

Universidade Federal Fluminense

MAEZIO PEREIRA DA SILVA

Otimização de um Modelo Computacional
para Simulação de Transporte de Poluentes em
Corpos Hídricos

Volta Redonda, 28 de Abril de 2023.

MAEZIO PEREIRA DA SILVA

Otimização de um Modelo Computacional para Simulação de Transporte de Poluentes em Corpos Hídricos

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia. Área de Concentração: Modelagem Computacional.

Orientador:

Professor D.Sc. Wagner Rambaldi Telles

Coorientador:

Professor D.Sc. Ricardo Silveira Sousa

UNIVERSIDADE FEDERAL FLUMINENSE

Volta Redonda, 28 de Abril de 2023.

Ficha catalográfica automática - SDC/BEM
Gerada com informações fornecidas pelo autor

S586o Silva, Maezio Pereira da
Otimização de um modelo computacional para simulação de
transporte de poluentes em corpos hídricos / Maezio Pereira
da Silva. - 2023.
131 p.: il.

Orientador: Wagner Rambaldi Telles.
Coorientador: Ricardo Silveira Souza.
Dissertação (mestrado)-Universidade Federal Fluminense,
Escola de Engenharia Industrial e Metalúrgica de Volta
Redonda, Volta Redonda, 2023.

1. Transporte de contaminantes. 2. Matrizes esparsas. 3.
Solução de sistemas de equações lineares. 4. Otimização
de tempo e alocação de memória. 5. Produção intelectual.
I. Telles, Wagner Rambaldi, orientador. II. Souza, Ricardo
Silveira, coorientador. III. Universidade Federal Fluminense.
Escola de Engenharia Industrial e Metalúrgica de Volta
Redonda. IV. Título.


CDD - XXX

Otimização de um Modelo Computacional para Simulação de Transporte de Poluentes em Corpos Hídricos


Maezio Pereira da Silva

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia. Área de Concentração: Modelagem Computacional.


Aprovada por:

Documento assinado digitalmente
 **WAGNER RAMBALDI TELLES**
Data: 28/04/2023 17:17:09-0300
Verifique em <https://validar.iti.gov.br>


Prof. Wagner Rambaldi Telles, D.Sc.
MCCT-UFF (Presidente)

Documento assinado digitalmente
 **NELSON MACHADO BARBOSA**
Data: 28/04/2023 15:59:30-0300
Verifique em <https://validar.iti.gov.br>

Prof. Nelson Machado Barbosa, D.Sc.
UENF

Documento assinado digitalmente
 **THIAGO JORDEM PEREIRA**
Data: 28/04/2023 16:05:12-0300
Verifique em <https://validar.iti.gov.br>

Prof. Thiago Jordem Pereira, D.Sc.
MCCT-UFF

Documento assinado digitalmente
 **FABIO FREITAS FERREIRA**
Data: 28/04/2023 15:55:13-0300
Verifique em <https://validar.iti.gov.br>

Prof. Fábio Freitas Ferreira, D.Sc.
MCCT-UFF

Volta Redonda, 28 de Abril de 2023.

Agradecimentos

Primeiramente gostaria de agradecer ao grande arquiteto do universo pelo equilíbrio das leis fundamentais que possibilitou a vida como a conhecemos. Gostaria de agradecer aos meus avós, pois foram a base de tudo. Agradeço também os meus pais e minha irmã que me deram educação, moldaram meu caráter e me ensinaram a nunca desistir dos meus objetivos e em nome deles, agradeço todos meus familiares. Agradeço, em especial, minha namorada Thais pelo companheirismo, pela paciência, motivação e cumplicidade em todos os momentos, atitudes sem as quais tornariam muito mais árduo o caminho e em seu nome, agradeço minha sogra, meu sogro e todos seus familiares. Não poderia deixar de agradecer a meu orientador Wagner Rambaldi pelo conhecimento compartilhado, pelo prazer de ensinar, não limitado a dia ou horário, em seu nome agradeço todos os professores que tive ao longo da minha vida. Por fim, agradeço todos meus amigos, sejam eles de infância, virtuais ou de trabalho e que de alguma forma contribuíram para que eu concluísse essa etapa acadêmica.

Esse estudo foi apoiado em parte pela Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (projeto com número de processo E47/2021-SEI260003/016517/2021-E26/210.107/2022).

*“Há três caminhos para o sucesso: Perguntar o que se ignora, ensinar o que se sabe e
viver o que se ensina.”*
(Beda)

Resumo

Com objetivo de aprimorar programas computacionais que podem servir de auxílio aos órgãos de fiscalização e controle no combate à poluição dos recursos hídricos, a proposta desta dissertação é implementar e analisar distintas metodologias computacionais para simular o transporte de contaminantes lançados em um trecho de rio, buscando aproveitar a esparsidade da estrutura matricial, originária da solução numérica do modelo matemático, através do Método dos Volumes Finitos, de forma a otimizar o tempo de cálculo da máquina e reduzir o espaço para a alocação de memória da mesma. Foram utilizados os métodos numéricos de Gauss-Seidel e o Algoritmo de Thomas nas implementações computacionais para a solução dos sistemas de equações, enquanto na realização dos cálculos matemáticos, assim como na representação gráfica dos resultados, utilizou-se o software MATLAB. Para validação dos resultados, os valores numéricos, obtidos através da simulação computacional, foram comparados com os dados de dois estudos de casos reais onde o primeiro refere-se a um experimento realizado no rio São Pedro, localizado na cidade de Nova Friburgo-RJ, e o segundo, em um trecho do rio Macaé, na cidade de Macaé-RJ.

Abstract

Intending to improve computational programs that can help inspection and control agencies in the fight against pollution of water resources, the purpose of this dissertation is to implement and analyze different computational methodologies to simulate the transport of contaminants released in a stretch of river, seeking to take advantage of the sparsity of the matrix structure, originating from the numerical solution of the mathematical model, through the Finite Volumes Method, in order to optimize the machine's calculation time and reduce the space for its memory allocation. The numerical methods of Gauss-Seidel and the Thomas Algorithm were used in the computational implementations for the solution of the systems of equations, while in the accomplishment of the mathematical calculations, as well as in the graphical representation of the results, the software MATLAB was used. To validate the results, numerical values obtained through computer simulation were compared with data from two studies of real cases, where the first refers to an experiment carried out on the São Pedro River, located in the city of Nova Friburgo-RJ, and the second, in a stretch of the Macaé River, in the city of Macaé-RJ.

Palavras-chave

1. Recursos Hídricos;
2. Transporte de Contaminantes;
3. Método dos Volumes Finitos;
4. Matrizes Esparsas;
5. Otimização de Tempo e Alocação de Memória.

Lista de Figuras

2.1	Visualização da matriz esparsa com variação do parâmetro γ	26
2.2	Visualização da matriz esparsa.	27
3.1	Exemplo da linha de programação para criação de estruturas matriciais no MATLAB.	43
3.2	Exemplo de multiplicação de matrizes no MATLAB.	44
4.1	Diagrama para resolução de um problema físico.	53
4.2	Processo de transporte de soluto em fluido.	54
4.3	Esquematização do problema de transporte de poluente.	55
5.1	Representação do domínio discretizado para uma dimensão.	63
5.2	Representação das distâncias no domínio discretizado.	65
5.3	Representação do volume fictício para cálculo do volume 1.	68
5.4	Representação do volume fictício para cálculo do volume n	69
5.5	Representação do domínio discretizado para duas dimensões.	71
5.6	Representação dos volumes internos no domínio discretizado para duas dimensões.	74
5.7	Representação das regiões do domínio físico.	75
5.8	Representação dos volumes fictícios para o cálculo da Região 1.	75
5.9	Representação dos volumes fictícios para o cálculo da Região 2.	77
5.10	Representação dos volumes fictícios para o cálculo da Região 3.	77
5.11	Representação dos volumes fictícios para o cálculo da Região 4.	78
5.12	Representação dos volumes fictícios para o cálculo da Região 6.	79
5.13	Representação dos volumes fictícios para o cálculo da Região 7.	80

5.14	Representação dos volumes fictícios para o cálculo da Região 8.	81
5.15	Representação dos volumes fictícios para o cálculo da Região 9.	82
6.1	Teste Unidimensional - Perfil da concentração para matriz de ordem 100×100	96
6.2	Teste Unidimensional - Perfil da concentração para matriz de ordem 500×500	97
6.3	Teste Unidimensional - Perfil da concentração para matriz de ordem 1.000×1.000	99
6.4	Teste Unidimensional - Perfil da concentração para matriz de ordem 1.500×1.500	101
6.5	Teste Unidimensional - Perfil da concentração para matriz de ordem 3.000×3.000	102
6.6	Teste Unidimensional - Perfil da concentração para matriz de ordem 40.000×40.000	104
6.7	Teste Unidimensional - Perfil da concentração para matriz de ordem $50.000.000 \times 50.000.000$	106
6.8	Comparação dos perfis de concentração para o caso unidimensional em relação ao aumento da ordem da matriz utilizando a Implementação 7.	107
6.9	Teste Unidimensional - Comparação dos tempos de criação das estruturas (Matriz e Vetores) com a execução das implementações 2 e 6 (gráfico em escala logarítmica).	108
6.10	Teste Unidimensional - Comparação dos tempos de criação das estruturas (Matriz e Vetores) com a execução das implementações 3 e 7 (gráfico em escala logarítmica).	109
6.11	Teste Bidimensional - Perfil da concentração para matriz de ordem 250×250	111
6.12	Teste Bidimensional - Perfil da concentração para matriz de ordem 1.000×1.000	113
6.13	Teste Bidimensional - Perfil da concentração para matriz de ordem 4.000×4.000	114
6.14	Teste Bidimensional - Perfil da concentração para matriz de ordem 9.000×9.000	116

6.15	Teste Bidimensional - Perfil da concentração para matriz de ordem 16.000×16.000	117
6.16	Teste Bidimensional - Perfil da concentração para matriz de ordem 40.000×40.000	119
6.17	Teste Unidimensional - Perfil da concentração para matriz de ordem $50.000.000 \times 50.000.000$	121
6.18	Comparação dos perfis de concentração para o caso bidimensional em relação ao aumento da ordem da matriz utilizando a Implementação 7. . . .	122
6.19	Teste Bidimensional - Comparação dos tempos de criação das estruturas (Matriz e Vetores) com a execução das implementações 2 e 6 (gráfico em escala logarítmica).	123
6.20	Teste Bidimensional - Comparação dos tempos de criação das estruturas (Matriz e Vetores) com a execução das implementações 3 e 7 (gráfico em escala logarítmica).	124

Lista de Tabelas

2.1	Quantidades (gramas) fornecidas por 100 g de ingrediente.	20
2.2	Variação do parâmetro γ para uma matriz de ordem $n = 150$	25
3.1	Conversão de números decimais em binário.	44
3.2	Relação entre a ordem da matriz quadrada e a memória necessária para seu armazenamento.	45
3.3	Economia no armazenamento de memória em relação à ordem da matriz. .	47
3.4	Economia no armazenamento de memória em relação ao grau de esparsidade da matriz.	48
4.1	Nível de formulação dos modelos.	55
4.2	Parâmetros do rio para caso unidimensional.	57
4.3	Parâmetros do rio para caso bidimensional.	60
6.1	Componentes físicos do computador em que os testes foram executados. . .	93
6.2	Tipos de implementações desenvolvidas na presente pesquisa.	93
6.3	Parâmetros utilizados nos testes unidimensionais.	95
6.4	Teste Unidimensional - Tempos de execução para matriz de ordem 100×100 . .	96
6.5	Teste Unidimensional - Consumo de memória para matriz de ordem 100×100 . .	97
6.6	Teste Unidimensional - Tempos de execução para matriz de ordem 500×500 . .	98
6.7	Teste Unidimensional - Consumo de memória para matriz de ordem 500×500 . .	98
6.8	Teste Unidimensional - Tempos de execução para matriz de ordem 1.000×1.000	99
6.9	Teste Unidimensional - Consumo de memória para matriz de ordem 1.000×1.000	100

6.10	Teste Unidimensional - Tempos de execução para matriz de ordem 1.500×1.500	101
6.11	Teste Unidimensional - Consumo de memória para matriz de ordem 1.500×1.500	102
6.12	Teste Unidimensional - Tempos de execução para matriz de ordem 3.000×3.000	103
6.13	Teste Unidimensional - Consumo de memória para matriz de ordem 3.000×3.000	103
6.14	Teste Unidimensional - Tempos de execução para matriz de ordem 40.000×40.000	105
6.15	Teste Unidimensional - Consumo de memória para matriz de ordem 40.000×40.000	105
6.16	Teste Unidimensional - Tempos de execução para matriz de ordem $50.000.000 \times 50.000.000$	106
6.17	Teste Unidimensional - Consumo de memória para matriz de ordem $50.000.000 \times 50.000.000$	107
6.18	Parâmetros utilizados nos testes bidimensionais.	110
6.19	Teste Bidimensional - Tempos de execução para matriz de ordem 250×250	111
6.20	Teste Bidimensional - Consumo de memória para matriz de ordem 250×250	112
6.21	Teste Bidimensional - Tempos de execução para matriz de ordem 1.000×1.000	113
6.22	Teste Bidimensional - Consumo de memória para ordem 1.000×1.000	113
6.23	Teste Bidimensional - Tempos de execução para matriz de ordem 4.000×4.000	115
6.24	Teste Bidimensional - Consumo de memória para matriz de ordem 4.000×4.000	115
6.25	Teste Bidimensional - Tempos de execução para matriz de ordem 9.000×9.000	116
6.26	Teste Bidimensional - Consumo de memória para matriz de ordem 9.000×9.000	117

6.27	Teste Bidimensional - Tempos de execução para matriz de ordem 16.000×16.000	118
6.28	Teste Bidimensional - Consumo de memória para matriz de ordem 16.000×16.000	118
6.29	Teste Bidimensional - Tempos de execução para matriz de ordem 40.000×40.000	120
6.30	Teste Bidimensional - Consumo de memória para matriz de ordem 40.000×40.000	120
6.31	Teste Bidimensional - Tempos de execução para matriz de ordem $50.000.000 \times 50.000.000$	121
6.32	Teste Bidimensional - Consumo de memória para matriz de ordem $50.000.000 \times 50.000.000$	122

Glossário

ONU	Organização das Nações Unidas
PNMA	Política Nacional de Meio Ambiente
MVF	Método dos Volumes Finitos
MATLAB	<i>Matrix Laboratory</i>
TDMA	<i>Tridiagonal Matrix Algorithm</i>
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
CSR	<i>Compressed Sparse Row</i>
CSC	<i>Compressed Sparse Column</i>
CSV	<i>Compressed Sparse Vector</i>
CDS	<i>Compressed Diagonal Storage</i>

Sumário

1	INTRODUÇÃO	16
1.1	Justificativa	17
1.1.1	Objetivo Geral	18
1.1.2	Objetivos Específicos	18
1.2	Organização do Trabalho	19
2	EMBASAMENTO TEÓRICO	20
2.1	Matrizes	21
2.1.1	Tipos de Matrizes	22
2.2	Sistema de Equações Lineares	28
2.3	Solução de um Sistema de Equações Lineares	29
2.3.1	Métodos Diretos	31
2.3.1.1	Eliminação de Gauss	32
2.3.1.2	Algoritmo de Thomas (TDMA)	33
2.3.2	Métodos Iterativos	35
2.3.2.1	Método de Gauss-Seidel	36
2.3.2.2	Algoritmo de Thomas Adaptado para Matriz Pentadiagonal	40
3	ASPECTOS COMPUTACIONAIS ENVOLVENDO MATRIZES	42
3.1	Software MATLAB	42
3.1.1	Vetores e Matrizes no MATLAB	43
3.1.2	Alocação de Matrizes no MATLAB	44

3.1.3	Alocação de Matrizes Esparsas no MATLAB	46
3.2	Compressão de Matrizes Esparsas	48
3.2.1	<i>Compressed Sparse Row</i> (CSR)	48
3.2.2	<i>Compressed Sparse Column</i> (CSC)	49
3.2.3	<i>Compressed Sparse Vector</i> (CSV)	50
3.2.4	<i>Compressed Diagonal Storage</i> (CDS)	50
4	DESCRIÇÃO E MODELAGEM DO PROBLEMA PROPOSTO	52
4.1	Caracterização e Modelagem do Problema Físico Proposto	53
4.2	Estudo de Caso 01: Problema Unidimensional	57
4.3	Estudo de Caso 02: Problema Bidimensional	59
5	SOLUÇÃO NUMÉRICA DO PROBLEMA PROPOSTO	62
5.1	Método dos Volumes Finitos	62
5.2	Solução Numérica do Estudo de Caso 01: Problema Unidimensional	63
5.3	Solução Numérica do Estudo de Caso 02: Problema Bidimensional	71
5.4	Implementação Computacional	84
6	RESULTADOS E DISCUSSÕES	93
6.1	Resultados do Estudo de Caso 01: Problema Unidimensional	94
6.2	Resultados do Estudo de Caso 02: Problema Bidimensional	109
7	CONCLUSÕES E TRABALHOS FUTUROS	125
7.1	Conclusões	125
7.2	Trabalhos Futuros	127
	Referências	129

Capítulo 1

INTRODUÇÃO

Segundo a Organização das Nações Unidas (ONU), a população mundial atingiu, em 2022, a marca de 8 bilhões de pessoas [20], resultados dos avanços científicos nas áreas de alimentação, saneamento e saúde pública, proporcionados pelo advento da revolução industrial [29]. Ainda, segundo a ONU, a taxa de crescimento populacional atual é de 1,2% ao ano. Com base nesses dados, estima-se que, em 2100, a humanidade tenha 10,9 bilhões de indivíduos [21]. Diante disto, Miller (1985) [17], faz a seguinte analogia:

Nosso planeta pode ser comparado a uma astronave, deslocando-se a cem mil quilômetros por hora pelo espaço sideral, sem possibilidade de parada para reabastecimento, mas dispondo de um eficiente sistema de aproveitamento de energia solar e de reciclagem de matéria. Existem atualmente, na astronave, ar, água e comida suficientes para manter seus passageiros. Tendo em vista o progressivo aumento do número desses passageiros, em forma exponencial, e a ausência de portos para reabastecimento, podem-se vislumbrar, a médio e longo prazos, problemas sérios para a manutenção de sua população [17].

Ou seja, esse crescimento exponencial da população pode afetar a disponibilidade de recursos naturais no planeta, os quais são definidos como quaisquer insumos necessários para a manutenção dos organismos vivos [3].

Dentre os recursos naturais essenciais para a vida humana, a água, talvez seja o mais importante. Perfazendo mais de 70% da superfície terrestre, ela é um dos elementos mais abundantes do planeta [13]. Contudo, 97,5% desse valor é composto por água salgada. Da quantidade de água doce presente nos 2,5% que restam, cerca de 69% se concentram em geleiras, 30% em águas subterrâneas e apenas 1% está presente em rios [28].

Somado ao crescimento populacional, outro fator que poderá comprometer a qualidade deste recurso natural, ou mesmo, impossibilitar a sua utilização para o consumo humano,

é a crescente degradação dos corpos hídricos, através de ações diretas do homem.

Em relação a essa degradação, os conceitos de poluição e contaminação são importantes para diferenciar causa e consequência. De acordo com Carapeto (1999) [5], tais conceitos podem ser definidos como segue:

Contaminação é definida como a presença de concentrações elevadas de substâncias na água, nos sedimentos ou nos organismos, isto é, concentrações que estão acima do nível base para uma dada área e um dado organismo. Poluição deve ser definida como a introdução pelo Homem, direta ou indiretamente, de substâncias ou energia no ambiente marinho, resultando em efeitos nocivos que prejudiquem os recursos vivos, sejam um perigo para a saúde humana, se tornem um obstáculo para as atividades marítimas, incluindo a pesca, diminuam a qualidade da água do mar para ser utilizada e reduzam a utilização da água do mar para amenidades [5].

O lançamento em rios, sem nenhum tipo de tratamento, de materiais orgânicos presentes em esgotos, adotado pelos órgãos responsáveis pelo sistema de saneamento, é uma das principais causas da poluição dos corpos hídricos [22]. A consequência direta da poluição dos rios é a contaminação da água. O descarte de poluentes gera uma série de reações biológicas e químicas que elevam o nível de concentração das substâncias, prejudicando a qualidade desse recurso natural.

Duas vertentes cruciais para o controle do nível de contaminação dos rios, são: o acompanhamento da qualidade da água e a estimativa dos possíveis impactos ambientais causados pela ação poluidora [24]. Essas vertentes são instrumentos da Política Nacional de Meio Ambiente (PNMA). Porém, devido ao alto custo de análises da qualidade da água e a exigência de equipamentos e material humano especializado, faz com que o programa encontre barreiras financeiras, políticas e operacionais, comprometendo sua execução [24].

Alternativamente, são utilizados modelos matemáticos e computacionais para analisar o comportamento de contaminantes em cursos d'água, em particular, nos rios, de forma que os mesmos possam contribuir para o processo de tomada de decisão no que se refere ao monitoramento de tais recursos hídricos.

1.1 Justificativa

Diante do cenário apresentado, uma forma de contornar o alto custo operacional e financeiro, exigido para o monitoramento da qualidade da água, é investir no desenvolvimento e no aprimoramento de implementações computacionais, capazes de simular ações

poluidoras em trechos de rios, a fim de se estimar o nível de dano que um poluente despejado no curso d'água pode causar. Os dados obtidos, quando devidamente validados, auxiliam no controle da contaminação dos rios, sendo uma importante ferramenta na tomada de decisões, em relação ao enfrentamento da poluição dos recursos hídricos.

Essas implementações computacionais trabalham com modelos matemáticos que envolvem equações diferenciais ordinárias e/ou parciais, e para algumas formulações, a discretização, oriunda da solução numérica, resulta em um sistema de equações lineares, os quais utilizam matrizes esparsas de grandes dimensões. Esses modelos, quando não consideram a esparsidade da matriz e realizam o cálculo na estrutura completa, exigem uma grande demanda computacional, tanto no tempo de execução dos cálculos, quanto no consumo de memória para armazenamento dos dados.

1.1.1 Objetivo Geral

Neste sentido, o objetivo geral do presente trabalho acadêmico é buscar estratégias para desenvolver implementações computacionais, as quais aproveitem a esparsidade da matriz dos coeficientes, gerada através da discretização do problema, de forma a utilizá-la em métodos numéricos diretos e iterativos, não computando no cálculo e nem armazenando na memória do programa, os elementos nulos da matriz.

1.1.2 Objetivos Específicos

Para atingir o objetivo geral, são elencados os seguintes objetivos específicos:

1. Compreender a definição de matriz esparsa, bem como analisar seus tipos e características;
2. Analisar dois tipos de métodos para solução de sistema de equações: diretos e iterativos;
3. Abordar métodos de compressão de matrizes, assim como sua alocação na memória do computador;
4. Descrever e modelar problemas de transporte de contaminantes, realizando estudos específicos de cada caso;
5. Solucionar os problemas de transporte de contaminantes, através do Método dos Volumes Finitos (MVF), criando implementações com diferentes tipos de estruturas;

6. Otimizar os tempos de execução dos métodos, evitando operações matemáticas desnecessárias, envolvendo elementos nulos da matriz dos coeficientes trabalhada;
7. Reduzir o consumo de memória do computador, uma vez que os elementos nulos não serão armazenados;
8. Apresentar os resultados na forma gráfica, com o intuito de comparar as implementações utilizadas.

1.2 Organização do Trabalho

No Capítulo 1, definido como Introdução, é apresentada a ideia do trabalho, os objetivos que se deseja alcançar e os resultados esperados.

Por outro lado, no Capítulo 2, intitulado Embasamento Teórico, são abordados os conceitos básicos para o desenvolvimento do trabalho. Definição e tipos de matrizes, os métodos de solução de um sistema de equações lineares e seus critérios de convergência.

No Capítulo 3, é apresentado o software utilizado para execução das implementações. Também é exemplificado a forma como o software armazena as estruturas matriciais completas e suas formas compactas.

A descrição dos problemas de transportes de contaminantes propostos e sua caracterização para os casos estudados, são demonstrados no Capítulo 4. Nele, também, são definidas as formulações matemáticas, bem como as condições iniciais e de contornos dos problemas sob análise.

No Capítulo 5, é apresentado o Método dos Volumes Finitos (MVF), bem como a aplicação do mesmo na solução dos problemas de transportes de contaminantes propostos.

Os resultados obtidos e as comparações entre os valores numéricos das concentrações e os dados experimentais são expostos no Capítulo 6.

Por fim, no Capítulo 7, são apresentadas as conclusões do trabalho e discutidas possíveis melhorias para trabalhos futuros.

Capítulo 2

EMBASAMENTO TEÓRICO

Tanto na matemática, quanto em outras áreas da ciência, é comum a organização de dados ou números em formato de tabelas, ou seja, em linhas e colunas. Um exemplo, é apresentado através do problema da dieta de Cambridge, popular nos anos 80 e que se propunha a montar um regime de alimentação equilibrado para perda de peso [15]. A fórmula desta dieta consistia em uma combinação precisa de carboidratos, proteínas, gorduras, vitaminas, minerais elementos traços e eletrólitos. Para se atingir as proporções corretas, os ingredientes foram dispostos com suas respectivas quantidades de nutrientes, demonstrados na Tabela 2.1.

Tabela 2.1: Quantidades (gramas) fornecidas por 100 g de ingrediente.

Nutrientes (gramas)	Leite desnatado	Farinha de soja	Soro de leite	Qtd. fornecidas pela dieta de Cambridge
Proteína	36	51	13	33
Carboidrato	52	34	74	45
Gordura	0	7	1,1	3

Fonte: Adaptado de Lamin (2000) [15].

O problema básico seria determinar, por exemplo, a combinação de leite desnatado, farinha de soja e soro de leite, de forma a atingir a quantidade de nutrientes diário, estipulado pela dieta.

Uma das maneiras de resolução desse problema, utilizado à época, consistia em agrupar os valores mostrados na Tabela 2.1 em linhas e colunas, onde as três primeiras colunas, representavam, respectivamente, os nutrientes do leite desnatado, farinha de soja e soro de leite. Na última coluna, constava os valores da dieta de Cambridge. Tais estruturas são chamadas de matrizes.

Nas próximas seções são apresentados conceitos importantes em relação a esse tipo de estrutura, assim como alguns exemplos relevantes para o desenvolvimento desta pesquisa.

2.1 Matrizes

Por definição, matriz é um conjunto de números ou dados, referenciados através de linhas e colunas, apresentada entre parênteses ou colchetes [8].

De maneira geral, uma matriz é denotada por uma letra maiúscula (por exemplo, A , B , M , etc. – nesse texto é adotada a nomenclatura para as matrizes considerando letras maiúsculas e em itálico). Quando é necessário especificar a ordem (tamanho) de uma matriz genérica M , conforme exposto na Equação (2.1), escreve-se $M_{m \times n}$, onde m é o número de linhas e n é o número de colunas da referida matriz [8]. Na Equação (2.2), tem-se um exemplo de uma matriz M de ordem 3×4 , ou seja, $M_{3 \times 4}$.

$$M = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & \cdots & a_{3,n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & a_{m,4} & \cdots & a_{m,n} \end{bmatrix} \quad (2.1)$$

$$M = \begin{bmatrix} 5 & 1 & 3 & 2 \\ 9 & 3 & 4 & 8 \\ 0 & 7 & 2 & 6 \end{bmatrix} \quad (2.2)$$

O posicionamento dos elementos da matriz pode ser indicado em termos de linhas e colunas. Para isso, usa-se o termo geral $(a_{i,j})$, em que i representa a posição na linha e j a posição na coluna [15]. Usando como exemplo o elemento 4, que pertence à matriz da Equação (2.2), o mesmo pode ser representado como $a_{2,3}$, ou seja, está localizado na linha 2 e coluna 3.

Em particular, quando o número de linhas da matriz é igual ao número de colunas ($m = n$), ela recebe o nome de matriz quadrada [8] e diz-se que a mesma é de ordem n . Em termos gerais, a matriz quadrada está representada na Equação (2.3).

$$M = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \quad (2.3)$$

2.1.1 Tipos de Matrizes

Alguns tipos de matrizes possuem características especiais que auxiliam na interpretação de dados e facilitam o cálculo de determinadas operações.

- **Matriz Linha**

É aquela formada por apenas uma linha, como exemplificado na Equação (2.4), onde é apresentada uma matriz linha L contendo 4 colunas, cuja ordem é dada por $L_{1 \times 4}$.

$$L = \begin{bmatrix} 5 & 1 & 3 & 2 \end{bmatrix} \quad (2.4)$$

Usualmente, a matriz linha também é chamada de vetor. Nesse texto, para diferenciar as matrizes que possuem o número de linhas e colunas maiores ou iguais a 2 em relação aos vetores, estes últimos são denotados por letras minúsculas e em negrito (em alguns casos particulares, são adotadas letras maiúsculas e em negrito, devidamente mencionadas no texto). Nesse contexto, o exemplo apresentado na Equação (2.4), passa a ser escrito como:

$$\mathbf{l} = \begin{bmatrix} 5 & 1 & 3 & 2 \end{bmatrix} \quad (2.5)$$

- **Matriz Coluna**

É aquela formada por apenas uma coluna. Um exemplo de matriz coluna C com 3 linhas, está representado na Equação (2.6), cuja ordem é dada por $C_{3 \times 1}$.

$$C = \begin{bmatrix} 5 \\ 9 \\ 0 \end{bmatrix} \quad (2.6)$$

Assim como no caso da matriz linha, em determinadas situações, tais como, para indicar um plano ou espaço, a matriz coluna pode se considerada um vetor. Seguindo a estrutura estabelecida no tópico anterior, o exemplo apresentado na Equação (2.6), passa a ser escrito como:

$$\mathbf{c} = \begin{bmatrix} 5 \\ 9 \\ 0 \end{bmatrix} \quad (2.7)$$

- **Matriz Diagonal**

É uma matriz quadrada em que todos os elementos fora da diagonal principal são nulos. Ou seja, $a_{i,j} = 0$, se $i \neq j$, como exemplificado na Equação (2.8).

$$D = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix} \quad (2.8)$$

- **Matriz Identidade**

É uma matriz diagonal, em que todos os elementos da diagonal principal são iguais a 1. Ou seja, $a_{i,j} = 0$, se $i \neq j$ e $a_{i,j} = 1$, se $i = j$. Um exemplo de matriz identidade é apresentado na Equação (2.9).

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.9)$$

- **Matriz Triangular**

É um tipo de matriz em que todos os elementos acima ou abaixo da diagonal principal são nulos. No presente trabalho, dois tipos de matrizes triangulares são importantes:

- Matriz triangular superior: Matriz de ordem n em que os elementos $a_{i,j}$ são nulos, para $i > j$. Esse tipo de matriz é mostrado na Equação (2.10).

$$T_S = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ 0 & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n,n} \end{bmatrix} \quad (2.10)$$

- Matriz triangular inferior: Matriz de ordem n em que os elementos $a_{i,j}$ são nulos, para $i < j$, como indicado na Equação (2.11).

$$T_I = \begin{bmatrix} a_{1,1} & 0 & \cdots & 0 \\ a_{2,1} & a_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \quad (2.11)$$

• Matriz Esparsa

É um tipo de matriz em que o número de elementos não nulos é muito menor quando comparado com o tamanho da matriz. Por definição:

- Matriz de ordem n é dita esparsa se o número de elementos não nulos, em cada linha for inferior a um valor k , relativamente pequeno em comparação a sua ordem [1].
- O número de elementos não nulos de uma matriz pode ser representado a partir da seguinte expressão, mostrada na Equação (2.12):

$$nz = n^{1+\gamma}, \quad -1 \leq \gamma \leq 1, \quad (2.12)$$

onde n representa a ordem da matriz e o parâmetro γ determina a quantidade de elementos não nulos. Quando $\gamma = -1$, a matriz de ordem n possui apenas 1 elemento não nulo e quando $\gamma = 1$, a matriz é completa, ou seja, o número de elementos não nulos é igual a ordem da matriz [6].

Para uma matriz esparsa, o número de elementos não nulos deve ser muito menor que a sua ordem e, em particular, nesse trabalho acadêmico, ele deve aparecer, ao menos, em uma diagonal da matriz completa. Nesse sentido, γ pode variar de zero a valores próximos de zero ($0 \leq \gamma \ll 1$) [1].

- Grau de esparsidade de uma matriz é a porcentagem de elementos nulos, em relação ao número total de elementos [1], calculado através da Equação (2.13), onde $n^{1+\gamma}$ representa o número de elementos não nulos e n a ordem da matriz.

$$E = \frac{n^2 - n^{1+\gamma}}{n^2} \cdot 100 \quad (2.13)$$

O parâmetro γ , também pode ser usado para especificar o grau de esparsidade de uma matriz [6]. Usando como exemplo uma matriz esparsa aleatória de ordem $n = 150$, são realizadas variações em γ , como especificado na Tabela 2.2.

