin{bmatrix} f_{1,1} \\ f_{2,1} \\ f_{3,1} \\ f_{1,2} \\ f_{2,2} \\ f_{3,2} \\ f_{1,3} \\ f_{2,3} \\ f_{3,3} \end{bmatrix} = \begin{bmatrix} -4f_{0,1} - 4f_{1,0} - f_{0,2} - f_{2,0} - f_{0,0} \\ -4f_{2,0} - f_{1,0} - f_{3,0} \\ -4f_{4,1} - 4f_{3,0} - f_{2,0} - f_{4,0} - f_{4,2} \\ -4f_{0,2} - f_{0,1} - f_{0,3} \\ 0 \\ -4f_{4,2} - f_{4,1} - f_{4,3} \\ -4f_{0,3} - 4f_{1,4} - f_{0,2} - f_{0,4} - f_{2,4} \\ -4f_{2,4} - f_{1,4} - f_{3,4} \\ -4f_{4,3} - 4f_{3,4} - f_{2,4} - f_{4,2} - f_{4,4} \end{bmatrix}$$

Análogo à fórmula de 5 pontos, para o mesmo exemplo, os pontos de fronteira com valores constantes conhecidos compõem o vetor $\mathbf{b}$ do lado direito da equação, e a matriz $\mathbf{A}$ apresenta os coeficientes da equação de 9 pontos.

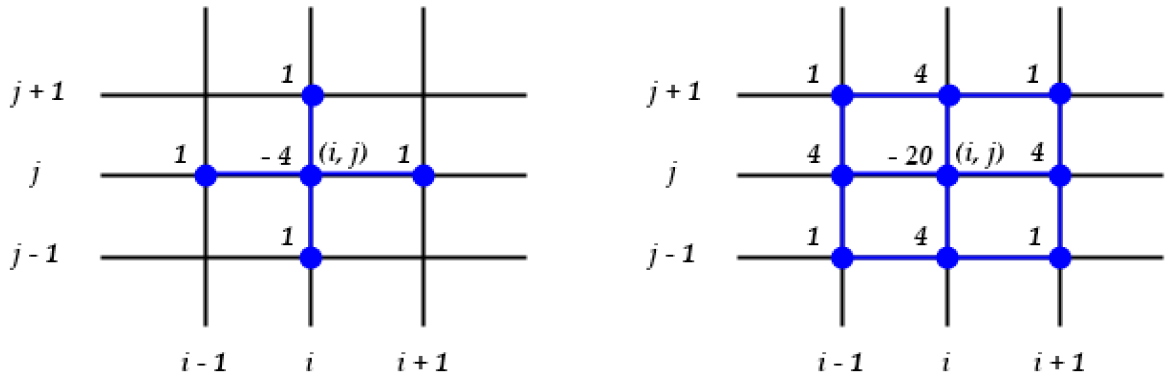
Aplicando os dados fornecidos pelo exemplo 2.5, o sistema matricial fica

$$\begin{bmatrix} -20 & 4 & 0 & 4 & 1 & 0 & 0 & 0 & 0 \\ 4 & -20 & 4 & 1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 4 & -20 & 0 & 1 & 4 & 0 & 0 & 0 \\ 4 & 1 & 0 & -20 & 4 & 0 & 4 & 1 & 0 \\ 1 & 4 & 1 & 4 & -20 & 4 & 1 & 4 & 1 \\ 0 & 1 & 4 & 0 & 4 & -20 & 0 & 1 & 4 \\ 0 & 0 & 0 & 4 & 1 & 0 & -20 & 4 & 0 \\ 0 & 0 & 0 & 1 & 4 & 1 & 4 & -20 & 4 \\ 0 & 0 & 0 & 0 & 1 & 4 & 0 & 4 & -20 \end{bmatrix} \begin{bmatrix} f_{1,1} \\ f_{2,1} \\ f_{3,1} \\ f_{1,2} \\ f_{2,2} \\ f_{3,2} \\ f_{1,3} \\ f_{2,3} \\ f_{3,3} \end{bmatrix} = \begin{bmatrix} -275 \\ -300 \\ -825 \\ 0 \\ 0 \\ -600 \\ -275 \\ -300 \\ -825 \end{bmatrix}$$

Nesta forma matricial, nota-se uma matriz $\mathbf{A}$ esparsa, semelhante àquela obtida para a formulação de 5 pontos.

Outro ponto interessante que pode-se observar é os diferentes pesos exercidos sobre o ponto

de interesse, pelos demais pontos, para as fórmulas de 5 pontos e 9 pontos [8]. Para isso, considere a Figura 2.6, nesta pode-se ver os pesos que cada solução dos pontos ao redor do ponto de interesse exerce na solução deste. Nota-se, também, que a soma de todos estes valores será zero.



(a) (b)

Figura 2.6: (a) Pesos no estêncil de 5 pontos e (b) pesos na fórmula de 9 pontos. [fonte: autora]

A precisão da fórmula utilizada pode ser observada pela ordem da aproximação obtida. O estêncil de 5 pontos fornece uma aproximação de segunda ordem, enquanto a fórmula de 9 pontos fornece uma aproximação de quarta ordem.

Sabe-se que quanto maior esta ordem, maior é o espaçamento da malha para se atingir uma mesma precisão. Isto é, para aumentar a precisão pode-se aumentar o número de pontos em uma malha ou aumentar a ordem da aproximação utilizada. Entretanto, ao aumentar-se a ordem de uma aproximação, o custo computacional para gerar os resultados aumenta bastante, muitas vezes tornando inviável a escolha de uma ordem muito alta.

Ao escolher a aproximação a ser usada, deve-se levar em consideração a precisão obtida através da equação mas, também, o custo computacional que será envolvido em todo o processo.

Em muitos casos, não se faz necessária a aproximação de quarta ordem, visto que essa corresponde a um alto custo computacional e uma ordem inferior pode ser suficiente para a resolução do problema. Mas, quando necessário, utiliza-se um método para aumentar a ordem da aproximação obtida com a fórmula dos 5 pontos.

2.5 CONSISTÊNCIA, ESTABILIDADE E CONVERGÊNCIA

A solução de um problema não será exata devido a diversos erros quanto à aproximação numérica das condições, à discretização das equações e também quanto aos arredondamentos contidos em todo o processo de cálculo. Um erro importante que surge, visto anteriormente, é o ELT.

O ELT indica quão próxima da solução correta o resultado da simulação numérica poderá chegar, ou, em outras palavras, o quanto o resultado obtido através da simulação consegue representar a solução real. Assim, o ELT acaba servindo como um indicador da qualidade da aproximação das derivadas parciais. Então, pode-se olhar para o ELT e observar seu comportamento para obter essa informação.

Este erro aparece com a aproximação por diferenças finitas para as derivadas parciais, a partir da expansão em série de Taylor da função de interesse. Além disso, foi visto que o ELT é reduzido com o refino da malha, isto é, com a redução do espaçamento entre os pontos, Δx.

Com isso, então, é esperado que através da redução de Δx a parcela do ELT tenda a zero, sendo, teoricamente, eliminada da equação. Esse processo recuperaria a aproximação contínua original da equação diferencial parcial, partindo da equação discreta de diferenças finitas. Neste caso, é dito que a aproximação de diferenças finitas é consistente com a equação diferencial discretizada por ela [9][1].

Em resumo, o conceito de consistência acaba por averiguar a qualidade da discretização realizada na equação diferencial. Caso consistente, a solução numérica da equação de diferenças finitas representa a solução da equação diferencial parcial de interesse. Caso contrário, a solução numérica não é representativa da solução desejada.

Outro conceito importante é o conceito de estabilidade de um método numérico. Esse conceito também é relacionado aos erros presentes em todo o processo de obtenção das equações até a obtenção do resultado final. O método será considerado estável quando, a cada resolução, fornecer resultados que se aproximam gradualmente da solução real das equações.

Para isso, erros na especificação das condições auxiliares devem ser evitados e erros de arredondamento devem ser controlados. Os últimos são erros inevitáveis e que, caso cresçam descontroladamente, podem acarretar na divergência da solução. Isto é, a cada passo de tempo será fornecido um resultado mais distante da solução real.

A forma de controlar o crescimento desses erros é estabelecer critérios de estabilidade ao método numérico utilizado.

Neste ponto é importante falar sobre os problemas transientes. Estes apresentam uma restrição quanto ao intervalo de tempo, Δt, que pode ser usado para gerar uma solução estável. Como exemplo, considere a equação transiente de difusão, dada por

$$\frac{\partial \phi}{\partial t} = \Gamma \frac{\partial \phi^2}{\partial x^2},$$

que discretizada com o método de Euler explícito e por diferenças centrais de segunda ordem, se torna

$$\frac{{T_i}^{n+1} - {T_i}^n}{\Delta t} = \alpha \frac{{T_{i-1}}^n - 2{T_i}^n + {T_{i+1}}^n}{(\Delta x)^2} + O[(\Delta t), (\Delta x^2)]. \tag{2.36}$$

Para essa equação de diferenças finitas, o critério de estabilidade é

$$s = \frac{\alpha(\Delta t)}{(\Delta x)^2} \leq \frac{1}{2}, \tag{2.37}$$

Com a equação (2.37), nota-se que o refino da malha torna necessária a redução também do parâmetro Δt, para que o valor do critério de estabilidade não extrapole o aceitável para gerar uma resposta estável.

Entretanto, nem sempre é possível determinar o critério de estabilidade como na equação anterior e, com isso, realiza-se a experimentação numérica e a comparação com equações mais simples que modelam fenômenos similares ao desejado.

Com o critério de estabilidade tão restrito para as metodologias explícitas aplicadas aos problemas transientes, o método implícito torna-se viável em alguns casos, para que um Δt maior possa ser utilizado sem desencadear um processo instável, com divergência no resultado final [9].

Em contrapartida, os problemas estacionários, que são o principal foco deste trabalho, não possuem esta restrição quanto ao tempo Δt, então os critérios de estabilidade para os problemas transientes e suas restrições não serão tratados a fundo, enquanto a estabilidade relacionada aos problemas estacionários será melhor apresentada na sequência.

Por fim, o conceito de convergência baseia-se nos dois conceitos anteriores apresentados. O método numérico será convergente se a solução fornecida por este convergir para a solução exata da EDP, com o decorrer dos passos de tempo.

Cabe ressaltar que, para a solução numérica se aproximar gradualmente da solução exata, necessariamente deve-se ser possível recuperar a EDP original através da redução do espaçamento Δx e consequente redução do ELT. Então, a consistência é uma condição necessária à convergência [9]. Entretanto, a consistência não é uma condição única para a convergência, sendo possível ter um modelo consistente porém não convergente.

Para determinar a convergência, portanto, é necessário olhar também para a estabilidade do método. De modo geral, segundo o Teorema de Equivalência de Lax [9], o método numérico será convergente caso seja, necessariamente, consistente e estável. Em resumo, a consistência em conjunto com a estabilidade de um método é condição necessária e suficiente para a convergência do método.

2.5.1 Estabilidade para equações elípticas

Como visto brevemente na seção anterior, o conceito de estabilidade pode ser difícil de determinar [5]. Para os problemas envolvendo equações elípticas, pode-se limitar o conceito da estabilidade à relação

$$\mathbf{A}^h \mathbf{E}^h = -\boldsymbol{\tau}^h, \tag{2.38}$$

onde $\mathbf{A}$ é a matriz dos coeficientes associada ao sistema linear gerado pela discretização, $h = \Delta x \Delta y$, E é o erro global dado por $E = u_{aprox} - u_{exata}$, isto é, o erro global é a solução aproximada menos a solução exata, e, por fim, $\tau = ELT$.

O termo $||(\mathbf{A}^h)^{-1}||$ deve ser limitado na expressão acima e essa restrição irá definir a estabilidade para esse caso.

Portanto, torna-se necessária a escolha de uma norma para o cálculo deste termo. A norma escolhida, por sua vez, não pode gerar inconsistências no espaço considerado, de dimensão finita. Assim, a limitação da estabilidade se torna associada à escolha da norma utilizada [8][14].

Considerando a norma 2, também denotada por $l2$, têm-se

$$||(\mathbf{A}^h)^{-1}||_2 = \rho(\mathbf{A}^h)^{-1}, \tag{2.39}$$

Aqui, para entender isso, torna-se importante olhar para os autovalores da matriz $\mathbf{A}$ [3]. Os autovalores de $\mathbf{A}$ têm a forma

$$\lambda_{p,q} = \frac{2}{h^2}[(cos(p\pi h) - 1) + (cos(q\pi h) - 1)], \tag{2.40}$$

onde $p, q = 1, 2, ..., m$.

Considerando a equação (2.40), observa-se que os autovalores de $\mathbf{A}$ são necessariamente negativos. Então, o maior valor, mais próximo à origem, é o autovalor dado por $\lambda_{1,1}$.

Para estes índices de p e q, têm-se

$$\lambda_{1,1} = \frac{4}{h^2}(cos(\pi h) - 1)$$

e expandindo $cos(\pi h)$, têm-se

$$cos(\pi h) = 1 - \frac{(\pi h)^2}{2} + \frac{(\pi h)^4}{4!} + ...$$

então

$$\lambda_{1,1} = \frac{4}{h^2}\left[1 - \frac{\pi^2 h^2}{2} + O(h^4) - 1\right] = -2\pi^2 + O(h^2)$$

E, com isso,

$$||(\mathbf{A}^h)^{-1}||_2 = \frac{1}{\rho(\mathbf{A}^h)} \simeq \frac{1}{2\pi^2} < 1, \tag{2.41}$$

assim, o método associado à matriz $\mathbf{A}$ é estável, sendo esta a nossa condição determinante da estabilidade.