Tabela 2.2: Variação do parâmetro γ para uma matriz de ordem $n = 150$.

γ	$n^{1+\gamma}$	E
0,01	157	99,3%
0,1	247	98,9%
0,5	1.836	91,84%
0,9	13.631	39,41%

Fonte: O Autor (2023).

Na Figura 2.1 é apresentada a estrutura, criada através do software MATLAB, das matrizes esparsas de ordem $n = 150$, em que os valores de γ são variados de acordo com os dados da Tabela 2.2. É possível perceber que, quanto mais perto de 1 for o valor de γ , menor é o grau de esparsidade da matriz. Da mesma forma, quanto mais próximo de 0 for o valor de γ , mais esparsa é a matriz.

Em particular, três tipos de matrizes esparsas, especialmente importantes para a presente pesquisa, são:

- **Matriz Tridiagonal**

Matriz esparsa, que possui elementos não nulos em apenas três diagonais da matriz, conforme Figura 2.2(a).

- **Matriz Pentadiagonal**

Possui elementos não nulos em apenas cinco diagonais da matriz, conforme Figura 2.2(b) ¹.

¹Alguns autores consideram matrizes pentadiagonais somente aquelas em que as diagonais dos elementos não nulos são adjacentes à diagonal principal.

• Matriz Heptadiagonal

Possui elementos não nulos em apenas sete diagonais da matriz, conforme Figura 2.2(c) ².

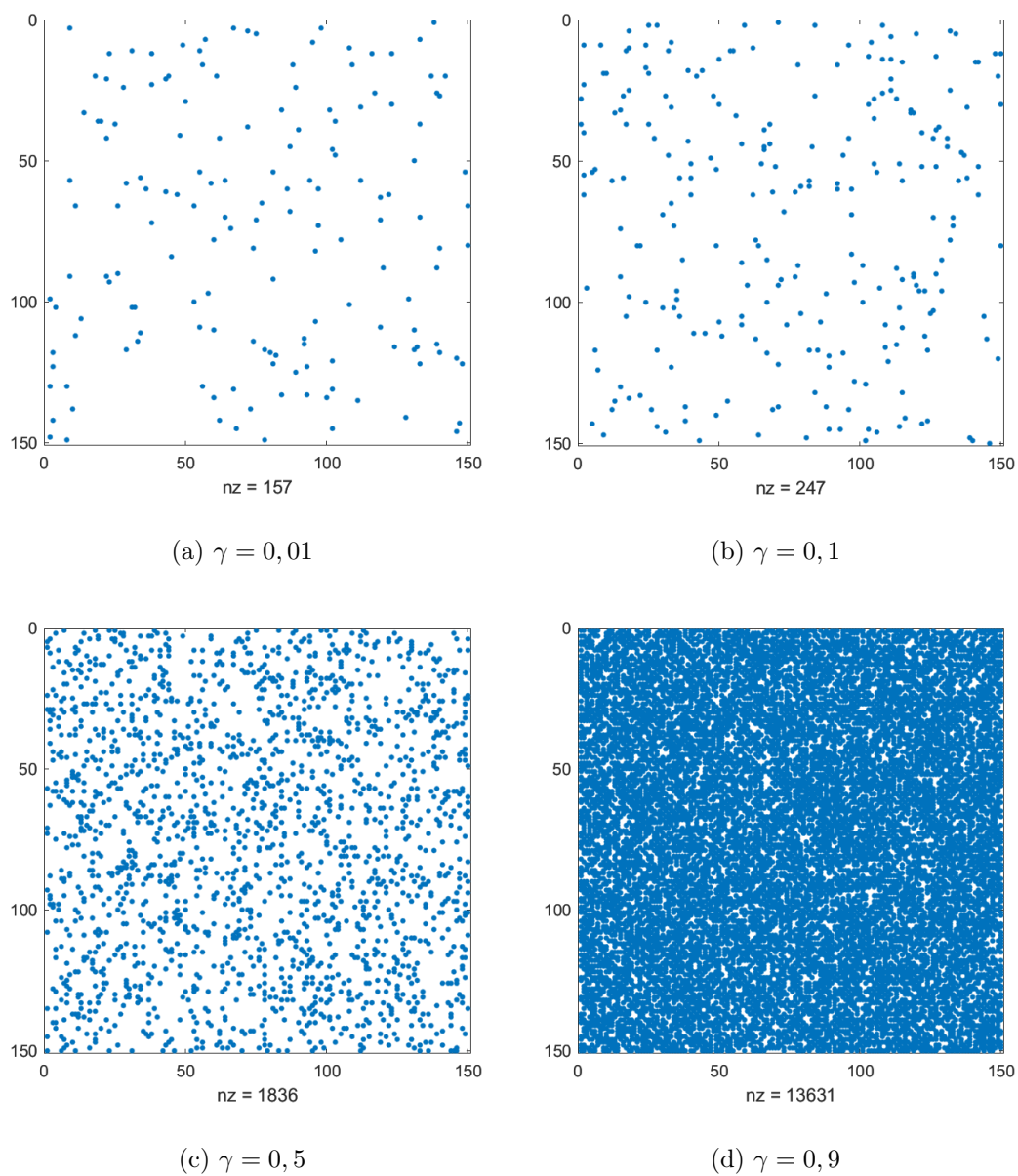


Figura 2.1: Visualização da matriz esparsa com variação do parâmetro γ .

Fonte: O Autor (2023).

²Alguns autores consideram matrizes heptatadiagonais somente aquelas em que as diagonais dos elementos não nulos são adjacentes à diagonal principal.

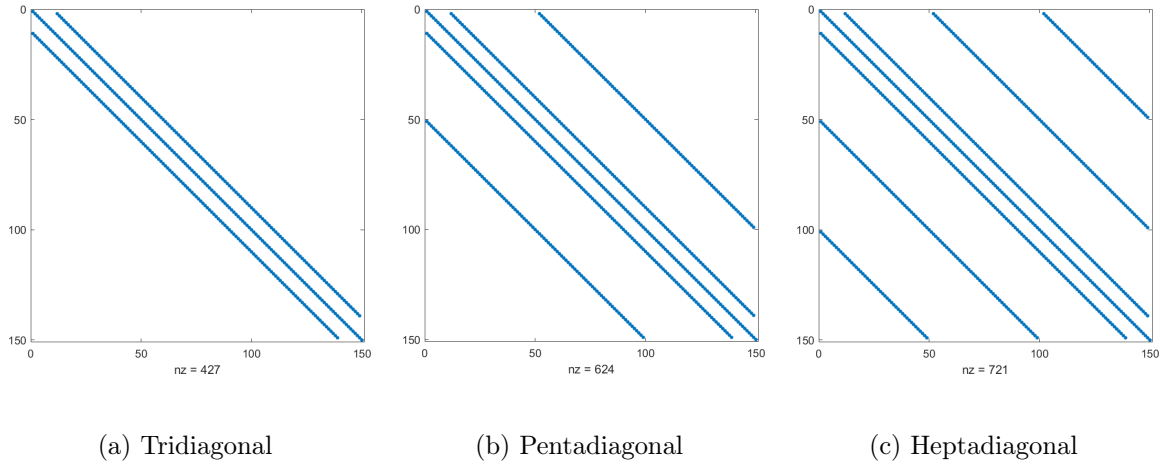


Figura 2.2: Visualização da matriz esparsa.

Fonte: O Autor (2023).

Nos tipos de matrizes, exemplificados nas Figuras 2.2(a), 2.2(b) e 2.2(c), os elementos não nulos crescem proporcionalmente à ordem da matriz, ou seja, para uma matriz quadrada, quanto maior for sua ordem, maior será o grau de esparsidade.

• Matriz Inversa

Considerando A uma matriz quadrada de ordem n , sua matriz inversa é aquela que multiplicada comutativamente por A , resulte na matriz identidade, ou seja:

$$A A^{-1} = I \text{ e } A^{-1} A = I \quad (2.14)$$

Satisfeita a Equação (2.14), diz-se que a matriz A é invertível. Algumas propriedades de matrizes inversas são apresentadas abaixo:

- A matriz inversa é única;

Sejam C e D matrizes inversas de A , então:

$$CA = I \text{ e } CD = I \quad (2.15)$$

Multiplicando D à direita em ambos os lados da primeira igualdade da Equação (2.15), tem-se:

$$CAD = ID \quad (2.16)$$

Como AD é igual I , então:

$$CI = ID \quad (2.17)$$

Ou seja,

$$C = D \quad (2.18)$$

- A matriz inversa da inversa de A é a própria matriz A ;

$$(A^{-1})^{-1} = A \quad (2.19)$$

- O inverso do produto escalar da matriz por um número (diferente de zero) é igual à matriz inversa multiplicada pelo inverso desse número.

$$(c \cdot A)^{-1} = \frac{1}{c} A^{-1} \quad (2.20)$$

2.2 Sistema de Equações Lineares

Uma equação que possui incógnitas com potência máxima igual a um, multiplicadas por constantes, é chamada de equação linear. Por definição matemática, uma equação linear, de n variáveis, pode ser escrita da seguinte forma [8]:

$$a_1 x_1 + a_2 x_2 + \cdots + a_{n-1} x_{n-1} + a_n x_n = b, \quad a_i \text{ e } b \in \mathbb{R}, \quad i = 1, 2, \cdots, n \quad (2.21)$$

Os elementos $a_1, a_2, \cdots, a_{n-1}, a_n$, são os coeficientes das variáveis $x_1, x_2, \cdots, x_{n-1}, x_n$, e b é o termo independente.

Por outro lado, um sistema de equações lineares, é um conjunto de equações lineares. De forma geral, um sistema linear com n equações e n incógnitas, está representado na Equação (2.22).

$$\begin{cases} a_{1,1} x_1 + a_{1,2} x_2 + \cdots + a_{1,n} x_n = b_1 \\ a_{2,1} x_1 + a_{2,2} x_2 + \cdots + a_{2,n} x_n = b_2 \\ \vdots \\ a_{n,1} x_1 + a_{n,2} x_2 + \cdots + a_{n,n} x_n = b_n \end{cases} \quad (2.22)$$

A Equação (2.22) pode ser representada na forma matricial, em que os coeficientes das variáveis formam uma matriz quadrada de ordem n . As variáveis formam um vetor, assim como os termos independentes, conforme mostrado na Equação (2.23).

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad (2.23)$$

O sistema de equações na forma matricial, como apresentado na Equação (2.23), é do tipo $A \mathbf{x} = \mathbf{b}$, onde, A é a matriz dos coeficientes, \mathbf{x} o vetor das incógnitas e \mathbf{b} o vetor dos termos independentes, ou seja:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

2.3 Solução de um Sistema de Equações Lineares

Vários problemas nas áreas da física e da engenharia resultam em um sistema de equações lineares. Neste sentido, é importante conhecer métodos para solucionar esse tipo de sistema. Por definição, dado um sistema de equações na forma matricial $A \mathbf{x} = \mathbf{b}$, como indicado na Equação (2.23), procura-se um vetor solução $\bar{\mathbf{x}}_j = [\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n]$ tal que satisfaça todas as equações do sistema [10].

Em relação ao número de soluções, o sistema linear pode ser classificado de três formas, são elas:

- **Compatível e determinado:** quando há uma única solução para o sistema;
- **Compatível e indeterminado:** quando há infinitas soluções para o sistema;

- **Incompatível:** quando não existe solução para o sistema.

De maneira formal, para resolver o sistema de equações, do tipo $A\mathbf{x} = \mathbf{b}$, basta calcular a matriz inversa de A , como demonstrado na Equação (2.24).

$$A^{-1} A\mathbf{x} = A^{-1} \mathbf{b} \quad \rightarrow \quad \mathbf{x} = A^{-1} \mathbf{b} \quad (2.24)$$

Uma vez comentados os conceitos importantes em relação à estrutura matricial, pode-se voltar ao problema da dieta de Cambridge, apresentado no início deste capítulo. Uma das possibilidades de resolução desse problema é escrevê-lo através de um sistema de equações do tipo $A\mathbf{x} = \mathbf{b}$, em que A é matriz dos nutrientes, x_1 , x_2 e x_3 representam o número de unidades (para cada 100 g) dos tipos alimentares e \mathbf{b} o vetor com o total de nutrientes da dieta de Cambridge, como demonstrado na Equação (2.25).

$$\begin{cases} 36x_1 + 51x_2 + 13x_n = 33 \\ 52x_1 + 34x_2 + 74x_n = 45 \\ 0x_1 + 7x_2 + 1,1x_n = 3 \end{cases} \quad (2.25)$$

O sistema demonstrado na Equação (2.25) calcula a quantidade de unidades, de cada tipo alimentar, necessária para se atingir os valores diários da dieta de Cambridge. Apresentando o sistema na forma matricial, tem-se:

$$\begin{bmatrix} 36 & 51 & 13 \\ 52 & 34 & 74 \\ 0 & 7 & 1,1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_n \end{bmatrix} = \begin{bmatrix} 33 \\ 45 \\ 3 \end{bmatrix} \quad (2.26)$$

Usando a ideia presente na Equação (2.24), tem-se:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_n \end{bmatrix} = \begin{bmatrix} 36 & 51 & 13 \\ 52 & 34 & 74 \\ 0 & 7 & 1,1 \end{bmatrix}^{-1} \begin{bmatrix} 33 \\ 45 \\ 3 \end{bmatrix} \quad (2.27)$$

Analisando a Equação (2.27), a resolução do sistema consiste em calcular a matriz inversa de A , resultando em:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_n \end{bmatrix} = \begin{bmatrix} \frac{2403}{77434} & \frac{-349}{154868} & \frac{-1190}{5531} \\ \frac{143}{38717} & \frac{-99}{38717} & \frac{710}{5531} \\ \frac{-130}{5531} & \frac{90}{5531} & \frac{510}{5531} \end{bmatrix} \begin{bmatrix} 33 \\ 45 \\ 3 \end{bmatrix} \quad (2.28)$$

Agora, basta multiplicar a matriz inversa presente na Equação (2.28) pelo vetor \mathbf{b} para se obter o resultado a seguir:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0,277 \\ 0,392 \\ 0,233 \end{bmatrix} \quad (2.29)$$

A dieta de Cambridge é um exemplo simples, de sistemas de equações lineares, que pode ser resolvido com o auxílio da notação matricial e do cálculo da matriz inversa de A . O grande problema de se obter a solução utilizando essa estratégia, é que nem sempre é fácil determinar de forma explícita A^{-1} e, para alguns casos, pode se mostrar bastante custoso computacionalmente [10]. Com objetivo de contornar esse problema, foram desenvolvidos métodos para solucionar sistemas de equações lineares sem a necessidade de se calcular, explicitamente A^{-1} , sendo os mesmos divididos em dois grupos: diretos e iterativos, os quais são descritos na sequência.

2.3.1 Métodos Diretos

A estratégia dos métodos diretos é transformar o sistema $A\mathbf{x} = \mathbf{b}$ em outro sistema equivalente e, quando não há erros de arredondamento, obter a solução exata através de um número finito de processos. A principal vantagem dos métodos diretos é que a matriz equivalente, criada a partir do sistema original, possui características que tornam sua solução mais simples [10].

Nesse trabalho, é utilizado o método direto Algoritmo de Thomas ou Algoritmo da Matriz Tridiagonal (o inglês *Tridiagonal Matrix Algorithm* - TDMA). Porém, antes de conceituar o referido método, torna-se necessária uma breve contextualização envolvendo a eliminação de Gauss, uma vez que, apesar de não ser usado nessa pesquisa, este último é a base para o desenvolvimento do TDMA.

2.3.1.1 Eliminação de Gauss

Um dos métodos diretos mais utilizados para solução de sistemas de equações lineares é a eliminação de Gauss. Tem por objetivo realizar operações no sistema, de forma a transformar a matriz dos coeficientes em uma matriz triangular superior. Nesta configuração, aplica-se a técnica de substituição reversa para solucionar o sistema de equações [2], como apresentado no pseudocódigo a seguir [4].

- 1: Pseudocódigo do método da eliminação de Gauss:
- 2: Dado um sistema de equações do tipo $A\mathbf{x} = \mathbf{b}$, onde $A = (a_{i,j})$ com $1 \leq i \leq n$
- 3: e $1 \leq j \leq n + 1$
- 4: **Passo 1:** para $i = 1$ até $n - 1$ **faça** Passo 2-4 (processo de eliminação)
- 5: **Passo 2:** seja p o menor inteiro com $1 \leq p \leq n$ e a_{pi}
- 6: **se** nenhum inteiro p puder se encontrado
- 7: **então** escreva "não existe solução única"
- 8: **Passo 3:** se $p \neq i$ **então** executar $E_p \leftrightarrow E_i$
- 9: **Passo 4:** para $j = i + 1$ até n **faça** Passo 5-6
- 10: **Passo 5:** seja $m_{ij} = a_{ji}/a_{ii}$
- 11: **Passo 6:** executar $(E_j - m_{ij} E_i) \rightarrow (E_j)$
- 12: **Passo 7:** se $a_{nn} = 0$ **então** escreva "não existe solução única"; **pare método**
- 13: **Passo 8:** executar $x_n = \frac{a_{n,n+1}}{a_{nn}}$ (começo da substituição regressiva)
- 14: **Passo 9:** para $i = n - 1$ até 1 executar $x_i = (a_{i,n+1} - \sum_{j=i+1}^n a_{ij} x_j)/a_{ii}$
- 15: **Passo 10:** Escreva (x_1, \dots, x_n) ; "Método completado com sucesso";
- 16: **pare método**

Em sistemas esparsos aleatórios, a eliminação de Gauss não se mostra vantajosa devido ao consumo de processamento do computador. Os passos realizados para transformar a matriz dos coeficientes em triangular superior, além de não aproveitar a esparsidade da matriz, alteram elementos nulos para valores que devem ser computados e armazenados, o que demanda um grande custo computacional [10].

Em alguns casos, onde a esparsidade da matriz segue algum padrão específico, como por exemplo, matrizes tridiagonal (Figura 2.2(a)) e pentadiagonal (Figura 2.2(b)), os elementos não nulos ocorrem em regiões específicas da matriz dos coeficientes, o que pode facilitar o processo de eliminação.

No presente trabalho, uma das implementações utilizada para solucionar o problema

proposto, é derivada do conceito da eliminação de Gauss, conhecida como Algoritmo de Thomas, e que torna a convergência para a solução mais eficiente.

2.3.1.2 Algoritmo de Thomas (TDMA)

O Algoritmo de Thomas (TDMA) é originalmente desenvolvido para solucionar sistemas com matrizes esparsas tridiagonais. Utiliza o mesmo princípio de eliminação de Gauss, o qual consiste em zerar termos da matriz dos coeficientes de forma a transformá-la em uma matriz triangular e, através de uma substituição regressiva, obter a solução do sistema [11].

Considere um sistema linear tridiagonal, apresentado na Equação (2.30), em que a_i ($1 \leq i \leq n$) representa os elementos da diagonal principal, c_i ($2 \leq i \leq n$) e b_i ($1 \leq i \leq n - 1$) representam os elementos das diagonais inferior e superior à diagonal principal, respectivamente, e d_i ($1 \leq i \leq n$) os termos independentes.

$$\begin{bmatrix} a_1 & b_1 & & & \\ c_2 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-1} & a_{n-1} & b_{n-1} \\ & & & c_n & a_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \quad (2.30)$$

Na sua forma geral, o sistema matricial representado na Equação (2.30), é dado por:

$$c_i x_{i-1} + a_i x_i + b_i x_{i+1} = d_i, \quad i = 1, \dots, n \quad (2.31)$$

O objetivo é transformar a matriz da Equação (2.30) em uma matriz triangular superior. Para isso, cria-se uma equação genérica (Equação (2.32)) que será substituída na equação geral (Equação (2.31)), de forma a deixá-la em função do termo x_{i+1} .

$$x_i = b'_i x_{i+1} + d'_i \quad (2.32)$$

Na Equação (2.32), b' é o novo coeficiente da variável x_{i+1} e d' o novo termo independente, ambos serão calculados a seguir. Deslocando a Equação (2.32), para a direção $i - 1$, tem-se:

$$x_{i-1} = b'_{i-1} x_i + d'_{i-1} \quad (2.33)$$

Substituindo a Equação (2.33) em (2.31), resulta em:

$$a_i x_i + b_i x_{i+1} + c_i (b'_{i-1} x_i + d'_{i-1}) = d_i \quad (2.34)$$

Isolando o termo x_i da Equação (2.34), tem-se:

$$x_i = \frac{-b_i}{a_i + c_i b'_{i-1}} x_{i+1} + \frac{d_i - c_i d'_{i-1}}{a_i + c_i b'_{i-1}} \quad (2.35)$$

Comparando as Equações (2.35) e (2.32), resulta em:

$$b'_i = \frac{-b_i}{a_i + c_i b'_{i-1}}; \quad d'_i = \frac{d_i - c_i d'_{i-1}}{a_i + c_i b'_{i-1}} \quad (2.36)$$

A resolução da Equação (2.32), resulta em uma matriz tridiagonal superior, como indicado na Equação (2.37).

$$\begin{bmatrix} 1 & b'_1 & & & \\ 0 & 1 & b'_2 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & 1 & b'_{n-1} \\ & & & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d'_1 \\ d'_2 \\ \vdots \\ d'_{n-1} \\ d'_n \end{bmatrix} \quad (2.37)$$

Analisando a Equação (2.37), o cálculo da incógnita x_n , localizada na última linha do sistema, sai de forma direta, ou seja:

$$x_n = d'_n \quad (2.38)$$

A partir do valor de x_n , realiza-se uma substituição regressiva para o cálculo das demais incógnitas.

$$x_i = b'_i x_{i+1} + d'_i, \quad i = n-1, n-2, \dots, 1 \quad (2.39)$$

Em resumo, as expressões utilizadas para o escalonamento da matriz dos coeficientes, de forma a transformá-la em triangular superior e, conseqüentemente, obter a resolução do sistema de equações, são:

- Atualização da diagonal superior à principal.

$$b'_i = \begin{cases} \frac{-b_i}{a_i}, & i = 1 \\ \frac{-b_i}{a_i + c_i b'_{i-1}}, & i = 2, 3, \dots, n-1 \end{cases} \quad (2.40)$$

- Atualização do termo independente.

$$d'_i = \begin{cases} \frac{d_i}{b_i}, & i = 1 \\ \frac{d_i - a_i d'_{i-1}}{b_i + a_i c'_{i-1}}, & i = 2, 3, \dots, n \end{cases} \quad (2.41)$$

- Solução obtida por substituição reversa.

$$\begin{aligned} x_n &= d'_n \\ x_i &= b'_i x_{i+1} + d'_i, \quad i = n-1, n-2, \dots, 1 \end{aligned} \quad (2.42)$$

2.3.2 Métodos Iterativos

Quando os problemas geram matrizes dos coeficientes que não são esparsas ou que não possuem um padrão definido de esparsidade, outras classes de métodos se mostram mais eficientes para se chegar à solução, os métodos iterativos.

Dado um sistema do tipo $A\mathbf{x} = \mathbf{b}$, a solução do problema, através dos métodos iterativos, é obtida realizando uma sequência de correções aplicadas a uma aproximação da solução inicial [10]. Essa sequência de correções é feita por meio da equação:

$$\mathbf{x}^{(k+1)} = F \mathbf{x}^{(k)} + \mathbf{G} \quad (2.43)$$

Na Equação (2.43), F é a matriz do método de iteração e \mathbf{G} é um vetor coluna, ambos construídos através de decomposição da matriz original A . O parâmetro k , representa o número de iterações. Esta equação é usada para criar uma sequência de vetores: $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^k$. Se a sequência satisfaz a Equação (2.44),

$$\lim_{k \rightarrow \infty} \mathbf{x}^{(k)} = \mathbf{x} \quad (2.44)$$

ou seja, se no limite, onde o número de iterações tende ao infinito, a solução aproximada for igual a solução exata, então o método converge. Devido a limitação computacional, é determinado um critério de parada do método, dentro de uma certa tolerância da solução

exata.

Os métodos iterativos apresentam uma grande vantagem em relação aos métodos diretos, eles não são tão afetados pelos erros de arredondamento da máquina, o que possibilita uma maior flexibilidade para se trabalhar com valores muito pequenos [10].

De uma maneira geral, os diferentes métodos iterativos resultam da forma como a matriz A é decomposta [10]. Na presente pesquisa é utilizado o método iterativo de Gauss-Seidel, bem como o algoritmo de Thomas adaptado para matrizes pentadiagonal.

2.3.2.1 Método de Gauss-Seidel

O método iterativo de Gauss-Seidel possui grande eficiência em sistemas com matrizes esparsas que possuem dominância diagonal, ou seja, os elementos da diagonal principal são maiores que a soma dos elementos das suas respectivas linhas [10].

Considere um sistema de equações mostrado na Equação (2.45) e sua respectiva representação matricial (Equação (2.46)).

$$\begin{cases} a_{1,1} x_1 + a_{1,2} x_2 + \cdots + a_{1,n} x_n = y_1 \\ a_{2,1} x_1 + a_{2,2} x_2 + \cdots + a_{2,n} x_n = y_2 \\ \vdots \\ a_{n,1} x_1 + a_{n,2} x_2 + \cdots + a_{n,n} x_n = y_n \end{cases} \quad (2.45)$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad (2.46)$$

Tomando como premissa um processo iterativo para o cálculo do vetor \mathbf{x} , o primeiro passo é isolar o termo x_1 .

$$x_1^{(k+1)} = \frac{y_1 - \left(a_{1,2} x_2^{(k)} + \cdots + a_{1,n} x_n^{(k)} \right)}{a_{1,1}} \quad (2.47)$$

Para o cálculo do termo $x_1^{(k+1)}$, utiliza-se os valores da solução na iteração anterior ou

valores da aproximação inicial, quando tratar-se da primeira iteração $(x_2^{(k)}, \dots, x_n^{(k)})$.

No que se refere ao do termo $x_2^{(k+1)}$, utiliza-se o valor de $x_1^{(k+1)}$ calculado na atual iteração $(x_1^{(k+1)}, \dots, x_n^{(k)})$.

$$x_2^{(k+1)} = \frac{y_1 - (a_{2,1} x_1^{(k+1)} + \dots + a_{2,n} x_n^{(k)})}{a_{2,2}} \quad (2.48)$$

Estendendo o procedimento para o enésimo termo, tem-se:

$$x_n^{(k+1)} = \frac{y_1 - (a_{2,1} x_1^{(k+1)} + \dots + a_{2,n} x_{n-1}^{(k+1)})}{a_{n,n}} \quad (2.49)$$

De forma compacta, o método de Gauss-Seidel pode ser escrito, como:

$$x_i^{(k+1)} = \frac{1}{a_{i,i}} \left(y_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(k+1)} + \sum_{j=i+1}^n a_{i,j} x_j^{(k)} \right), \quad i = 1, 2, \dots, n \quad (2.50)$$

A Equação (2.50) é usada para resolver o sistema de equações lineares apresentado na Equação (2.46) percorrendo todos os elementos da matriz dos coeficientes. É possível notar que no primeiro somatório estão presentes as incógnitas calculadas na iteração atual $(x_i^{(k+1)})$ e, no segundo somatório, as incógnitas calculadas na iteração anterior $(x_i^{(k)})$.

Na forma matricial, o método de Gauss-Seidel pode ser representado como:

$$\mathbf{x}^{(k+1)} = L \mathbf{x}^{(k+1)} + U \mathbf{x}^{(k)} + \mathbf{B} \quad (2.51)$$

em que:

$$L = \begin{bmatrix} 0 & 0 & \dots & 0 \\ -\frac{a_{2,1}}{a_{2,2}} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{a_{n,1}}{a_{n,n}} & -\frac{a_{n,2}}{a_{n,n}} & \dots & 0 \end{bmatrix} \quad U = \begin{bmatrix} 0 & -\frac{a_{1,2}}{a_{1,1}} & \dots & -\frac{a_{1,n}}{a_{1,1}} \\ 0 & 0 & \dots & -\frac{a_{2,n}}{a_{2,2}} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} \frac{y_1}{a_{1,1}} \\ \frac{y_1}{a_{2,2}} \\ \vdots \\ \frac{y_n}{a_{n,n}} \end{bmatrix}$$

De acordo com a Equação (2.51), os elementos que estão abaixo da diagonal principal, ou seja, na matriz triangular inferior, representado pela letra L , são os coeficientes das incógnitas calculadas no passo de iteração atual $(x^{(k+1)})$. Já a matriz dos elementos que estão acima da diagonal principal, representados pela letra U , são os coeficientes das

incógnitas calculadas no do passo de iteração anterior $(x^{(k)})$.

A solução da Equação (2.51) é obtida realizando uma série de atualizações no vetor solução, a partir de um vetor inicial $\mathbf{x}^{(0)}$. O método termina quando o vetor solução converge para a solução exata ou uma boa estimativa da mesma, dentro de um critério previamente estabelecido. A quantidade de iterações do método, até a convergência, irá depender da aproximação inicial escolhida, a qual pode ser qualquer uma. Porém, quanto mais distante for a aproximação inicial da solução exata, maior será o número de iterações do método [10].

Por outro lado, a principal desvantagem dos métodos iterativos é a convergência. Em alguns casos, o sistema pode convergir lentamente para a solução demandando um custo computacional muito alto. Em outros casos, podem nem convergir.

Para a convergência do método de Gauss-Seidel à solução exata, caso ela exista, é preciso que a matriz dos coeficientes do sistema atenda alguns critérios [23], conforme descrito na sequência:

- **Critério das Linhas**

De acordo com o critério das linhas, o método converge, se:

$$|a_{i,i}| > \sum_{j=1}^n |a_{i,j}| \quad \forall i = 1, 2, \dots, n. \quad \text{Para } j \neq i. \quad (2.52)$$

O critério das linhas, descrito na Equação (2.52), sugere que a matriz dos coeficientes deve possuir predominância diagonal, ou seja, qualquer elemento da diagonal principal deve ser, em módulo, maior que a soma de todos os elementos da sua linha [23].

- **Critério de Sassenfeld**

De acordo com o critério de Sassenfeld, calcula-se um vetor β , que envolve os elementos da matriz dos coeficientes, como descrito na Equação (2.53):

$$\beta_1 = \sum_{j=2}^n \frac{|a_{1,j}|}{|a_{1,1}|},$$

$$\beta_i = \sum_{j=1}^{i-1} \frac{|a_{i,j}| \beta_j}{|a_{i,i}|} + \sum_{j=i+1}^n \frac{|a_{i,j}|}{|a_{i,i}|}, \quad \forall i = 1, 2, \dots, n \quad (2.53)$$

Se, $\beta_{max} = \max\{\beta_i\} < 1$, $1 \leq i \leq n$, o método converge para qualquer valor inicial escolhido [23].

• Critério do Raio Espectral

É o critério necessário para que o método de Gauss-Seidel convirja. Calcular o raio espectral de uma matriz é achar o máximo valor absoluto entre seus autovalores (ρ). O critério aplica-se à matriz de iteração, ou seja, na matriz dos coeficientes modificada. Exemplificando, tem-se a Equação (2.51), que representa a estrutura, na forma matricial, do método de Gauss-Seidel [23].

Juntando os termos com índices $(k + 1)$, tem-se:

$$\mathbf{x}^{(k+1)} - L \mathbf{x}^{(k+1)} = U \mathbf{x}^{(k)} + \mathbf{B} \quad (2.54)$$

Colocando o termo $\mathbf{x}^{(k+1)}$ em evidência:

$$(I - L)\mathbf{x}^{(k+1)} = U \mathbf{x}^{(k)} + \mathbf{B} \quad (2.55)$$

Isolando o termo $\mathbf{x}^{(k+1)}$:

$$\mathbf{x}^{(k+1)} = (I - L)^{-1}U \mathbf{x}^{(k)} + (I - L)^{-1}\mathbf{B} \quad (2.56)$$

Comparando com a Equação (2.43), que representa a forma genérica da equação dos métodos iterativos, tem-se:

$$\mathbf{x}^{(k+1)} = F \mathbf{x}^{(k)} + \mathbf{G}$$

em que:

$$F = (I - L)^{-1}U \quad (2.57)$$

$$\mathbf{G} = (I - L)^{-1}\mathbf{B} \quad (2.58)$$

onde, F é conhecida como a matriz de iteração do método de Gauss-Seidel. É nela que o critério do raio espectral é aplicado.

Se, $\rho(F) = \max\{|\lambda_i|\} < 1, i \in \{1, 2, \dots, n\}$, então o método converge. Ou seja, se o maior valor absoluto, entre os autovalores da matriz F for menor que 1, o método irá convergir.

Em relação aos critérios de convergência, é importante ressaltar que:

- Os critérios das Linhas e de Sassenfeld são suficientes para o método de Gauss-Seidel convergir, mas não necessários. Em outras palavras, se os critérios forem atendidos, o método irá convergir. Caso não sejam atendidos, o método poderá ou não convergir.
- O método do Raio Espectral não é só suficiente para o método de Gauss-Seidel convergir, mas necessário. Caso o critério não seja atendido, o método irá divergir.

2.3.2.2 Algoritmo de Thomas Adaptado para Matriz Pentadiagonal

Para a utilização do algoritmo de Thomas em matrizes pentadiagonais, se faz necessário ajustes no referido algoritmo, uma vez que sua formulação é pensada para uma matriz tridiagonal. Para o sistema com matriz pentadiagonal, a equação geral é dada por:

$$f_i x_{i-r} + c_i x_{i-1} + a_i x_i + b_i x_{i+1} + e_i x_{i+r} = d_i, \quad i = 1, 2, \dots, n \quad (2.59)$$

em que r é a distância das diagonais mais externas à diagonal principal.

Usando a mesma ideia da Seção 2.3.1.2, é preciso transformar a matriz dos coeficientes oriunda da Equação (2.59), em uma matriz triangular superior. Para isso, é criado um sistema de equações, em função dos coeficientes dos termos x_{i+1} e x_{i+r} .

$$\begin{cases} x_i = b'_i x_{i+1} + d'_i \\ x_i = e'_i x_{i+r} + g'_i \end{cases} \quad (2.60)$$

Deslocando as equações do sistema (Equação (2.60)), para as direções $i - 1$ e $x - r$, respectivamente, tem-se:

$$x_{i-1} = b'_{i-1} x_i + d'_{i-1} \quad (2.61)$$

$$x_{i-r} = e'_{i-r} x_i + g'_{i-r} \quad (2.62)$$

Substituindo as Equações (2.61) e (2.62), na Equação (2.59), resulta em:

$$a_i x_i + b_i x_{i+1} + c_i (b'_{i-1} x_i + d'_{i-1}) + e_i x_{i+r} + f_i (e'_{i-r} x_i + g'_{i-r}) = d_i \quad (2.63)$$

Reagrupando a Equação (2.63) e isolando o termo x_i , tem-se:

$$x_i = \frac{1}{a_i + c_i b'_{i-1} + f_i e'_{i-r}} (-b_i x_{i+1} + d_i - e_i x_{i+r} - c_i d'_{i-1} - f_i g'_{i-r}) \quad (2.64)$$

Somando o sistema de equações (2.60), resulta em:

$$x_i = \frac{1}{2} (b'_i x_{i+1} + e'_i x_{i+r} + d'_i + g'_i) \quad (2.65)$$

Comparando as Equações (2.65) e (2.64), tem-se:

$$b'_i = \frac{-2 b_i}{a_i + c_i b'_{i-1} + f_i e'_{i-r}} \quad (2.66)$$

$$e'_i = \frac{-2 e_i}{a_i + c_i b'_{i-1} + f_i e'_{i-r}} \quad (2.67)$$

$$d'_i + g'_i = 2 \frac{d_i - c_i d'_{i-1} - f_i g'_{i-r}}{a_i + c_i b'_{i-1} + f_i e'_{i-r}} \quad (2.68)$$

Na Equação (2.66), os termos d'_{i-1} e g'_{i-r} não são conhecidos, mas podem ser obtidos a partir das Equações (2.61) e (2.62), resultando em:

$$d'_i + g'_i = 2 \frac{d_i - c_i(x_{i-1} - b'_{i-1} x_i) - f_i(x_{i-r} - e'_{i-r} x_i)}{a_i + c_i b'_{i-1} + f_i e'_{i-r}} \quad (2.69)$$

A resolução da Equação (2.65) transforma a matriz dos coeficientes da Equação (5.90), em uma matriz triangular superior, e por substituição regressiva, calcula-se o sistema, ou seja:

$$\begin{cases} x_i = d'_i + g'_i, & i = n \\ x_i = \frac{1}{2}(b'_i x_{i+1} + e'_i x_{i+r} + d'_i + g'_i), & i = n-1, n-2, \dots, 1 \end{cases} \quad (2.70)$$

É importante analisar as regiões da matriz dos coeficientes, onde as Equações (2.66), (2.67) e (2.70) fazem sentido.

Capítulo 3

ASPECTOS COMPUTACIONAIS ENVOLVENDO MATRIZES

Com o advento do computador moderno, muitos problemas que envolviam extensas quantidades de cálculos repetitivos passaram a ser solucionados com grande precisão, através de modelos computacionais. Um modelo computacional é a ferramenta utilizada para realizar cálculos de métodos matemáticos, quando estes, devido a seu tamanho, são inviáveis de serem realizados manualmente. Um computador armazena e executa cálculos numéricos muito rapidamente, mas para isso, é preciso fornecer uma série de instruções, através de uma linguagem que a máquina possa interpretar.

Tanto para as operações matemáticas quanto para o armazenamento, o computador executa as tarefas de forma binária. Neste sentido, a linguagem utilizada para as instruções deve ser escrita da mesma forma [11]. Existem diversos programas que possuem interfaces intuitivas, permitindo um melhor contato entre a comunicação humana e a linguagem do computador. Na presente pesquisa, é utilizado o *MATLAB*®, um programa escrito em linguagens de alto nível que permite ao operador focar no problema a ser resolvido, sem demandar excessivo tempo com linhas de programação [11].

3.1 Software MATLAB

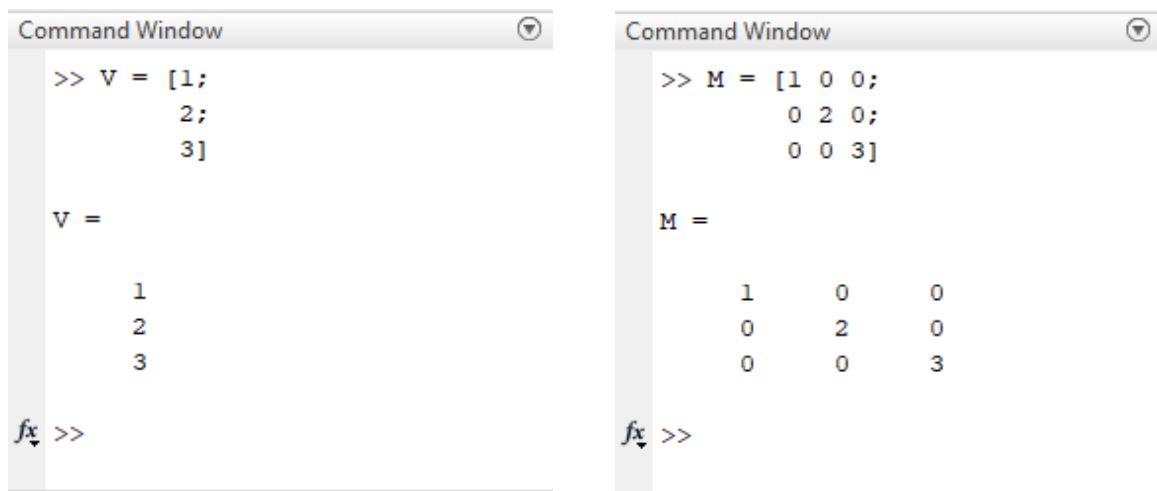
Programa desenvolvido com linguagem de alto nível, utilizado para análise e visualização de dados, de forma avançada. O MATLAB, aglutinação da expressão *MATrix LABoratory*, possui uma linguagem de programação própria, que dispensa tarefas como: declaração de variáveis, utilização de ponteiros nas estruturas e alocação de memória. A grande vantagem da linguagem MATLAB, em relação às demais linguagens de programa-

ção, é que os problemas são expressados quase exatamente como são escritos matematicamente [27], o que otimiza o tempo de criação do programa.

Na sequência, são expostas algumas estruturas que podem ser construídas no MATLAB, as quais são utilizadas para o desenvolvimento dessas pesquisas.

3.1.1 Vetores e Matrizes no MATLAB

Como o próprio nome sugere, o MATLAB foi criado e desenvolvido para trabalhar com vetores e matrizes. Possui ferramentas que otimizam os cálculos e armazenamentos desse tipo de estrutura. Na Figura 3.1 é mostrada a linha de programação para se criar um vetor e uma matriz no ambiente do programa.



(a) Vetor \mathbf{v} .

(b) Matriz M .

Figura 3.1: Exemplo da linha de programação para criação de estruturas matriciais no MATLAB.

Fonte: O Autor (2023).

Diferentemente de outras linguagens de programação, em que as operações algébricas com matrizes demandam linhas de comando para se percorrer toda estrutura, no ambiente do MATLAB esse processo é mais simples, pois os comandos utilizam operadores comuns na escrita matemática. Um exemplo de multiplicação de matrizes, é demonstrado na Figura 3.2, onde é feita a multiplicação da matriz M com o vetor \mathbf{v} , ambos representados na Figura 3.1.

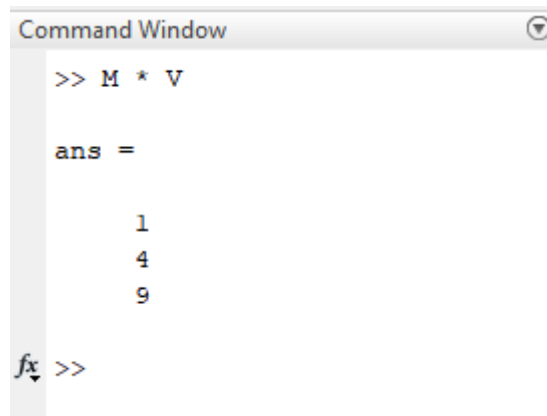


Figura 3.2: Exemplo de multiplicação de matrizes no MATLAB.
Fonte: O Autor (2023).

Outra grande vantagem do MATLAB é que não há necessidade de se reservar um espaço, na memória do programa, para alocar as estruturas matriciais que serão utilizadas. Essa alocação é feita de forma automática pelo software, uma vez que toda matriz, à princípio, é considerada adimensional pelo programa.

3.1.2 Alocação de Matrizes no MATLAB

O computador armazena números de forma binária, ou seja, representado por sequências de 0 e 1, multiplicados por potências de base 2 [11]. Na Tabela 3.1 é mostrada a conversão dos números decimais para representação binária de base 2.

Tabela 3.1: Conversão de números decimais em binário.

Decimal	2^3	2^2	2^1	2^0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	1	0
9	1	0	0	1
10	1	0	1	0

Fonte: Adaptado de Gilat (2009) [11].

A aritmética binária define posições para os algarismos binários. A posição "li-

gada" corresponde ao Algarismo 1 e, a posição "desligada", ao Algarismo 0. Cada um desses algarismos recebe o nome de *bit*, e um conjunto de 8 *bits* é chamado de *byte*. De acordo com a norma 754, do *Instituto de Engenheiros Eletricistas e Eletrônicos*, os computadores modernos podem armazenar números de duas maneiras diferentes:

- **Precisão simples:** Neste tipo de precisão, os números são armazenados em cadeia de 32 *bits* ou 4 *bytes*.
- **Precisão dupla:** Neste tipo de precisão, os dados são armazenados em cadeia de 64 *bits* ou 8 *bytes*.

Em particular, todos os testes desenvolvidos na presente pesquisa foram realizados em um sistema operacional com arquitetura de armazenamento de precisão dupla, ou seja, em cadeias de 8 *bytes*. Nesta estrutura, o menor e o maior número que o computador pode expressar, são:

- Menor número: $2^{-1023} \approx 1,1 \times 10^{308}$.
- Maior número: $2^{1024} \approx 1,8 \times 10^{308}$.

Quando o MATLAB armazena os valores de uma matriz, para cada elemento, são utilizados 8 *bytes* de memória da máquina. Na Tabela 3.2 é mostrada a relação entre a ordem da matriz e a quantidade de memória necessária para alocação da mesma.

Tabela 3.2: Relação entre a ordem da matriz quadrada e a memória necessária para seu armazenamento.

Ordem da matriz	Memória utilizada (<i>bytes</i>)
10	800
100	80.000
1.000	8.000.000
10.000	800.000.000

Fonte: O Autor (2023).

Ao analisar os dados da Tabela 3.2, verifica-se que a demanda de memória cresce exponencialmente em relação à ordem da matriz, o que dificulta a resolução de sistemas extensos. Esse crescimento independe do algarismo armazenado, dentro dos limites de máximo e mínimo, estabelecido pela precisão dupla. Ou seja, para uma matriz de ordem

$n = 100$, se todos os seus elementos forem 0, ainda sim, necessitaria de 80.000 *bytes* para armazená-la.

Outro grande problema, quando se trabalha com matrizes extensas, é o tempo necessário para que o computador crie, realize operações e leia essas matrizes. Na computação, o cálculo do tempo de um algoritmo depende dos parâmetros de *hardware*, os quais são os componentes físicos do computador, podendo variar de máquina para máquina. Diante disso, com objetivo de padronizar as análises de tempo, independente dos componentes do computador, utiliza-se um parâmetro conhecido como **complexidade do tempo**, representado pela notação O [23].

O grau de complexidade não calcula diretamente o tempo de execução de um programa, mas a quantidade de operações elementares $O(1)$, onde o tempo de execução é constante, que são realizadas pelo algoritmo. Através desse grau é possível intuir o tempo total de execução [25].

Nos métodos diretos de resolução de sistemas lineares, como por exemplo, a eliminação de Gauss, o grau de complexidade é $O(n^3)$, ou seja, o tempo de execução do programa é proporcional ao cubo da ordem [25]. Assim, se um computador precisa de 1×10^{-7} segundos para executar uma operação elementar em uma matriz de ordem $n = 100$, esse tempo será próximo de 0,1 segundos, uma vez que esse cálculo será feito n^3 vezes.

Em relação aos métodos iterativos, a complexidade do tempo não é trivial de ser determinada, uma vez que não é possível prever o número de iterações para convergência do método. Porém, como o cálculo da complexidade é baseado no número de operações que são realizadas no programa e, nos métodos iterativos essas operações são executadas percorrendo toda a estrutura, é de se esperar que quanto maior for a ordem da matriz maior será o tempo de execução do método.

3.1.3 Alocação de Matrizes Esparsas no MATLAB

Como já discutido, uma matriz esparsa possui um número muito grande de elementos nulos. Em matrizes completas ou densas, cada elemento deverá ser armazenado na memória do computador independente do valor. Porém, para a resolução de sistemas lineares do tipo $A\mathbf{x} = \mathbf{b}$, os elementos nulos podem ser dispensados do cálculo, evitando consumo de memória da máquina e desperdício no tempo de execução.

O MATLAB possui funções que aproveitam a esparsidade da matriz, armazenando somente os elementos diferentes de zero, reduzindo significativamente a quantidade neces-

sária de memória para a resolução do problema [12]. Nas Tabelas 3.3 e 3.4, são apresentados os dados de consumo de memória para armazenamento de matrizes esparsas, variando parâmetros como: a ordem da matriz e grau de esparsidade da mesma através de γ .

Tabela 3.3: Economia no armazenamento de memória em relação à ordem da matriz.

$\gamma = 0,01$ n	Memória Utilizada (<i>byte</i>)		Economia de Memória
	Completa	Esparsa	
10	800	104	87%
100	80.000	968	98,79%
1.000	8.000.000	24.008	99,69%
10.000	800.000.000	1.679.128	99,80%

Fonte: O Autor (2023).

De acordo com a Tabela 3.3, fixado o parâmetro em $\gamma = 0,01$ e alterando os valores da ordem da matriz, fica evidente que em sistemas com alto grau de esparsidade a utilização da função do MATLAB apresenta uma grande economia de memória do computador. Ainda, de acordo com a referida tabela, quanto maior for a ordem da matriz, mais vantajosa é a utilização da função.

Uma desvantagem da função do MATLAB para matrizes esparsas é que, além do gasto de memória para armazenar os elementos não nulos da matriz, ela também utiliza a memória para armazenar os índices de linhas e colunas desses elementos, o que inviabiliza sua utilização para matrizes com baixo grau de esparsidade. Na Tabela 3.4, onde a ordem da matriz é fixada em $n = 100$ e variado os valores de γ , pode-se observar que a diminuição do grau de esparsidade aumenta o consumo de memória do programa. Para valores de γ muito próximos de 1, esse consumo apresenta um deficit em relação ao armazenamento da matriz completa.

A função nativa do MATLAB armazena, de forma automática, os índices referentes ao posicionamento de cada elemento não nulo. Com isso, as operações algébricas que envolvem as matrizes esparsas também apresentam um ganho em relação ao tempo de cálculo, uma vez que ele é realizado somente nos elementos com índices armazenados [12].

Tabela 3.4: Economia no armazenamento de memória em relação ao grau de esparsidade da matriz.

$n = 100$ γ	Memória Utilizada (<i>byte</i>)		Economia de Memória
	Completa	Esparsa	
0,01	80.000	2.360	97,05%
0,1	80.000	16.136	79,83%
0,5	80.000	63.512	20,61%
0,9	80.000	95.608	-19,51%

Fonte: O Autor (2023).

3.2 Compressão de Matrizes Esparsas

Como forma de aproveitar a esparsidade da matriz, na literatura, é possível encontrar modelos de compressão que tem por objetivo criar estruturas menores, onde são armazenados apenas os elementos não nulos do sistema, os quais são descritos sucintamente nas próximas seções.

3.2.1 *Compressed Sparse Row (CSR)*

A metodologia empregada no método *Compressed Sparse Row* (CRS), consiste em substituir a matriz esparsa por três vetores: **aa**, **jr** e **cr**.

No vetor **aa** são armazenados os valores dos elementos não nulos da matriz esparsa na ordem em que aparecem nas linhas. A dimensão do vetor é igual a quantidade de elementos não nulos, representado por nz . Em outras palavras: **aa** $\in \mathbb{R}^{nz}$ [18].

Já no vetor **jr** estão armazenados a posição na coluna de cada elemento não nulo. A dimensão do vetor corresponde ao número de elementos não nulos. Ou seja, **jr** $\in \mathbb{N}^{nz}$ [18].

Por fim, o último vetor **jc**, contém a posição no vetor **aa**, do primeiro elemento não nulo de cada linha da matriz. Como esse vetor percorre todas as linhas da estrutura matricial, sua dimensão é n . De forma sintética: **jc** $\in \mathbb{N}^n$ [18]. Um exemplo desse tipo de compressão é apresentado na Equação (3.1).

$$\begin{aligned}
M &= \begin{bmatrix} 5 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 1 & 0 & 8 \end{bmatrix} \\
\mathbf{aa} &= [5 \quad 3 \quad 6 \quad 2 \quad 1 \quad 8] \\
\mathbf{jr} &= [1 \quad 3 \quad 2 \quad 4 \quad 2 \quad 4] \\
\mathbf{jc} &= [1 \quad 3 \quad 4 \quad 5]
\end{aligned} \tag{3.1}$$

3.2.2 *Compressed Sparse Column (CSC)*

O método *Compressed Sparse Column* (CSC) apresenta o mesmo princípio do método CSR, a diferença se dá na ordem em que os valores dos elementos não nulos são armazenados no vetor **aa**. Enquanto no método CSR eles são armazenados na ordem em que aparecem nas linhas, no método CSC o armazenamento é na ordem em que aparecem nas colunas [18].

A posição na linha da matriz, de cada elemento não nulo, é armazenada no vetor **jr**. Já no vetor **jc**, é colocado o primeiro elemento não nulo de cada coluna da matriz.

Usando como exemplo a matriz M exposta na Equação (3.1), a estrutura CSC pode ser caracterizada conforme Equação (3.2), ou seja:

$$\begin{aligned}
M &= \begin{bmatrix} 5 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 1 & 0 & 8 \end{bmatrix} \\
\mathbf{aa} &= [5 \quad 6 \quad 1 \quad 3 \quad 2 \quad 8] \\
\mathbf{jr} &= [1 \quad 2 \quad 4 \quad 1 \quad 3 \quad 4] \\
\mathbf{jc} &= [1 \quad 2 \quad 4 \quad 5]
\end{aligned} \tag{3.2}$$

3.2.3 *Compressed Sparse Vector (CSV)*

O método *Compressed Sparse Vector* (CSV) tem por objetivo armazenar a matriz esparsa em dois vetores: **aa** e **ia**. No vetor **aa**, será armazenado os elementos não nulos da matriz, na ordem em que aparecem nas linhas. Já o vetor **ia** armazena as posições dos elementos não nulos na matriz [18]. Essa posição não se dará em termos de linhas e colunas, mas em uma sequência que será de 1 até n^2 , ou seja, **aa** $\in \mathbb{R}^{nz}$ e **ai** $\in \mathbb{N}^{nz}$. O exemplo de compressão, pelo método *Compressed Sparse Vector*, está representado na Equação (3.3).

$$M = \begin{bmatrix} 5 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 1 & 0 & 8 \end{bmatrix} \quad (3.3)$$

$$\mathbf{aa} = [5 \quad 3 \quad 6 \quad 2 \quad 1 \quad 8]$$

$$\mathbf{ia} = [1 \quad 3 \quad 6 \quad 12 \quad 14 \quad 16]$$

3.2.4 *Compressed Diagonal Storage (CDS)*

O método *Compressed Diagonal Storage* (CDS) é desenvolvido para matrizes diagonais. Neste tipo de matriz, existe duas constantes positivas p e q , as quais definem a posição dos elementos à direita e à esquerda do elemento da diagonal principal, respectivamente, para $a_{i,j}$, com o índice j variando entre $i - p$ e $i + q$. Nesta configuração, pode-se armazenar os elementos das diagonais em vetores distintos, da seguinte maneira: **VAL**(1 : n , $-p : q$). Essa estrutura é exemplificada na Equação (3.4), onde $p = 1$ e $q = 1$.

Cada posição dos vetores **VAL**(:, -1) e **VAL**(:, $+1$) é referente ao seu posicionamento na linha da matriz. Neste sentido, para o vetor **VAL**(:, -1), que representa a diagonal inferior à principal, a primeira posição é 0, pois não há elementos dessa diagonal na primeira linha da matriz. De forma análoga, a última posição do vetor **VAL**(:, $+1$), superior à diagonal principal, também é 0, uma vez que não há elementos dessa diagonal na última linha da matriz [18].

$$N = \begin{bmatrix} 5 & 7 & 0 & 0 & 0 \\ 8 & 1 & 3 & 0 & 0 \\ 0 & 1 & 7 & 9 & 0 \\ 0 & 0 & 4 & 2 & 4 \\ 0 & 0 & 0 & 2 & 9 \end{bmatrix} \quad (3.4)$$

$$\mathbf{VAL}(:, -1) = \begin{bmatrix} 0 & 8 & 1 & 4 & 2 \end{bmatrix}$$

$$\mathbf{VAL}(:, 0) = \begin{bmatrix} 5 & 1 & 7 & 2 & 9 \end{bmatrix}$$

$$\mathbf{VAL}(:, +1) = \begin{bmatrix} 7 & 3 & 9 & 4 & 0 \end{bmatrix}$$

Por fim, é importante ressaltar que a compressão de estruturas matriciais, representada pelos métodos CSR, CSC, CSV e CDS, busca economizar o consumo de memória do computador no que diz respeito ao armazenamento de matrizes esparsas para resoluções de sistemas lineares extensos.

Capítulo 4

DESCRIÇÃO E MODELAGEM DO PROBLEMA PROPOSTO

A descoberta de um fenômeno físico, químico ou natural é muito importante para o avanço científico. Porém, para reproduzi-lo, testar diferentes cenários, buscando prever seu comportamento, é preciso quantificá-lo. Não raro, muitos desses problemas recaem em um sistema de equações lineares, envolvendo um número muito grande de incógnitas, o que torna sua resolução analítica impraticável ou até mesmo impossível. Nesses casos, para se chegar à solução exata, ou uma boa estimativa da mesma, são utilizadas implementações computacionais baseadas em algum método matemático. A origem da ideia central da modelagem computacional, quando devidamente convalidada, é entender e explicar, com diferentes graus de precisão, fatos e fenômenos que são observados na vida real [2]. O diagrama da Figura 4.1, representa os passos utilizados para resolução de um problema físico.

Na presença de um fenômeno, inicialmente são realizadas observações, com objetivo de compreender suas causas e efeitos. Através de métodos experimentais, são feitos testes e medições que visam desenvolver um conceito teórico do problema. Depois, a partir de leis conhecidas (na maioria das vezes, conservação de massa e quantidade de movimento), busca-se traduzir o conceito teórico desenvolvido em um modelo matemático. A partir do modelo matemático, são estudadas técnicas para resolução do problema, as quais podem ser de forma analítica (solução exata) ou numérica (aproximação da solução exata). Por fim, com objetivo de validar o modelo matemático, é realizada a comparação dos resultados numéricos com os obtidos através dos experimentos.

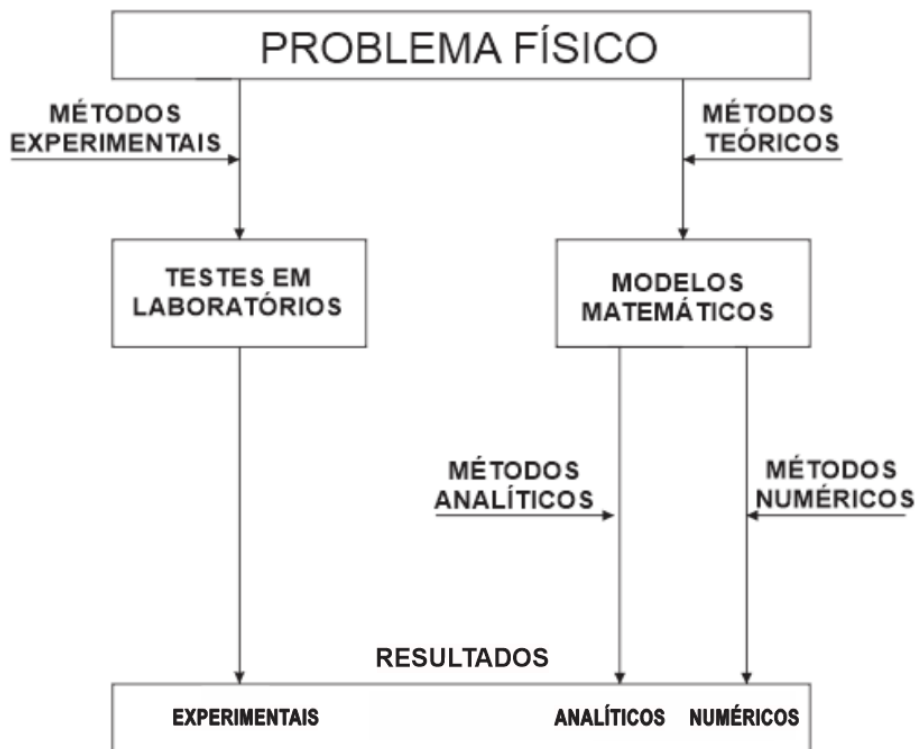


Figura 4.1: Diagrama para resolução de um problema físico.

Fonte: Adaptado de Maliska (1995) [16].

Nesse trabalho é feita a modelagem de dois problemas de transporte de contaminantes, os quais são modelados matematicamente utilizando equações diferenciais parciais, com abordagens uni e bidimensionais.

4.1 Caracterização e Modelagem do Problema Físico Proposto

Quando um soluto é colocado em contato com algum fluido, ele tende a se diluir, misturando-se com este fluido [7]. Isso ocorre, devido aos processos de advecção, difusão molecular e dispersão mecânica.

No que diz respeito a advecção, esta refere-se ao transporte da substância, de um ponto ao outro, sem alteração de suas características, causado pelo movimento do fluido onde a mesma foi inserida [7]. Já a difusão molecular consiste no espalhamento da substância no fluido, seguindo um gradiente de concentração [7]. Por fim, define-se dispersão mecânica, como a mistura da substância no fluido devido às diferenças de velocidades no mesmo [7], ou seja:

A dispersão mecânica tem o mesmo efeito da difusão, com a mistura resultando do movimento físico da água (diferenças de velocidade). Portanto, onde a concentração difere, o soluto é predominantemente transportado por difusão molecular quando a velocidade é relativamente baixa ($1,6 \times 10^{-10} \text{ m/s}$) e por dispersão mecânica quando a velocidade é considerada alta [7].

Na Figura 4.2 é representado o processo de transporte e mistura do soluto em um fluido de velocidade u .

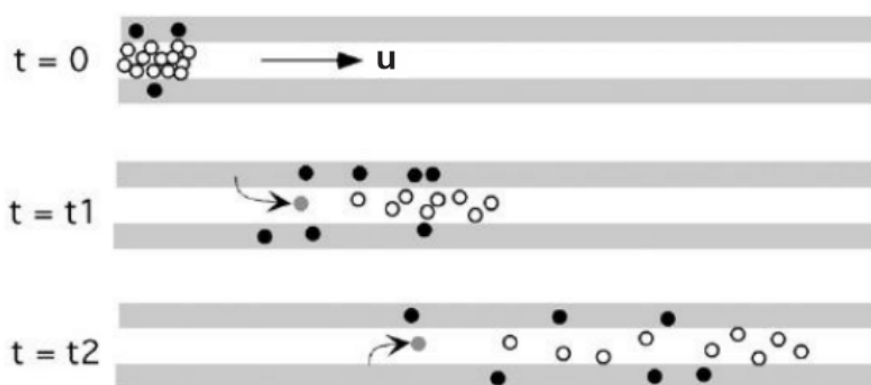


Figura 4.2: Processo de transporte de soluto em fluido.
Fonte: Adaptado de Potter e Wiggert (2002) [19].

Neste trabalho, o processo descrito na Figura 4.2 foi analisado em um trecho de rio. Um programa computacional foi desenvolvido para simular o lançamento de um poluente e sua dispersão ao longo do curso d'água. A esquematização do problema está demonstrada na Figura 4.3, em que: L_x representa o comprimento do trecho do rio a ser analisado, L_y a largura do trecho do rio, u é a velocidade do rio ao longo da direção longitudinal, P_L é o ponto no domínio, onde o poluente é lançado e P_C é o ponto de coleta da amostra de água, a jusante do ponto de lançamento.

Uma vez caracterizado o problema físico, é preciso definir um modelo matemático para solucioná-lo. Para escolha do modelo matemático, o primeiro passo é a definição do nível de balanços de conservação das grandezas a serem utilizadas. Este nível pode ser molecular, ou seja, as equações são aplicadas a cada molécula, o que inviabilizaria computacionalmente a resolução, ou através de volumes de controles, que em determinadas direções, podem até coincidir com o domínio de solução [16]. Na Tabela 4.1 são mostrados os níveis de balanço de conservação.

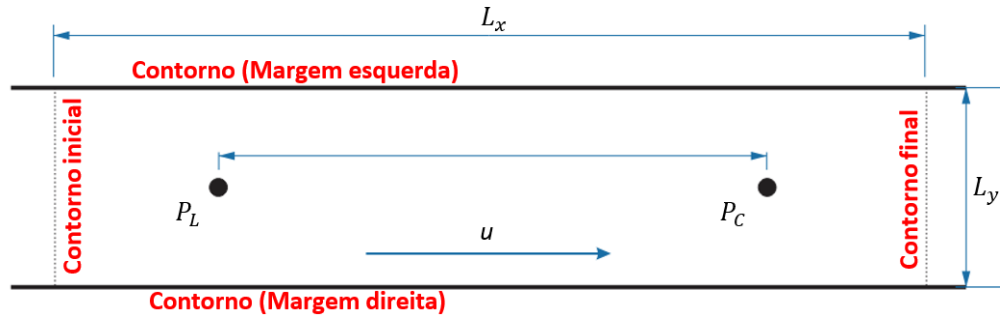


Figura 4.3: Esquematização do problema de transporte de poluente.

Fonte: O Autor (2023).

Tabela 4.1: Nível de formulação dos modelos.

Nível dos balanços de conservação	Informações necessárias	Tipo de equação resultante
I - Conservação para cada molécula $V \ll L_m^3$	Massa molecular; Campos de forças elétricos, magnéticos, etc	Equação para cada molécula
II - Balanços onde: $t_m \ll t \ll t_t$ $L_m \ll L \ll L_t$	Propriedades refletindo o comportamento molecular	Conjunto de equações diferenciais parciais
III - Balanços onde: $t \gg t_t$ $L \gg L_t$	Fornecer o comportamento molecular e as tensões de transferência de calor e massa turbulenta	Conjunto de equações diferenciais parciais
IV - Balanços onde o volume de controle coincide com o domínio de solução em alguma(s) direção(ões)	Fornecer as condições de contorno nas direções onde o volume de controle coincide com o domínio da solução	Equações diferenciais, parciais, ordinárias ou algébricas

Fonte: Adaptado de Maliska (1995) [16].

Em relação aos dados da Tabela 4.1, tem-se que:

 t = Tempo médio sobre os quais os balanços de conservação são realizados; t_m = Tempo entre colisões moleculares;

t_t = Escala de tempo para turbulência;

L = Comprimento médio sobre os quais os balanços de conservação são realizados;

L_m = Livre caminho médio entre moléculas;

L_t = Escala de comprimento para turbulência.

Com o grau tecnológico acessível atualmente, os níveis I e II (Tabela 4.1) se tornam impraticáveis, pois trabalham com as equações aplicadas em cada molécula, o que exigiria uma grande demanda computacional de processamento. Geralmente, os problemas de transporte de calor e massa são solucionados através dos níveis III e IV [16].