Além disso, associada aos conceitos de estabilidade, convergência e consistência, a resolução de sistemas lineares também possui como variável importante o tempo que se leva à atingir a convergência, devido ao processo computacional envolvido na resolução do sistema.

Sabendo disso, note o raio espectral de $||\mathbf{A}^h||_2 = \frac{8}{h^2}$, lembrando que o raio espectral é o máximo dos módulos dos autovalores. Então, têm-se

$$k_2(\mathbf{A}^h) = ||\mathbf{A}^h||_2||(\mathbf{A}^h)^{-1}||_2 = \frac{1}{2\pi^2} + \frac{8}{h^2}, \tag{2.42}$$

onde $k_2(\mathbf{A})$ é o número de condição na norma $l2$.

Este número tende a um - $k_2(\mathbf{A}) \to 1$ - à medida em que a malha é refinada com a redução do valor de h. Este valor é responsável por informar o quanto uma perturbação é capaz de

interferir no resultado final de uma aproximação.

Quanto menor o número de condição de um problema, diz-se que este é um problema bem condicionado e uma alteração nas variáveis independentes do problema não irá interferir fortemente no resultado final. Em contrapartida, um problema com alto número de condição é dito um problema mal condicionado, e uma perturbação nas variáveis independentes poderá se propagar fortemente e gerar grande alteração nas variáveis dependentes e no resultado final [15].

3. Metodologia

Após conhecer as técnicas de discretização que podem ser aplicadas para a resolução de EDPs, deve-se conhecer as técnicas numéricas. Para resolver o sistema linear originado dessa discretização.

Os sistemas lineares a serem resolvidos possuem a forma

$$\mathbf{A}\ \mathbf{x} = \mathbf{b}, \tag{3.1}$$

onde $\mathbf{A}$ é a matriz de coeficientes e $\mathbf{b}$ é o vetor das constantes. E $\mathbf{x}$ é o vetor das variáveis, $\mathbf{A} \in \mathbb{R}^{n,n}$, $\mathbf{b}$ e $\mathbf{x} \in \mathbb{R}^n$.

3.1 Métodos de Resolução Numérica para Sistemas Lineares

Os métodos podem ser divididos em dois grupos: métodos diretos e métodos iterativos. Os métodos diretos são aqueles que obtêm uma solução exata para o problema através de operações realizadas diretamente nas equações dos sistemas lineares. Mas apesar de dita exata, o acúmulo dos erros de arredondamento ao decorrer das operações, que ocorre na prática, faz com que a solução seja, ainda assim, diferente da solução exata [12]. Apesar de existirem formas de amenizar esse efeito, a propagação dos erros ao longo das operações é inevitável e pode ser considerável dependendo do sistema linear.

Já os métodos iterativos são esquemas que partem de uma aproximação inicial $\mathbf{x}_0$ para obter-se aproximações $\mathbf{x}_k$, onde k é a variável que percorre da iteração inicial à iteração na qual se atinge a convergência do método e o resultado final aproximado do sistema linear. A convergência, nesses casos, precisa ser garantida adotando-se algumas condições específicas ao método. Apesar disso, são métodos eficientes e indicados para a resolução de sistemas lineares com matrizes esparsas (com muitos valores iguais a zero)[9][11] ou matrizes de banda.

O foco será mantido nos métodos iterativos para a resolução dos sistemas lineares esparsos, nos quais pode-se facilmente resolver um grande sistema com boa economia de armazenamento de memória.

Adiante, serão apresentados quatro métodos iterativos comumente encontrados na literatura: Método de Gauss Jacobi (MGJ), Método de Gauss Seidel (MGS), Método de Sobre-Relaxação Sucessiva (MSOR) e Método do Gradiente Conjugado (MGC), sendo este o maior

foco do estudo.

3.1.1 MÉTODO DE GAUSS JACOBI

Para conhecer como o MGJ funciona, primeiramente deve-se conhecer a matriz **A** dos coeficientes. Essa matriz pode ser escrita como a soma de três outras matrizes, na forma

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}, \tag{3.2}$$

onde **L** é a matriz triangular inferior de **A**, **D** é a matriz diagonal e **U** é a matriz triangular superior [9], como

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ a_{2,1} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & 0 \end{bmatrix} + \begin{bmatrix} a_{1,1} & 0 & \dots & 0 \\ 0 & a_{2,2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{n,n} \end{bmatrix} + \begin{bmatrix} 0 & a_{1,2} & \dots & a_{1,n} \\ 0 & 0 & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

Assim, o sistema visto anteriormente $\mathbf{A}\ \mathbf{x} = \mathbf{b}$ torna-se

$$(\mathbf{L} + \mathbf{D} + \mathbf{U})\mathbf{x} = \mathbf{b}$$

$$\mathbf{D}\ \mathbf{x} = -(\mathbf{L} + \mathbf{U})\mathbf{x} + \mathbf{b}$$

$$\mathbf{x} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\mathbf{x} + \mathbf{D}^{-1}\mathbf{b},$$

então, para um valor inicial $\mathbf{x}^0$ e $k \geq 0$, tem-se

$$\mathbf{x}^{(k+1)} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b}. \tag{3.3}$$

Dessa forma, o processo iterativo para o MGJ possue a forma

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{\substack{j=1, \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right], \tag{3.4}$$

para $i = 1, ..., n$. Ao olhar para a equação (3.4), nota-se que no cálculo de cada iteração utiliza-se apenas informações da iteração anterior.

3.1.2 MÉTODO DE GAUSS SEIDEL

O Método de Gauss Seidel é semelhante ao MGJ e sua dedução é análoga e encontrada na literatura [9][3][7][14]. De forma direta, para um valor inicial $\mathbf{x}^0$ e $k \geq 0$, tem-se

$$\mathbf{x}^{(k+1)} = -(\mathbf{D} + \mathbf{L})^{-1}\mathbf{U}\mathbf{x}^{(k)} + (\mathbf{D} + \mathbf{L})^{-1}\mathbf{b}. \tag{3.5}$$

E o processo iterativo para o MGS será da forma

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right], \tag{3.6}$$

para $i = 1, ..., n$. Observando a equação (3.6), vê-se que em cada componente de $\mathbf{x}$ na iteração $k+1$ é utilizada informações mais recentes obtidas na iteração atual. Assim, o MGS converge mais rápido do que o MGJ, desde que as condições para se ter convergência sejam satisfeitas.

Considere o sistema linear $\mathbf{Ax} = \mathbf{b}$ possível e determinado com $a_{i,i} \neq 0$, para $i = 1, 2, ..., n$.

(i) Se uma matriz for estritamente diagonal dominante, ou seja

$$|a_{i,i}| > \sum_{\substack{j=1, \\ j \neq i}}^{n} |a_{i,j}| \, \forall i = 1, 2, ..., n, \tag{3.7}$$

então há garantia de convergência para o MGS independente da aproximação inicial,

(ii) se

$$|a_{i,i}| = \sum_{\substack{j=1, \\ j \neq i}}^{n} |a_{i,j}|, \tag{3.8}$$

para quase todo i e existir pelo menos uma linha k, com $1 \leq k \leq n$, onde

$$|a_{k,k}| > \sum_{\substack{j=1, \\ j \neq k}}^{n} |a_{k,j}|,$$

então também há garantia de convergência para o MGS independente da aproximação inicial, e

(iii) se o método iterativo definido por

$$\mathbf{x}^{(k+1)} = \mathbf{B}\mathbf{x}^{(k)} + \mathbf{g}, \tag{3.9}$$

apresentar $||\mathbf{B}|| < 1$, para alguma norma matricial, há garantia de convergência para o método.

3.1.3 Método de Sobre-Relaxação Sucessiva

Os métodos iterativos em geral mostram que, com o passar das iterações, a diferença entre as aproximações $\mathbf{x}^{(k+1)}$ e $\mathbf{x}^{(k)}$ diminuem até que seja atingida a convergência e obtida a solução adequada. Isso faz com que sejam necessárias muitas iterações para atingir-se a convergência do método, sendo necessária a resolução de muitas equações ao longo do processo de resolução, levando a um maior tempo de processamento.

Esse problema pode ser melhorado através do método apresentado a seguir. O Método SOR consiste em sobre-relaxar essa diferença mencionada entre $\mathbf{x}^{(k+1)}$ e $\mathbf{x}^{(k)}$, extrapolando o valor obtido de $\mathbf{x}^{(k+1)}$, fazendo com que a diferença não seja tão reduzida com o passar das iterações [9][14]. Isso leva a uma convergência mais rápida do método pois $\mathbf{x}^{(k+1)}$ se aproximará mais da

solução esperada.

Para extrapolar o valor de $\mathbf{x}^{(k+1)}$ a cada iteração, utiliza-se a seguinte equação

$$\mathbf{x}_{SOR}^{(k+1)} = \mathbf{x}_{GS}^{(k+1)} + \lambda \left[\mathbf{x}_{GS}^{(k+1)} - \mathbf{x}^{(k)} \right], \tag{3.10}$$

onde $\mathbf{x}_{SOR}^{(k+1)}$ é o valor extrapolado pelo MSOR, $\mathbf{x}_{GS}^{(k+1)}$ é o valor pelo Método de Gauss Seidel, $\mathbf{x}^{(k)}$ é o valor na iteração anterior e λ é a constante que será ajustada, responsável pela extrapolação do resultado obtido por MGS.

Considera-se λ um valor entre 0 e 1, pois, para extrapolarmos o resultado, então λ deve ser maior que zero. Mas, para evitar que o resultado cresça indefinidamente e divirja, restringe-se λ menor do que 1.

A relaxação do método SOR é definida, usualmente, pelo fator de relaxação $\omega = 1 + \lambda$. Com isso, obtém-se uma nova equação

$$\mathbf{x}_{SOR}^{(k+1)} = \omega \mathbf{x}_{GS}^{(k+1)} + (1 - \omega)\mathbf{x}^{(k)}, \tag{3.11}$$

onde $0 < \omega < 2$. Quando $\omega = 1$, têm-se o mesmo resultado obtido por MGS, quando $0 < \omega < 1$, diz-se que o sistema está sub-relaxado e, quando $1 < \omega < 2$ o sistema está propriamente sobre-relaxado e os valores calculados são extrapolados [9].

3.1.4 Método do Gradiente Conjugado

O Método do Gradiente Conjugado é o último método abordado aqui e também o método de maior foco deste trabalho. Para compreendê-lo e entender o porquê de ser o método de mais rápida convergência e melhor resolução para sistemas lineares esparsos, deve-se primeiro entender em que consiste o Método dos Gradientes (MG) e o que são direções conjugadas.

O Método dos Gradientes é aplicável apenas quando a matriz dos coeficientes $\mathbf{A}$ é simétrica definida positiva [13][7]. Uma matriz simétrica obedece a relação $\mathbf{A}^t = \mathbf{A}$, ou seja, os elementos da matriz respeitam a igualdade dos elementos $a_{i,j} = a_{j,i}$, para todos os índices i e j.

Além disso, a matriz é dada como definida positiva quando todas as matrizes em $\mathbf{A}$, a partir de seu primeiro elemento, possuírem determinante positivo. Isto é, é necessário que:

- o canto superior esquerdo 1x1 de $\mathbf{A}$ possua determinante positivo,
- o canto superior esquerdo 2x2 de $\mathbf{A}$ possua determinante positivo,

⋮

- a própria matriz $\mathbf{A}$ possua determinante positivo.

Cumpridos esses requisitos, o método pode ser utilizado.

Considerando o sistema linear já mencionado anteriormente $\mathbf{A}\ \mathbf{x} = \mathbf{b}$, em que $\mathbf{x}$ é a solução almejada, toma-se $\mathbf{x}^{(k)}$, onde $k = 0$, como a aproximação inicial da solução do problema. Assim como os demais métodos vistos até aqui, o MG trabalhará com diversas iterações (k) até que esse vetor $\mathbf{x}^{(k)}$ se aproxime gradualmente da solução.

Com este raciocínio, é possível visualizar que entre o resultado fornecido pela iteração atual e o resultado da iteração anterior, existirá um resíduo. A ideia principal é que este resíduo

reduza gradualmente a cada iteração, até atingir-se a convergência.

Considere o resíduo dado por

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}, \tag{3.12}$$

o qual quer-se reduzir até o critério de parada, no qual possa ser considerado nulo.

Para a correção do resíduo, toma-se uma determinada direção $\vec{\mathbf{d}}$ e corrige-se $\mathbf{x}^{(k)}$ nesta direção, de forma que

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha\vec{\mathbf{d}}, \tag{3.13}$$

e, portanto, $\mathbf{r}^{(k+1)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k+1)} < \mathbf{r}^{(k)}$.

Assim, o resíduo reduz-se a cada iteração convergindo para zero e a sequência $\mathbf{x}^{(k)}$ converge para a solução do sistema.

Para entender melhor a tomada de direção e a redução gradual deste resíduo, considera-se a função $f(\mathbf{x})$ originária do sistema $\mathbf{A}\mathbf{x} = \mathbf{b}$.

Sabe-se que o ponto de mínimo de $f(\mathbf{x})$ é a solução do sistema, pois este ponto é um ponto crítico, onde

$$f'(\mathbf{x}) = \nabla f(\mathbf{x}) = 0$$

$$\nabla f(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} \Rightarrow \mathbf{A}\mathbf{x} = \mathbf{b}$$

Desta forma, este ponto é tal que $\mathbf{A}\mathbf{x} = \mathbf{b}$. Além disso, sabe-se que o ponto é mínimo se $\mathbf{A}$ for uma matriz positiva definida.

Como dito anteriormente, para a possível aplicação do método, $\mathbf{A}$ é uma matriz positiva definida e, portanto, o ponto crítico é ponto de mínimo da função.

Com essa informação, sabe-se que em um determinado ponto $\mathbf{x}$, o gradiente $\nabla f(\mathbf{x})$ irá apontar para a direção de maior crescimento da função. Com isso, têm-se uma direção a tomar e resta determinar o quanto deve-se caminhar nesta direção. O coeficiente que dará essa medida, será o α, visto na equação (3.13). Para determiná-lo, considera-se então

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha\mathbf{r}^{(k)} \tag{3.14}$$

e sabe-se que o coeficiente α irá minimizar a função quando a derivada direcional $\dfrac{\partial f(\mathbf{x}^{(k+1)})}{\partial \alpha} = 0$.

A derivada direcional, por sua vez, pode ser escrita como

$$\frac{\partial f(\mathbf{x}^{(k+1)})}{\partial \alpha} = \left(f'(\mathbf{x}^{(k+1)})\right)^T \frac{\partial \mathbf{x}^{(k+1)}}{\partial \alpha} = \left(f'(\mathbf{x}^{(k+1)})\right)^T \mathbf{r}^{(k)} = 0 \tag{3.15}$$

Então, os termos $\mathbf{r}^{(k)}$ e $f'(\mathbf{x}^{(k+1)})$ são ortogonais, segundo a escolha de α. Para finalmente

determinar α, então, utiliza-se que $f'(\mathbf{x}^{(k+1)}) = -\mathbf{r}^{(k+1)}$. Assim, e relembrando as equações (3.12) e (3.14), determina-se α

$$(\mathbf{r}^{(k+1)})^T\mathbf{r}^{(k)} = 0$$

$$(\mathbf{b} - \mathbf{A}(\mathbf{x}^{(k)} + \alpha\mathbf{r}^{(k)}))^T\mathbf{r}^{(k)} = 0$$

$$(\mathbf{b} - \mathbf{A}\mathbf{x}^{(k)})^T\mathbf{r}^{(k)} = \alpha(\mathbf{A}\mathbf{r}^{(k)})^T\mathbf{r}^{(k)}$$

$$(\mathbf{r}^{(k)})^T\mathbf{r}^{(k)} = \alpha(\mathbf{r}^{(k)})^T\mathbf{A}\mathbf{r}^{(k)},$$

então

$$\alpha = \frac{(\mathbf{r}^{(k)})^T\mathbf{r}^{(k)}}{(\mathbf{r}^{(k)})^T\mathbf{A}\mathbf{r}^{(k)}}. \tag{3.16}$$

Este raciocínio construído até aqui pode ser visto com mais detalhes na literatura [13].

Agora, têm-se todos os termos necessários ao Método dos Gradientes. Entretanto, um inconveniente do MG é o fato de uma mesma direção $\vec{d}$ poder ser adotada novamente em uma dada iteração k, mesmo tendo sido adotada anteriormente em outra iteração.

Uma forma de evitar que a mesma direção seja adotada de forma repetida, é proposto, finalmente, pelo Método dos Gradientes Conjugados.

A proposta consiste em tomar-se um conjunto de direções conjugadas $(p_0, p_1, ..., p_{n-1})$ em até, no máximo, n iterações, para a aproximação inicial $\mathbf{x}^{(0)}$. Observe que o MGC convergirá em, no máximo, n iterações.

Para entender isso, relembre que dois vetores x e y são conjugados quando

$$\mathbf{x}^T\mathbf{A}\mathbf{y} = \mathbf{y}^T\mathbf{A}\mathbf{x} = 0. \tag{3.17}$$

Para a aproximação inicial $\mathbf{x}$ considera-se um vetor $\mathbf{p}^{(k)}$ de forma que

$$\mathbf{p}^{(0)} = \mathbf{r}^{(0)}, \tag{3.18}$$

e

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{p}^{(k)}. \tag{3.19}$$

Assim, o valor $\alpha^{(k)}$ será determinado de forma análoga à vista para o Método dos Gradientes, a seguir

$$\frac{d(f(\mathbf{x}^{(1)}))}{d\alpha} = f'(\mathbf{x}^{(1)})^T\frac{d\mathbf{x}^{(1)}}{d\alpha} = -\mathbf{r}^{(1)T}\mathbf{p}^{(0)}, \tag{3.20}$$

e para minimizar $f(\mathbf{x}^{(k+1)})$

$$\mathbf{r}^{(k+1)T}\mathbf{p}^{(k)} = 0 \rightarrow \mathbf{p}^{(k)T}\mathbf{r}^{(k+1)} = 0 \rightarrow \mathbf{p}^{(k)T}(\mathbf{b} - \mathbf{A}\mathbf{x}^{(k+1)}) = 0. \tag{3.21}$$

Substitui-se $\mathbf{x}^{(k+1)}$ como em (3.19) nessa expressão e determina-se $\alpha^{(k)}$

$$\alpha^{(k)} = \frac{\mathbf{r}^{(k)T}\mathbf{r}^{(k)}}{\mathbf{p}^{(k)T}\mathbf{A}\mathbf{p}^{(k)}} \tag{3.22}$$

Então, sabe-se o tamanho do passo α, mas ainda precisa-se que $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha^{(k)}\mathbf{A}\mathbf{p}^{(k)}$ seja ortogonal à uma nova direção adotada, dada por

$$\mathbf{p}^{(k+1)} = \mathbf{r}^{(k+1)} + \beta^{(k+1)}\mathbf{p}^{(k)} \tag{3.23}$$

Com isso, vê-se necessário determinar um valor para $\beta^{(k+1)}$ de forma conveniente.

Também análogo à estratégia anterior, faz-se

$$\mathbf{r}^{(k+1)T}\mathbf{p}^{(k+1)} = 0 \rightarrow \left(\mathbf{r}^{(k)} - \frac{\mathbf{r}^{(k)T}\mathbf{r}^{(k)}}{\mathbf{p}^{(k)T}\mathbf{A}\mathbf{p}^{(k)}}\mathbf{A}\mathbf{p}^{(k)}\right)^T (\mathbf{r}^{(k+1)} + \beta^{(k+1)}\mathbf{p}^{(k)}) = 0$$

$$\mathbf{r}^{(k)T}\mathbf{r}^{(k+1)} + \beta^{(k+1)}\mathbf{r}^{(k)T}\mathbf{p}^{(k)} - \mathbf{p}^{(k)T}\mathbf{A}\frac{\mathbf{r}^{(k)T}\mathbf{r}^{(k)}}{\mathbf{p}^{(k)T}\mathbf{A}\mathbf{p}^{(k)}}\mathbf{r}^{(k+1)} - \beta^{(k+1)}\mathbf{p}^{(k)T}\mathbf{A}\frac{\mathbf{r}^{(k)T}\mathbf{r}^{(k)}}{\mathbf{p}^{(k)T}\mathbf{A}\mathbf{p}^{(k)}}\mathbf{p}^{(k)} = 0.$$

Simplicando a expressão de forma a obter-se $\beta^{(k+1)}$, têm-se

$$\beta^{(k+1)} = -\frac{\mathbf{p}^{(k)T}\mathbf{A}\mathbf{r}^{(k+1)}}{\mathbf{p}^{(k)T}\mathbf{A}\mathbf{p}^{(k)}} = -\frac{\mathbf{r}^{(k+1)T}\mathbf{A}\mathbf{p}^{(k)}}{\mathbf{p}^{(k)T}\mathbf{A}\mathbf{p}^{(k)}}. \tag{3.24}$$

Com essa equação (3.24) e sabendo que $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha^{(k)}\mathbf{A}\mathbf{p}^{(k)}$, simplifica-se ainda mais $\beta^{(k+1)}$ de forma conveniente. Para isso, inicialmente isolam-se termos de interesse como a seguir

$$\mathbf{A}\mathbf{p}^{(k)} = -\frac{1}{\alpha^{(k)}}(\mathbf{r}^{(k+1)} - \mathbf{r}^{(k)}), \tag{3.25}$$

$$\mathbf{r}^{(k+1)T}\mathbf{A}\mathbf{p}^{(k)} = -\frac{1}{\alpha^{(k)}}(\mathbf{r}^{(k+1)T}\mathbf{r}^{(k+1)} - \mathbf{r}^{(k+1)T}\mathbf{r}^{(k)}) = -\frac{1}{\alpha^{(k)}}(\mathbf{r}^{(k+1)T}\mathbf{r}^{(k+1)}), \tag{3.26}$$

e

$$\mathbf{p}^{(k)T}\mathbf{A}\mathbf{p}^{(k)} = -\frac{1}{\alpha^{(k)}}(\mathbf{p}^{(k)T}\mathbf{r}^{(k+1)} - \mathbf{p}^{(k)T}\mathbf{r}^{(k)}) = -\frac{1}{\alpha^{(k)}}(-\mathbf{p}^{(k)T}\mathbf{r}^{(k)}) = \frac{1}{\alpha^{(k)}}\mathbf{p}^{(k)T}\mathbf{r}^{(k)}. \tag{3.27}$$

Por fim, substitui-se (3.26) e (3.27) em (3.24), obtendo uma melhor simplificação para $\beta^{(k+1)}$

$$\beta^{(k+1)} = \frac{\mathbf{r}^{(k+1)T}\mathbf{r}^{(k+1)}}{\mathbf{r}^{(k)T}\mathbf{r}^{(k)}} \tag{3.28}$$

Com este resultado em mãos, portanto, têm-se o tamanho do passo β necessário para a

definição da nova direção $\mathbf{p}$.

Essa estratégia como é proposta, fornece a convergência do método de forma mais eficiente do que os demais, pois como visto, caminha-se na direção do gradiente de forma a reduzir o resíduo a cada iteração, sem tomar-se direções repetidas entre as iterações. Mais informações podem ser vistas nas referências [13], [7] e [11].

3.2 ALGORITMO COMPUTACIONAL

Conhecendo o Método do Gradiente Conjugado, torna-se possível agora a criação do código em linguagem C referente ao método, para resolver o sistema linear resultante da discretização de uma EDP elíptica via métodos de diferenças finitas (MDF).

O algoritmo a seguir mostra, resumidamente, a estratégia a ser implementada [16]:

(i) inicialmente, determina-se o vetor resíduo inicial $\mathbf{r}_0$, a direção inicial $\mathbf{p}_0$ e o número inicial de iterações k

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

$\mathbf{p}_0 = \mathbf{r}_0$

$k = 0$

(ii) com isso, pode-se calcular o passo α_k

$$\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A}\mathbf{p}_k}$$

(iii) e, com este resultado, pode-se calcular $\mathbf{x}_{k+1}$ e o resíduo $\mathbf{r}_{k+1}$

$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$

$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$

aqui, é conferido se este valor $\mathbf{r}_{k+1}$ é menor do que a tolerância adotada (neste trabalho, de 1×10^{-10}); se for, o algoritmo pode ser finalizado neste ponto, retornando o resultado $\mathbf{x}_{k+1}$ como solução,

(iv) caso contrário, faz o cálculo de β_{k+1} e da nova direção $\mathbf{p}_{k+1}$, e atualiza o contador da iteração

$$\beta_{k+1} = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$$

$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_{k+1} \mathbf{p}_k$

$k = k + 1$

(v) repete-se os itens de (ii) a (v), quantas iterações forem necessárias até atingir-se a convergência e até no máximo n (tamanho do sistema linear) iterações.

Todo o código desenvolvido neste trabalho poderá ser encontrado no apêndice A.

4. Verificação do código

Para validar o algoritmo implementado, utilizou-se um problema de Poisson com solução analítica conhecida. A função $u(x, y)$ utilizada para validação foi a seguinte

$$u(x, y) = sen(x)cos(y). \tag{4.1}$$

A equação (4.1) possui forma conhecida e limites superior e inferior conhecidos, sendo limitada entre -1 e 1. Derivando parcialmente duas vezes com relação a x e a y, obtém-se o termo fonte de $\nabla^2 u$, como

$$\nabla^2 u(x, y) = f(x, y) = \frac{\partial u^2}{\partial x^2} + \frac{\partial u^2}{\partial y^2}. \tag{4.2}$$

Aplicando (4.2) em (4.1), têm-se o termo fonte

$$f(x, y) = -2sen(x)cos(y). \tag{4.3}$$

Além disso, determina-se as condições de contorno para as fronteiras do problema, como condições de Dirichlet da forma de $u(x, y) = sen(x)cos(y)$.