O modelo matemático, utilizado para simular o transporte de contaminantes em rios, é desenvolvido a partir de equações conservativas, que contemplam a taxa de variação espacial e temporal [24]. Em sua forma geral e tridimensional, a equação de conservação é dada por:

$$\frac{\partial(\rho\phi)}{\partial t} + \frac{\partial(\rho u\phi)}{\partial x} + \frac{\partial(\rho v\phi)}{\partial y} + \frac{\partial(\rho w\phi)}{\partial z} = \frac{\partial}{\partial x} \left(E_l \frac{\partial\phi}{\partial x} \right) + \frac{\partial}{\partial y} \left(E_t \frac{\partial\phi}{\partial y} \right) + \frac{\partial}{\partial z} \left(E_z \frac{\partial\phi}{\partial z} \right) + S, \quad (4.1)$$

onde ϕ é a grandeza a ser conservada; ρ a densidade do fluido; t o tempo; x, y, z são os domínios espaciais em coordenadas cartesianas; u, v, w são as velocidades do fluido nas direções x, y e z , respectivamente; E_l, E_t, E_z são os coeficientes de dispersão nas respectivas direções x, y e z ; e S o termo fonte.

Para a resolução do problema físico proposto, deve-se fazer ajustes na Equação (4.1), em relação à grandeza a ser conservada. Na presente pesquisa, a propriedade de balanço conservada é a massa. O balanço dessa propriedade nos volumes de controle são aplicadas considerando os fenômenos de advecção, difusão molecular e dispersão mecânica, processos que ocorrem quando um soluto ou contaminante é colocado em contato com a água.

A equação de conservação é formada por derivadas parciais, que pode ou não, possuir uma solução, e mesmo que tenha, nem sempre é possível chegar a essa solução pela forma analítica [30]. Neste cenário, é preciso utilizar métodos numéricos para obtenção de curvas com pontos distintos, cujas coordenadas são aproximações da curva de solução. As aproximações são obtidas discretizando o domínio espacial e temporal, no qual a solução faz sentido. Existem muitos métodos, amplamente validados, para solucionar a equação de conservação. No presente trabalho acadêmico é utilizado o Método dos Volumes Finitos (MVF). O problema aqui proposto é dividido em dois casos distintos, um para a formulação unidimensional e o outro, para a formulação bidimensional, os quais são descritos nas

próximas seções deste capítulo.

4.2 Estudo de Caso 01: Problema Unidimensional

Para validação dos resultados, os dados numéricos são comparados com os dados experimentais, obtidos através de um trabalho de campo, realizado por Souza (2009) [22], no rio São Pedro, localizado na parte superior da bacia hidrográfica do rio Macaé, região serrana do Estado do Rio de Janeiro, no município de Nova Friburgo [22].

• Descrição do Experimento

O experimento realizado dia 26 de janeiro de 2009, simulou um cenário acidental de poluição no curso d'água. Foram feitos lançamentos pontuais ou instantâneos, utilizando um traçador de solução salina, com objetivo de se determinar o coeficiente de dispersão longitudinal do rio. Na solução salina, foram utilizados 2 *kg* de cloreto de sódio, diluídos em 15 *l* de água. Essa solução foi injetada em um ponto da corrente central do rio. As coletas de amostra para medição, foram retiradas a 100 *m* do ponto de lançamento, com intervalo de 15 *s* entre elas. A concentração de base do rio foi medida antes da injeção da solução, a qual foi considerada como condição inicial do corpo hídrico ao iniciar as simulações [22]. As características do rio, obtidas através do experimento realizado, estão demonstradas na Tabela 4.2.

Tabela 4.2: Parâmetros do rio para caso unidimensional.

Parâmetros	Valores
Velocidade do rio (u)	0,59 m/s
Coeficiente de dispersão longitudinal (E_L)	1,82 m^2/s
Massa (M)	2.000 g
Comprimento do trecho analisado (x_{total})	500 m
Concentração inicial (C_0)	15,5 mg/l
Posição de lançamento (x_{lanc})	100 m
Posição de coleta (x_{lanc})	200 m

Fonte: Adaptado de Sousa (2009) [22].

• Modelagem do Problema

Devido às características da região de estudos, para a modelagem unidimensional do problema proposto são realizados ajustes na Equação (4.1), de forma a desconsiderar os processos advectivos e dispersivos nas direções y e z . Já a velocidade do rio (u) e o coeficiente de dispersão ao longo da longitudinal (E_L), são constantes e conhecidos. Com isso, obtém-se a Equação (4.2), conhecida como Equação Advecção-Dispersão na formulação unidimensional:

$$\frac{\partial C}{\partial t} + u \frac{\partial C}{\partial x} = E_L \frac{\partial}{\partial x} \left(\frac{\partial C}{\partial x} \right), \quad 0 < x < L_x, \quad t > 0, \quad (4.2)$$

onde C é a concentração em mg/l .

Na Equação (4.2), o primeiro termo representa a parte transiente, onde a concentração só depende do tempo. O segundo termo é a parte advectiva, que depende da velocidade do rio. O último termo é conhecido como dispersivo, e depende da dispersão ao longo da longitudinal.

Para finalizar a caracterização do problema, são dadas as condições iniciais e de contorno, do trecho de rio a ser analisado.

A condição inicial determina uma concentração de base em todo o domínio analisado no instante de tempo $t = 0$, acrescido de um lançamento pontual do poluente, como demonstrado na Equação (4.3), em que S representa a área da seção transversal do rio ao longo do trecho investigado.

$$C(x, 0) = C_0 + \frac{M}{S} \delta(x), \quad 0 < x < L_x, \quad t = 0 \quad (4.3)$$

De acordo com a Equação (4.3), a função de *Dirac* $\delta(x)$, define um ponto no domínio onde o poluente é lançado. Em todo o restante do trecho de rio, a concentração é dada por C_0 , para o instante inicial da simulação.

Já a condição de contorno determina as concentrações nos limites das margens inicial e final do domínio analisado, para os instantes de tempo $t > 0$.

Para o contorno inicial ao longo da longitudinal, tem-se:

$$C(0, t) = C_0, \quad t > 0. \quad (4.4)$$

Para o contorno final ao longo da longitudinal, tem-se:

$$\frac{\partial C(L_x, t)}{\partial x} = 0, \quad t > 0. \quad (4.5)$$

Na Equação (4.4), é dado o valor da função no contorno inicial, essa condição de contorno é conhecida como *Dirichlet*. Por outro lado, na Equação (4.5), não é dado o valor da função no contorno final, mas sua derivada, essa condição de contorno é conhecida como *Neumann*.

4.3 Estudo de Caso 02: Problema Bidimensional

Para o problema bidimensional analisado nesse trabalho, o perfil da concentração dos resultados numéricos foram comparados com os dados experimentais, obtidos através de um trabalho de campo realizado por Telles (2009) [24], em um trecho do rio Macaé, situado na cidade de mesmo nome, próximo à Usina Termoelétrica Mário Lago. O rio Macaé localiza-se na faixa costeira central do norte do Estado do Rio de Janeiro e compreende cerca de 1.765 km^2 [24].

• Descrição do Experimento

De forma análoga ao executado no experimento unidimensional, foram feitos lançamentos pontuais de uma solução salina no curso d'água, simulando um cenário acidental de poluição.

Na solução, foram utilizados 20 kg de cloreto de sódio, diluídos em 10 l de água, totalizando aproximadamente 11 l de volume total. Essa mistura foi repetida até se formarem dois recipientes com volume total de, aproximadamente, 55 l cada. A concentração salina total foi próxima de 174 g/l em cada recipiente.

Já as coletas das amostras para medição foram feitas em um ponto 50 m a jusante do lançamento e $0,5 \text{ m}$ da margem direita do rio [24], enquanto o tempo total do experimento foi de 360 s .

Por fim, cabe ressaltar que realização do trabalho de campo para a obtenção dos parâmetros e demais informações ocorreu no dia 29 de maio de 2008 e estão demonstrados na Tabela 4.3.

Tabela 4.3: Parâmetros do rio para caso bidimensional.

Parâmetros		Valores
Velocidade do rio	(u)	0,36 m/s
Coeficiente de dispersão longitudinal	(E_L)	0,33 m^2/s
Coeficiente de dispersão transversal	(E_T)	0,008 m^2/s
Massa	(M)	20 kg
Comprimento do trecho analisado	(x_{total})	182 m
Largura do trecho analisado	(y_{total})	42 m
Concentração inicial	(C_0)	37 mg/l
Posição de lançamento em x	(x_{lanc})	50 m
Posição de lançamento em y	(y_{lanc})	0,5 m
Posição de coleta em x	(x_{col})	100 m
Posição de coleta em y	(y_{col})	0,5 m

Fonte: Adaptado de Telles (2009) [24].

• Modelagem do Problema

São realizados ajustes na Equação (4.1) para o caso bidimensional. Devido às características da região de estudo, são desconsiderados os termos advectivos nas direções y e z , assim como o termo dispersivo na direção z . A velocidade do rio (u), o coeficiente de dispersão ao longo da longitudinal (E_L) e o coeficiente de dispersão ao longo da transversal (E_T) são constantes e conhecidos. Na Equação (4.6) é representada a formulação do problema proposto, para o caso bidimensional.

$$\frac{\partial C}{\partial t} + u \frac{\partial C}{\partial x} = E_L \frac{\partial}{\partial x} \left(\frac{\partial C}{\partial x} \right) + E_T \frac{\partial}{\partial y} \left(\frac{\partial C}{\partial y} \right), \quad 0 \leq x \leq L_x, \quad 0 \leq y \leq L_y, \quad t > 0 \quad (4.6)$$

Comparando a Equação (4.6) do modelo bidimensional com a Equação (4.2), do modelo unidimensional, nota-se o acréscimo do termo dispersivo, na direção transversal.

No modelo bidimensional, a condição inicial também pode ser determinada através da função *Dirac*, o ponto de lançamento é definido no domínio (x, y) , conforme Equação (4.7).

$$C(x, y, 0) = C_0 + \frac{M}{S} \delta(x, y), \quad 0 < x < L_x, \quad 0 < y < L_y, \quad t = 0 \quad (4.7)$$

As condições de contorno do modelo bidimensional, consideram as margens longitudinais, bem como os limites a esquerda e a direita do trecho sob análise, conforme já exposto na Figura 4.3.

Para os contornos inicial e final, ao longo da longitudinal, tem-se:

$$C(0, y, t) = C0, \quad 0 \leq y \leq L_y, \quad t > 0 \quad (4.8)$$

$$\frac{\partial C(L_x, y, t)}{\partial x} = 0, \quad 0 \leq y \leq L_y, \quad t > 0 \quad (4.9)$$

Já, para as margens direita ($y = 0$) e esquerda ($y = L_y$), ao longo da transversal, tem-se:

$$\frac{\partial C(x, 0, t)}{\partial y} = 0, \quad 0 \leq x \leq L_x, \quad t > 0 \quad (4.10)$$

$$\frac{\partial C(x, L_y, t)}{\partial y} = 0, \quad 0 \leq x \leq L_x, \quad t > 0 \quad (4.11)$$

Capítulo 5

SOLUÇÃO NUMÉRICA DO PROBLEMA PROPOSTO

Nesse capítulo são apresentadas as soluções numéricas dos estudos de caso da presente pesquisa através do Método dos Volumes Finitos, assim como o desenvolvimento das implementações computacionais.

5.1 Método dos Volumes Finitos

O Método dos Volumes Finitos pode ser utilizado para resolução dos problemas que envolvem transporte de poluentes em fluidos. Se baseia no balanço de massa e quantidade de movimento sobre um determinado volume de controle, onde o fluxo das variáveis que se pretende calcular atravessa as faces do volume [14]. Ou seja, a solução do problema consiste em particionar o domínio espacial em vários volumes de controle, processo conhecido como discretização, e em cada um deles, integrar a equação na forma conservativa para se obter as equações de aproximação.

Discretizar o domínio de solução é subdividi-lo em pequenos volumes de controle, criando uma grade ou malha, delimitada pelas faces dos volumes e que possui em seu centro geométrico, o nó computacional [9].

A equação algébrica para cada volume de controle é obtida através de interpolações entre os valores em cada face do volume e o nó central. Para que isso aconteça, é importante que não haja sobreposição de volumes, ou seja, em toda a malha, cada face deverá ser única para dois volumes de controle que se encontram a cada lado dela [9]. Esta característica da malha garante a lei da conservação, pois mantém igual o fluxo de massa que sai de um volume ao que entra em outro.

5.2 Solução Numérica do Estudo de Caso 01: Problema Unidimensional

Para a aplicação do modelo unidimensional, apresentado na Seção 4.2, deve-se discretizar o domínio físico, através do Método dos Volumes Finitos, como demonstrado na Figura 5.1.

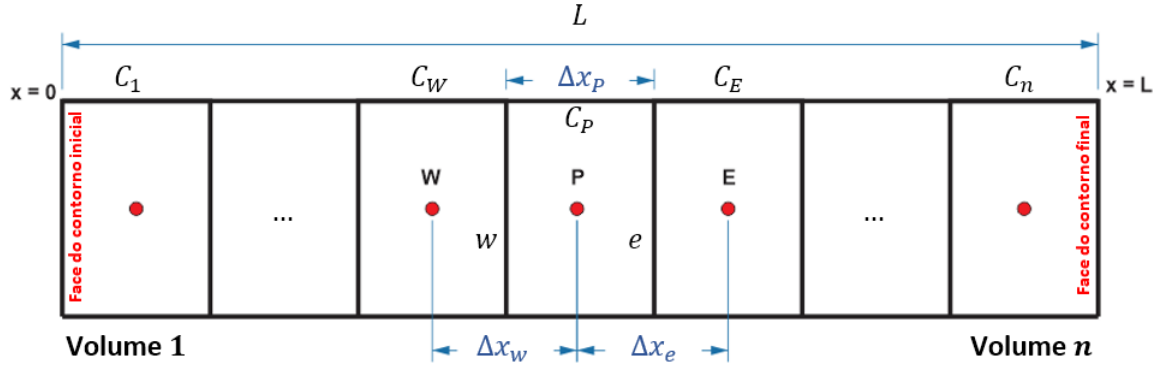


Figura 5.1: Representação do domínio discretizado para uma dimensão.
Fonte: O Autor (2023).

Na Figura 5.1 é representado o particionamento do trecho de rio, esquematizado na Figura 4.3, em n volumes de controle, onde C_W , C_P e C_E são as concentrações nos nós centrais dos volumes W , P e E , respectivamente. As faces entre os volumes $W - P$ e $P - E$, são representadas, respectivamente, por w e e . C_1 e C_n são as concentrações no primeiro e último volume, respectivamente. Supondo volumes regulares, então: $\Delta x_w = \Delta x_e = \Delta x_p = \Delta x$.

Para a obtenção das equações de aproximação, em cada volume de controle, basta integrar a Equação (4.2) sobre o volume elementar, no espaço e no tempo, em toda área de interesse ($0 < x < L$), conforme Equação (5.1).

$$\int_t^{t+\Delta t} \int_w^e \frac{\partial C}{\partial t} dx dt + \int_t^{t+\Delta t} \int_w^e u \frac{\partial C}{\partial x} dx dt = \int_t^{t+\Delta t} \int_w^e E_L \frac{\partial^2 C}{\partial x^2} dx dt \quad (5.1)$$

Invertendo a ordem de integração do primeiro termo da Equação (5.1), conhecido como *transiente*, o qual descreve o comportamento da variação da concentração ao longo do tempo, tem-se a Equação (5.2).

$$\int_w^e \int_t^{t+\Delta t} \frac{\partial C}{\partial t} dt dx = \int_w^e (C_P^{t+\Delta t} - C_P^t) dx \quad (5.2)$$

Integrando a Equação (5.2) no espaço de w a e , chega-se a Equação (5.3).

$$\int_w^e (C_P^{t+\Delta t} - C_P^t) dx = (C_P^{t+\Delta t} - C_P^t) (x_e - x_w) \quad (5.3)$$

Como a diferença espacial entre as posições e e w (faces dos volumes) é igual ao tamanho da grade, a Equação (5.3) resulta na Equação (5.4).

$$\int_w^e (C_P^{t+\Delta t} - C_P^t) dx = (C_P^{t+\Delta t} - C_P^t) \Delta x \quad (5.4)$$

O próximo passo é integrar o segundo termo da Equação (5.1), chamado de *advectivo*. Uma vez que é considerada constante, a velocidade do rio (u) não depende do tempo nem do espaço e pode ser retirada da integral, logo, a integração no tempo é direta, conforme Equação (5.5).

$$\int_t^{t+\Delta t} \int_w^e u \frac{\partial C}{\partial x} dx dt = u \int_w^e \Delta t \frac{\partial C}{\partial x} dx \quad (5.5)$$

Restando a integração da concentração no espaço de w a e , resultando na Equação (5.6).

$$\int_t^{t+\Delta t} \int_w^e u \frac{\partial C}{\partial x} dx dt = u \int_w^e \Delta t \frac{\partial C}{\partial x} dx = u \Delta t (C_e^\theta - C_w^\theta) \quad (5.6)$$

A integração resulta em uma diferença de concentrações nas faces ($C_e^\theta - C_w^\theta$), que são valores desconhecidos. Para solucionar essa inadequação utiliza-se o esquema **upwind** ou **célula doadora**, que leva em consideração a direção do fluxo para determinar os valores das concentrações nas faces [26]. Analisando as Figuras 4.3 e 5.1 e considerando o fluxo indo de W para E , ou seja, na direção positiva em relação ao eixo longitudinal, a contribuição da concentração para a face w é muito maior no volume central W . De forma análoga, a face e recebe uma contribuição maior da concentração do volume central P , logo: $C_w^\theta = C_W^\theta$ e $C_e^\theta = C_P^\theta$. Substituindo os valores na Equação (5.6), tem-se a Equação (5.7).

$$\int_t^{t+\Delta t} \int_w^e u \frac{\partial C}{\partial x} dx dt = u \int_w^e \Delta t \frac{\partial C}{\partial x} dx = u \Delta t (C_P^\theta - C_W^\theta) \quad (5.7)$$

Por fim, é preciso integrar o último termo da Equação (5.1), conhecido como termo *dispersivo*. Como E_L é constante e a derivada não depende do tempo, tem-se a Equação (5.8).

$$\int_t^{t+\Delta t} \int_w^e E_L \frac{\partial^2 C}{\partial x^2} dx dt = \int_w^e E_L \Delta t \frac{\partial}{\partial x} \left(\frac{\partial C}{\partial x} \right) dx \quad (5.8)$$

Como a Equação (5.8) não depende do tempo, a primeira integração é direta e calculada de maneira similar à Equação (5.5). Por fim, é realizada a integração no espaço, resultando na Equação (5.9).

$$\int_t^{t+\Delta t} \int_w^e E_L \frac{\partial^2 C}{\partial x^2} dx dt = \int_w^e E_L \Delta t \frac{\partial}{\partial x} \left(\frac{\partial C}{\partial x} \right) dx = E_L \Delta t \left(\frac{\partial C^\theta}{\partial x} \Big|_e - \frac{\partial C^\theta}{\partial x} \Big|_w \right) \quad (5.9)$$

O problema, agora, consiste em encontrar as derivadas da concentração nas faces w e e . Supondo uma malha regular, em que a distância da face até o ponto central do volume é igual $\frac{\Delta x}{2}$, como indicado na Figura 5.2, pode-se obter uma aproximação das derivadas utilizando uma expansão em séries de Taylor, em torno das faces w e e .

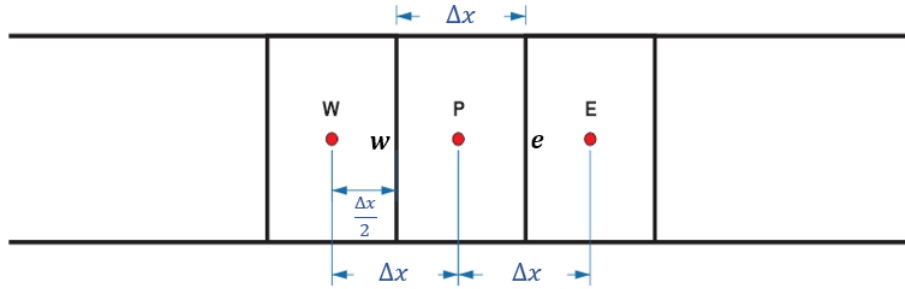


Figura 5.2: Representação das distâncias no domínio discretizado.
Fonte: O Autor (2023).

Começando com a concentração a montante da face w , tem-se a Equação (5.10).

$$C_{w-\frac{\Delta x}{2}}^\theta = C_W^\theta = C_w^\theta - C_w'^\theta \frac{\Delta x}{2} + C_w''^\theta \frac{\left(\frac{\Delta x}{2}\right)^2}{2!} - C_w'''^\theta \frac{\left(\frac{\Delta x}{2}\right)^3}{3!} + \dots \quad (5.10)$$

Calculando a concentração a jusante da face w , tem-se Equação (5.11).

$$C_{w+\frac{\Delta x}{2}}^\theta = C_P^\theta = C_w^\theta + C_w'^\theta \frac{\Delta x}{2} + C_w''^\theta \frac{\left(\frac{\Delta x}{2}\right)^2}{2!} + C_w'''^\theta \frac{\left(\frac{\Delta x}{2}\right)^3}{3!} + \dots \quad (5.11)$$

Subtraindo a Equação (5.10) da Equação (5.11), resulta na Equação (5.12).

$$C_P^\theta - C_W^\theta = C_w'^\theta \Delta x + 2 C_w'''^\theta \frac{\left(\frac{\Delta x}{2}\right)^3}{3!} + \dots \quad (5.12)$$

Isolando a primeira derivada na Equação (5.12), chega-se à Equação (5.13).

$$C'_w{}^\theta = \frac{C_P^\theta - C_W^\theta}{\Delta x} + \varphi(\Delta x)^2 \quad (5.13)$$

A Equação (5.13) é uma aproximação da primeira derivada com erro de truncamento de ordem quadrática.

Fazendo o mesmo processo para a concentração a montante da face e , tem-se a Equação (5.14).

$$C_{e-\frac{\Delta x}{2}}^\theta = C_P^\theta = C_e^\theta - C'_e{}^\theta \frac{\Delta x}{2} + C''_e{}^\theta \frac{\left(\frac{\Delta x}{2}\right)^2}{2!} - C'''_e{}^\theta \frac{\left(\frac{\Delta x}{2}\right)^3}{3!} + \dots \quad (5.14)$$

Calculando a concentração a jusante da face e , tem-se a Equação (5.15).

$$C_{e+\frac{\Delta x}{2}}^\theta = C_E^\theta = C_e^\theta + C'_e{}^\theta \frac{\Delta x}{2} + C''_e{}^\theta \frac{\left(\frac{\Delta x}{2}\right)^2}{2!} + C'''_e{}^\theta \frac{\left(\frac{\Delta x}{2}\right)^3}{3!} + \dots \quad (5.15)$$

Subtraindo a Equação (5.14) da Equação (5.15), obtem-se a Equação (5.16).

$$C_P^\theta - C_E^\theta = C'_e{}^\theta \Delta x + 2 C'''_e{}^\theta \frac{\left(\frac{\Delta x}{2}\right)^3}{3!} + \dots \quad (5.16)$$

Isolando a primeira derivada na Equação (5.16), obtem-se a Equação (5.17).

$$C'_e{}^\theta = \frac{C_E^\theta - C_P^\theta}{\Delta x} + \varphi(\Delta x)^2 \quad (5.17)$$

Usando as aproximações das Equações (5.13) e (5.17) na Equação (5.9), tem-se a Equação (5.18).

$$E_L \Delta t \left(\left. \frac{\partial C^\theta}{\partial x} \right|_e - \left. \frac{\partial C^\theta}{\partial x} \right|_w \right) = \quad (5.18)$$

$$E_L \Delta t \left(\frac{C_E^\theta - C_P^\theta}{\Delta x} - \frac{C_P^\theta - C_W^\theta}{\Delta x} \right) = E_L \Delta t \left(\frac{-2 C_P^\theta + C_E^\theta + C_W^\theta}{\Delta x} \right)$$

Substituindo as Equações (5.4), (5.7) e (5.18) na Equação (5.1), resulta na Equação (5.19).

$$(C_P^{t+\Delta t} - C_P^t) \Delta x + u \Delta t (C_E^\theta - C_W^\theta) = E_L \Delta t \left(\frac{-2 C_P^\theta + C_E^\theta + C_W^\theta}{\Delta x} \right) \quad (5.19)$$

O termo transiente, especificado na Equação (5.4), possui as concentrações no ponto de referência com instantes de tempo t e $t + \Delta t$. Falta definir o valor do parâmetro θ na Equação (5.19), o qual está relacionado com o instante de tempo das concentrações obtidas através dos termos advectivo e dispersivo, conforme exposto na Equação (5.20).

$$C^\theta = \theta C^{t+\Delta t} + (1 - \theta) C^t \quad (5.20)$$

Usando a função de interpolação apresentada na Equação (5.20), pode-se definir o tipo de formulação a ser adotada no Método dos Volumes Finitos. Quando $\theta = 1$, tem-se a formulação **implícita**, conforme Equação (5.21), a qual é adotada nesse trabalho.

$$C^\theta = 1 C^{t+\Delta t} + (1 - 1) C^t \quad \rightarrow \quad C^\theta = C^{t+\Delta t} \quad (5.21)$$

Substituindo a Equação (5.21) na Equação (5.19), tem-se a Equação (5.22).

$$(C_P^{t+\Delta t} - C_P^t) \Delta x + u \Delta t (C_P^{t+\Delta t} - C_W^{t+\Delta t}) = E_L \Delta t \left(\frac{-2 C_P^{t+\Delta t} + C_E^{t+\Delta t} + C_W^{t+\Delta t}}{\Delta x} \right) \quad (5.22)$$

Isolando a concentração de referência com instante de tempo igual a t e reorganizando os termos da Equação (5.22), chega-se à Equação (5.23).

$$\left(-u \frac{\Delta t}{\Delta x} - E_L \frac{\Delta t}{\Delta x^2} \right) C_W^{t+\Delta t} + \left(1 + u \frac{\Delta t}{\Delta x} + 2 E_L \frac{\Delta t}{\Delta x^2} \right) C_P^{t+\Delta t} + \left(-E_L \frac{\Delta t}{\Delta x^2} \right) C_E^{t+\Delta t} = C_P^t \quad (5.23)$$

A Equação (5.23) é válida para todos os **volumes internos** da malha. Para os volumes das extremidades, nesse trabalho, é adotada uma estratégia que consiste em criar volumes fictícios, de forma a tornar os volumes 1 e n , volumes internos. A representação do volume fictício para o contorno à esquerda do domínio é mostrada na Figura 5.3.

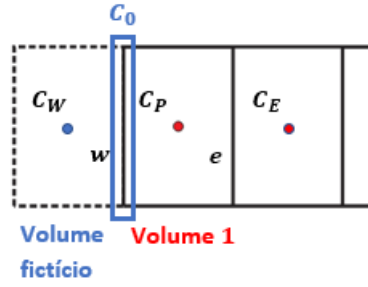


Figura 5.3: Representação do volume fictício para cálculo do volume 1.

Fonte: O Autor (2023).

Como representado na Figura 5.3, foi criado um volume fictício C_W , a montante do volume 1. O valor da concentração na face w é conhecido e vale C_0 . A concentração no volume da face w pode ser calculado usando uma média entre as concentrações C_W e C_P , como demonstrado na Equação (5.24).

$$C_w^{t+\Delta t} = \frac{C_W^{t+\Delta t} + C_P^{t+\Delta t}}{2} \quad (5.24)$$

Isolando a concentração do volume fictício e substituindo o valor da concentração na face w , tem-se a Equação (5.25).

$$C_W^{t+\Delta t} = 2C_w^{t+\Delta t} - C_P^{t+\Delta t} \rightarrow C_W^{t+\Delta t} = 2C_0 - C_P^{t+\Delta t} \quad (5.25)$$

Aplicando a Equação (5.23) no volume 1 e substituindo a concentração $C_W^{t+\Delta t}$ pelo valor encontrado na Equação (5.25), resulta na Equação (5.26).

$$C_P^t = \left(2E_L \frac{\Delta t}{\Delta x^2} + u \frac{\Delta t}{\Delta x} + 1 \right) C_P^{t+\Delta t} - \left(2E_L \frac{\Delta t}{\Delta x^2} + u \frac{\Delta t}{\Delta x} \right) (2C_0 - C_P^{t+\Delta t}) - E_L \frac{\Delta t}{\Delta x^2} C_E^{t+\Delta t} \quad (5.26)$$

Reorganizando os termos da Equação (5.26), tem-se a equação para o **volume 1**, dada pela Equação (5.27).

$$\left(1 + 2u \frac{\Delta t}{\Delta x} + 3E_L \frac{\Delta t}{\Delta x^2} \right) C_P^{t+\Delta t} + \left(-E_L \frac{\Delta t}{\Delta x^2} \right) C_E^{t+\Delta t} = C_P^t + 2C_0 \left(u \frac{\Delta t}{\Delta x} + E_L \frac{\Delta t}{\Delta x^2} \right) \quad (5.27)$$

De forma análoga, calcula-se o valor do volume n . Na Figura 5.4 é demonstrado o volume fictício criado para a extremidade a jusante do ponto de referência.

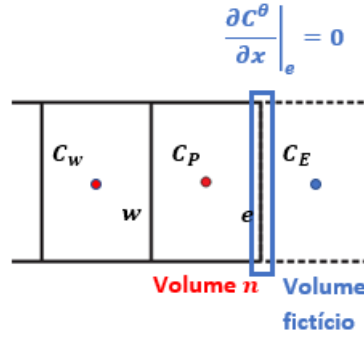


Figura 5.4: Representação do volume fictício para cálculo do volume n .

Fonte: O Autor (2023).

Como a concentração na face e , do último volume de controle, é dada por uma taxa de variação, faz-se uma expansão em séries de Taylor em torno desta face, resultando na Equação (5.28). Igualando o valor a zero, tem-se a Equação (5.28).

$$C'_e{}^\theta = \frac{C_E^\theta - C_P^\theta}{\Delta x} + \varphi(\Delta x)^2 = 0 \rightarrow C_E^\theta = C_P^\theta \quad (5.28)$$

Substituindo a Equação (5.28) na Equação (5.23) e reorganizando os termos, tem-se a equação para o **volume n** , conforme Equação (5.29).

$$\left(-u \frac{\Delta t}{\Delta x} - E_L \frac{\Delta t}{\Delta x^2}\right) C_W^{t+\Delta t} + \left(1 + u \frac{\Delta t}{\Delta x} + E_L \frac{\Delta t}{\Delta x^2}\right) C_P^{t+\Delta t} = C_P^t \quad (5.29)$$

Considerando, $E_L \frac{\Delta t}{\Delta x^2} = \beta$ e $u \frac{\Delta t}{\Delta x} = \alpha$, em resumo, a aplicação do Método dos Volumes Finitos resulta nas Equações (5.30)-(5.32).

- **volume 1:**

$$(1 + 2\alpha + 3\beta) C_P^{t+\Delta t} + (-\beta) C_E^{t+\Delta t} = C_P^t + 2C_0(\alpha + \beta) \quad (5.30)$$

- **volumes internos ($2, \dots, n-1$):**

$$(-\alpha - \beta) C_W^{t+\Delta t} + (1 + \alpha + 2\beta) C_P^{t+\Delta t} + (-\beta) C_E^{t+\Delta t} = C_P^t \quad (5.31)$$

• **Volume n :**

$$(-\alpha - \beta) C_W^{t+\Delta t} + (1 + \alpha + \beta) C_P^{t+\Delta t} = C_P^t \quad (5.32)$$

Logo, a formulação unidimensional do problema proposto, utilizando o Método dos Volumes Finitos, resulta em um sistema de equações lineares, do tipo $A \mathbf{x} = \mathbf{b}$. Uma característica importante desse tipo de sistema, é que a matriz dos coeficientes, é tridimensional.