A geometria utilizada para essa validação será uma região quadrada de dimensões $[0, L] \times [0, L]$, onde $L = 2\pi$. As quantidades de pontos na malha computacional para cada direção x e y, sendo Nx e Ny, respectivamente, também são conhecidas.

Com estes dados, torna-se possível conhecer os espaçamentos Δx e Δy da malha, dados por

$$\Delta x = \frac{L}{Nx - 1} \tag{4.4}$$

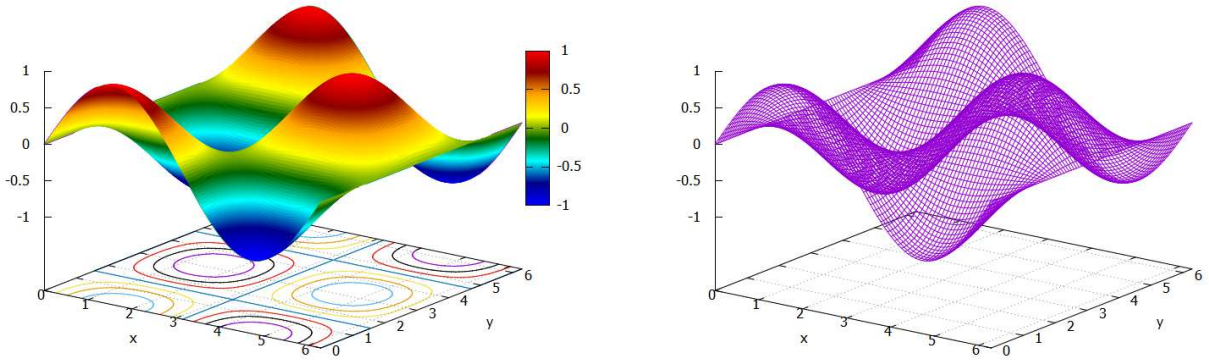
e

$$\Delta y = \frac{L}{Ny - 1}. \tag{4.5}$$

Serão testadas 5 malhas com refinamentos diferentes, $Nx = Ny = 21, 41, 81, 161$ e 321, e o resultado numérico obtido será comparado com o resultado analítico deste problema. Serão analisados os erros e a ordem de convergência da aproximação utilizada.

A Figura 4.1 exibe uma comparação entre as soluções numérica e analítica, com $Nx = Ny = 41$. Observa-se que as curvas geradas são visualmente idênticas e, a seguir, são apresentados os dados referentes aos erros e à ordem de convergência.

Quanto à ordem, devido a aproximação pela fórmula de 5 pontos, espera-se uma ordem de



(a) (b)

Figura 4.1: Perfis (a) numérico e (b) analítico de $u(x, y) = sen(x)cos(y)$, com $Nx = Ny = 41$. [fonte: autora]

convergência igual a 2. Para a obtenção da ordem considera-se

$$O = \left| \frac{log\left(\frac{Erro_{\frac{\Delta x}{2}}}{Erro_{\Delta x}}\right)}{log(2)} \right|, \tag{4.6}$$

onde

$$Erro_{\Delta x} = \frac{||u_{aprox} - u_{exato}||}{||u_{exato}||}, \tag{4.7}$$

é o erro relativo em uma malha com espaçamento Δx, u_{aprox} é a solução numérica, e u_{exato} é a solução analítica.

Estes resultados são obtidos com a escolha e uso de uma determinada norma e, aqui, serão utilizadas duas: a norma infinita e a norma 2 ou euclidiana. Então, para cada uma destas, a equação (4.7) torna-se, respectivamente

$$||Erro||_\infty = \frac{||u_{aprox} - u_{exato}||_\infty}{||u_{exato}||_\infty}, \tag{4.8}$$

onde

$$||u||_\infty = \max_{1 \leq i \leq n} |u_i|, \tag{4.9}$$

e

$$||Erro||_2 = \frac{||u_{aprox} - u_{exato}||_2}{||u_{exato}||_2}, \tag{4.10}$$

onde

$$||u||_2 = \sqrt{\sum_{i=1}^{n} u_i^2}. \tag{4.11}$$

Assim, pode-se calcular a ordem de convergência da aproximação, através da equação (4.6), com as normas infinita e euclidiana. Os resultados são apresentados na Tabela 4.1.

Tabela 4.1: Resultados obtidos para o erro e a ordem de convergência referentes à validação.

Malha	Nx	Ny	$\|\|Erro\|\|_\infty$	$\|\|Erro\|\|_2$	O_∞	O_2
1	21	21	8.9968×10^{-3}	6.8821×10^{-3}	—	—
2	41	41	2.2374×10^{-3}	1.7114×10^{-3}	2.0076	2.0077
3	81	81	5.5863×10^{-4}	4.2728×10^{-4}	2.0019	2.0019
4	161	161	1.3961×10^{-4}	1.0679×10^{-4}	2.0005	2.0005
5	321	321	3.4900×10^{-5}	2.6694×10^{-5}	2.0001	2.0002

Os erros obtidos são pequenos e a ordem de convergência do método foi próxima de 2, valor esperado para a ordem dessa aproximação.

Uma vez realizada a verificação, torna-se possível expandir a aplicação do código desenvolvido para outros problemas de interesse, com diferentes funções e geometrias. Isso será trabalhado no próximo capítulo.

5. Aplicações

Considere um escoamento laminar totalmente desenvolvido em regime permanente, através de um duto de geometria conhecida e comprimento longitudinal L na direção z. O escoamento é laminar ao possuir pouca agitação entre as camadas (lâminas) do fluido, e é totalmente desenvolvido na região onde todo campo de velocidade é invariante ao longo de z, apresentando o perfil de velocidade de interesse deste trabalho [6][10]. A Figura 5.1 exibe as regiões do escoamento em desenvolvimento e totalmente desenvolvido. Sob as hipóteses consideradas, a velocidade na direção z é uma função $w = w(x, y)$.

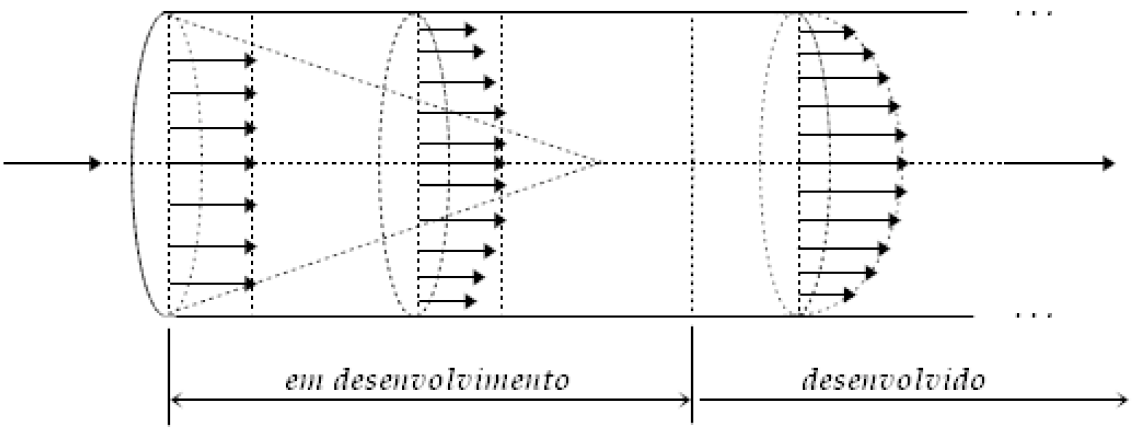


Figura 5.1: Regiões do escoamento em desenvolvimento e totalmente desenvolvido. [fonte: autora]

Em escoamentos totalmente desenvolvidos, a pressão é constante em toda seção transversal do duto e varia linearmente ao longo do comprimento. Nesta região, um problema 3D como o proposto neste trabalho, portanto, pode ser reduzido a um problema 2D [10].

Considere a equação de quantidade de movimento na direção z

$$\rho\left(\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z}\right) = \rho g_z + \mu\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) - \frac{\partial p}{\partial z}, \tag{5.1}$$

onde ρ é a densidade do fluido, g_z é a componente gravitacional em z e μ é a viscosidade dinâmica do fluido [6]. Por hipótese, as componentes de velocidade nas direções x e y são

nulas, ou seja, $u = 0$ e $v = 0$, $\dfrac{\partial w}{\partial z} = 0$ e considerando $g_z = 0$, a equação (5.1) será resumida em

$$\mu \left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} \right) - \frac{\partial p}{\partial z} = 0. \tag{5.2}$$

Assumindo que a dimensão da seção transversal é D $[m]$, as variáveis adimensionais podem ser escritas como

$$X = \frac{x}{D},$$

$$Y = \frac{y}{D},$$

e

$$W = w \frac{\mu}{D^2(-dp/dz)}.$$

Assim, a equação (5.2) se desenvolve da seguinte forma

$$-\nabla^2 W = -\left(\frac{\partial^2 W}{\partial X^2} + \frac{\partial^2 W}{\partial Y^2} \right) = 1. \tag{5.3}$$

Essa é a equação principal a ser resolvida, assim o problema se reduz a resolver uma equação de Poisson. Vale ressaltar que diferentes perfis de velocidade podem ser obtidos considerando valores diferentes de viscosidade e gradiente de pressão. O sinal do gradiente de pressão é responsável por alterar a orientação do escoamento.

Define-se as condições de contorno para a fronteira como

$$W|_{\delta\Omega} = 0, \tag{5.4}$$

onde Ω é a região de interesse e $\delta\Omega$ representa a fronteira desta região. Tal condição dita que na parede do duto há uma característica de não deslizamento do fluido e, por este motivo, a velocidade na fronteira é nula.

Com estas informações, pode-se obter o perfil de velocidade do escoamento através da seção transversal escolhida, e, também, pode-se obter outras informações sobre o escoamento, como a vazão do fluido, o coeficiente de atrito α e a tensão cisalhante τ_ω.

A vazão Q é calculada da seguinte forma

$$Q = \int_\Omega -W \, d\Omega = -W_{mdio}\Omega, \tag{5.5}$$

onde Ω é a área da região. O coeficiente de atrito, por sua vez, é dado por

$$fRe = \alpha, \tag{5.6}$$

onde Re é o número de Reynolds e f é o fator de atrito. Esta relação é adimensional e utilizada para informar a perda de carga devido ao atrito, e se aplica a escoamentos laminares como o

estudado neste trabalho [10].

O fator de atrito f é dado pela equação

$$f = \frac{\Delta p}{L}\frac{2}{\rho U^2}D_h, \tag{5.7}$$

em que Δp é a diferença entre as pressões de entrada e saída, U é a velocidade média e D_h é chamado diâmetro hidráulico, muito utilizado em questões de escoamentos em dutos. Este parâmetro é dado por

$$D_h = \frac{4|\Omega|}{Pm}, \tag{5.8}$$

onde Pm é o perímetro molhado, isto é, o perímetro da região pela qual passa o escoamento.

A velocidade média é dada pela razão entre a vazão e a área

$$U = \frac{Q}{|\Omega|}. \tag{5.9}$$

Por fim, o número de Reynolds pode ser obtido através da equação

$$Re = \frac{UD_h\rho}{\mu}. \tag{5.10}$$

Aplicando as equações (5.7) e (5.10) em (5.6), têm-se

$$fRe = \frac{\Delta p}{L}\frac{2D_h^2}{U\mu}.$$

Substituindo U como dado em (5.9), obtêm-se a equação do coeficiente de atrito de forma mais simplificada como a seguir

$$fRe = \frac{2D_h^2|\Omega|}{\mu Q}\frac{\Delta p}{L}. \tag{5.11}$$

Já a tensão cisalhante τ_ω é a força gerada pela fronteira e aplicada sobre o escoamento em direção oposta a este, e é dada por

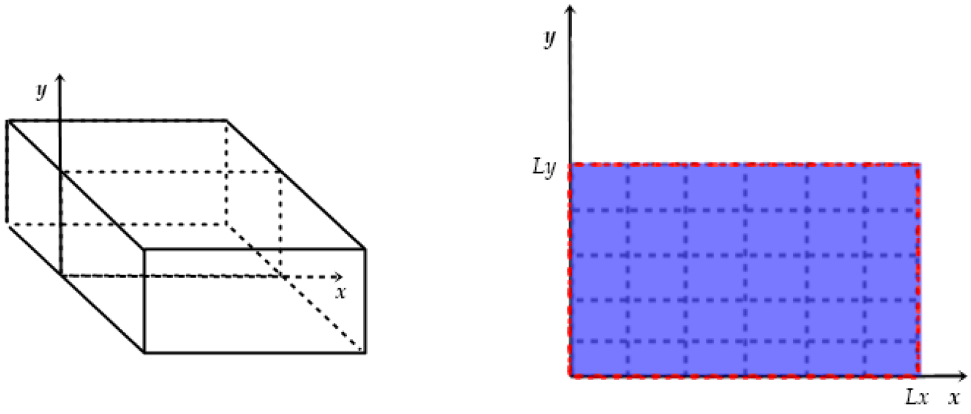
$$\tau_\omega = \mu\frac{\partial\omega}{\partial\mathbf{n}}, \tag{5.12}$$

onde $\mathbf{n}$ é o vetor normal exterior à fronteira. Assim torna-se possível obter o perfil da tensão de cisalhamento na seção transversal do duto.

A seguir será apresentado um estudo numérico sobre escoamentos em dutos com três formatos de seção transversal. São representados os perfis de velocidade obtidos para cada caso, bem como os perfis da tensão de cisalhamento, a vazão e o fator de atrito gerado para cada geometria. Em todo o estudo numérico, foi considerado $\mu = 1$ e $\dfrac{dp}{dz} = 1$.

5.1 GEOMETRIA DE SEÇÃO RETANGULAR

A geometria adotada aqui é de seção transversal retangular 2x1, com comprimento $Lx = 2$ e $Ly = 1$, como mostrado na Figura 5.2.



(a) (b)

Figura 5.2: (a) Geometria do tubo e (b) seção transversal retangular da geometria utilizada. [fonte: autora]

Com as equações (5.5) e (5.6), pode-se calcular a vazão e o fator de atrito. Aqui serão observados resultados para três malhas, todas regulares, com $Nx = Ny = 41, 81$ e 161. Os resultados são apresentados na Tabela 5.1.

Tabela 5.1: Resultados de vazão e coeficiente de atrito referentes à geometria retangular.

Nx	Ny	Vazão (Q)	fRe
41	41	0.11408	62.33215
81	81	0.11428	62.22723
161	161	0.11432	62.20098

O perfil da velocidade W, com $Nx = 81$, é exibido na Figura 5.3.

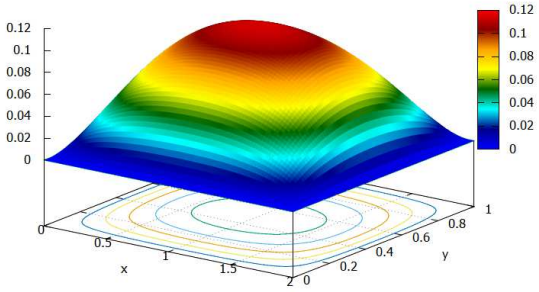


Figura 5.3: Perfil de velocidade do escoamento em duto de seção retangular, com $Nx = Ny = 81$.
[fonte: autora]

Com a equação (5.12), obtém-se o gráfico da tensão de cisalhamento aplicada sobre o escoamento, exibido na Figura 5.4.

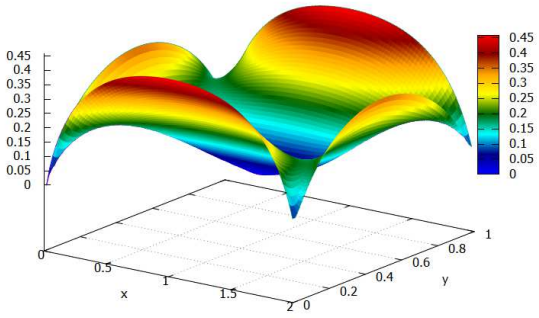
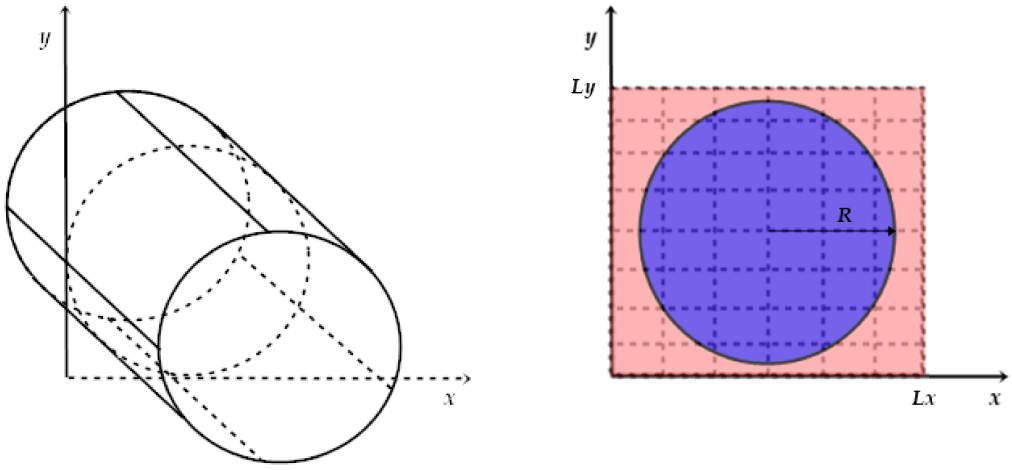


Figura 5.4: Tensão de cisalhamento para escoamento em duto de seção retangular, com $Nx = Ny = 81$.
[fonte: autora]

5.2 GEOMETRIA DE SEÇÃO CIRCULAR

As características do escoamento proposto em um duto de seção transversal circular constante são conhecidas pela literatura [17]. As equações para obter o perfil teórico de velocidade, o valor da vazão esperado e o valor para o coeficiente de atrito teórico são conhecidas e estes valores podem ser calculados.

A geometria escolhida aqui é um duto longo de seção transversal circular constante com raio $R = 0.5$. O esboço da geometria utilizada é apresentado na Figura 5.5.



(a) (b)

Figura 5.5: (a) Geometria do tubo e (b) seção transversal circular da geometria utilizada. [fonte: autora]

A equação de Poiseuille descreve o perfil de velocidade para um duto de seção transversal circular, que pode ser encontrada em [17], dada por

$$u(r) = \frac{R^2 - r^2}{4}, \tag{5.13}$$

onde r é a coordenada de interesse.

Para o problema em questão, o domínio de cálculo é mantido quadrado e em coordenadas cartesianas, entretanto, as células de interesse, interiores ao círculo, são indicadas como células válidas, enquanto as células exteriores ao círculo são indicadas como células de fronteira. Na Figura 5.5(b) é possível ver essa estratégia, na qual a região azul é a região válida para cálculo e a região vermelha é dada como fronteira.

Para o cálculo, r será definido por

$$r = \sqrt{\left(x - \frac{Lx}{2}\right)^2 + \left(y - \frac{Ly}{2}\right)^2}. \tag{5.14}$$

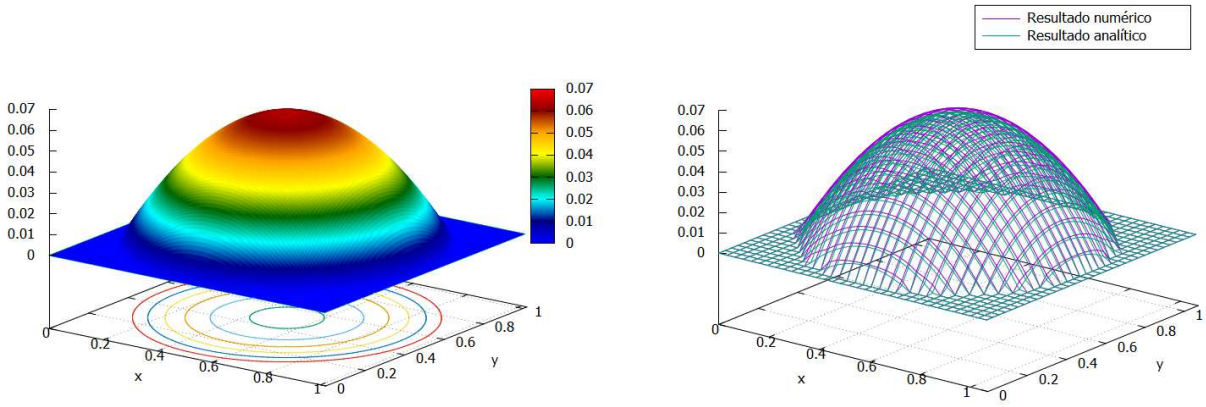
A vazão também tem forma conhecida para o escoamento laminar totalmente desenvolvido em duto cilíndrico, dada por

$$Q = \pi R^2 \frac{u_{max}}{2}, \tag{5.15}$$

onde u_{max} é o valor máximo de velocidade obtido.

Na Figura 5.6(a) é exibido o perfil da velocidade Wobtido numericamente, já na Figura 5.6(b), é apresentada uma comparação entre o resultado numérico e o resultado analítico. Nota-se que o resultado obtido numericamente foi fiel ao esperado e que o perfil de velocidade para este escoamento é uma superfície parabólica, como esperado para o escoamento de Poiseuille.

A tensão de cisalhamento exercida sobre o escoamento pode ser vista na Figura 5.7 e também possui o formato esperado. Além disso, pode-se comparar a vazão e o coeficiente de atrito



(a) (b)

Figura 5.6: Perfil de velocidade do escoamento em duto de seção circular, com $Nx = Ny = 128$.
[fonte: autora]

numérico com o analítico.

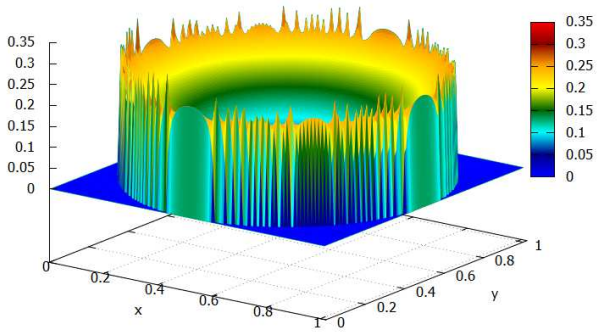


Figura 5.7: Tensão de cisalhamento para escoamento em duto de seção circular, com $Nx = Ny = 128$.
[fonte: autora]

Na Tabela 5.2, foram organizados resultados para três malhas regulares, sendo estas com $Nx = Ny = 32, 64$ e 128. Os resultados são para a vazão e o coeficiente de atrito numéricos, bem como os erros obtidos em comparação ao resultado analítico.

Observa-se que os erros referentes ao cálculo da velocidade do escoamento são pequenos e, de modo geral, os erros diminuem consideravelmente com o refinamento da malha.

O coeficiente de atrito também é conhecido e pode ser obtido com a equação (5.6). Relembrando, esse coeficiente é dependente da área da seção transversal, da vazão e do diâmetro hidráulico. Mas este é dependente do perímetro molhado, que será diferente para cada geometria.

Tabela 5.2: Resultados obtidos referentes à geometria de seção transversal circular.

Nx	Ny	Vazão (Q)	fRe	Erro (Q)	Erro (u)
32	32	0.02601	60.3926	0.0599	3.7800×10^{-2}
64	64	0.02539	61.8655	0.0344	1.7260×10^{-2}
128	128	0.02503	62.7505	0.0198	9.9429×10^{-3}

Para um duto cilíndrico, a área da seção transversal Ω será

$$\Omega = \pi R^2, \tag{5.16}$$

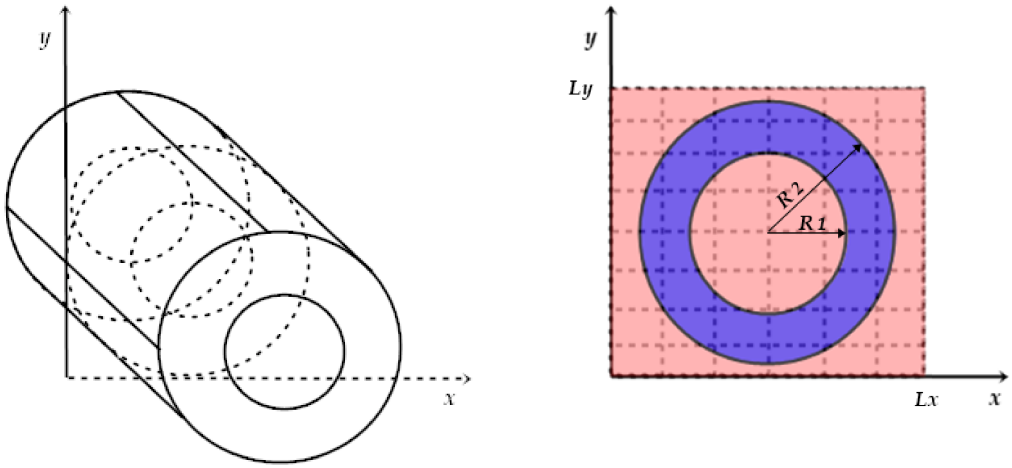
e o perímetro molhado será

$$Pm = 2\pi R. \tag{5.17}$$

Assim, o valor analítico de $fRe = 64$. Apesar da boa concordância entre os resultados teóricos e numéricos, parte do erro na estimativa se deve à própria forma da seção transversal, que já carrega consigo algum erro em decorrência do sistema de coordenadas utilizado, resultando em uma malha computacional retangular.

5.3 GEOMETRIA DE SEÇÃO ANULAR

Semelhante à resolução para a geometria de seção transversal circular, faz-se agora a resolução para um escoamento laminar totalmente desenvolvido escoando em um duto longo de seção transversal anular. A coroa circular terá raio externo $R_2 = 0.5$ e raio interno $R_1 = 0.15$. A Figura 5.8 mostra a geometria utilizada.



(a) (b)

Figura 5.8: (a) Geometria do tubo e (b) seção transversal anular da geometria utilizada. [fonte: autora]

A equação de Poiseuille para este caso também fornece o perfil de velocidade [17], dado por

$$u(r) = \frac{G}{4\mu}(R_1^2 - r^2) + \frac{G}{4\mu}(R_2^2 - R_1^2)\frac{ln\left(\frac{r}{R_1}\right)}{ln\left(\frac{R_2}{R_1}\right)},$$

onde μ é a viscosidade dinâmica do fluido, considerada como $\mu = 1$, e G é dado por

$$G = \frac{-dp}{dz},$$

sendo $G = 1$.