Em particular, na Equação (5.33) é exemplificado um sistema de equações, gerado através da discretização de um domínio com 6 volumes de controle. Para facilitar a representação, é adotada a seguinte notação:

- $d_1 \rightarrow$ Para os coeficientes da concentração no volume de referência $C_P^{t+\Delta t}$;
 $d_2 \rightarrow$ Para os coeficientes da concentração no volume $C_E^{t+\Delta t}$;
 $d_3 \rightarrow$ Para os coeficientes da concentração no volume $C_W^{t+\Delta t}$;
 $F = 2 C_0 (d_1 + d_2)$.

$$\left\{ \begin{array}{lcl} d_1 C_1^{t+\Delta t} + d_2 C_2^{t+\Delta t} & & = C_1^t + F \\ d_3 C_1^{t+\Delta t} + d_1 C_2^{t+\Delta t} + d_2 C_3^{t+\Delta t} & & = C_2^t \\ & d_3 C_2^{t+\Delta t} + d_1 C_3^{t+\Delta t} + d_2 C_4^{t+\Delta t} & = C_3^t \\ & & d_3 C_3^{t+\Delta t} + d_1 C_4^{t+\Delta t} + d_2 C_5^{t+\Delta t} & = C_4^t \\ & & & d_3 C_4^{t+\Delta t} + d_1 C_5^{t+\Delta t} + d_2 C_6^{t+\Delta t} & = C_5^t \\ & & & & d_3 C_5^{t+\Delta t} + d_1 C_6^{t+\Delta t} & = C_6^t \end{array} \right. \quad (5.33)$$

O sistema dado na Equação (5.33), pode ser representado na forma matricial, conforme Sistema (5.34).

$$\begin{bmatrix} d_1 & d_2 & & & & \\ d_3 & d_1 & d_2 & & & \\ & d_3 & d_1 & d_2 & & \\ & & d_3 & d_1 & d_2 & \\ & & & d_3 & d_1 & d_2 \\ & & & & d_3 & d_1 \end{bmatrix} \begin{bmatrix} C_1^{t+\Delta t} \\ C_2^{t+\Delta t} \\ C_3^{t+\Delta t} \\ C_4^{t+\Delta t} \\ C_5^{t+\Delta t} \\ C_6^{t+\Delta t} \end{bmatrix} = \begin{bmatrix} C_1^t + F \\ C_2^t \\ C_3^t \\ C_4^t \\ C_5^t \\ C_6^t \end{bmatrix} \quad (5.34)$$

5.3 Solução Numérica do Estudo de Caso 02: Problema Bidimensional

Para a solução numérica do estudo de caso 02, descrito na Seção 4.3, o trecho de rio analisado também é discretizado através do Método dos Volumes Finitos. A discretização do domínio físico foi apresentada na Figura 4.3.

No modelo unidimensional, o processo dispersivo ao longo da transversal do sentido do fluxo é desconsiderado. Com isso, os pontos onde o poluente é lançado e coletado não levam em consideração a largura do rio, diferentemente do caso bidimensional, conforme mostrado na Figura 5.5. Usando como referência o volume P , cuja concentração é C_P , tem-se que C_N e C_S são as concentrações nos volumes acima e abaixo de P , respectivamente; C_E e C_W são as concentrações dos volumes à frente e atrás de P . Assim como no modelo unidimensional, considera-se, devido à geometria da região de interesse, uma malha regular, ou seja, $\Delta x = \Delta y$. As faces de fronteira do volume P são: e , w , n e s .

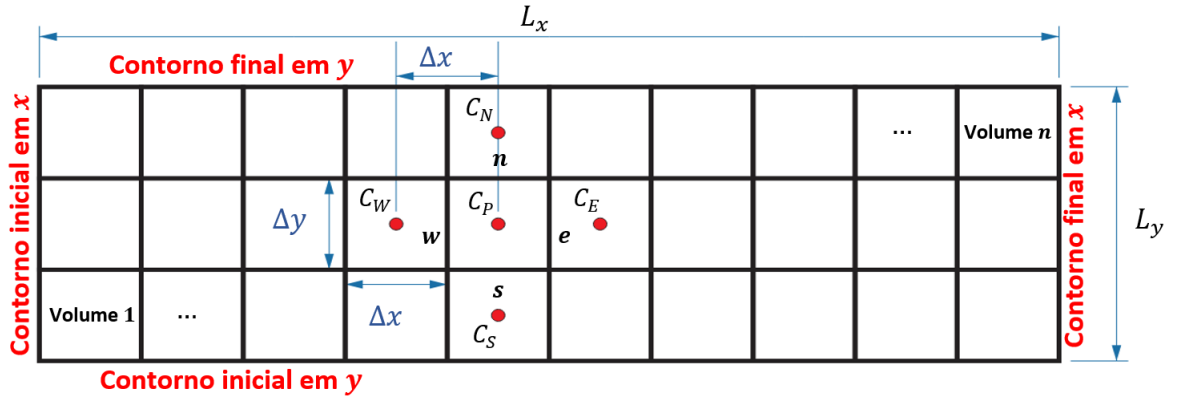


Figura 5.5: Representação do domínio discretizado para duas dimensões.

Fonte: O Autor (2023).

Novamente, para obtenção das equações de aproximação dos volumes de controle, integra-se sobre o volume elementar, a Equação (4.6), no espaço e no tempo, conforme Equação (5.35).

$$\int_t^{t+\Delta t} \int_s^n \int_w^e \frac{\partial C}{\partial t} dx dy dt + \int_t^{t+\Delta t} \int_s^n \int_w^e u \frac{\partial C}{\partial x} dx dy dt =$$

$$\int_t^{t+\Delta t} \int_s^n \int_w^e E_L \frac{\partial}{\partial x} \left(\frac{\partial C}{\partial x} \right) dx dy dt + \int_t^{t+\Delta t} \int_s^n \int_w^e E_T \frac{\partial}{\partial y} \left(\frac{\partial C}{\partial y} \right) dx dy dt$$
(5.35)

Integrando o termo transiente da Equação (5.35) no tempo, tem-se a Equação (5.36).

$$\int_t^{t+\Delta t} \int_s^n \int_w^e \frac{\partial C}{\partial t} dx dy dt = \int_s^n \int_w^e (C_P^{t+\Delta t} - C_P^t) dx dy \quad (5.36)$$

Integrando a Equação (5.36) no espaço de s a n e de w a e , resulta na Equação (5.37).

$$\int_s^n \int_w^e (C_P^{t+\Delta t} - C_P^t) dx dy = (C_P^{t+\Delta t} - C_P^t) (x_e - x_w) (x_n - x_s) \quad (5.37)$$

Por se tratar de uma grade regular, onde todos os volumes possuem o mesmo tamanho, as diferenças de comprimento e altura entre as faces é igual às do volume, chega-se à Equação (5.38).

$$\int_s^n \int_w^e (C_P^{t+\Delta t} - C_P^t) dx dy = (C_P^{t+\Delta t} - C_P^t) \Delta x \Delta y \quad (5.38)$$

Em relação ao termo advectivo da Equação (5.35), a integração em y e em t sai direto, pois a concentração só depende de x , resultando na Equação (5.39).

$$\int_t^{t+\Delta t} \int_s^n \int_w^e u \frac{\partial C}{\partial x} dx dy dt = u \Delta t \Delta y \int_w^e \frac{\partial C}{\partial x} dx \quad (5.39)$$

Realizando a integração no espaço da Equação (5.39), tem-se a Equação (5.40).

$$u \Delta t \Delta y \int_w^e \frac{\partial C}{\partial x} dx = u \Delta t \Delta y (C_e^\theta - C_w^\theta) \quad (5.40)$$

Exatamente como na formulação unidimensional, a integração do termo advectivo resulta em uma diferença de concentrações nas faces. Nesse trabalho, é usado, novamente, o esquema **upwind** para determinar as concentrações nas faces ($C_w^\theta = C_W^\theta$ e $C_e^\theta = C_P^\theta$), então, obtém-se a Equação (5.41).

$$u \Delta t \Delta y \int_w^e \frac{\partial C}{\partial x} dx = u \Delta t \Delta y (C_P^\theta - C_W^\theta) \quad (5.41)$$

Falta integrar os termos dispersivos da Equação (5.35). Começando pela dispersão ao longo do eixo x , tem-se a Equação (5.42).

$$\int_t^{t+\Delta t} \int_s^n \int_w^e E_L \frac{\partial}{\partial x} \left(\frac{\partial C}{\partial x} \right) dx dy dt = E_L \Delta t \Delta y \left(\frac{\partial C^\theta}{\partial x} \Big|_e - \frac{\partial C^\theta}{\partial x} \Big|_w \right) \quad (5.42)$$

Já, para a dispersão ao longo do eixo y , tem-se a Equação (5.43).

$$\int_t^{t+\Delta t} \int_s^n \int_w^e E_T \frac{\partial}{\partial y} \left(\frac{\partial C}{\partial y} \right) dx dy dt = E_T \Delta t \Delta x \left(\frac{\partial C^\theta}{\partial y} \Big|_n - \frac{\partial C^\theta}{\partial y} \Big|_s \right) \quad (5.43)$$

Mais uma vez, assim como na formulação unidimensional, para encontrar as derivadas nas faces dos volumes de controle, são utilizadas expansões em séries de Taylor, em torno das faces w , e , s e n , resultando nas Equações (5.44)-(5.47).

$$\frac{\partial C^\theta}{\partial x} \Big|_w = \frac{C_P^\theta - C_W^\theta}{\Delta x} + \varphi(\Delta x)^2 \quad (5.44)$$

$$\frac{\partial C^\theta}{\partial x} \Big|_e = \frac{C_E^\theta - C_P^\theta}{\Delta x} + \varphi(\Delta x)^2 \quad (5.45)$$

$$\frac{\partial C^\theta}{\partial y} \Big|_s = \frac{C_P^\theta - C_S^\theta}{\Delta y} + \varphi(\Delta y)^2 \quad (5.46)$$

$$\frac{\partial C^\theta}{\partial y} \Big|_n = \frac{C_N^\theta - C_P^\theta}{\Delta y} + \varphi(\Delta y)^2 \quad (5.47)$$

Substituindo as Equações (5.44) e (5.45), na Equação (5.42), tem-se a Equação (5.48).

$$\begin{aligned} E_L \Delta t \Delta y \left(\frac{\partial C^\theta}{\partial x} \Big|_e - \frac{\partial C^\theta}{\partial x} \Big|_w \right) = \\ E_L \Delta t \Delta y \left(\frac{C_E^\theta - C_P^\theta}{\Delta x} - \frac{C_P^\theta - C_W^\theta}{\Delta x} \right) = E_L \Delta t \Delta y \left(\frac{C_W^\theta - 2C_P^\theta + C_E^\theta}{\Delta x} \right) \end{aligned} \quad (5.48)$$

Substituindo as Equações (5.46) e (5.47), na Equação (5.43), tem-se a Equação (5.49).

$$\begin{aligned} E_T \Delta t \Delta x \left(\frac{\partial C^\theta}{\partial y} \Big|_n - \frac{\partial C^\theta}{\partial y} \Big|_s \right) = \\ E_T \Delta t \Delta x \left(\frac{C_N^\theta - C_P^\theta}{\Delta y} - \frac{C_P^\theta - C_S^\theta}{\Delta y} \right) = E_T \Delta t \Delta x \left(\frac{C_S^\theta - 2C_P^\theta + C_N^\theta}{\Delta y} \right) \end{aligned} \quad (5.49)$$

Juntando as Equações (5.38), (5.41), (5.48) e (5.49), tem-se a Equação (5.50).

$$(C_P^{t+\Delta t} - C_P^t) \Delta x \Delta y + u \Delta t \Delta y (C_P^\theta - C_W^\theta) = E_L \Delta t \Delta y \left(\frac{C_W^\theta - 2C_P^\theta + C_E^\theta}{\Delta x} \right) + E_T \Delta t \Delta x \left(\frac{C_S^\theta - 2C_P^\theta + C_N^\theta}{\Delta y} \right) \quad (5.50)$$

De forma análoga à formulação unidimensional, define-se o parâmetro θ presente na Equação (5.20), através da função de interpolação na formulação **implícita**, ou seja, $C^\theta = C^{t+\Delta t}$. Assim, a Equação (5.50), é reescrita como apresentado na Equação (5.51).

$$(C_P^{t+\Delta t} - C_P^t) \Delta x \Delta y + u \Delta t \Delta y (C_P^{t+\Delta t} - C_W^{t+\Delta t}) = E_L \Delta t \Delta y \left(\frac{C_W^{t+\Delta t} - 2C_P^{t+\Delta t} + C_E^{t+\Delta t}}{\Delta x} \right) + E_T \Delta t \Delta x \left(\frac{C_S^{t+\Delta t} - 2C_P^{t+\Delta t} + C_N^{t+\Delta t}}{\Delta y} \right) \quad (5.51)$$

Isolando a concentração de referência e reagrupando as demais concentrações da Equação (5.51), é obtido o seguinte resultado, conforme Equação (5.52).

$$\begin{aligned} & \left(-E_T \frac{\Delta x}{\Delta y} \right) C_S^{t+\Delta t} + \left(-u \Delta y - E_L \frac{\Delta y}{\Delta x} \right) C_W^{t+\Delta t} + \\ & \left(\frac{\Delta x \Delta y}{\Delta t} + u \Delta y + 2 E_L \frac{\Delta y}{\Delta x} + 2 E_T \frac{\Delta x}{\Delta y} \right) C_P^{t+\Delta t} + \left(-E_L \frac{\Delta y}{\Delta x} \right) C_E^{t+\Delta t} + \\ & \left(-E_T \frac{\Delta x}{\Delta y} \right) C_N^{t+\Delta t} = \left(\frac{\Delta x \Delta y}{\Delta t} \right) C_P^t \end{aligned} \quad (5.52)$$

A Equação (5.52) pode ser utilizada para todos os volumes internos do domínio, como indicado na Figura 5.6.

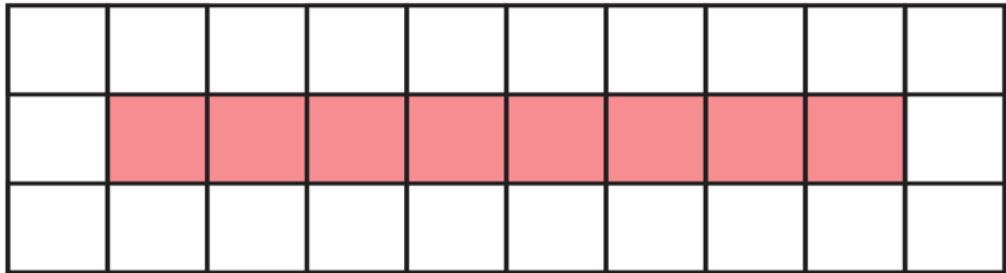


Figura 5.6: Representação dos volumes internos no domínio discretizado para duas dimensões.

Fonte: O Autor (2023).

Todos os demais volumes do domínio da Figura 5.5, são de fronteira, ou seja, para determinar seus valores é preciso conhecer as condições de contorno. Para isso, é utili-

zada a mesma estratégia da formulação unidimensional, a qual consiste em criar volumes fictícios, de forma a transformar os volumes de fronteira, em volumes internos.

Por simplicidade e para facilitar o entendimento, o domínio físico é dividido por regiões, como demonstrado na Figura 5.7. Na sequência é mostrada aplicação do Método dos Volumes Finitos para as 8 regiões definidas referentes ao contorno do domínio analisado, uma vez que a Região 5 refere-se aos volumes internos do domínio.



Figura 5.7: Representação das regiões do domínio físico.

Fonte: O Autor (2023).

• Região 1

Para determinar a concentração da Região 1, utiliza-se como referência o volume C_P , destacado na Figura 5.8, C_W e C_S representam os volumes fictícios.

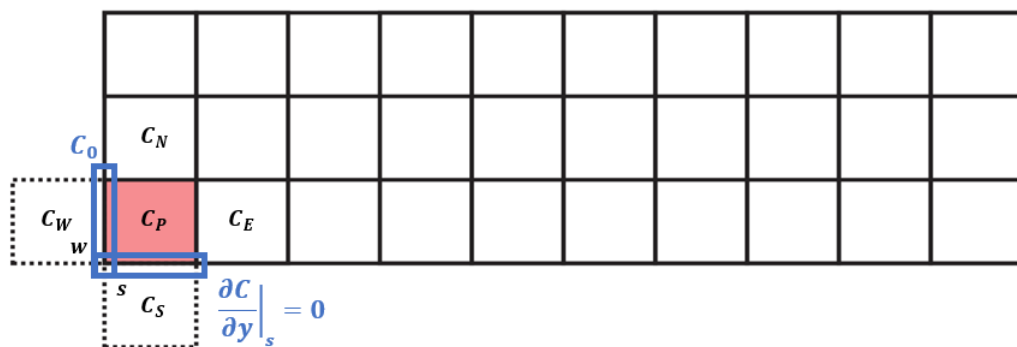


Figura 5.8: Representação dos volumes fictícios para o cálculo da Região 1.

Fonte: O Autor (2023).

Para utilização da Equação (5.52), basta achar o valor dos volumes fictícios, em função do volume interno. Assumindo o valor da concentração nas faces de fronteira (w e s) como a média entre as concentrações nos nós centrais dos volumes, exatamente como na formulação unidimensional, tem-se a concentração na face w dada na Equação (5.53).

$$C_w^{t+\Delta t} = \frac{C_W^{t+\Delta t} - C_P^{t+\Delta t}}{2} = C_0 \rightarrow C_W^{t+\Delta t} = 2C_0 - C_P^{t+\Delta t} \quad (5.53)$$

Para a face s , utiliza-se um expansão em série de Taylor em torno da mesma, resultando na Equação (5.54).

$$\left. \frac{\partial C^\theta}{\partial y} \right|_s = \frac{C_P^{t+\Delta t} - C_S^{t+\Delta t}}{\Delta y} + \varphi(\Delta y)^2 = 0 \rightarrow C_S^{t+\Delta t} = C_P^{t+\Delta t} \quad (5.54)$$

Substituindo as Equações (5.53) e (5.54) na Equação (5.52), tem-se a Equação (5.55).

$$\begin{aligned} \left(\frac{\Delta x \Delta y}{\Delta t} \right) C_P^t &= \left(-E_T \frac{\Delta x}{\Delta y} \right) C_P^{t+\Delta t} + \left(-E_T \frac{\Delta x}{\Delta y} \right) C_N^{t+\Delta t} + \left(-E_L \frac{\Delta y}{\Delta x} \right) C_E^{t+\Delta t} + \\ &\left(\frac{\Delta x \Delta y}{\Delta t} + u \Delta y + 2 E_L \frac{\Delta y}{\Delta x} + 2 E_T \frac{\Delta x}{\Delta y} \right) C_P^{t+\Delta t} + \left(-u \Delta y - E_L \frac{\Delta y}{\Delta x} \right) (2C_0 - C_P^{t+\Delta t}) \end{aligned} \quad (5.55)$$

Isolando as concentrações da Equação (5.55) e reorganizando seus termos, o resultado é dado na Equação (5.56).

$$\begin{aligned} \left(\frac{\Delta x \Delta y}{\Delta t} + 2u \Delta y + 3E_L \frac{\Delta y}{\Delta x} + E_T \frac{\Delta x}{\Delta y} \right) C_P^{t+\Delta t} &+ \left(-E_L \frac{\Delta y}{\Delta x} \right) C_E^{t+\Delta t} + \\ \left(-E_T \frac{\Delta x}{\Delta y} \right) C_N^{t+\Delta t} &= \left(\frac{\Delta x \Delta y}{\Delta t} \right) C_P^t + \left(u \Delta y + E_L \frac{\Delta y}{\Delta x} \right) 2C_0 \end{aligned} \quad (5.56)$$

A Equação (5.56) é utilizada para calcular a concentração do volume na Região 1, destacada na Figura 5.8.

• Região 2

Para a Região 2, o único volume fictício a ser criado é C_S , representado na Figura 5.9.

Substituindo o valor da concentração do volume fictício $C_S^{t+\Delta t}$, já calculado na Equação (5.54), na Equação (5.52), tem-se a Equação (5.57).

$$\begin{aligned} \left(\frac{\Delta x \Delta y}{\Delta t} \right) C_P^t &= \left(-E_T \frac{\Delta x}{\Delta y} \right) C_P^{t+\Delta t} + \left(-E_T \frac{\Delta x}{\Delta y} \right) C_N^{t+\Delta t} + \left(-E_L \frac{\Delta y}{\Delta x} \right) C_E^{t+\Delta t} + \\ &\left(\frac{\Delta x \Delta y}{\Delta t} + u \Delta y + 2 E_L \frac{\Delta y}{\Delta x} + 2 E_T \frac{\Delta x}{\Delta y} \right) C_P^{t+\Delta t} + \left(-u \Delta y - E_L \frac{\Delta y}{\Delta x} \right) C_W^{t+\Delta t} \end{aligned} \quad (5.57)$$

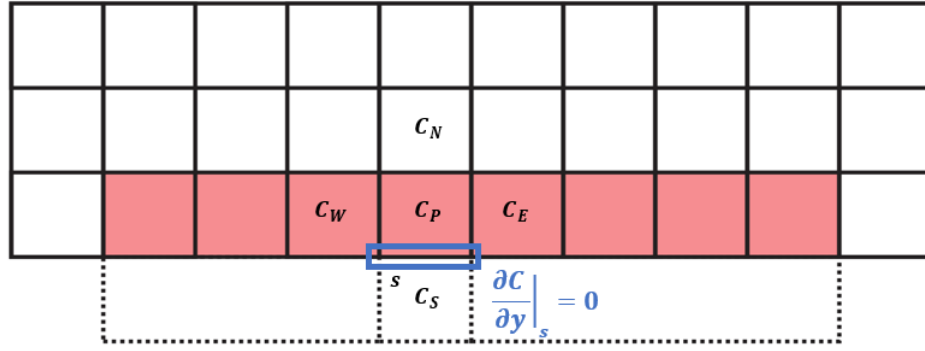


Figura 5.9: Representação dos volumes fictícios para o cálculo da Região 2.
Fonte: O Autor (2023).

Isolando os valores das concentrações da Equação (5.57) e reorganizando seus termos, chega-se à Equação (5.58).

$$\begin{aligned} \left(-u \Delta y - E_L \frac{\Delta y}{\Delta x}\right) C_W^{t+\Delta t} + \left(\frac{\Delta x \Delta y}{\Delta t} + u \Delta y + 2 E_L \frac{\Delta y}{\Delta x} + E_T \frac{\Delta x}{\Delta y}\right) C_P^{t+\Delta t} + \\ \left(-E_L \frac{\Delta y}{\Delta x}\right) C_E^{t+\Delta t} + \left(-E_T \frac{\Delta x}{\Delta y}\right) C_N^{t+\Delta t} = \left(\frac{\Delta x \Delta y}{\Delta t}\right) C_P^t \end{aligned} \quad (5.58)$$

A Equação (5.58), vale para todos os volumes da Região 2.

• Região 3

Para a Região 3, são criados dois volumes fictícios C_S e C_E , demonstrado na Figura 5.10.

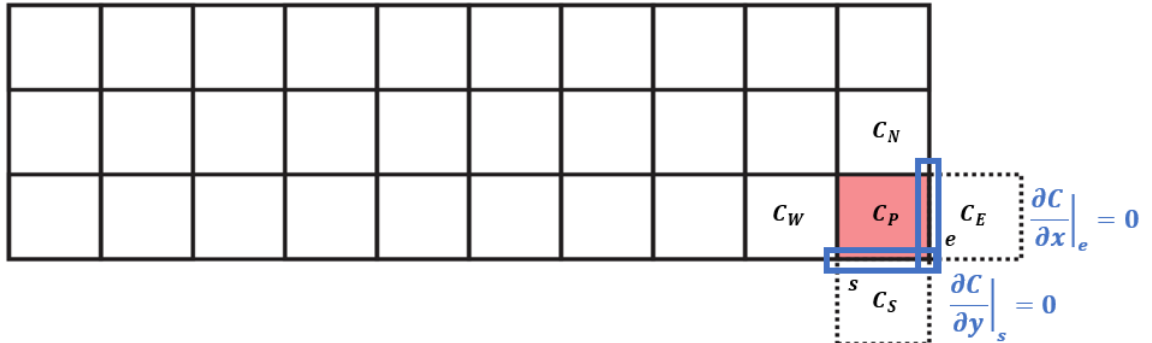


Figura 5.10: Representação dos volumes fictícios para o cálculo da Região 3.
Fonte: O Autor (2023).

Com isso, a concentração na face s é dada pela Equação (5.54) e a concentração na

face e é dada na Equação (5.59).

$$\left. \frac{\partial C^\theta}{\partial x} \right|_e = \frac{C_P^{t+\Delta t} - C_E^{t+\Delta t}}{\Delta x} + \varphi(\Delta x)^2 = 0 \rightarrow C_E^{t+\Delta t} = C_P^{t+\Delta t} \quad (5.59)$$

Substituindo as Equações (5.54) e (5.59), na Equação (5.52), resulta na Equação (5.60).

$$\begin{aligned} \left(\frac{\Delta x \Delta y}{\Delta t} \right) C_P^t &= \left(-E_T \frac{\Delta x}{\Delta y} \right) C_P^{t+\Delta t} + \left(-E_T \frac{\Delta x}{\Delta y} \right) C_N^{t+\Delta t} + \left(-E_L \frac{\Delta y}{\Delta x} \right) C_P^{t+\Delta t} + \\ &\left(\frac{\Delta x \Delta y}{\Delta t} + u \Delta y + 2 E_L \frac{\Delta y}{\Delta x} + 2 E_T \frac{\Delta x}{\Delta y} \right) C_P^{t+\Delta t} + \left(-u \Delta y - E_L \frac{\Delta y}{\Delta x} \right) C_W^{t+\Delta t} \end{aligned} \quad (5.60)$$

Colocando as concentrações em evidência, a Equação (5.60) resulta na Equação (5.61).

$$\begin{aligned} \left(-u \Delta y - E_L \frac{\Delta y}{\Delta x} \right) C_W^{t+\Delta t} + \left(\frac{\Delta x \Delta y}{\Delta t} + u \Delta y + E_L \frac{\Delta y}{\Delta x} + E_T \frac{\Delta x}{\Delta y} \right) C_P^{t+\Delta t} + \\ \left(-E_T \frac{\Delta x}{\Delta y} \right) C_N^{t+\Delta t} = \left(\frac{\Delta x \Delta y}{\Delta t} \right) C_P^t \end{aligned} \quad (5.61)$$

A Equação (5.61) é válida para o volume da Região 3.

• Região 4

Na Região 4, só há necessidade de um volume fictício C_W , indicado na Figura 5.11.

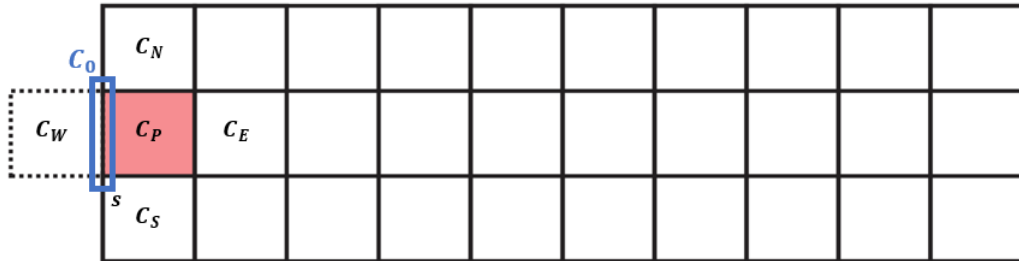


Figura 5.11: Representação dos volumes fictícios para o cálculo da Região 4.

Fonte: O Autor (2023).

Usando o valor de C_W , calculado na Equação (5.53), na Equação (5.52), resulta na Equação (5.62).

$$\begin{aligned} \left(\frac{\Delta x \Delta y}{\Delta t} \right) C_P^t = & \left(-E_T \frac{\Delta x}{\Delta y} \right) C_S^{t+\Delta t} + \left(-E_T \frac{\Delta x}{\Delta y} \right) C_N^{t+\Delta t} + \left(-E_L \frac{\Delta y}{\Delta x} \right) C_E^{t+\Delta t} + \\ & \left(\frac{\Delta x \Delta y}{\Delta t} + u \Delta y + 2 E_L \frac{\Delta y}{\Delta x} + 2 E_T \frac{\Delta x}{\Delta y} \right) C_P^{t+\Delta t} + \left(-u \Delta y - E_L \frac{\Delta y}{\Delta x} \right) (2 C_0 - C_P^{t+\Delta t}) \end{aligned} \quad (5.62)$$

Reorganizado a Equação (5.62), tem-se a Equação (5.63).

$$\begin{aligned} & \left(-E_T \frac{\Delta x}{\Delta y} \right) C_S^{t+\Delta t} + \left(\frac{\Delta x \Delta y}{\Delta t} + 2 u \Delta y + 3 E_L \frac{\Delta y}{\Delta x} + 2 E_T \frac{\Delta x}{\Delta y} \right) C_P^{t+\Delta t} + \\ & \left(-E_L \frac{\Delta y}{\Delta x} \right) C_E^{t+\Delta t} + \left(-E_T \frac{\Delta x}{\Delta y} \right) C_N^{t+\Delta t} = \left(\frac{\Delta x \Delta y}{\Delta t} \right) C_P^t + \left(u \Delta y + E_L \frac{\Delta y}{\Delta x} \right) 2 C_0 \end{aligned} \quad (5.63)$$

A Equação (5.63), calcula as concentrações nos volumes da Região 4.

• Região 6

De forma análoga à Região 4, para o volume da Região 6, utiliza-se apenas um volume fictício C_E , conforme mostrado na Figura 5.12.

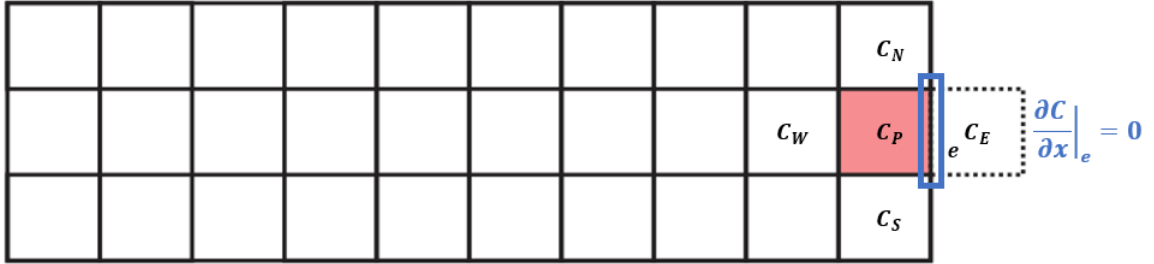


Figura 5.12: Representação dos volumes fictícios para o cálculo da Região 6.

Fonte: O Autor (2023).

O valor do volume fictício C_E , já foi calculado na Equação (5.59). Substituindo esse valor na Equação (5.52), tem-se a Equação (5.64).

$$\begin{aligned} \left(\frac{\Delta x \Delta y}{\Delta t} \right) C_P^t = & \left(-E_T \frac{\Delta x}{\Delta y} \right) C_S^{t+\Delta t} + \left(-E_T \frac{\Delta x}{\Delta y} \right) C_N^{t+\Delta t} + \left(-E_L \frac{\Delta y}{\Delta x} \right) C_P^{t+\Delta t} + \\ & \left(\frac{\Delta x \Delta y}{\Delta t} + u \Delta y + 2 E_L \frac{\Delta y}{\Delta x} + 2 E_T \frac{\Delta x}{\Delta y} \right) C_P^{t+\Delta t} + \left(-u \Delta y - E_L \frac{\Delta y}{\Delta x} \right) C_W^{t+\Delta t} \end{aligned} \quad (5.64)$$

Colocando em evidência os termos de concentração da Equação (5.64), resulta na Equação (5.65), utilizada para calcular as concentrações nos volumes da Região 6.

$$\begin{aligned} & \left(-E_T \frac{\Delta x}{\Delta y} \right) C_S^{t+\Delta t} + \left(-u \Delta y - E_L \frac{\Delta y}{\Delta x} \right) C_W^{t+\Delta t} + \\ & \left(\frac{\Delta x \Delta y}{\Delta t} + u \Delta y + E_L \frac{\Delta y}{\Delta x} + 2 E_T \frac{\Delta x}{\Delta y} \right) C_P^{t+\Delta t} + \left(-E_T \frac{\Delta x}{\Delta y} \right) C_N^{t+\Delta t} = \\ & \left(\frac{\Delta x \Delta y}{\Delta t} \right) C_P^t \end{aligned} \quad (5.65)$$

• Região 7

Na Região 7 se faz necessário a criação de dois volumes fictícios C_W e C_N , mostrado na Figura 5.13. Assim, utiliza-se o cálculo da concentração na face w realizado na Equação (5.53) e, a concentração na face n , obtida conforme Equação (5.66).