Com estas considerações, a equação de velocidade fica na forma

$$u(r) = \frac{1}{4}(R_1^2 - r^2) + \frac{1}{4}(R_2^2 - R_1^2)\frac{ln\left(\frac{r}{R_1}\right)}{ln\left(\frac{R_2}{R_1}\right)}. \tag{5.18}$$

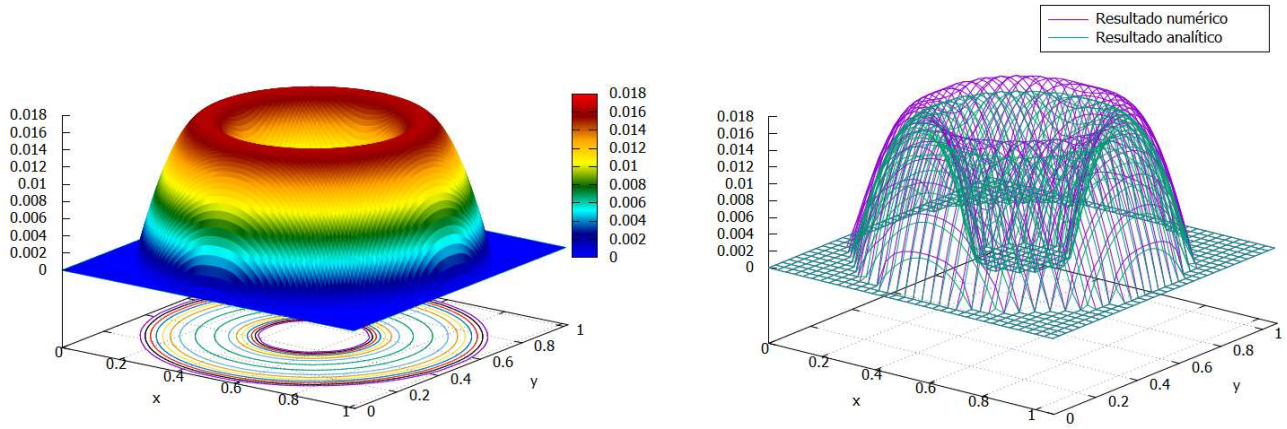
A vazão também é conhecida, dada pela seguinte equação

$$Q = \frac{G\pi}{8\mu}\left[R_2^4 - R_1^4 - \frac{(R_2^2 - R_1^2)^2}{ln\left(\frac{R_2}{R_1}\right)}\right], \tag{5.19}$$

em que $\mu = 1$ e $G = 1$,

$$Q = \frac{1\pi}{8}\left[R_2^4 - R_1^4 - \frac{(R_2^2 - R_1^2)^2}{ln\left(\frac{R_2}{R_1}\right)}\right]. \tag{5.20}$$

O perfil de velocidade é estimado resolvendo a equação 5.3, podendo ser observado na Figura 5.9(a), e comparado com o perfil de velocidade teórico, como apresentado na Figura 5.9(b). O resultado ainda manteve-se fiel ao resultado analítico, mas é possível observar uma maior diferença entre os perfis numérico e teórico, quando comparado ao obtido para a seção transversal circular. Entre os raios externo e interno, na região em que ocorre o escoamento, observa-se um perfil próximo ao parabólico como esperado.



(a) (b)

Figura 5.9: Perfil de velocidade do escoamento em duto de seção anular, com $Nx = Ny = 128$.
[fonte: autora]

A tensão de cisalhamento foi obtida e seu perfil pode ser visto na Figura 5.10. Note que a tensão de cisalhamento agora aparece próxima às paredes externa e interna, sendo exercida por elas sobre o escoamento.

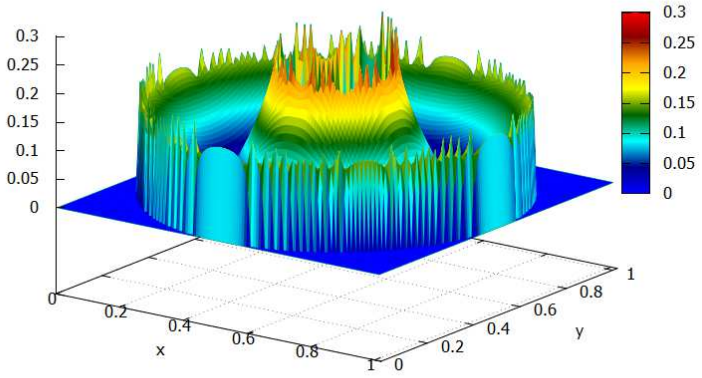


Figura 5.10: Tensão de cisalhamento para escoamento em duto de seção anular, com $Nx = Ny = 128$.
[fonte: autora]

Análogo à análise para o escoamento em um duto de seção transversal circular, foram observadas três malhas regulares, sendo $Nx = Ny = 32, 64$ e 128. Os resultados obtidos numericamente para a vazão e coeficiente de atrito são mostrados na Tabela 5.3, bem como os erros obtidos com relação à vazão teórica e aos valores de velocidade teóricos.

Tabela 5.3: Resultados referentes à geometria de seção transversal anular.

Nx	Ny	Vazão (Q)	fRe	Erro (Q)	Erro (u)
32	32	8.6531×10^{-3}	94.9966	0.1537	1.1088×10^{-1}
64	64	8.2082×10^{-3}	100.1466	0.0944	7.3403×10^{-2}
128	128	7.7881×10^{-3}	105.5475	0.0384	3.0037×10^{-2}

Os erros reduzem de forma considerável com o aumento no número de pontos da malha. Para a velocidade, os erros obtidos foram maiores do que aqueles obtidos para a seção transversal circular, o que é esperado uma vez em que a seção anular é mais complexa do que a seção circular, reforçando que a geometria complexa em coordenadas cartesianas acarretará em um determinado acúmulo de erro nos cálculos.

Ainda de forma análoga ao tópico anterior, faz-se o cálculo do coeficiente de atrito fRe, e, para isso, a área e o perímetro molhado serão determinados.

A área para esta geometria é dada por

$$\Omega = \pi(R_2^2 - R_1^2), \tag{5.21}$$

e o perímetro molhado é dado por

$$Pm = 8\pi R_1. \tag{5.22}$$

Com a equação do cálculo do coeficiente de atrito (5.6) determina-se $fRe = 109$. Novamente observa-se que, com o refinamento da malha, o valor deste coeficiente aproxima-se do valor teórico. Mas nesse caso, a discrepância foi ainda um pouco maior, devido à maior complexidade da geometria e acúmulo de erros nos resultados obtidos para a velocidade, vazão, área transversal e diâmetro hidráulico.

6. Considerações

Os resultados numéricos, apesar de próximos do teórico, mostraram um acúmulo de erros por diversos motivos. Como observado, estes erros são mais evidentes nas geometrias de área transversal de formato circular. Isso se deve ao sistema de coordenadas cartesiano, que irá representar a geometria circular apenas de forma aproximada.

Na Figura 6.1 pode ser observada a maneira que foi representada a geometria circular, tomando como exemplo a seção transversal anular. Os pontos em vermelho, como já é sabido, são considerados pontos de fronteira e não precisam ser calculados, neles estão impostas as condições de contorno. Os pontos em azul são os pontos de interesse que serão calculados, chamados de pontos válidos.

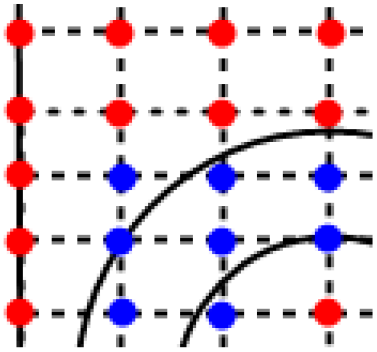


Figura 6.1: Representação do domínio de cálculo para figuras geométricas não retangulares. [fonte: autora]

Uma vez em que a coordenada cartesiana funcionará com base nos índices das direções horizontal e vertical, separados pelos espaçamentos Δx e Δy, ela não conseguirá representar a geometria perfeitamente. A geometria, mesmo que se aproxime da geometria circular, ainda terá o aspecto quadriculado das células.

Quanto mais refinada a malha, menores serão os espaçamentos e a região mais aproximada da geometria real. Então, com uma malha mais refinada, o erro referente à essa questão será menor, como pode ser observado nos resultados.

Uma boa forma de corrigir este problema é alterar o sistema de coordenadas utilizado, de coordenadas cartesianas para coordenadas polares. Este sistema consegue aproximar-se bem de figuras geométricas circulares, pois não se baseia apenas nas direções retas horizontal e vertical, mas em uma coordenada linear r e na rotação de um ângulo θ, conseguindo varrer toda a área de uma região circular. A Figura 6.2 mostra como um ponto P qualquer, pertencente à mesma geometria de seção transversal anular, é representado em coordenadas polares.

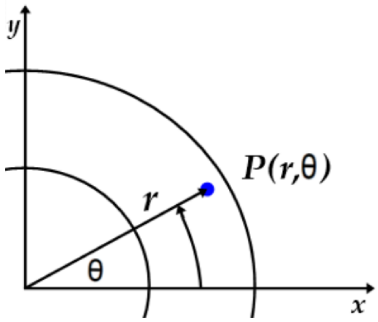


Figura 6.2: Representação de um ponto em coordenadas polares.
[fonte: autora]

Isto é, em coordenadas cartesianas têm-se um ponto $P(x, y)$, enquanto em coordenadas polares este ponto tem a forma $P(r, \theta)$ e o problema pode ser melhorado fazendo uma alteração de coordenadas cartesianas para polares, para a resolução de problemas com área transversal circular [7]. Com uma noção básica de trigonometria, sabe-se que

$$x = rcos(\theta),$$

$$y = rsen(\theta),$$

e

$$r = \sqrt{x^2 + y^2}.$$

Com isso, fazendo a relação entre os sistemas de coordenadas, pode-se reescrever as equações relativas aos escoamentos em dutos de seção transversal circular e anular. Essa estratégia melhora a estimativa resultados, no entanto, a matriz do sistema linear resultante da discretização da equação de Poisson perde a simetria.

Outro tópico que pode ser trabalhado para a obtenção de resultados melhores, é o uso de pré-condicionadores no MGC [11].

O uso de pré-condicionadores reduz consideravelmente o número de iterações do MGC e, consequentemente, o tempo de processamento.

7. Conclusões

Neste trabalho, foram apresentadas as principais classes de equações diferenciais parciais, responsáveis pela modelagem de diversos problemas físicos encontrados na engenharia, e ver que cada classe possui características a serem consideradas na escolha do método de discretização e método numérico a serem utilizados. No contexto das EDPs elípticas, a discretização resultará em um sistema linear esparso a ser resolvido.

Os métodos iterativos são comumente aplicados a sistemas lineares grandes com matriz dos coeficientes esparsa. Dentre esses métodos, destaca-se o Método dos Gradientes Conjugados (MGC), enfatizado e utilizado neste estudo. Foi visto que o MGC é capaz de fornecer uma boa aproximação para a solução do sistema, com um número máximo de iterações igual ao número de equações do sistema e com boa velocidade de convergência.

Toda formulação foi implementada em linguagem C. Uma validação computacional foi realizada, mostrando que a ordem numérica está de acordo com a ordem teórica.

Considerando um conjunto de hipóteses para escoamentos em dutos, as equações de Navier-Stokes foram simplificadas em uma equação de Poisson, para prover o perfil de velocidade em uma seção transversal. Foram investigadas numericamente três seções transversais, em que os resultados obtidos apresentaram boa concordância com os da literatura. Essa foi mais uma evidência da correta implementação computacional.

Não foi mencionado, mas o código permite resolver a equação de Poisson com qualquer condição de contorno (Dirichlet, Neumann ou Robin).

Por fim, foi discutida a limitação da formulação em resolver problemas com domínios complexos.

Referências Bibliográficas

[1] Arnold, D. N.: *Lectures notes on Numerical Analysis of Partial Differential Equations*. https://www-users.cse.umn.edu/~arnold/8445-8446.14-15/notes.pdf, notas disponibilizadas online, 2014-2015.

[2] Boyce, W. E. e Diprima, R. C.: *Equações Diferenciais Elementares e Problemas de Valores de Contorno*. LTC, 10ª ed., 2015.

[3] Burden, R. L. e Faires, J. D.: *Numerical Analysis*. Cengage, 9ª ed., 2010.

[4] Falk, R. S.: *Lecture 1: Finite Difference Methods for Elliptic Problems*. https://sites.math.rutgers.edu/~falk/math575/lecture-notes17.html, notas disponibilizadas online, 2017.

[5] Falk, R. S.: *Lecture 2: Stability and error estimates*. https://sites.math.rutgers.edu/~falk/math575/lecture-notes17.html, notas disponibilizadas online, 2017.

[6] Fox, R. W., McDonald, A. T. e Mitchell, J. W.: *Fox and McDonald's Introduction to Fluid Mechanics*. Wiley, 10ª ed., 2020.

[7] Franco, N. M. B.: *Cálculo Numérico*. Pearson, 1ª ed., 2006.

[8] LeVeque, R. J.: *Finite Difference Methods for Ordinary and Partial Differential Equations*. SIAM, 1ª ed., 2007.

[9] O. Fortuna, A. de: *Técnicas Computacionais para Dinâmica dos Fluidos*. Edusp, 1ª ed., 2000.

[10] Patankar, S. V.: *Numerical Heat Transfer and Fluid Flow*. CRC Press, 1ª ed., 1980.

[11] Saad, Y.: *Iterative Methods for Sparse Linear Systems*. SIAM, 2ª ed., 2003.

[12] Santana, A. A. e Figueiredo, R. A.: *Notas de Cálculo Numérico - Métodos Iterativos*. notas de aula utilizados na disciplina Cálculo Numérico, 2017.

[13] Shewchuk, J. R.: *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213, 1 1/4 ed., 1994.

[14] Smith, G. D.: *Numerical Solution of Partial Differential Equations: Finite Difference Methods*. Clarendon Press - Oxford, 3ª ed., 1985.

[15] Wikipedia: *Condition Number*, 2021. https://en.wikipedia.org/wiki/Condition_number, acessado em 20/07/2021.

[16] Wikipedia: *Conjugate Gradient Method*, 2021. https://en.wikipedia.org/wiki/Conjugate_gradient_method, acessado em 10/06/2021.

[17] Wikipedia: *Hagen–Poiseuille equation*, 2021. https://en.wikipedia.org/wiki/HagenE28093Poiseuille_equation#Poiseuille_flow_through_some_non-circular_cross-sections, acessado em 12/06/2021.