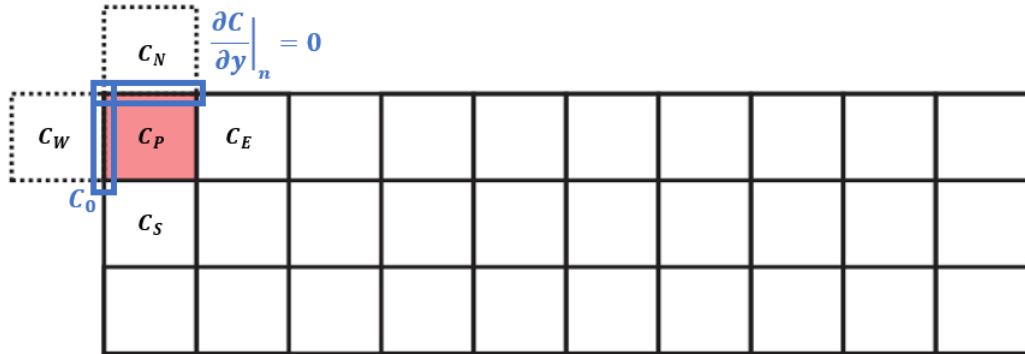


Figura 5.13: Representação dos volumes fictícios para o cálculo da Região 7.
Fonte: O Autor (2022).

$$\left. \frac{\partial C}{\partial y} \right|_n = \frac{C_P^{t+\Delta t} - C_N^{t+\Delta t}}{\Delta y} + \varphi(\Delta y)^2 = 0 \rightarrow C_N^{t+\Delta t} = C_P^{t+\Delta t} \quad (5.66)$$

Substituindo as Equações (5.53) e (5.66), na Equação (5.52), tem-se a Equação (5.67).

$$\begin{aligned} & \left(\frac{\Delta x \Delta y}{\Delta t} \right) C_P^t = \left(-E_T \frac{\Delta x}{\Delta y} \right) C_S^{t+\Delta t} + \left(-E_T \frac{\Delta x}{\Delta y} \right) C_P^{t+\Delta t} + \left(-E_L \frac{\Delta y}{\Delta x} \right) C_E^{t+\Delta t} + \\ & \left(\frac{\Delta x \Delta y}{\Delta t} + u \Delta y + 2 E_L \frac{\Delta y}{\Delta x} + 2 E_T \frac{\Delta x}{\Delta y} \right) C_P^{t+\Delta t} + \left(-u \Delta y - E_L \frac{\Delta y}{\Delta x} \right) (2 C_0 - C_P^{t+\Delta t}) \end{aligned} \quad (5.67)$$

Colocando em evidência as concentrações da Equação (5.67) e reorganizando seus termos, resulta na Equação (5.68).

$$\begin{aligned} \left(-E_T \frac{\Delta x}{\Delta y}\right) C_S^{t+\Delta t} + \left(\frac{\Delta x \Delta y}{\Delta t} + 2u \Delta y + 3E_L \frac{\Delta y}{\Delta x} + E_T \frac{\Delta x}{\Delta y}\right) C_P^{t+\Delta t} \\ \left(-E_L \frac{\Delta y}{\Delta x}\right) C_E^{t+\Delta t} = \left(\frac{\Delta x \Delta y}{\Delta t}\right) C_P^t + \left(u \Delta y + E_L \frac{\Delta y}{\Delta x}\right) 2C_0 \end{aligned} \quad (5.68)$$

A Equação (5.68) é válida para o cálculo no volume da Região 7.

• Região 8

Para a Região 8, é criado um volume fictício C_N , representado na Figura 5.14.

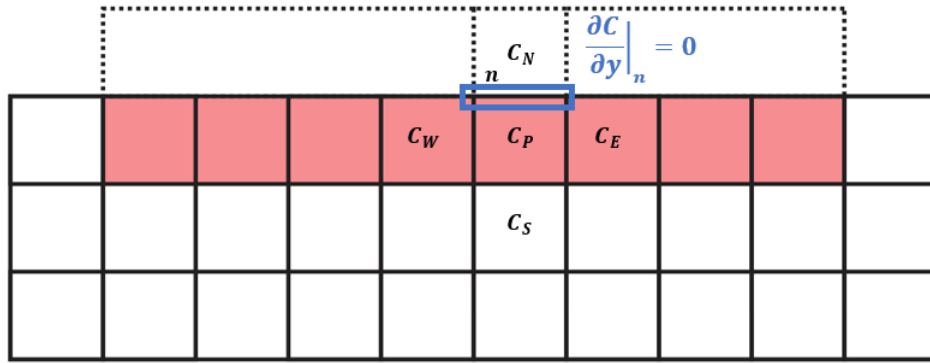


Figura 5.14: Representação dos volumes fictícios para o cálculo da Região 8.
Fonte: O Autor(2023).

Substituindo o valor fictício da concentração C_N , calculado na Equação (5.66), na Equação (5.52), resulta na Equação (5.69).

$$\begin{aligned} \left(\frac{\Delta x \Delta y}{\Delta t}\right) C_P^t = \left(-E_T \frac{\Delta x}{\Delta y}\right) C_S^{t+\Delta t} + \left(-E_T \frac{\Delta x}{\Delta y}\right) C_P^{t+\Delta t} + \left(-E_L \frac{\Delta y}{\Delta x}\right) C_E^{t+\Delta t} + \\ \left(\frac{\Delta x \Delta y}{\Delta t} + u \Delta y + 2E_L \frac{\Delta y}{\Delta x} + 2E_T \frac{\Delta x}{\Delta y}\right) C_P^{t+\Delta t} + \left(-u \Delta y - E_L \frac{\Delta y}{\Delta x}\right) C_W^{t+\Delta t} \end{aligned} \quad (5.69)$$

Reorganizando os termos da Equação (5.69), resulta na Equação (5.70), utilizada para o cálculo das concentrações nos volumes da Região 8.

$$\begin{aligned} & \left(-E_T \frac{\Delta x}{\Delta y}\right) C_S^{t+\Delta t} + \left(-u \Delta y - E_L \frac{\Delta y}{\Delta x}\right) C_W^{t+\Delta t} + \\ & \left(\frac{\Delta x \Delta y}{\Delta t} + u \Delta y + 2 E_L \frac{\Delta y}{\Delta x} + E_T \frac{\Delta x}{\Delta y}\right) C_P^{t+\Delta t} + \left(-E_L \frac{\Delta y}{\Delta x}\right) C_E^{t+\Delta t} = \\ & \left(\frac{\Delta x \Delta y}{\Delta t}\right) C_P^t \end{aligned} \quad (5.70)$$

A solução da formulação bidimensional do problema proposto, através dos Método dos Volumes Finitos, recai em um sistema de equações lineares, do tipo $A\mathbf{x} = \mathbf{b}$. É importante notar, que a matriz dos coeficientes gerada, é pentadiagonal.

Em particular, exemplificando a formulação bidimensional descrita acima, para um domínio dividido em 3 volumes na longitudinal e 3 volumes na transversal, tem-se o sistema de equações lineares dado no Sistema (5.73). Para facilitar a representação, é adotada a seguinte notação:

$d_1 \rightarrow$ Para os coeficientes da concentração no volume de referência $C_P^{t+\Delta t}$;

$d_2 \rightarrow$ Para os coeficientes da concentração no volume $C_E^{t+\Delta t}$;

$d_3 \rightarrow$ Para os coeficientes da concentração no volume $C_W^{t+\Delta t}$;

$d_4 \rightarrow$ Para os coeficientes da concentração no volume $C_S^{t+\Delta t}$;

$d_5 \rightarrow$ Para os coeficientes da concentração no volume $C_N^{t+\Delta t}$;

$F = u \Delta y + E_L \frac{\Delta y}{\Delta x}$.

$$\left\{ \begin{array}{llll} d_1 C_1^{t+\Delta t} + d_2 C_2^{t+\Delta t} & & + d_5 C_4^{t+\Delta t} & = d_1 C_1^t + F \\ d_3 C_1^{t+\Delta t} + d_1 C_2^{t+\Delta t} + d_2 C_3^{t+\Delta t} & & + d_5 C_5^{t+\Delta t} & = d_1 C_2^t \\ & d_3 C_2^{t+\Delta t} + d_1 C_3^{t+\Delta t} & & + d_5 C_6^{t+\Delta t} = d_1 C_3^t \\ d_4 C_1^{t+\Delta t} & & d_1 C_4^{t+\Delta t} + d_2 C_5^{t+\Delta t} & + d_5 C_7^{t+\Delta t} = d_1 C_4^t + F \\ & d_4 C_2^{t+\Delta t} & d_3 C_4^{t+\Delta t} + d_1 C_5^{t+\Delta t} + d_2 C_6^{t+\Delta t} & + d_5 C_8^{t+\Delta t} = d_1 C_5^t \\ & & d_4 C_3^{t+\Delta t} & d_3 C_5^{t+\Delta t} + d_1 C_6^{t+\Delta t} & + d_5 C_9^{t+\Delta t} = d_1 C_6^t \\ & & & d_4 C_4^{t+\Delta t} & d_1 C_7^{t+\Delta t} + d_2 C_8^{t+\Delta t} = d_1 C_7^t + F \\ & & & & d_4 C_5^{t+\Delta t} & d_3 C_7^{t+\Delta t} + d_1 C_8^{t+\Delta t} + d_2 C_9^{t+\Delta t} = d_1 C_8^t \\ & & & & & d_4 C_6^{t+\Delta t} & d_3 C_8^{t+\Delta t} + d_1 C_9^{t+\Delta t} = d_1 C_9^t \end{array} \right. \quad (5.73)$$

O Sistema (5.73), pode ser escrito na forma matricial, representado no Sistema (5.74).

$$\begin{bmatrix}
d_1 & d_2 & & & d_5 \\
d_3 & d_1 & d_2 & & d_5 \\
& d_3 & d_1 & & d_5 \\
d_4 & & & d_1 & d_2 & & d_5 \\
& d_4 & & d_3 & d_1 & d_2 & & d_5 \\
& & d_4 & & d_3 & d_1 & & d_5 \\
& & & d_4 & & d_1 & d_2 & \\
& & & & d_4 & & d_3 & d_1 & d_2 \\
& & & & & d_4 & & d_3 & d_1
\end{bmatrix}
\begin{bmatrix}
C_1^{t+\Delta t} \\
C_2^{t+\Delta t} \\
C_3^{t+\Delta t} \\
C_4^{t+\Delta t} \\
C_5^{t+\Delta t} \\
C_6^{t+\Delta t} \\
C_7^{t+\Delta t} \\
C_8^{t+\Delta t} \\
C_9^{t+\Delta t}
\end{bmatrix}
=
\begin{bmatrix}
d_1 C_1^t + F \\
d_1 C_2^t \\
d_1 C_3^t \\
d_1 C_4^t + F \\
d_1 C_5^t \\
d_1 C_6^t \\
d_1 C_7^t + F \\
d_1 C_8^t \\
d_1 C_9^t
\end{bmatrix} \quad (5.74)$$

Cabe ressaltar, como característica geral desse tipo de formulação, que os coeficientes das diagonais mais afastadas d_4 e d_5 , são das concentrações nos volumes C_S e C_N , respectivamente. Como esses volumes estão abaixo e acima do volume de referência, eles possuem uma distância do mesmo, da ordem de $n_x + 1$, em que n_x representa o número de volumes ao longo do eixo x , do domínio físico. Esse afastamento se reflete na matriz dos coeficientes, ou seja, as diagonais d_4 e d_5 , se distanciam da diagonal principal, na ordem de $n_x + 1$.

5.4 Implementação Computacional

Como demonstrado nos capítulos anteriores, a discretização do domínio espacial do problema de transporte de contaminantes, abordagem unidimensional e bidimensional, proposto nessa pesquisa, resulta em um sistema de equações algébricas do tipo $A \mathbf{x} = \mathbf{b}$, em que A é uma matriz esparsa tridiagonal para o caso unidimensional e uma matriz esparsa pentadiagonal para o caso bidimensional.

Em ambos os problemas, o sistema possui um grande número de incógnitas, o que torna impraticável a solução analítica. Para esses casos são necessárias implementações computacionais, as quais utilizam métodos numéricos diretos ou iterativos, a fim de se chegar à solução exata ou uma aproximação da mesma, que satisfaça algum critério pré-estabelecido [9].

Nesse trabalho, tanto para o caso unidimensional quanto para o bidimensional são realizadas sete implementações computacionais que buscam resolver os respectivos sistemas matriciais. As implementações possuem diferenças estruturais em relação ao armazenamento da matriz dos coeficientes e ao método numérico utilizado na resolução. O objetivo

é analisar modelos que aproveitem a esparsidade da estrutura, de forma a não realizar cálculos nos elementos nulos da matriz, evitando gastos desnecessários de processamento, consumo de memória e tempo de execução dos métodos. Na sequência, são apresentadas as sete implementações desenvolvidas.

• Implementação 1

A Implementação 1 é usada como referência para as outras implementações, pois é a única que, além de trabalhar com o armazenamento da matriz completa, também realiza os cálculos em todos os seus elementos, inclusive os nulos. Utiliza o método iterativo de Gauss-Seidel na forma geral, ou seja, desconsiderando a esparsidade da matriz. A formulação base desse método está demonstrada na Seção 2.3.2.1 e é aplicada para ambos os casos do problema proposto, unidimensional e bidimensional, da mesma maneira.

• Implementação 2

A Implementação 2 também armazena a matriz completa, porém, são realizadas modificações na estrutura do método de Gauss-Seidel, de modo a não computar durante a execução dos cálculos, os elementos nulos da matriz. A nova formulação é apresentada, tanto para o caso unidimensional quanto para o bidimensional da seguinte maneira:

Unidimensional

Toma-se, como exemplo, um sistema de equações que possui uma matriz esparsa tridiagonal, típico da discretização do problema proposto em uma dimensão e onde os elementos não nulos estão dispostos exclusivamente na diagonal principal, diagonal inferior à principal e diagonal superior à principal, como demonstrado na Equação (5.75).

$$\begin{bmatrix} a_{1,1} & a_{1,2} & & & \\ a_{2,1} & a_{2,2} & a_{2,3} & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ & & & a_{n,n-1} & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} \quad (5.75)$$

Nesse tipo de sistema, excetuando a primeira e a última linha da matriz dos coeficientes, todas as outras linhas possuem elementos agrupados de três em três, tendo

como referência a diagonal principal. Fazendo alguns ajustes no método de Gauss-Seidel, tem-se:

- Para o primeiro elemento: $i = 1$

$$x_1^{(k+1)} = \frac{1}{a_{1,1}} \left(y_1 - a_{1,2} x_2^{(k)} \right) \quad (5.76)$$

- Para os elementos: $i = 2$ até $(n - 1)$

$$x_i^{(k+1)} = \frac{1}{a_{i,i}} \left(y_i - a_{i,i-1} x_{i-1}^{(k+1)} - a_{i,i+1} x_{i+1}^{(k)} \right) \quad (5.77)$$

- Para o último elemento: $i = n$

$$x_n^{(k+1)} = \frac{1}{a_{n,n}} \left(y_n - a_{n,n-1} x_{n-1}^{(k+1)} \right) \quad (5.78)$$

As Equações (5.76)-(5.78) descrevem o método de Gauss-Seidel adaptado para uma matriz esparsa tridiagonal. O número de cálculos do método é reduzido, uma vez que não se computa os elementos nulos da matriz dos coeficientes.

Bidimensional

Para o caso bidimensional, o sistema de equações possui uma matriz pentadiagonal, em que os elementos não nulos estão presentes em cinco diagonais da matriz, representado na Equação (5.79).

$$\begin{bmatrix} a_{1,1} & a_{1,2} & & a_{1,n_x+1} & & \\ a_{2,1} & a_{2,2} & a_{2,3} & & \ddots & \\ & a_{3,2} & a_{3,3} & a_{2,3} & & a_{n-n_x,n} \\ a_{n_x+1,1} & & \ddots & \ddots & \ddots & \\ & \ddots & & a_{n-2,n-1} & a_{n-1,n-1} & a_{n-1,n} \\ & & a_{n,n-n_x} & & a_{n,n-1} & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} \quad (5.79)$$

Novamente, é possível realizar ajustes no método computacional de Gauss-Seidel, de forma a computar somente os termos não nulos, assim como no caso unidimensional. Para o modelo bidimensional, os dados da matriz dos coeficientes são divididos em arranjos, que podem ser agrupados de três em três ou de cinco em cinco termos, como descrito abaixo:

- Para o primeiro elemento: $i = 1$

$$x_1^{(k+1)} = \frac{1}{a_{1,1}} \left(y_1 - a_{1,2} x_2^{(k)} - a_{1,n_x+1} x_{n_x+1}^{(k)} \right) \quad (5.80)$$

- Para os elementos: $i = 2$ até n_x

$$x_i^{(k+1)} = \frac{1}{a_{i,i}} \left(y_i - a_{i,i-1} x_{i-1}^{(k+1)} - a_{i,i+1} x_{i+1}^{(k)} - a_{i,i+n_x} x_{i+n_x}^{(k)} \right) \quad (5.81)$$

- Para os elementos: $i = (n_x + 1)$ até $(n - n_x)$

$$x_i^{(k+1)} = \frac{1}{a_{i,i}} \left(y_i - a_{i,i-n_x} x_{i-n_x}^{(k+1)} - a_{i,i-1} x_{i-1}^{(k+1)} - a_{i,i+1} x_{i+1}^{(k)} - a_{i,i+n_x} x_{i+n_x}^{(k)} \right) \quad (5.82)$$

- Para os elementos: $i = [(n - n_x) + 1]$ até $(n - 1)$

$$x_i^{(k+1)} = \frac{1}{a_{i,i}} \left(y_i - a_{i,i-n_x} x_{i-n_x}^{(k+1)} - a_{i,i-1} x_{i-1}^{(k+1)} - a_{i,i+1} x_{i+1}^{(k)} \right) \quad (5.83)$$

- Para o último elemento: $i = n$

$$x_n^{(k+1)} = \frac{1}{a_{n,n}} \left(y_n - a_{n,n-n_x} x_{n-n_x}^{(k+1)} - a_{n,n-1} x_{n-1}^{(k+1)} \right) \quad (5.84)$$

As Equações (5.80)-(5.84) representam uma adaptação do método de Gauss-Seidel para resolução de um sistema com matriz pentadiagonal, de forma a não computar no cálculo os elementos nulos.

• Implementação 3

A estrutura armazenada na Implementação 3 é a matriz completa, porém, o método utilizado na solução numérica é o algoritmo de Thomas, que tanto para o caso unidimensional (descrito na Seção 2.3.1.2) quanto para o caso bidimensional (descrito na Seção 2.3.2.2), devido a característica do método, é desenvolvido para não computar no cálculo os elementos nulos da matriz.

Dessa forma, esta terceira implementação segue a metodologia descrita na segunda implementação, ou seja, armazenamento da matriz completa e solução numérica dada pelo algoritmo de Thomas (Implementação 3), em vez do método Gauss-Seidel (Implementação 2). Em ambas, não são computados os elementos nulos nos cálculos realizados pelos métodos.

• Implementação 4

Na Implementação 4, em vez da utilização da matriz completa, a estrutura armazenada é a matriz comprimida através do método *Compressed Sparse Column* (CSC), utilizando para isso uma função nativa do MATLAB. Para solução numérica é utilizado o método de Gauss-Seidel modificado, conforme descrito na Implementação 2 para as formulações unidimensional e bidimensional, de maneira a não computar os elementos nulos nos cálculos realizados.

• Implementação 5

A Implementação 5 utiliza o método de compressão *Compressed Sparse Column* (CSC), em vez da matriz completa, assim como na Implementação 4. Porém, a solução numérica é obtida pelo algoritmo de Thomas, descrito nas Seções 2.3.1.2 (para o caso unidimensional) e 2.3.2.2 (para o caso bidimensional), em substituição ao método de Gauss-Seidel (Implementação 4).

Até agora, excetuando a primeira (referência), todas as demais implementações apresentadas possuem mudanças na forma de se armazenar a matriz esparsa (completa ou CSC) ou na estrutura de cálculo em relação aos elementos nulos (Gauss-Seidel original, Gauss-Seidel modificado ou algoritmo de Thomas).

Embora essas modificações contribuam na otimização do tempo de execução do método e no armazenamento da estrutura matricial, ainda há o inconveniente de todas partirem do princípio da necessidade de criação da matriz no início do código.

A contribuição de maior relevância do presente trabalho está nas duas últimas implementações, ou seja, Implementações 6 e 7, uma vez que não há a criação da estrutura matricial, a qual é substituída por vetores, como demonstrado na sequência desse texto.

• Implementação 6

Na Implementação 6, a criação da matriz é substituída por vetores, os quais representam as diagonais não nulas da matriz relacionada aos casos unidimensional e bidimensional. Para a solução do problema, é utilizado o método de Gauss-Seidel.

Como não há matrizes, foi necessário realizar mudanças na estrutura de cálculo do método Gauss-Seidel, de forma a trabalhar com vetores em substituição à matriz dos coeficientes, conforme exposto na sequência.

Unidimensional

Para o caso unidimensional, são criados três vetores: **c**, **a** e **b**, que representam as diagonais dos elementos não nulos, como demonstrado na Equação (5.85).

$$\begin{bmatrix} a_1 & b_1 & & & \\ c_2 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-1} & a_{n-1} & b_{n-1} \\ & & & c_n & a_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} \quad (5.85)$$

O vetor **c**, contém os elementos da diagonal inferior à diagonal principal. O vetor **a**, contém os elementos da diagonal principal. Por fim, o vetor **b** contém elementos da diagonal superior à diagonal principal. Como a diagonal inferior não possui elemento na primeira linha da matriz dos coeficientes, assim como a diagonal superior não possui elemento na última linha da referida matriz, foi adotado valores nulos, respectivamente, para que os vetores tivessem o mesmo tamanho, com indicado na Equação (5.86).

$$\mathbf{c} = \begin{bmatrix} 0 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} 0 \\ a_{2,1} \\ \vdots \\ a_{n,n-1} \\ a_{n-1,n-2} \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{n-1} \\ a_n \end{bmatrix} = \begin{bmatrix} a_{1,1} \\ a_{2,2} \\ \vdots \\ a_{n-1,n-1} \\ a_{n,n} \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ 0 \end{bmatrix} = \begin{bmatrix} a_{1,2} \\ a_{2,3} \\ \vdots \\ a_{n-1,n} \\ 0 \end{bmatrix} \quad (5.86)$$

Usando os vetores **c**, **a** e **b** nas Equações (5.76)-(5.78), a aplicação do método de Gauss-Seidel é estruturada conforme segue:

- Para o primeiro elemento: $i = 1$

$$x_1^{(k+1)} = \frac{1}{a_1} \left(y_1 - b_1 x_2^{(k)} \right) \quad (5.87)$$

- Para os elementos: $i = 2$ até $(n - 1)$

$$x_i^{(k+1)} = \frac{1}{a_i} \left(y_i - c_i x_{i-1}^{(k+1)} - b_i x_{i+1}^{(k)} \right) \quad (5.88)$$

- Para o último elemento: $i = n$

$$x_n^{(k+1)} = \frac{1}{a_n} \left(y_n - c_n x_{n-1}^{(k+1)} \right) \quad (5.89)$$

Bidimensional

Para o modelo bidimensional, por se tratar de uma matriz pentadiagonal, são criados cinco vetores: **f**, **c**, **a**, **b** e **e**, conforme demonstrado na Equação (5.90).

$$\begin{bmatrix} a_1 & b_1 & e_1 & & \\ c_2 & a_2 & b_2 & & \ddots \\ & c_2 & a_3 & b_3 & e_{n-n_x} \\ f_{n_x+1} & & \ddots & \ddots & \ddots \\ & \ddots & & c_{n-1} & a_{n-1} & b_{n-1} \\ & & f_n & & c_n & a_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d'_1 + g'_1 \\ d'_2 + g'_2 \\ d'_3 + g'_3 \\ \vdots \\ d'_{n-1} + g'_{n-1} \\ d'_n + g'_n \end{bmatrix} \quad (5.90)$$

A diferença para o modelo unidimensional é o acréscimo dos vetores **f** e **e**, mais afastados da diagonal principal. Para que os vetores ficassem do mesmo tamanho, o vetor **f** recebeu o valor 0 para os índices de 1 até n_x . De forma análoga, o vetor **e** recebeu o valor 0 para os índices de $n - n_x$ até n , como demonstrado na Equação (5.91).

$$\mathbf{f} = \begin{bmatrix} 0 \\ \vdots \\ f_{n_x+1} \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ a_{n_x+1,1} \\ \vdots \\ a_{n,n-n_x} \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} 0 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} 0 \\ a_{2,1} \\ \vdots \\ a_{n-1,n-2} \\ a_{n,n-1} \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{n-1} \\ a_n \end{bmatrix} = \begin{bmatrix} a_{1,1} \\ a_{2,2} \\ \vdots \\ a_{n-1,n-1} \\ a_{n,n} \end{bmatrix} \quad (5.91)$$

$$\mathbf{b} = \begin{bmatrix} a_{1,2} \\ a_{2,3} \\ \vdots \\ a_{n-1,n} \\ 0 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ 0 \end{bmatrix} \quad \mathbf{e} = \begin{bmatrix} e_1 \\ \vdots \\ e_{n-n_x} \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} a_{1,n_x+1} \\ \vdots \\ a_{n-n_x,n} \\ \vdots \\ 0 \end{bmatrix}$$

Substituindo os vetores nas Equações (5.80)-(5.84), a aplicação do método de Gauss-Seidel é estruturada conforme segue:

- Para o primeiro elemento: $i = 1$

$$x_1^{(k+1)} = \frac{1}{a_1} \left(y_1 - b_1 x_2^{(k)} - e_1 x_{n_x+1}^{(k)} \right) \quad (5.92)$$

- Para os elementos: $i = 2$ até n_x

$$x_i^{(k+1)} = \frac{1}{a_i} \left(y_i - c_i x_{i-1}^{(k+1)} - b_i x_{i+1}^{(k)} - e_i x_{i+n_x}^{(k)} \right) \quad (5.93)$$

- Para os elementos: $i = (n_x + 1)$ até $(n - n_x)$

$$x_i^{(k+1)} = \frac{1}{a_i} \left(y_i - f_i x_{i-n_x}^{(k+1)} - c_i x_{i-1}^{(k+1)} - b_i x_{i+1}^{(k)} - e_i x_{i+n_x}^{(k)} \right) \quad (5.94)$$

- Para os elementos: $i = [(n - n_x) + 1]$ até $(n - 1)$

$$x_i^{(k+1)} = \frac{1}{a_i} \left(y_i - f_i x_{i-n_x}^{(k+1)} - c_i x_{i-1}^{(k+1)} - b_i x_{i+1}^{(k)} \right) \quad (5.95)$$

- Para o último elemento: $i = n$

$$x_n^{(k+1)} = \frac{1}{a_n} \left(y_n - f_n x_{n-n_x}^{(k+1)} - c_n x_{n-1}^{(k+1)} \right) \quad (5.96)$$

• Implementação 7

Na Implementação 7 também não há a criação da estrutura matricial. A solução numérica é obtida através do Algoritmo de Thomas, que também utiliza vetores em substituição à matriz dos coeficientes, conforme procedimento descrito na Implementação 6. A Implementação 7 é aplicada para os modelos unidimensional e bidimensional.

Unidimensional

Para o caso unidimensional, em que a matriz dos coeficientes é tridiagonal, após a construção dos vetores contendo os elementos da matriz, já descrito na Implementação 6, utiliza-se o algoritmo de Thomas, conforme formulação descrita na Seção 2.3.1.2.

Bidimensional

No caso bidimensional, após a construção dos vetores contendo os elementos da matriz, já descrito na Implementação 6, utiliza-se o algoritmo de Thomas modificado para matriz pentadiagonal, conforme formulação descrita na Seção 2.3.2.2.

Com a utilização dos vetores, em substituição a matriz dos coeficientes, espera-se uma redução do número de cálculos, bem como do consumo de memória, realizados pela máquina, uma vez que os elementos nulos da matriz dos coeficientes, além de não serem computados no cálculo, também não são armazenados pelo programa.

Capítulo 6

RESULTADOS E DISCUSSÕES

Todos os resultados apresentados na sequência deste capítulo foram realizados em um computador com arquitetura de armazenamento em precisão dupla, composto pelos componentes descritos na Tabela 6.1.

Tabela 6.1: Componentes físicos do computador em que os testes foram executados.

Componentes	Descrição
Processador	intel® Core™ i7 – 2.4 GHz com Turbo Boost 3.0 GHz
Placa de vídeo	NVIDIA® GeForce® GTX1060 com 6 GB de VRAM
Memória	16 GB DDR4 L
HDD	SSD de 1.000GB

Fonte: O Autor (2023).

Para a execução dos testes, foram utilizadas sete implementações computacionais, as quais foram descritas na Seção 5.4, e apresentadas resumidamente, conforme Tabela 6.2.

Tabela 6.2: Tipos de implementações desenvolvidas na presente pesquisa.

	Estrutura Armazenada	Método Utilizado	Estrutura de Cálculo
Implementação 1	Matriz completa	Gauss-Seidel	Todos elementos
Implementação 2	Matriz completa	Gauss-Seidel	Elementos não nulos
Implementação 3	Matriz completa	Thomas (TDMA)	Elementos não nulos
Implementação 4	Matriz CSC	Gauss-Seidel	Elementos não nulos
Implementação 5	Matriz CSC	Thomas (TDMA)	Elementos não nulos
Implementação 6	Vetores	Gauss-Seidel	Elementos não nulos
Implementação 7	Vetores	Thomas (TDMA)	Elementos não nulos

Fonte: O Autor (2023).

Com objetivo de facilitar a interpretação dos dados gráficos, é adotada a seguinte nomenclatura para as implementações:

Implementação 1 \rightarrow GS - Matriz Completa: Todos Elementos;

Implementação 2 \rightarrow GS - Matriz Completa: Elementos não nulos;

Implementação 3 \rightarrow AT - Matriz Completa: Elementos não nulos;

Implementação 4 \rightarrow GS - Matriz CSC: Elementos não nulos;

Implementação 5 \rightarrow AT - Matriz CSC: Elementos não nulos;

Implementação 6 \rightarrow GS - Vetores: Elementos não nulos;

Implementação 7 \rightarrow AT - Vetores: Elementos não nulos;

Todos os resultados apresentados nos testes, em relação ao tempo de execução das implementações, foram obtidos executando o programa dez vezes e tirando a média de tempo.

Cabe ressaltar que, apesar do MATLAB possuir diversas funções já implementadas internamente, as quais possibilitam agilidade em algumas etapas de criação de um código, neste trabalho, todo o desenvolvimento computacional de construção de matrizes, bem como dos métodos adotados para a solução das mesmas, foi realizado pelo próprio autor, sendo utilizadas apenas a função nativa do referido software para o cálculo do tempo de execução e alocação de memória. Além disso, também foi utilizada de forma intencional, a função *sparse*, visando a comparação entre uma rotina pré-definida do próprio software com as demais estruturas desenvolvidas.

6.1 Resultados do Estudo de Caso 01: Problema Unidimensional

Para fins de comparação das implementações, foram realizados 7 testes com as mesmas condições iniciais e de contorno, alterando apenas o número de volumes no espaço (n_x), que variou seguindo um critério pessoal do autor, resultando em sistemas matriciais com as seguintes ordens: 100×100 , 500×500 , 1.000×1.000 , 1.500×1.500 , 3.000×3.000 , 40.000×40.000 e $50.000.000 \times 50.000.000$.

Na Tabela 6.3 são apresentados os parâmetros usados nas implementações, no que se refere ao experimento realizado por Sousa (2009) [22], o qual contemplou um período total de coleta de dados igual a 500 s.

Tabela 6.3: Parâmetros utilizados nos testes unidimensionais.

Valores Adotados					
u	=	0,59	m/s	C_0	= 15,5 mg/L
E_L	=	1,82	m^2/s	x_{lanc}	= 100 m
M	=	2.000	g	x_{col}	= 200 m
x_f	=	500	m	n_t	= 550

Fonte: O Autor (2023).

Os dados apresentados na Tabela 6.3, podem ser definidos como:

- u = Velocidade do rio;
- E_L = Coeficiente de dispersão longitudinal;
- M = Massa do poluente;
- x_f = Comprimento do trecho analisado;
- n_t = número de passos no tempo;
- C_0 = Concentração inicial do rio;
- x_{lanc} = Posição de lançamento do poluente;
- x_{col} = Posição de coleta do poluente.

• **Teste 1 - Para $n_x = 100$ (Ordem da Matriz: 100×100)**

No primeiro teste realizado foi adotada uma discretização do domínio espacial considerando 100 volumes, ou seja, $n_x = 100$ e, com isso, obteve-se uma matriz quadrada de ordem 100 (10.000 elementos).

O gráfico da Figura 6.1 representa a concentração do rio no ponto de coleta ao longo do tempo. O resultado indica que todas as implementações convergiram e apresentaram o mesmo comportamento, com pontos de máximos e mínimos, semelhantes.

Por outro lado, na Tabela 6.4 são apresentados os resultados do tempo de execução das implementações e sua relação com o tempo total. Os dados da referida tabela foram obtidos executando todas as implementações em um mesmo programa e o tempo total é o somatório de tempo de todas implementações. Não estão inclusos o tempo de criação das estruturas (matriz e vetores).

Para a primeira ordem testada, com uma matriz relativamente pequena, as Implementações 4 e 5, que utilizaram o método de compressão CSC, apresentaram o pior desempenho. O comprometimento do tempo para essas implementações ocorre porque a cada cálculo realizado o programa percorre toda estrutura comprimida até chegar aos elementos necessários para sua execução.