A. Apêndice

```
//---------------------------------------------------------------------
/*
 Universidade Federal de Uberlandia
 Faculdade de Matematica
 Disciplina: Trabalho de Conclusao de Curso
 Data: 29/04/2021
 Autor: Rafael Figueiredo e Daniella Faria

 Resolve a equacao de Poisson 2D:
 p_xx + p_yy = f(x,y)
 em um dominio [0,Lx] x [0, Ly]
 usando o metodo de diferencas finitas
 e Gradientes Conjugados
 */
//---------------------------------------------------------------------
// Bibliotecas do sistema

#include<stdio.h>
#include<stdlib.h>
#include<math.h>

//---------------------------------------------------------------------
// Macros

#define PI acos(-1.0)
#define POINT_INVALID 0
#define POINT_VALID 1
#define POINT_DIRICHLET 2 // p = cte
#define POINT_NEUMANN 4    // dp/dx = 0 || dp/dy = 0

//---------------------------------------------------------------------
typedef struct {
  int *type; // Tipo de ponto
  int *map;
```

```
35    double *p;
36  } sv_t; // vetor de estado (state vector)
37
38  typedef struct {
39    int Nx;
40    int Ny;
41    int caseMask;
42    double Lx;
43    double Ly;
44    double Rad;
45    double rad;
46  } params_t;
47
48  params_t *params; // informacao global
49
50  //---------------------------------------------------------------------
51  // Declaracao das funcoes utilizadas
52
53  double *ConjugateGradient(double **A, double *b, double *x, int n);
54  double ip(double *u, double *v, int n);
55  double *matVec(double *dest, double **A, double *x, int size);
56  double *xpay(double *dest, double *x, double alpha, double *y, int
        size);
57  void freeMatrix(int n, double **A);
58  double **allocMatrix(int n);
59  double *allocVector(int n);
60  sv_t *sv_alloc();
61  void sv_free(sv_t *sv);
62  void sv_set_type(sv_t *sv, int i, int j, int value);
63  int sv_get_type(sv_t *sv, int i, int j);
64  double sv_get_p(sv_t *sv, int i, int j);
65  void sv_set_p(sv_t *sv, int i, int j, double value);
66  void sv_set_p_BC(sv_t *sv, int i, int j, double value);
67  double sv_get_p_BC(sv_t *sv, int i, int j);
68  void sv_set_map(sv_t *sv, int i, int j, int value);
69  int sv_get_map(sv_t *sv, int i, int j);
70  int point(int i, int j);
71  int map_points(sv_t *sv);
72  void check_boundary(double *v, double P, double h1, double *vC, int
        size);
73  sv_t *SetDomain();
74  sv_t *Poisson(sv_t *sv);
75  double f(double x, double y);
```

```
76 double perim(int caseMask, double Lx, double Ly, double Rad, double
      rad);
77 double area(int caseMask, double Lx, double Ly, double Rad, double rad
      );
78 void SaveOutput(sv_t *sv);
79
80 //---------------------------------------------------------------------
81 int main ( ) {
82
83   sv_t *sv;
84   params = calloc(1, sizeof(params_t)); // Nx, Ny, Lx, Ly sao dados
        globais
85
86   sv = SetDomain();
87   sv = Poisson(sv);
88   SaveOutput(sv);
89   sv_free(sv);
90   free(params);
91
92   return 0;
93 }
94
95 //---------------------------------------------------------------------
96 // Metodo dos Gradientes Conjugados
97
98 // Mais detalhes podem ser encontrados em:
99 // https://en.wikipedia.org/wiki/Conjugate_gradient_method
100 double *ConjugateGradient(double **A, double *b, double *x, int size)
      {
101
102   int k, i;
103   double rsold, rsnew, alpha;
104   double *r, *p, *Ap;
105
106   // Aloca memoria
107   r   = allocVector(size);
108   p   = allocVector(size);
109   Ap  = allocVector(size);
110
111   // Calculo do residuo
112   r = xpay(r, b, -1.0, matVec(Ap, A, x, size), size);
113
114   for ( i = 0 ; i < size ; i++ )
```

```
115         p[i] = r[i];
116
117   rsold = ip(r, r, size);
118
119   for ( k = 0 ; k <= size ; k++ ) {
120
121         Ap = matVec(Ap, A, p, size);
122         alpha = rsold / ip(p, Ap, size);
123         x = xpay(x, x,  alpha,  p, size);
124         r = xpay(r, r, -alpha, Ap, size);
125         rsnew = ip(r, r, size);
126
127         // Criterio de parada
128         if ( sqrt(rsnew) < 1.0e-10 ) break;
129
130         p = xpay(p, r, rsnew/rsold, p, size);
131         rsold = rsnew;
132   }
133   printf("O metodo dos GC gastou %i iteracoes\n", k);
134
135   if ( k == size ) {
136         printf("\n\nAVISO: no metodo dos GC, k == n\n\n");
137         exit(1);
138   }
139
140   // Libera memoria alocada
141   free(Ap);
142   free(p);
143   free(r);
144
145   // Imprime a solucao na tela
146   printf("Solucao de Ax=b via metodo dos GC\n");
147   for ( k = 0 ; k < size ; k++ )
148         printf("x[%i] = %lf\n", k, x[k]);
149
150   printf("\n\n");
151
152   return x;
153 }
154 //--------------------------------------------------------------------
155 // Produto interno: <u,v>, inner product (ip)
156
157 double ip(double *u, double *v, int size) {
```

```
158   double soma = 0.0;
159   int i;
160   for ( i = 0 ; i < size ; i++ )
161         soma += u[i] * v[i];
162   return soma;
163 }
164
165 //---------------------------------------------------------------------
166 // matVec = A*x
167
168 double *matVec(double *dest, double **A, double *x, int size) {
169
170   double soma;
171   int i, j;
172
173   for ( i = 0 ; i < size ; i++ ) {
174         soma = 0.0;
175         for ( j = 0 ; j < size ; j++ ) {
176           soma += A[i][j] * x[j];
177     }
178         dest[i] = soma;
179   }
180
181   return dest;
182 }
183
184 //---------------------------------------------------------------------
185 // dest = x + alpha * y
186
187 double *xpay(double *dest, double *x, double alpha, double *y, int
      size) {
188
189   int i;
190
191   for (i = 0 ; i < size ; i++ )
192         dest[i] = x[i] + alpha * y[i];
193
194   return dest;
195 }
196
197 //---------------------------------------------------------------------
198 void freeMatrix(int size, double **A) {
199
```

```
200   int i;
201
202   for ( i = 0 ; i < size ; i++ )
203         free(A[i]);
204
205   free(A);
206 }
207
208 //---------------------------------------------------------------------
209 double **allocMatrix(int size) {
210
211   // Aloca memoria para a matriz
212   double **A;
213   int i;
214
215   A = (double **) malloc(size * sizeof(double *));
216
217   for ( i = 0 ; i < size ; i++ ) {
218         A[i] = (double *) malloc(size * sizeof(double));
219   }
220   // Verifica se a memoria foi alocada com sucesso
221   if ( A == NULL ) {
222         printf("Erro: memoria nao alocada\n");
223         exit(1);
224   }
225
226   return A;
227 }
228
229 //---------------------------------------------------------------------
230 double *allocVector(int size) {
231
232   double *vec;
233
234   // Aloca memoria para armazenar vetor
235   vec = (double *) calloc(size, sizeof(double)); // calloc inicia com
      o vetor nulo
236   // Verifica se a memoria foi alocada com sucesso
237   if ( vec == NULL ) {
238         printf("Erro: memoria nao alocada\n");
239         exit(1);
240   }
241
```

```
242   return vec;
243 }
244
245 //---------------------------------------------------------------------
246 // Aloca memoria para o vetor de estado
247
248 sv_t *sv_alloc() {
249   sv_t *sv = calloc(1, sizeof(sv_t));
250   sv->type = calloc(params->Nx * params->Ny, sizeof(int));
251   sv->map  = calloc(params->Nx * params->Ny, sizeof(int));
252   sv->p    = calloc(params->Nx * params->Ny, sizeof(double));
253
254   return sv;
255 }
256
257 //---------------------------------------------------------------------
258 // Libera memoria do vetor de estado
259
260 void sv_free(sv_t *sv) {
261   free(sv->type);
262   free(sv->map);
263   free(sv->p);
264   free(sv);
265 }
266
267 //---------------------------------------------------------------------
268 int point(int i, int j) {
269
270   if ( i >= 0 && i < params->Nx && j >= 0 && j < params->Ny ) {
271         return i + params->Nx * j;
272   } else {
273         printf("Erro: point(%i,%i)\n", i, j);
274   }
275 }
276
277 //---------------------------------------------------------------------
278 void sv_set_type(sv_t *sv, int i, int j, int value) {
279
280   if ( i < 0 || i >= params->Nx || j < 0 || j >= params->Ny ) {
281         printf("sv_set_type error: %i %i\n", i, j);
282         exit(1);
283   }
284   sv->type[point(i,j)] = value;
```

```
285
286 }
287 //---------------------------------------------------------------------
288 int sv_get_type(sv_t *sv, int i, int j) {
289
290   if ( i >= 0 && i < params->Nx && j >= 0 && j < params->Ny )
291         return sv->type[point(i,j)];
292
293   return POINT_INVALID;
294 }
295
296 //---------------------------------------------------------------------
297 void sv_set_p(sv_t *sv, int i, int j, double value) {
298   if ( sv_get_type(sv, i, j) & POINT_VALID ) {
299         sv->p[point(i,j)] = value;
300   } else {
301         printf("sv_set_p error: %i %i\n", i, j);
302         exit(1);
303   }
304 }
305
306 //---------------------------------------------------------------------
307 double sv_get_p(sv_t *sv, int i, int j) {
308
309   if ( sv_get_type(sv, i, j) & POINT_VALID ) {
310         return sv->p[point(i,j)];
311   }
312         printf("sv_get_p error: %i %i\n", i, j);
313         exit(1);
314 }
315
316 //---------------------------------------------------------------------
317 void sv_set_p_BC(sv_t *sv, int i, int j, double value) {
318
319   if ( sv_get_type(sv, i, j) & POINT_DIRICHLET ) {
320         sv->p[point(i,j)] = value;
321   } else {
322         printf("sv_set_p_BC error: %i %i\n", i, j);
323         exit(1);
324   }
325 }
326
327 //---------------------------------------------------------------------
```

```
328 double sv_get_p_BC(sv_t *sv, int i, int j) {
329
330   if ( sv_get_type(sv, i, j) & POINT_DIRICHLET ) {
331         return sv->p[point(i,j)];
332   }
333         printf("sv_get_p_BC error: %i %i\n", i, j);
334         exit(1);
335 }
336
337 //---------------------------------------------------------------------
338 void sv_set_map(sv_t *sv, int i, int j, int value) {
339
340   if ( i < 0 || i >= params->Nx || j < 0 || j >= params->Ny ) {
341         printf("sv_set_map error: %i %i\n", i, j);
342         exit(1);
343   }
344   sv->map[point(i,j)] = value;
345 }
346
347 //---------------------------------------------------------------------
348 int sv_get_map(sv_t *sv, int i, int j) {
349
350   if ( i >= 0 || i < params->Nx || j >= 0 || j < params->Ny ) {
351         return sv->map[point(i,j)];
352   }
353   printf("sv_get_map error: %i %i\n", i, j);
354   exit(1);
355 }
356
357 //---------------------------------------------------------------------
358 // Termo fonte
359
360 double f(double x, double y) {
361   return -1;
362   // validacao: -2.0 * sin(x) * cos(y);
363 }
364
365 //---------------------------------------------------------------------
366 sv_t *SetDomain() {
367
368   sv_t *sv;
369   params->caseMask = 1;
370
```

```
371   printf("\n\n# Set Domain: Mask %i\n\n", params->caseMask);
372
373   // Definir a geometria de acordo com o problema
374   if ( params->caseMask == 1 ) {
375
376     // Retangulo: [0,Lx] x [0,Ly]
377     params->Nx = 161;
378     params->Ny = 161;
379     params->Lx = 2;
380     params->Ly = 1;
381     params->Rad = 0;
382     params->rad = 0;
383
384     sv = sv_alloc();
385
386     double dx = params->Lx / ( params->Nx - 1 );
387     double dy = params->Ly / ( params->Ny - 1 );
388     double x, y;
389     int i, j;
390
391     for ( i = 0 ; i < params->Nx ; i++ ) {
392       for ( j = 0 ; j < params->Ny ; j++ ) {
393         if ( i == 0 || i == params->Nx-1 || j == 0 || j == params->Ny
              -1 ) {
394           sv_set_type(sv, i, j, POINT_DIRICHLET);
395           x = i * dx;
396           y = j * dy;
397           sv_set_p_BC(sv, i, j, 0);
398           // validacao: sin(x) * cos(y));
399         } else {
400           sv_set_type(sv, i, j, POINT_VALID);
401         }
402       }
403     }
404   } else if ( params->caseMask == 2 ) {
405
406     // Cilindro: raio R
407     int i, j;
408     params->Rad = 0.5;
409     params->rad = 0;
410     params->Nx = 34;
411     params->Ny = 34;
412     params->Lx = 2.0*params->Rad;
```

```
413     params->Ly = 2.0*params->Rad;
414
415     double dx = params->Lx / ( params->Nx - 1 );
416     double dy = params->Ly / ( params->Ny - 1 );
417
418     params->Nx += 2;
419     params->Ny += 2;
420     params->Lx += 2.0*dx;
421     params->Ly += 2.0*dy;
422
423     sv = sv_alloc();
424
425     double x, y; // Coordenada de cada celula computacional
426     double Cx, Cy; // Centro do tubo circular
427     Cx = params->Lx / 2.0;
428     Cy = params->Ly / 2.0;
429
430     for ( i = 0 ; i < params->Nx ; i++ ) {
431         for ( j = 0 ; j < params->Ny ; j++ ) {
432                 x = i * dx;
433                 y = j * dy;
434         if ( i == 0 || i == params->Nx-1 || j == 0 || j == params->Ny
               -1 ) {
435                 sv_set_type(sv, i, j, POINT_DIRICHLET);
436         } else if ( pow(x - Cx, 2.0) + pow(y - Cy, 2.0) < params->Rad*
               params->Rad ) {
437         // Verifica se o ponto (i,j) esta dentro do circulo
438                 sv_set_type(sv, i, j, POINT_VALID);
439         } else {
440                 sv_set_type(sv, i, j, POINT_DIRICHLET);
441         }
442         }
443     }
444 } else if ( params->caseMask == 3 ) {
445
446     // Anular: raio externo Rad e raio interno rad
447     int i, j;
448     params->Rad = 0.5;
449     params->rad = 0.15;
450     params->Nx = 34;
451     params->Ny = 34;
452     params->Lx = 2.0*params->Rad;
453     params->Ly = 2.0*params->Rad;
```

```
454
455     double dx = params->Lx / ( params->Nx - 1 );
456     double dy = params->Ly / ( params->Ny - 1 );
457
458     params->Nx += 2;
459     params->Ny += 2;
460     params->Lx += 2.0*dx;
461     params->Ly += 2.0*dy;
462
463     sv = sv_alloc();
464
465     double x, y; // Coordenada de cada celula computacional
466     double Cx, Cy; // Centro do tubo circular
467     Cx = params->Lx / 2.0;
468     Cy = params->Ly / 2.0;
469
470     for ( i = 0 ; i < params->Nx ; i++ ) {
471         for ( j = 0 ; j < params->Ny ; j++ ) {
472                 x = i * dx;
473                 y = j * dy;
474         if ( i == 0 || i == params->Nx-1 || j == 0 || j == params->Ny
               -1 ) {
475                  sv_set_type(sv, i, j, POINT_DIRICHLET);
476         } else if ( pow(x - Cx, 2.0) + pow(y - Cy, 2.0) < params->Rad*
               params->Rad ) {
477         // Verifica se o ponto (i,j) esta dentro do circulo maior
478              if ( pow(x - Cx, 2.0) + pow(y - Cy, 2.0) < params->rad*
                   params->rad ) {
479                  // Verifica se o ponto (i, j) esta dentro do circulo
                       menor
480                   sv_set_type(sv, i, j, POINT_DIRICHLET);
481                 } else {
482                 // O ponto (i, j) esta dentro da coroa circular
483                 sv_set_type(sv, i, j, POINT_VALID);
484                 }
485              } else {
486                  sv_set_type(sv, i, j, POINT_DIRICHLET);
487              }
488         }
489     }
490     } else {
491
492         printf("SetDomain error\n");
```

```
493         exit(1);
494   }
495   return sv;
496 }
497
498 //---------------------------------------------------------------------
499 // Mapeia quais pontos do dominio serao calculados
500
501 int map_points(sv_t *sv) {
502
503   int i, j, map, type, n = 0;
504
505   for ( j = 0 ; j < params->Ny ; j++ ) {
506     for ( i = 0 ; i < params->Nx ; i++ ) {
507           type = sv_get_type(sv, i, j);
508
509           if ( type & POINT_VALID ) {
510                 map = n++;
511           } else if ( type & POINT_NEUMANN ) {
512                 map = -2;
513           } else {
514                 map = -1;
515           }
516           sv_set_map(sv, i, j, map);
517         }
518   }
519   return n;
520 }
521
522 //---------------------------------------------------------------------
523 void check_boundary(double *v, double P, double h1, double *vC, int
      size) {
524
525   (*v)   = ( P >= 0 && P < size ) ? h1 : 0.0;
526   (*vC) -= ( P >= 0 || P == -1 ) ? h1 : 0.0;
527 }
528
529 //---------------------------------------------------------------------
530 double perim (int caseMask, double Lx, double Ly, double Rad, double
      rad) {
531
532         if ( caseMask == 1 ) {
533                 return 2 * params->Lx + 2 * params->Ly;
```

```
534         }
535         else if ( caseMask == 2 ) {
536                 return 2 * params->Rad * PI;
537         }
538         else if ( caseMask == 3 ) {
539                 return 8 * PI * params->rad;
540         }
541 }
542
543 //---------------------------------------------------------------------
544 double area(int caseMask, double Lx, double Ly, double R, double r) {
545
546         if ( caseMask == 1 ) {
547                 return params->Lx * params->Ly;
548         }
549         else if ( caseMask == 2 ) {
550                 return PI * params->Rad * params->Rad;
551         }
552         else if ( caseMask == 3 ) {
553                 return PI * ((params->Rad*params->Rad) - (params->rad*
                    params->rad));
554         }
555 }
556
557 //---------------------------------------------------------------------
558 sv_t *Poisson(sv_t *sv) {
559
560   int i, j, map, map_E, map_W, map_N, map_S;
561   double dx = params->Lx / ( params->Nx - 1.0 );
562   double dy = params->Ly / ( params->Ny - 1.0 );
563   double x_coord, y_coord;
564   double vC, vE, vW, vN, vS;
565   int size = map_points(sv); // size eh o numero de eqs do sistema
      linear
566
567   printf("\n# Solve Poisson equation: size = %i\n\n", size);
568   printf("\nLx = %lf, \tLy = %lf\nNx = %i, \t Ny = %i\ndx = %lf, \tdy
      = %lf\n\n", params->Lx, params->Ly, params->Nx, params->Ny, dx,
      dy);
569
570   // Aloca memoria para armazenar os dados do sistema linear Ax = b
571   double **A = allocMatrix(size);
572   double  *b = allocVector(size);
```

```
573   double  *x = allocVector(size); // vetor nulo como chute inicial
          para o solver ConjugateGradient
574
575   // Calculo da matriz A e do vetor b (rhs)
576   for ( j = 0 ; j < params->Ny ; j++ ) {
577     for ( i = 0 ; i < params->Nx ; i++ ) {
578
579           map = sv_get_map(sv, i, j);
580
581           if ( map < 0 || map >= size ) continue;
582
583           // rhs
584           x_coord = i * dx;
585           y_coord = j * dy;
586           b[map] = -f(x_coord, y_coord); // Termo fonte
587
588           // Matriz A
589         vC = 0.0;
590
591           map_E = sv_get_map(sv, i+1, j);
592           map_W = sv_get_map(sv, i-1, j);
593           map_N = sv_get_map(sv, i, j+1);
594           map_S = sv_get_map(sv, i, j-1);
595
596       check_boundary(&vE, map_E, 1.0 / ( dx * dx ), &vC, size);
597       check_boundary(&vW, map_W, 1.0 / ( dx * dx ), &vC, size);
598       check_boundary(&vN, map_N, 1.0 / ( dy * dy ), &vC, size);
599       check_boundary(&vS, map_S, 1.0 / ( dy * dy ), &vC, size);
600
601       A[map][map] = - vC;
602
603           if ( map_E >= 0 && map_E < size ) {
604                 A[map][map_E] = - vE;
605           } else if ( map_E == -1 ) {
606                 b[map] += sv_get_p_BC(sv, i+1, j) / ( dx * dx );
607           }
608
609           if ( map_W >= 0 && map_W < size ) {
610                 A[map][map_W] = - vW;
611           } else if ( map_W == -1 ) {
612                 b[map] += sv_get_p_BC(sv, i-1, j) / ( dx * dx );
613           }
614
```

```
615           if ( map_N >= 0 && map_N < size ) {
616                 A[map][map_N] = - vN;
617           } else if ( map_N == -1 ) {
618                 b[map] += sv_get_p_BC(sv, i, j+1) / ( dy * dy );
619           }
620
621           if ( map_S >= 0 && map_S < size ) {
622                 A[map][map_S] = - vS;
623           } else if ( map_S == -1 ) {
624                 b[map] += sv_get_p_BC(sv, i, j-1) / ( dy * dy );
625           }
626         }
627   }
628
629   // Resolve Ax = b
630   x = ConjugateGradient(A, b, x, size);
631
632   double Q = 0, fRe;
633
634   // Atualiza a solucao da eq. de Poisson
635   for ( j = 0 ; j < params->Ny ; j++ ) {
636     for ( i = 0 ; i < params->Nx ; i++ ) {
637
638           map = sv_get_map(sv, i, j);
639
640           if ( map < 0 || map >= size ) continue;
641
642           sv_set_p(sv, i, j, x[map]);
643
644           // Calculo da vazao Q = - W * (area) + C
645           Q += x[map] * dx * dy;
646         }
647   }
648
649   double Area = area(params->caseMask, params->Lx, params->Ly, params
        ->Rad, params->rad);
650   double Perim = perim(params->caseMask,params->Lx,params->Ly,params->
        Rad,params->rad);
651
652   // Calculo do coeficiente de atrito fRe
653   fRe = (2.0 * pow((4*Area/Perim),2) * Area) / Q;
654
655   printf("\n\nVazao: %e\nfRe = %e\n\n", Q, fRe);
```

```
656
657   // Libera memoria
658   freeMatrix(size, A);
659   free(b);
660   free(x);
661
662   return sv;
663 }
664
665 //---------------------------------------------------------------------
666 void SaveOutput(sv_t *sv) {
667
668   FILE *fp;
669   char file_name[128];
670   double dx = params->Lx / ( params->Nx - 1 );
671   double dy = params->Ly / ( params->Ny - 1 );
672   double x, y, xc, yc, Dx, Dy, tau;
673   int i, j;
674
675   // Output para o perfil de velocidade
676   sprintf(file_name, "output_Poisson_Nx_%i_Ny_%i_Lx_%.02lf_Ly_%.02lf.
      txt", params->Nx, params->Ny, params->Lx, params->Ly);
677
678   fp = fopen(file_name, "w");
679
680   for ( j = 0 ; j < params-> Ny ; j++ ) {
681     for ( i = 0 ; i < params-> Nx ; i++ ) {
682           x = i * dx;
683           y = j * dy;
684           fprintf(fp, "%lf %lf %lf \n", x, y, sv->p[point(i,j)]);
685         }
686         fprintf(fp, "\n");
687   }
688   fclose(fp);
689
690   // Output para tensao de cisalhamento
691   sprintf(file_name, "output_Poisson_Nx_%i_Ny_%i_Lx_%.02lf_Ly_%.02
      lf_tau.txt", params->Nx, params->Ny, params->Lx, params->Ly);
692
693   fp = fopen(file_name, "w");
694
695   for ( j = 0 ; j < params-> Ny - 1 ; j++ ) {
696     for ( i = 0 ; i < params-> Nx - 1 ; i++ ) {
```

```
697             xc = ( i + 0.5 ) * dx;
698             yc = ( j + 0.5 ) * dy;
699             Dx = (sv->p[point(i+1,j)] - sv->p[point(i,j)]) / dx;
700             Dy = (sv->p[point(i,j+1)] - sv->p[point(i,j)]) / dy;
701 
702             tau = sqrt( Dx*Dx + Dy*Dy);
703 
704             fprintf(fp, "%lf %lf %lf %lf\n", xc, yc, tau);
705           }
706           fprintf(fp, "\n");
707   }
708   fclose(fp);
709 //-------------------------------------------------------------
```