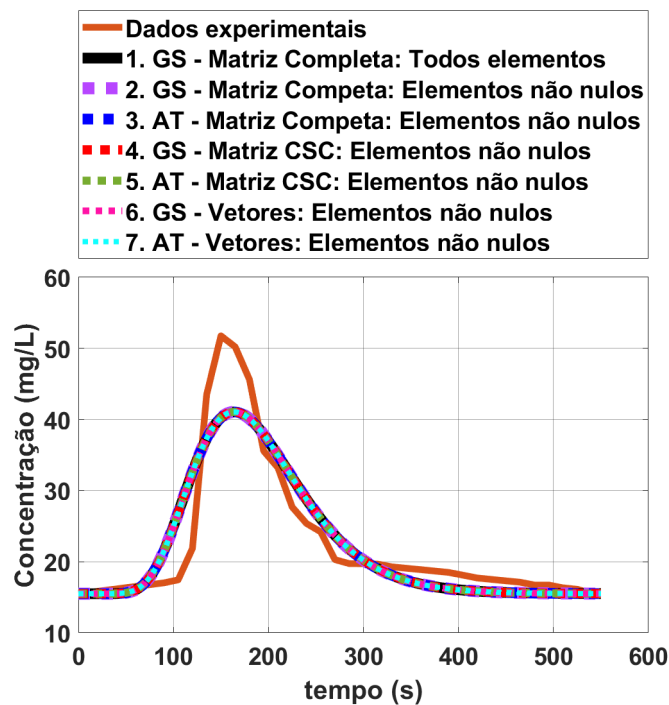


Figura 6.1: Teste Unidimensional - Perfil da concentração para matriz de ordem 100×100 .
Fonte: O Autor (2023).

Tabela 6.4: Teste Unidimensional - Tempos de execução para matriz de ordem 100×100 .

Implementação Computacional	Tempo de Execução (s)	% do Tempo Total
1. GS - Matriz Completa: Todos Elementos	0,116	5,649%
2. GS - Matriz Completa: Elementos não nulos	0,011	0,536%
3. AT - Matriz Completa: Elementos não nulos	0,013	0,609%
4. GS - Matriz CSC: Elementos não nulos	1,312	63,891%
5. AT - Matriz CSC: Elementos não nulos	0,584	28,459%
6. GS - Vetores: Elementos não nulos	0,009	0,438%
7. AT - Vetores: Elementos não nulos	0,009	0,419%

Fonte: O Autor (2023).

Em contrapartida, os melhores tempos foram das Implementações 6 e 7, ambas utilizaram vetores em substituição à matriz dos coeficientes. As Implementações 2 e 3 apresentaram tempos de execução semelhantes às implementações que utilizaram vetores, isso porque, embora a estrutura armazenada seja a matriz completa, os cálculos são realizados apenas nos elementos não nulos.

Quanto aos dados de alocação de memória, demonstrados na Tabela 6.5, as implementações que utilizaram a matriz completa apresentaram um consumo superior a 90% da memória do programa. Os melhores resultados foram para as implementações que

utilizaram vetores em substituição à matriz dos coeficientes (3% da memória total) e as implementações que utilizaram o sistema de compressão CSC (7% da memória total).

Tabela 6.5: Teste Unidimensional - Consumo de memória para matriz de ordem 100×100 .

Tipo de estrutura $\gamma = 0,23$	Memória Utilizada (<i>bytes</i>)	% do Consumo Total
Matriz Completa	80.000	90,93%
Matriz Comprimida CSC	5.576	6,34 %
Vetores	2.400	2,73 %

Fonte: O Autor (2023).

• **Teste 2 - Para $n_x = 500$ (Ordem da Matriz: 500×500)**

No segundo teste realizado, foi adotada uma discretização do domínio espacial considerando 500 volumes, ou seja, $n_x = 500$ e, com isso, obteve-se uma matriz quadrada de ordem 500 (25.000 elementos). Todas as implementações convergiram para a solução, apresentando o mesmo comportamento ao longo do tempo, conforme demonstrado na Figura 6.2.

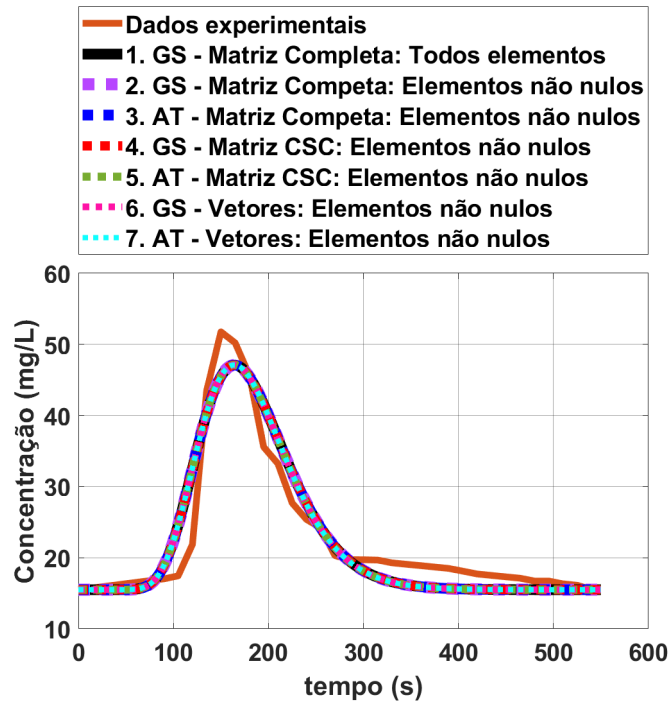


Figura 6.2: Teste Unidimensional - Perfil da concentração para matriz de ordem 500×500 .

Fonte: O Autor (2023).

Em relação ao tempo de execução, de acordo com a Tabela 6.6, as Implementações 3 e 7 obtiveram os melhores desempenhos, seguidos pelas Implementações 2 e 6. As

Implementações 4 e 5, embora tenham apresentado uma melhora em relação ao Teste 1, permaneceram com os piores tempos.

Tabela 6.6: Teste Unidimensional - Tempos de execução para matriz de ordem 500×500 .

Implementação Computacional	Tempo de Execução (s)	% do Tempo Total
1. GS - Matriz Completa: Todos Elementos	5,675	17,104%
2. GS - Matriz Completa: Elementos não nulos	0,069	0,208%
3. AT - Matriz Completa: Elementos não nulos	0,025	0,075%
4. GS - Matriz CSC: Elementos não nulos	24,454	73,704%
5. AT - Matriz CSC: Elementos não nulos	2,867	8,640%
6. GS - Vetores: Elementos não nulos	0,069	0,208%
7. AT - Vetores: Elementos não nulos	0,020	0,066%

Fonte: O Autor (2023).

Quanto ao consumo de memória do programa, demonstrado na Tabela 6.7, as implementações que trabalharam com a matriz completa consumiram mais de 98% da memória total, enquanto que as implementações que utilizaram vetores, ocuparam 0,59%.

Tabela 6.7: Teste Unidimensional - Consumo de memória para matriz de ordem 500×500 .

Tipo de estrutura $\gamma = 0, 17$	Memória Utilizada (bytes)	% do Consumo Total
Matriz Completa	2.000.000	98,04%
Matriz Comprimida CSC	27.976	1,37 %
Vetores	12.000	0,59 %

Fonte: O Autor (2023).

• **Teste 3 - Para $n_x = 1.000$ (Ordem da Matriz: 1.000×1.000)**

No terceiro teste realizado, foi adotada uma discretização do domínio espacial considerando 1.000 volumes, ou seja, $n_x = 1.000$ e, com isso, obteve-se uma matriz quadrada de ordem 1.000 (1.000.000 elementos).

Novamente, todas as implementações convergiram para a solução, apresentando o mesmo padrão de comportamento ao longo do tempo. O resultado é apresentado no gráfico da Figura 6.3.

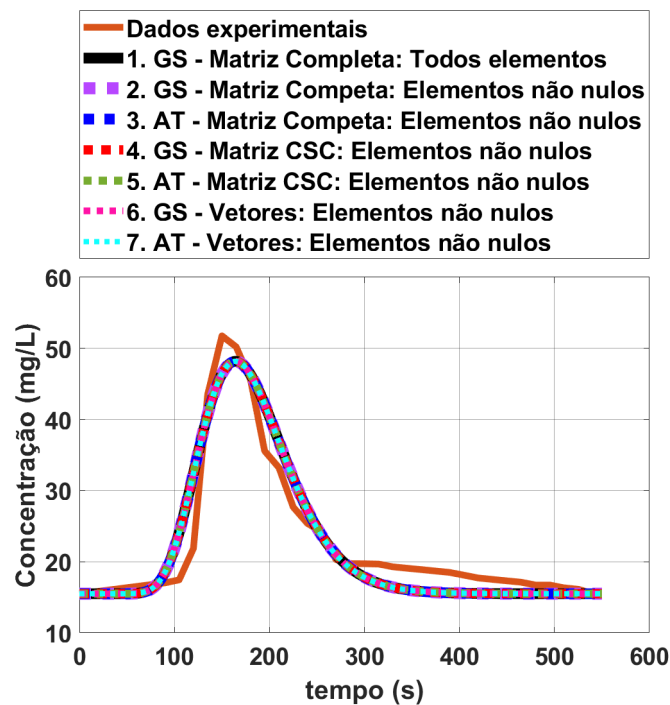


Figura 6.3: Teste Unidimensional - Perfil da concentração para matriz de ordem 1.000×1.000 .

Fonte: O Autor (2023).

Os tempos de execução são demonstrados na Tabela 6.8. Outra vez, as Implementações 3 e 7 foram as que mais rápido convergiram para a solução. As Implementações 4 e 5 continuam apresentando melhora de desempenho, porém, ainda com os piores tempos de execução.

Tabela 6.8: Teste Unidimensional - Tempos de execução para matriz de ordem 1.000×1.000 .

Implementação Computacional	Tempo de Execução (s)	% do Tempo Total
1. GS - Matriz Completa: Todos Elementos	60.038	29,420%
2. GS - Matriz Completa: Elementos não nulos	0,259	0,127%
3. AT - Matriz Completa: Elementos não nulos	0,042	0,020%
4. GS - Matriz CSC: Elementos não nulos	137,673	67,464%
5. AT - Matriz CSC: Elementos não nulos	5,770	2,828%
6. GS - Vetores: Elementos não nulos	0,252	0,123%
7. AT - Vetores: Elementos não nulos	0,035	0,017%

Fonte: O Autor (2023).

De acordo com os dados de consumo de memória, representados na Tabela 6.9, mais de 99% da memória utilizada no programa foram para as implementações que utilizaram a matriz completa. O melhor desempenho ficou para as implementações que utilizaram

vetores, com 0,3% da memória total. As implementações que trabalham com a matriz comprimida, através do método CSC, também apresentaram um consumo de memória inferior a 1%.

Tabela 6.9: Teste Unidimensional - Consumo de memória para matriz de ordem 1.000×1.000 .

Tipo de Estrutura $\gamma = 0, 16$	Memória Utilizada (<i>bytes</i>)	% do Consumo Total
Matriz Completa	8.000.000	99,01%
Matriz Comprimida CSC	55.976	0,69 %
Vetores	24.000	0,30 %

Fonte: O Autor (2023).

• **Teste 4 - Para $n_x = 1.500$ (Ordem da Matriz: 1.500×1.500)**

No quarto teste realizado, foi adotada uma discretização do domínio espacial considerando 1.500 volumes, ou seja, $n_x = 1.500$ e, com isso, obteve-se uma matriz quadrada de ordem 1.500 (2.250.000 elementos). De acordo com o gráfico de concentração da Figura 6.4, mais uma vez, todas as implementações convergiram, apresentando pontos de concentrações máximos e mínimos muito próximos.

Em relação ao tempo de execução, os resultados apresentados na Tabela 6.10 indicam tendências importantes. A Implementação 1 (referência) passou a ter o pior desempenho, superando, negativamente, as Implementações 4 e 5 (matriz comprimida pelo método CSC). A explicação está na estrutura de cálculo dos métodos. Nas Implementações 4 e 5, a cada cálculo executado, o programa percorre todo o vetor dos elementos não nulos. Já na Implementação 1, a cada iteração o programa faz cálculos em toda matriz, inclusive nos elementos nulos. Com isso, de acordo com que se aumenta a ordem da matriz, aumenta também seu grau de esparsidade e, conseqüentemente, o número de elementos nulos da estrutura matricial, tornando mais custoso computacionalmente percorrer toda matriz a cada iteração (Implementação 1), do que percorrer o vetor dos elementos não nulos a cada execução de cálculo (Implementações 4 e 5).

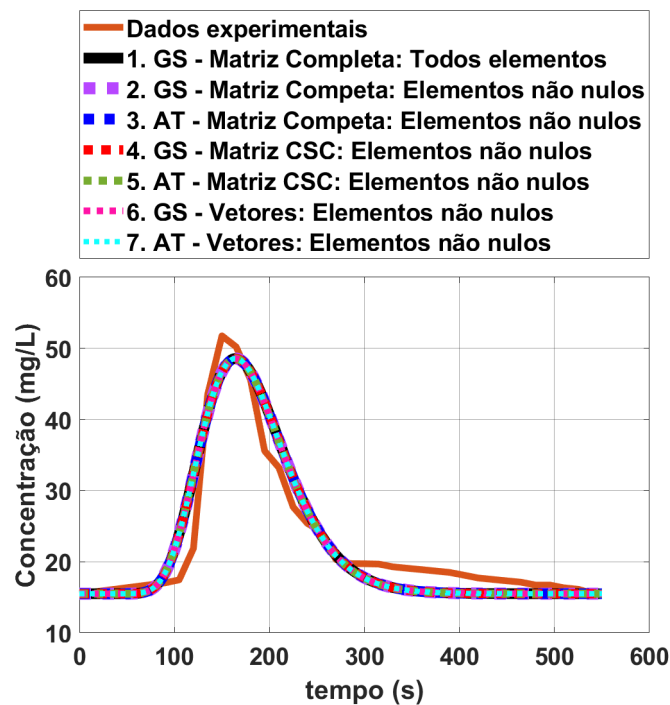


Figura 6.4: Teste Unidimensional - Perfil da concentração para matriz de ordem 1.500×1.500 .

Fonte: O Autor (2023).

Tabela 6.10: Teste Unidimensional - Tempos de execução para matriz de ordem 1.500×1.500 .

Implementação Computacional	Tempo de Execução (s)	% do Tempo Total
1. GS - Matriz Completa: Todos Elementos	531,086	58,370%
2. GS - Matriz Completa: Elementos não nulos	0,699	0,077%
3. AT - Matriz Completa: Elementos não nulos	0,063	0,009%
4. GS - Matriz CSC: Elementos não nulos	367,466	40,387%
5. AT - Matriz CSC: Elementos não nulos	9,833	1,081%
6. GS - Vetores: Elementos não nulos	0,631	0,069%
7. AT - Vetores: Elementos não nulos	0,085	0,007%

Fonte: O Autor (2023).

Quanto ao consumo de memória do programa, demonstrado na Tabela 6.11, manteve-se a prevalência de economia para as implementações que utilizaram vetores. As implementações que trabalharam com a matriz completa apresentaram o pior desempenho.

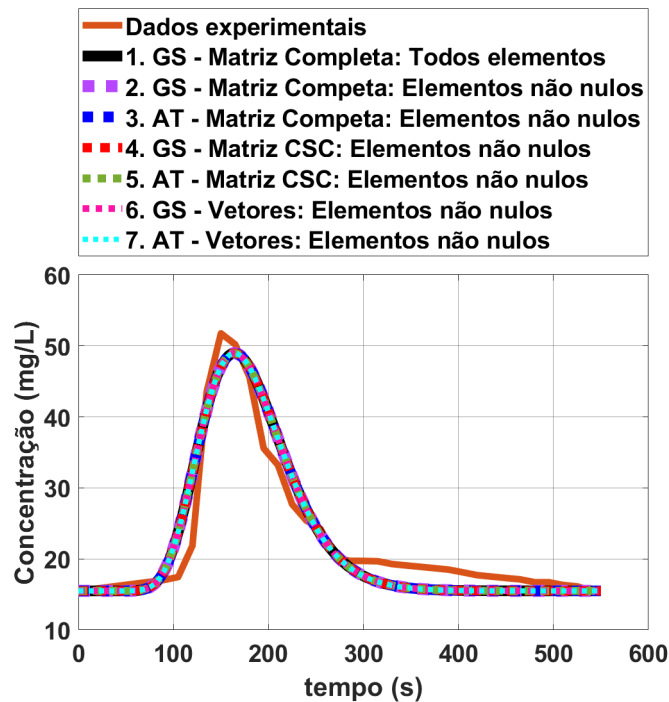
Tabela 6.11: Teste Unidimensional - Consumo de memória para matriz de ordem 1.500×1.500 .

Tipo de Estrutura $\gamma = 0, 15$	Memória Utilizada (<i>bytes</i>)	% do Consumo Total
Matriz Completa	18.000.000	99,34%
Matriz Comprimida CSC	83.976	0,46 %
Vetores	36.000	0,20 %

Fonte: O Autor (2023).

• **Teste 5 - Para $n_x = 3.000$ (Ordem da Matriz: 3.000×3.000)**

No quinto teste realizado, foi adotada uma discretização do domínio espacial considerando 3.000 volumes, ou seja, $n_x = 3.000$ e, com isso, obteve-se uma matriz quadrada de ordem 3.000 (9.000.000 elementos). De acordo com a Figura 6.5, assim como nos testes anteriores, todas as implementações convergiram.

Figura 6.5: Teste Unidimensional - Perfil da concentração para matriz de ordem 3.000×3.000 .

Fonte: O Autor (2023).

Também é importante destacar, com base nas Figuras 6.4 e 6.5, que o aumento no número de volumes do Teste 4 (1.500 volumes) para o Teste 5 (3.000 volumes) não impactou em uma variação significativa no perfil da concentração entre ambos os testes. Sendo

assim, já é possível ver uma estabilidade do método no que se refere ao refinamento da malha utilizada.

Além disso, os dados mostrados na Tabela 6.12, que faz referência aos tempos de execução, seguem o mesmo resultado do teste anterior. As Implementações 3 e 7 apresentaram o melhor desempenho e a Implementação 1 se consolida como o pior tempo de execução.

Tabela 6.12: Teste Unidimensional - Tempos de execução para matriz de ordem 3.000×3.000 .

Implementação Computacional	Tempo de Execução (s)	% do Tempo Total
1. GS - Matriz Completa: Todos Elementos	6.353,00	73,302%
2. GS - Matriz Completa: Elementos não nulos	4,324	0,051%
3. AT - Matriz Completa: Elementos não nulos	0,100	0,001%
4. GS - Matriz CSC: Elementos não nulos	2.058,00	24,393%
5. AT - Matriz CSC: Elementos não nulos	17,841	0,211%
6. GS - Vetores: Elementos não nulos	3,399	0,040%
7. AT - Vetores: Elementos não nulos	0,088	0,001%

Fonte: O Autor (2023).

Novamente, como indicado na Tabela 6.13, as implementações que utilizaram vetores no lugar de matrizes, apresentaram uma grande economia de memória do programa. O pior resultado, como nos outros testes, foram das implementações que trabalharam com a matriz completa.

Tabela 6.13: Teste Unidimensional - Consumo de memória para matriz de ordem 3.000×3.000 .

Tipo de estrutura $\gamma = 0, 13$	Memória Utilizada (<i>bytes</i>)	% do Consumo Total
Matriz Completa	72.000.000	99,67%
Matriz Comprimida CSC	167.976	0,23 %
Vetores	72.000	0,10 %

Fonte: O Autor (2023).

• Teste 6 - Para $n_x = 40.000$ (Ordem da Matriz: 40.000×40.000)

No sexto teste realizado, teve-se como objetivo avaliar o limite máximo de cálculo e armazenamento da máquina, uma vez que os Testes 4 e 5 indicaram uma estabilidade do

método quanto ao ajuste entre o perfil das concentrações numéricas e dados experimentais. Para isso, foi adotada uma discretização do domínio espacial considerando 40.000 volumes, ou seja, $n_x = 40.000$ e, com isso, obteve-se uma matriz quadrada de ordem 40.000 (1.600.000.000 elementos).

As implementações que utilizaram a matriz completa apresentaram erros ao compilar o programa, pois exigiram uma capacidade de memória livre, para armazenamento da matriz, superior à disponível pelo computador onde os testes foram feitos. Com isso, apenas as Implementações 6 e 7 convergiram para a solução na compilação dos códigos.

Na Figura 6.16 é apresentado o gráfico da concentração ao longo do tempo, onde é possível verificar que as Implementações 6 e 7 convergiram para a solução.

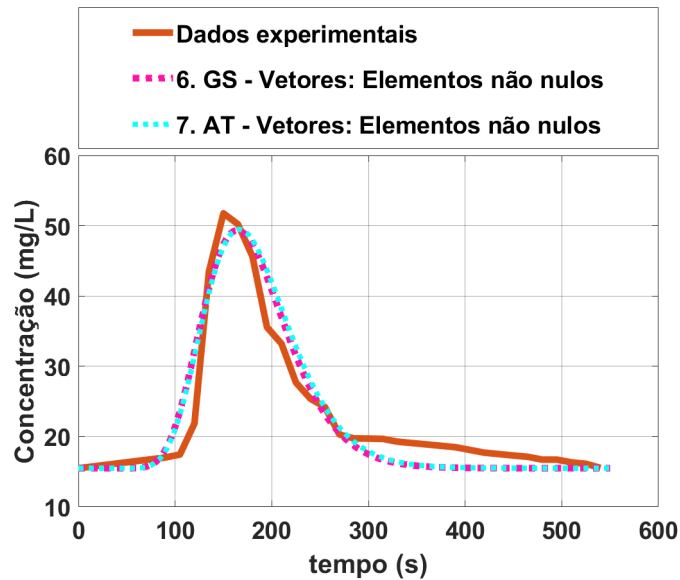


Figura 6.6: Teste Unidimensional - Perfil da concentração para matriz de ordem 40.000×40.000 .

Fonte: O Autor (2023).

Por outro lado, na Tabela 6.14 são demonstrados os dados referentes aos tempos de execução. A Implementação 6 foi responsável por mais de 99% do tempo total do programa, o que era esperado, devido ao grande número de iterações necessárias pelo método de Gauss-Seidel para convergir à solução.

Em relação ao consumo de memória, apresentado na Tabela 6.15, o resultado foi bastante significativo. As implementações que utilizaram vetores demandaram do computador 960.000 *bytes* (0,00096 *GB*) de memória para sua execução. Caso fosse utilizada alguma implementação que dependesse da criação matriz, o consumo de memória seria de 12.800.000.000 *bytes* ou 12,8 *GB*.

Tabela 6.14: Teste Unidimensional - Tempos de execução para matriz de ordem 40.000×40.000 .

Implementação Computacional	Tempo de Execução (s)	% do Tempo Total
1. GS - Matriz Completa: Todos Elementos	-	-
2. GS - Matriz Completa: Elementos não nulos	-	-
3. AT - Matriz Completa: Elementos não nulos	-	-
4. GS - Matriz CSC: Elementos não nulos	-	-
5. AT - Matriz CSC: Elementos não nulos	-	-
6. GS - Vetores: Elementos não nulos	3.824,9	99,99%
7. AT - Vetores: Elementos não nulos	0,56	0,001%

Fonte: O Autor (2023).

Tabela 6.15: Teste Unidimensional - Consumo de memória para matriz de ordem 40.000×40.000 .

Tipo de Estrutura	Memória Utilizada (<i>bytes</i>)	% do Consumo Total
Matriz Completa	-	-
Matriz Comprimida CSC	-	-
Vetores	960.000	100 %

Fonte: O Autor (2023).

- **Teste 7 - Para $n_x = 50.000.000$ (Ordem da Matriz: $50.000.000 \times 50.000.000$)**

Com objetivo de testar a eficiência da substituição da matriz dos coeficientes pelos vetores, foi adotada uma discretização do domínio espacial considerando 50.000.000 volumes, ou seja, $n_x = 50.000.000$ e, com isso, obteve-se uma matriz quadrada de ordem 50.000.000 ($2,5 \times 10^{15}$ elementos). Para a solução do sistema foi utilizada somente a Implementação 7, isso porque, embora a Implementação 6 apresente a mesma exigência de memória da Implementação 7, por se tratar de um método iterativo, os ajustes realizados no critério de parada para se chegar à solução desejada, aumentou demais o número de iterações, comprometendo o tempo de execução da implementação. O gráfico com perfil da concentração é apresentado na Figura 6.7.

No que diz respeito ao tempo de execução da Implementação 7, o valor é apresentado na Tabela 6.16.

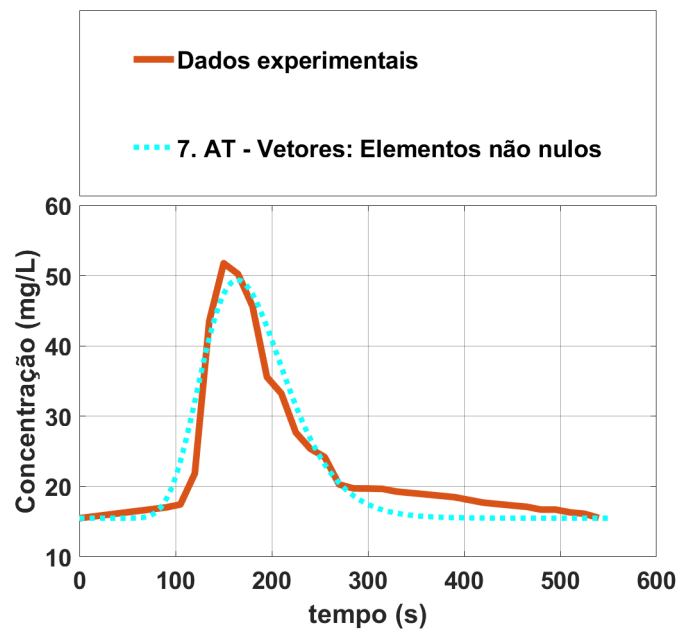


Figura 6.7: Teste Unidimensional - Perfil da concentração para matriz de ordem $50.000.000 \times 50.000.000$.

Fonte: O Autor (2023).

Tabela 6.16: Teste Unidimensional - Tempos de execução para matriz de ordem $50.000.000 \times 50.000.000$.

Implementação Computacional	Tempo de Execução (s)	% do Tempo Total
1. GS - Matriz Completa: Todos Elementos	-	-
2. GS - Matriz Completa: Elementos não nulos	-	-
3. AT - Matriz Completa: Elementos não nulos	-	-
4. GS - Matriz CSC: Elementos não nulos	-	-
5. AT - Matriz CSC: Elementos não nulos	-	-
6. GS - Vetores: Elementos não nulos	-	-
7. AT - Vetores: Elementos não nulos	970,445	100%

Fonte: O Autor (2023).

Em relação ao consumo de memória, apresentado na Tabela 6.17, os vetores demandaram do computador $1,2 \times 10^9$ bytes (1,2 GB) de memória. Caso fosse utilizada alguma implementação que dependesse da criação da matriz, o consumo de memória seria de 2×10^{16} bytes ou 20.000.000 GB.

Tabela 6.17: Teste Unidimensional - Consumo de memória para matriz de ordem $50.000.000 \times 50.000.000$.

Tipo de Estrutura	Memória Utilizada (<i>bytes</i>)	% do Consumo Total
Matriz Completa	-	-
Matriz Comprimida CSC	-	-
Vetores	$1,2 \times 10^9$	100 %

Fonte: O Autor (2023).

Diante dos testes executados foi possível observar que, com o crescimento da ordem da matriz, houve um refinamento no perfil da concentração, aproximando o valor numérico ao valor dos dados experimentais. Sendo assim, na Figura 6.8 é apresentada a comparação dos gráficos da concentração de todas as malhas utilizadas, onde foi usada a Implementação 7 para na geração dos referidos resultados, pois foi a implementação que obteve o melhor desempenho em todos os testes.

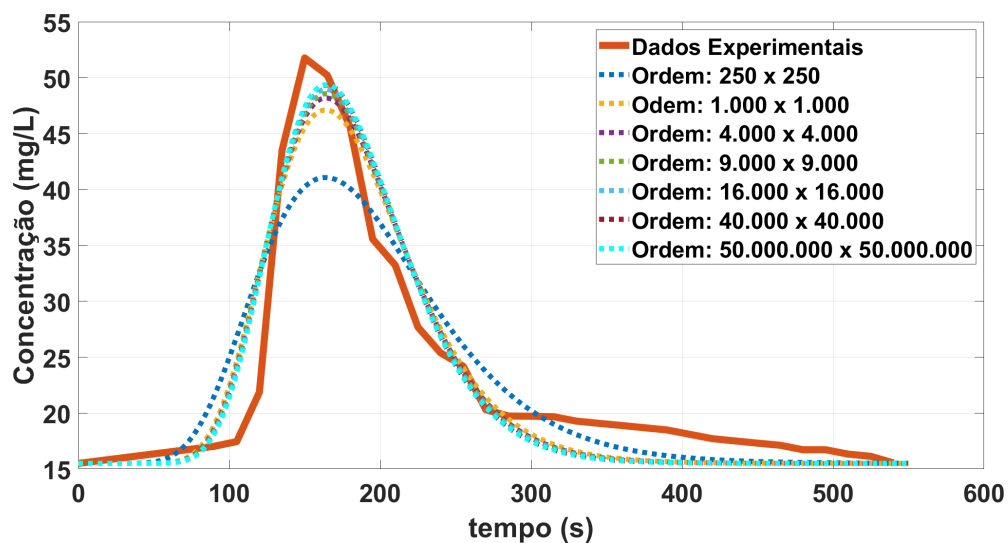


Figura 6.8: Comparação dos perfis de concentração para o caso unidimensional em relação ao aumento da ordem da matriz utilizando a Implementação 7.

Fonte: O Autor (2023).

- **Análise referente ao tempo de criação das estruturas**

Por fim, buscou-se uma relação de tempo entre a execução do método numérico (Gauss-Seidel e algoritmo de Thomas) e a criação das estruturas (matrizes ou vetores) com o intuito de verificar o impacto de ambas no tempo total das implementações.

No gráfico da Figura 6.9 é apresentada a comparação do tempo de execução das Implementações 2 e 6 com o tempo de criação das estruturas (matriz e vetores), ambas utilizam o método iterativo de Gauss-Seidel.

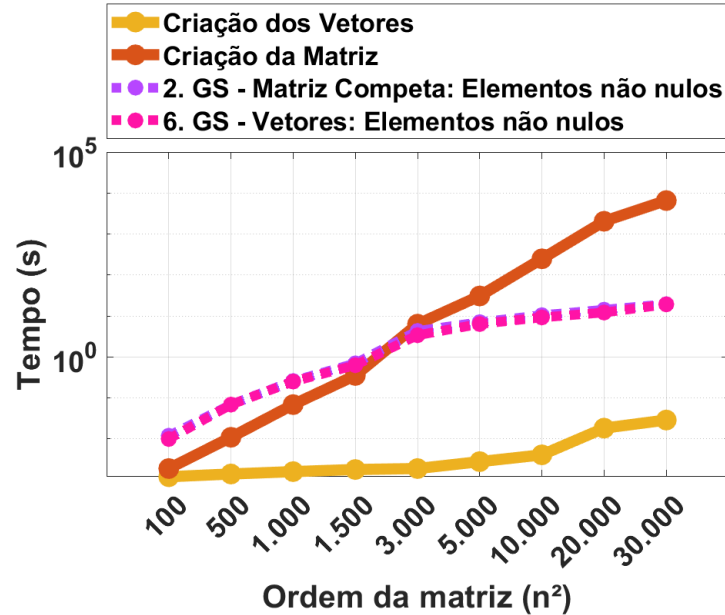


Figura 6.9: Teste Unidimensional - Comparação dos tempos de criação das estruturas (Matriz e Vetores) com a execução das implementações 2 e 6 (gráfico em escala logarítmica).

Fonte: O Autor (2023).

De acordo com a Figura 6.9, até a malha de ordem $n^2 = 1.500$, o tempo que criação da estrutura matricial é menor que o tempo de execução das implementações. Porém, a partir dessa ordem, há um crescimento significativo no tempo de criação da matriz, comprometendo o tempo total do programa. Em todas as ordens testadas, o tempo de criação dos vetores foi significativamente menor que o tempo de execução das implementações.

No gráfico da Figura 6.10 é apresentada a comparação do tempo de execução das Implementações 3 e 7 com o tempo de criação das estruturas (matriz e vetores), ambas utilizaram o algoritmo de Thomas.

O padrão se repete com o gráfico da Figura 6.10. O tempo de criação da matriz começa menor que o tempo de execução das implementações para as primeiras ordens testadas, porém, de acordo com o aumento da ordem da matriz, o tempo de criação da mesma ultrapassa o tempo de execução das implementações. Novamente, em nenhuma ordem testada o tempo de criação dos vetores foi maior que o tempo de execução das implementações.

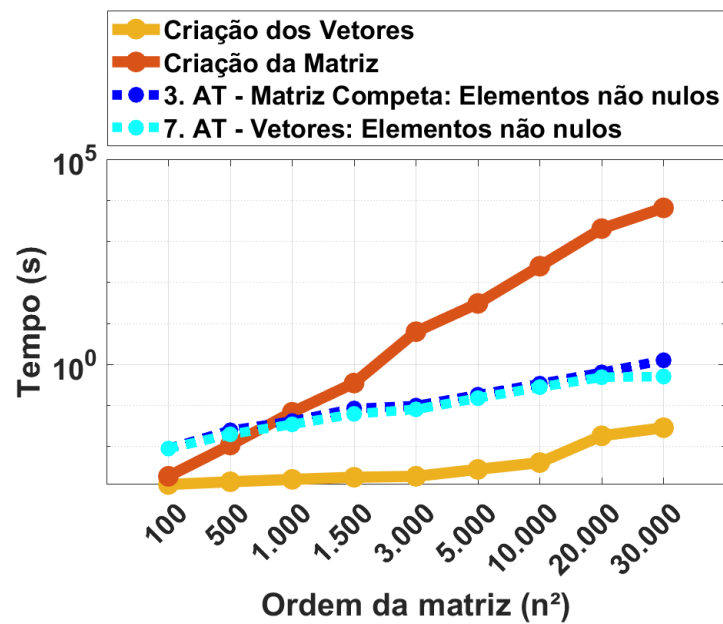


Figura 6.10: Teste Unidimensional - Comparação dos tempos de criação das estruturas (Matriz e Vetores) com a execução das implementações 3 e 7 (gráfico em escala logarítmica).

Fonte: O Autor (2023).

Analisando os resultados apresentados nas Figuras 6.9 e 6.10 é possível constatar que, embora as Implementações 2 e 3 tenham obtido tempos de execução próximos ou até melhores que as implementações que utilizaram vetores, sua aplicação para grandes sistemas se torna inviável, uma vez que dependem da criação da matriz, comprometendo o tempo total de execução do programa, além do aumento no consumo de memória.

6.2 Resultados do Estudo de Caso 02: Problema Bidimensional

Para fins de comparação das implementações, de forma análoga aos testes unidimensionais, foram realizados 7 testes com as mesmas condições iniciais e de contorno, alterando o número de volumes no espaço (n_x) e (n_y), que variou seguindo um critério pessoal do autor, resultando em sistemas matriciais com as seguintes ordens: 250×250 , 1.000×1.000 , 4.000×4.000 , 9.000×9.000 , 16.000×16.000 , 40.000×40.000 e $50.000.000 \times 50.000.000$.

Os parâmetros iniciais estão apresentados na Tabela 6.18, no que se refere ao experimento realizado por Telles (2009) [24], o qual contemplou um período total de coleta de dados igual a 352 s. Como há uma diferença entre o comprimento do rio e sua largura, foi mantido nos testes, uma proporção nos números de volumes no eixo x (n_x) em relação

aos volumes no eixo y (n_y).

Tabela 6.18: Parâmetros utilizados nos testes bidimensionais.

Valores Adotados					
u	=	0,359	m/s	x_{lanc}	= 50 m
E_L	=	0,33	m^2/s	y_{lanc}	= 0,5 m
E_T	=	0,008	m^2/s	x_{col}	= 100 m
x_f	=	182	m	y_{col}	= 0,5 m
y_f	=	42	m	M	= 20 kg
C_0	=	37,00	mg/L	n_t	= 176 $-$

Fonte: O Autor (2023).

Define-se, os dados da Tabela 6.18, como:

- u = Velocidade do rio;
- E_L = Coeficiente de dispersão longitudinal;
- E_T = Coeficiente de dispersão transversal;
- M = Massa do poluente;
- x_f = Comprimento do trecho analisado;
- y_f = largura do trecho analisado;
- C_0 = Concentração inicial do rio;
- x_{lanc} = Posição longitudinal de lançamento do poluente;
- y_{lanc} = Posição vertical de lançamento do poluente;
- x_{col} = Posição longitudinal de coleta do poluente;
- y_{col} = Posição transversal de coleta do poluente;
- n_t = número de passos no tempo.

• **Teste 1 - Para $n_x = 50$ e $n_y = 5$ (Ordem da Matriz: 250×250)**

No primeiro teste realizado, foi adotada uma discretização do domínio espacial considerando 50 volumes em relação ao eixo x (comprimento do rio no trecho analisado) e 5 volumes em relação ao eixo y (largura do rio no trecho analisado), ou seja, $n_x = 50$ e $n_y = 5$. Com isso, obteve-se uma matriz quadrada de ordem 250 (62.500 elementos). Todas as implementações convergiram, o resultado é apresentado na Figura 6.11.

Os tempos de execução das implementações estão representados na Tabela 6.19. Houve uma repetição do padrão quando comparado com o teste unidimensional. As Implementações 4 e 5, que usam o sistema de compressão CSC, apresentaram os piores

desempenhos, seguidas pela Implementação 1 (referência). Novamente, os melhores tempos de execução foram das Implementações 6 e 7, que utilizam vetores como estrutura de cálculo. As Implementações 2 e 3, que trabalham com a matriz completa, mas realizam os cálculos somente nos elementos não nulos, apresentaram tempos de execução próximos às Implementações 6 e 7.

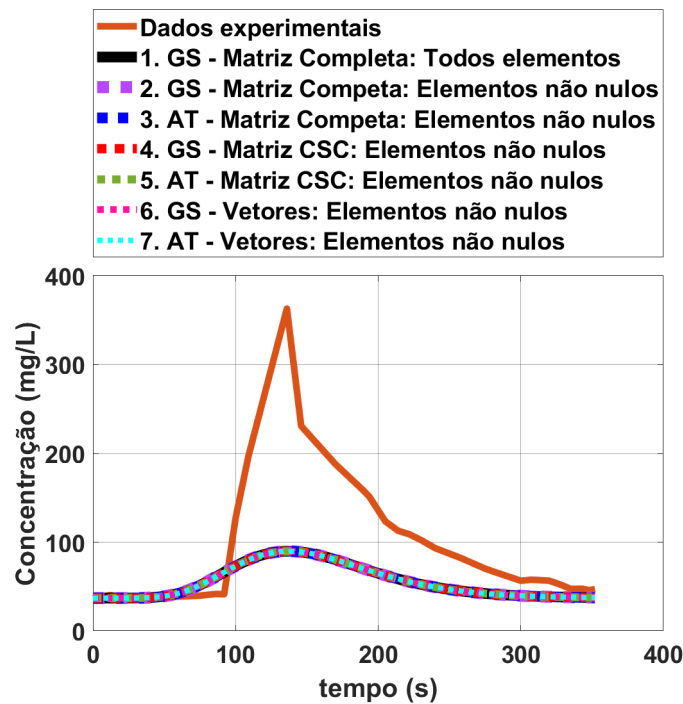


Figura 6.11: Teste Bidimensional - Perfil da concentração para matriz de ordem 250×250 .
Fonte: O Autor (2023).

Tabela 6.19: Teste Bidimensional - Tempos de execução para matriz de ordem 250×250 .

Implementação Computacional	Tempo de Execução (s)	% do Tempo Total
1. GS - Matriz Completa: Todos Elementos	0,338	12,693%
2. GS - Matriz Completa: Elementos não nulos	0,013	0,499%
3. AT - Matriz Completa: Elementos não nulos	0,014	0,529%
4. GS - Matriz CSC: Elementos não nulos	1,488	57,068%
5. AT - Matriz CSC: Elementos não nulos	0,731	28,020%
6. GS - Vetores: Elementos não nulos	0,013	0,499%
7. AT - Vetores: Elementos não nulos	0,011	0,422%

Fonte: O Autor (2023).

Em relação ao consumo de memória, demonstrados na Tabela 6.20, como nos testes unidimensionais, as implementações que trabalharam com a matriz completa consumiram mais de 90% da memória do programa. As implementações que utilizaram vetores apre-

sentaram o melhor resultado, seguidos pelas implementações que utilizaram o sistema de compressão CSC.

Tabela 6.20: Teste Bidimensional - Consumo de memória para matriz de ordem 250×250 .

Tipo de Estrutura $\gamma = 0,27$	Memória Utilizada (<i>bytes</i>)	% do Consumo Total
Matriz Densa	500.000	90,57%
Matriz Comprimida CSC	42.088	7,62 %
Vetores	10.000	1,81 %

Fonte: O Autor (2023).

• **Teste 2 - Para $n_x = 100$ e $n_y = 10$ (Ordem da Matriz: 1.000×1.000)**

No segundo teste realizado, foi adotada uma discretização do domínio espacial considerando 100 volumes em relação ao eixo x (comprimento do rio no trecho analisado) e 10 volumes em relação ao eixo y (largura do rio no trecho analisado), ou seja, $n_x = 100$ e $n_y = 10$. Com isso, obteve-se uma matriz quadrada de ordem 1.000 (1.000.000 elementos). Novamente, todas as implementações convergiram para a solução. O resultado é apresentado na Figura 6.12.

Os dados referentes ao tempo de execução das implementações, representados na Tabela 6.21, demonstram o mesmo padrão de resultado. As Implementações 6 e 7 sendo as que mais rápido convergiram para a solução, seguidas pelas Implementações 3 e 4. Os piores desempenhos ficaram, novamente, com as Implementações 4 e 5. É importante destacar que em relação ao Teste 1, assim como no caso unidimensional, houve uma melhora no tempo de execução das Implementações 4 e 5 quando comparado com a Implementação 1 (referência).

Como nos testes anteriores, o consumo de memória das implementações que utilizaram a matriz completa foi muito superior às demais implementações. Os melhores desempenhos ficaram com as implementações que utilizaram os vetores em substituição à matriz dos coeficientes. O segundo melhor desempenho foram das implementações que utilizaram a matriz comprimida através no método de compressão CSC. Os resultados são indicados na Tabela 6.22.

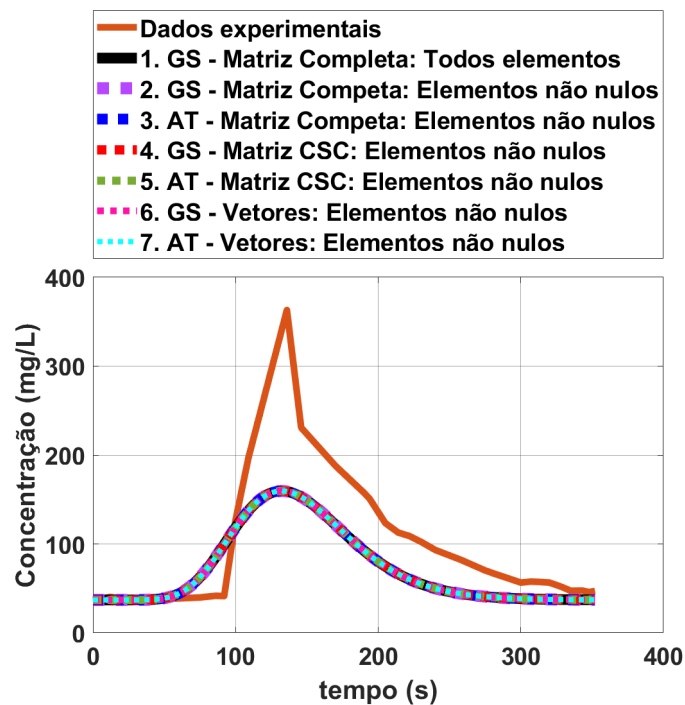


Figura 6.12: Teste Bidimensional - Perfil da concentração para matriz de ordem 1.000×1.000 .

Fonte: O Autor (2023).

Tabela 6.21: Teste Bidimensional - Tempos de execução para matriz de ordem 1.000×1.000 .

Implementação Computacional	Tempo de Execução (s)	% do Tempo Total
1. GS - Matriz Completa: Todos Elementos	3,235	22,813%
2. GS - Matriz Completa: Elementos não nulos	0,046	0,324%
3. AT - Matriz Completa: Elementos não nulos	0,025	0,179%
4. GS - Matriz CSC: Elementos não nulos	7,948	56,048%
5. AT - Matriz CSC: Elementos não nulos	2,873	20,261%
6. GS - Vetores: Elementos não nulos	0,034	0,240%
7. AT - Vetores: Elementos não nulos	0,019	0,135%

Fonte: O Autor (2023).

Tabela 6.22: Teste Bidimensional - Consumo de memória para ordem 1.000×1.000 .

Tipo de Estrutura $\gamma = 0,23$	Memória Utilizada (bytes)	% do Consumo Total
Matriz Densa	8.000.000	98,47%
Matriz Comprimida CSC	84.488	1,04 %
Vetores	40.000	0,49 %

Fonte: O Autor (2023).

• **Teste 3 - Para $n_x = 200$ e $n_y = 20$ (Ordem da Matriz: 4.000×4.000)**

No terceiro teste realizado, foi adotada uma discretização do domínio espacial considerando 200 volumes em relação ao eixo x (comprimento do rio no trecho analisado) e 20 volumes em relação ao eixo y (largura do rio no trecho analisado), ou seja, $n_x = 200$ e $n_y = 20$. Com isso, obteve-se uma matriz quadrada de ordem 4.000 (16.000.000 elementos). Todas as implementações convergiram para solução, conforme demonstrado na Figura 6.13.

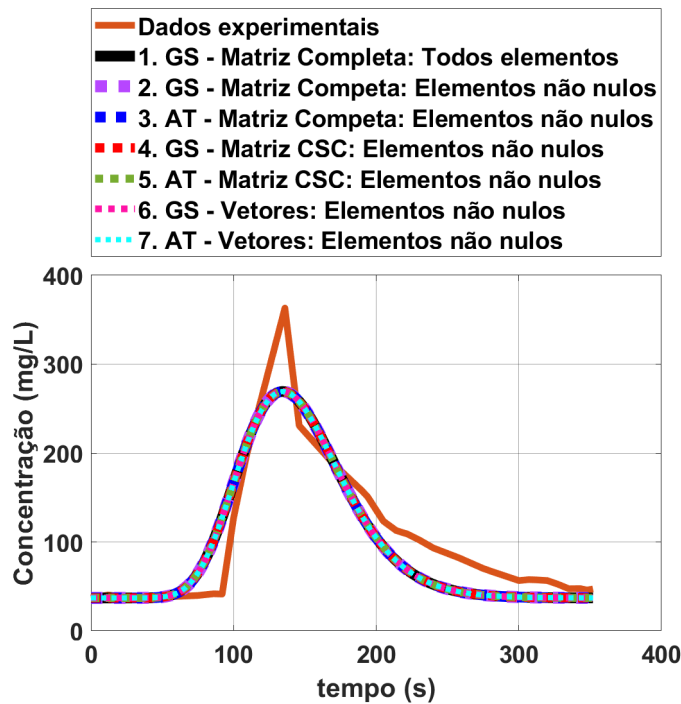


Figura 6.13: Teste Bidimensional - Perfil da concentração para matriz de ordem 4.000×4.000 .

Fonte: O Autor (2023).

De forma análoga ao teste unidimensional, com o aumento da ordem da matriz dos coeficientes, houve uma melhora no tempo de execução das Implementações 4 e 5 (método de compressão CSC) em relação à Implementação 1 (referência), evidenciado na Tabela 6.23. Os melhores desempenhos permaneceram com as Implementações 6 e 7 (utilizam vetores em substituição à matriz dos coeficientes). Novamente, as Implementações 2 e 3 (utilizam a matriz completa mas realiza os cálculos somente nos elementos não nulos) apresentaram um tempo de execução próximo às implementações com os melhores desempenhos.

O resultado, em relação ao consumo de memória, está demonstrado na Tabela 6.24. Para as implementações que utilizaram a matriz completa, como nos outros testes, houve

uma grande economia de memória quando comparado com as implementações que utilizaram matriz.

Tabela 6.23: Teste Bidimensional - Tempos de execução para matriz de ordem 4.000×4.000 .

Implementação Computacional	Tempo de Execução (s)	% do Tempo Total
1. GS - Matriz Completa: Todos Elementos	60,038	28,569%
2. GS - Matriz Completa: Elementos não nulos	0,259	0,123%
3. AT - Matriz Completa: Elementos não nulos	0,110	0,052%
4. GS - Matriz CSC: Elementos não nulos	137,673	65,512%
5. AT - Matriz CSC: Elementos não nulos	11,758	5,595%
6. GS - Vetores: Elementos não nulos	0,252	0,120%
7. AT - Vetores: Elementos não nulos	0,060	0,029%

Fonte: O Autor (2023).

Tabela 6.24: Teste Bidimensional - Consumo de memória para matriz de ordem 4.000×4.000 .

Tipo de Estrutura $\gamma = 0, 19$	Memória Utilizada (<i>bytes</i>)	% do Consumo Total
Matriz Densa	8.000.000	98,47%
Matriz Comprimida CSC	84.488	1,04 %
Vetores	40.000	0,49 %

Fonte: O Autor (2023).

• **Teste 4 - Para $n_x = 300$ e $n_y = 30$ (Ordem da Matriz: 9.000×9.000)**

No quarto teste realizado, foi adotada uma discretização do domínio espacial considerando 300 volumes em relação ao eixo x (comprimento do rio no trecho analisado) e 30 volumes em relação ao eixo y (largura do rio no trecho analisado), ou seja, $n_x = 300$ e $n_y = 30$. Com isso, obteve-se uma matriz quadrada de ordem 9.000 (81.000.000 elementos). Outra vez, todas as implementações convergiram e o resultado gráficos é mostrado na Figura 6.14.

Em relação aos tempos de execução, houve uma mudança importante. A Implementação 1 passou a ter o pior desempenho, superando (negativamente) as Implementações 4 e 5. Novamente, os melhores tempos ficaram com as Implementações 6 e 7. Os dados são demonstrados na Tabela 6.25.

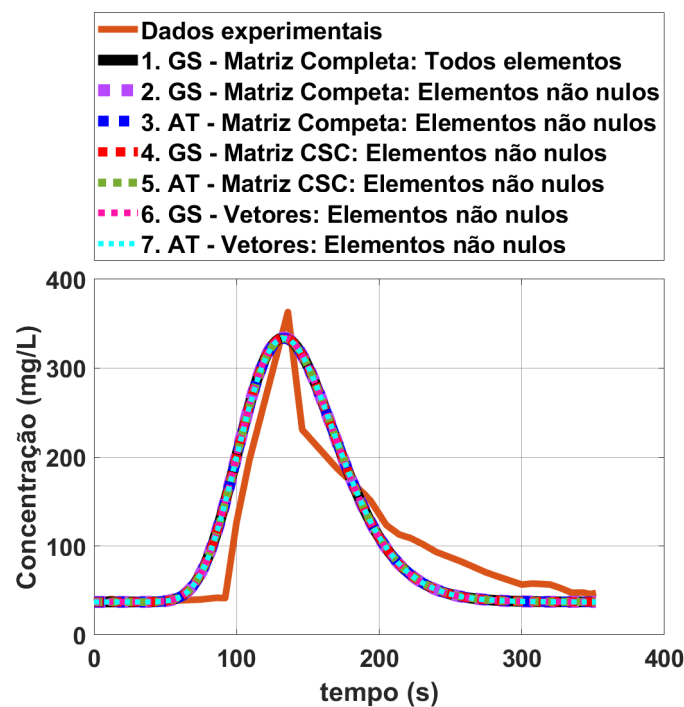


Figura 6.14: Teste Bidimensional - Perfil da concentração para matriz de ordem 9.000×9.000 .

Fonte: O Autor (2023).

Tabela 6.25: Teste Bidimensional - Tempos de execução para matriz de ordem 9.000×9.000 .

Implementação Computacional	Tempo de Execução (s)	% do Tempo Total
1. GS - Matriz Completa: Todos Elementos	6.353,000	96,040%
2. GS - Matriz Completa: Elementos não nulos	0,962	0,015%
3. AT - Matriz Completa: Elementos não nulos	0,241	0,004%
4. GS - Matriz CSC: Elementos não nulos	233,918	3,356%
5. AT - Matriz CSC: Elementos não nulos	26,280	0,397%
6. GS - Vetores: Elementos não nulos	0,419	0,006%
7. AT - Vetores: Elementos não nulos	0,149	0,002%

Fonte: O Autor (2023).

No que se refere ao consumo de memória, como nos testes anteriores, as implementações que utilizaram vetores apresentaram uma grande economia, quando comparados com as implementações que trabalharam com a matriz completa, os dados estão demonstrados na Tabela 6.26.

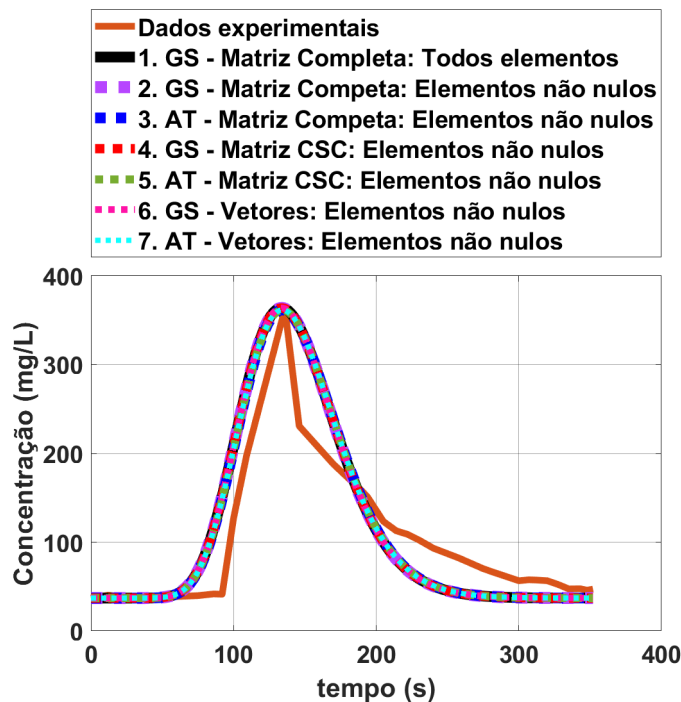
Tabela 6.26: Teste Bidimensional - Consumo de memória para matriz de ordem 9.000×9.000 .

Tipo de Estrutura $\gamma = 0,17$	Memória Utilizada (<i>bytes</i>)	% do Consumo Total
Matriz Densa	648.000.000	99,82%
Matriz Comprimida CSC	781.488	0,12 %
Vetores	360.000	0,06 %

Fonte: O Autor (2023).

• **Teste 5 - Para $n_x = 400$ e $n_y = 40$ (Ordem da Matriz: 16.000×16.000)**

No quinto teste realizado, foi adotada uma discretização do domínio espacial considerando 400 volumes em relação ao eixo x (comprimento do rio no trecho analisado) e 40 volumes em relação ao eixo y (largura do rio no trecho analisado), ou seja, $n_x = 400$ e $n_y = 40$. Com isso, obteve-se uma matriz quadrada de ordem 16.000 (256.000.000 elementos). Como em outros testes, houve convergência em todos as implementações. Os resultados são evidenciados no gráfico da Figura 6.15.

Figura 6.15: Teste Bidimensional - Perfil da concentração para matriz de ordem 16.000×16.000 .

Fonte: O Autor (2023).

Os dados demonstrados na Tabela 6.27, confirmam a alta eficiência das Implementações 6 e 7 (utilizam vetores), permanecendo com os melhores desempenhos em relação

ao tempo de execução do programa. A Implementação 1 (referência) manteve-se com o pior tempo de execução, seguida pelas Implementações 4 e 5 (matriz comprimida pelo métodos CSC).

Tabela 6.27: Teste Bidimensional - Tempos de execução para matriz de ordem 16.000×16.000 .

Implementação Computacional	Tempo de Execução (s)	% do Tempo Total
1. GS - Matriz Completa: Todos Elementos	8.555,000	92,382%
2. GS - Matriz Completa: Elementos não nulos	4,324	0,047%
3. AT - Matriz Completa: Elementos não nulos	0,412	0,004%
4. GS - Matriz CSC: Elementos não nulos	654,218	7,065%
5. AT - Matriz CSC: Elementos não nulos	45,283	0,489%
6. GS - Vetores: Elementos não nulos	1,052	0,011%
7. AT - Vetores: Elementos não nulos	0,152	0,002%

Fonte: O Autor (2023).

Em relação ao consumo de memória do programa, prevalece a grande economia das implementações que utilizaram vetores, conforme demonstrado na Tabela 6.28.

Tabela 6.28: Teste Bidimensional - Consumo de memória para matriz de ordem 16.000×16.000 .

Tipo de Estrutura $\gamma = 0, 16$	Memória Utilizada (bytes)	% do Consumo Total
Matriz Densa	648.000.000	99,82%
Matriz Comprimida CSC	781.488	0,12 %
Vetores	360.000	0,06 %

Fonte: O Autor (2023).

De forma análoga ao modelo unidimensional, a partir do Teste 5, o modelo começa a apresentar uma tendência de estabilidade no que se refere à variação da malha computacional, ou seja, o perfil das concentrações não sofre alterações significativas com o refinamento da mesma.

- **Teste 6) Para $n_x = 800$ e $n_y = 50$ - Ordem da Matriz: 40.000×40.000**

Neste sexto teste realizado foi adotada uma discretização do domínio espacial com 800 volumes em relação ao eixo x (comprimento do rio no trecho analisado) e 50 volumes

em relação ao eixo y (largura do rio no trecho analisado), ou seja, $n_x = 800$ e $n_y = 50$. Com isso, obteve-se uma matriz quadrada de ordem 40.000 (1.600.000.000 elementos).

Como esperado, as implementações que utilizaram a matriz completa apresentaram erros ao compilar o programa devido a insuficiência de memória. Com isso, apenas as Implementações 6 e 7 convergiram para a solução na compilação dos códigos. O resultado é apresentado na Figura 6.16.

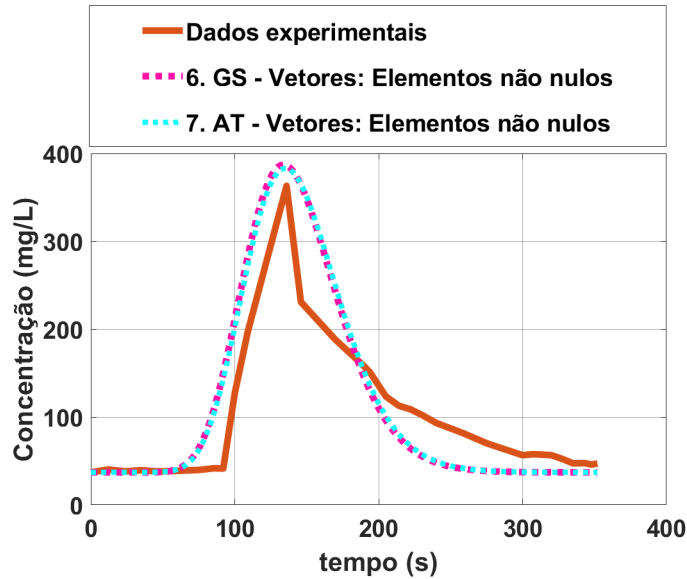


Figura 6.16: Teste Bidimensional - Perfil da concentração para matriz de ordem 40.000×40.000 .

Fonte: O Autor (2023).

Na Tabela 6.29 são apresentados os dados referentes ao tempo de execução das implementações. Por trabalhar com método parcialmente direto ¹ e não depender de aproximações para se chegar a solução, a Implementação 7 apresentou o melhor desempenho.

Os dados referentes ao consumo de memória são apresentados na Tabela 6.30. As implementações que utilizaram vetores demandaram do computador 1.600.000 *bytes* ou 0,0016 *GB* de memória para sua execução. Caso fosse utilizada alguma implementação que dependesse da criação matriz, o consumo de memória seria de 12.800.000.000 *bytes* ou 12,8 *GB*.

¹O algoritmo de Thomas é direto para o caso unidimensional, o qual recai em um sistema com matriz tridiagonal. Já no caso bidimensional, a adaptação do modelo matemático para a matriz pentadiagonal torna parte do método iterativo.

Tabela 6.29: Teste Bidimensional - Tempos de execução para matriz de ordem 40.000×40.000 .

Implementação Computacional	Tempo de Execução (s)	% do Tempo Total
1. GS - Matriz Completa: Todos Elementos	-	-
2. GS - Matriz Completa: Elementos não nulos	-	-
3. AT - Matriz Completa: Elementos não nulos	-	-
4. GS - Matriz CSC: Elementos não nulos	-	-
5. AT - Matriz CSC: Elementos não nulos	-	-
6. GS - Vetores: Elementos não nulos	51,604	99,13%
7. AT - Vetores: Elementos não nulos	0,454	0,87%

Fonte: O Autor (2023).

Tabela 6.30: Teste Bidimensional - Consumo de memória para matriz de ordem 40.000×40.000 .

Tipo de Estrutura	Memória Utilizada (<i>bytes</i>)	% do Consumo Total
Matriz Densa	-	-
Matriz Comprimida CSC	-	-
Vetores	1.600.000	100%

Fonte: O Autor (2023).

De forma análoga ao caso unidimensional, diante dos resultados apresentados, foi possível observar um refinamento no perfil da concentração de acordo com o crescimento da ordem da matriz, aproximando o valor numérico ao valor dos dados experimentais. Para o teste em que se utilizou $n_x = 800$ e $n_y = 50$, ocorreu um perfil sobrestimado da concentração numérica em relação à experimental. Fato esse que instiga uma maior investigação dos valores dos coeficientes de dispersão adotados por Telles (2009). Ressalta-se, ainda, que no trabalho do referido autor, o mesmo reforça a inviabilidade de utilização de uma malha mais refinada devido à limitações de *hardware* disponível durante a pesquisa.

- **Teste 7) Para $n_x = 1.000.000$ e $n_y = 50$ - Ordem da Matriz: $50.000.000 \times 50.000.000$**

Neste sétimo teste realizado foi adotada uma discretização do domínio espacial com 1.000.000 volumes em relação ao eixo x (comprimento do rio no trecho analisado) e 50 volumes em relação ao eixo y (largura do rio no trecho analisado), ou seja, $n_x = 1.000.000$ e $n_y = 50$. Com isso, obteve-se uma matriz quadrada de ordem 50.000.000 ($2,5 \times 10^{15}$

elementos). Para a solução do sistema foi utilizada a Implementação 7, pois apresentou o melhor desempenho nos testes anteriores. O gráfico com perfil da concentração é apresentado na Figura 6.17.

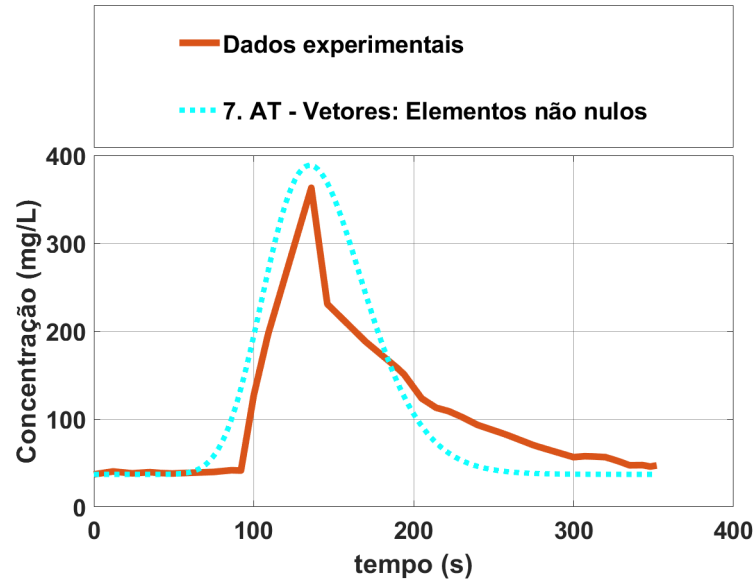


Figura 6.17: Teste Unidimensional - Perfil da concentração para matriz de ordem $50.000.000 \times 50.000.000$.

Fonte: O Autor (2023).

No que diz respeito ao tempo de execução da Implementação 7, o valor é apresentado na Tabela 6.31.

Tabela 6.31: Teste Bidimensional - Tempos de execução para matriz de ordem $50.000.000 \times 50.000.000$.

Implementação Computacional	Tempo de Execução (s)	% do Tempo Total
1. GS - Matriz Completa: Todos Elementos	-	-
2. GS - Matriz Completa: Elementos não nulos	-	-
3. AT - Matriz Completa: Elementos não nulos	-	-
4. GS - Matriz CSC: Elementos não nulos	-	-
5. AT - Matriz CSC: Elementos não nulos	-	-
6. GS - Vetores: Elementos não nulos	-	-
7. AT - Vetores: Elementos não nulos	0,454	100%

Fonte: O Autor (2023).

Novamente, em relação ao consumo de memória demonstrado na Tabela 6.32, o resultado foi bastante significativo. Os vetores demandaram do computador 2×10^9 bytes ou

2 GB de memória. Caso fosse utilizada alguma implementação que dependesse da matriz, o consumo de memória seria de 2×10^{16} bytes ou 20.000.000 GB.

Tabela 6.32: Teste Bidimensional - Consumo de memória para matriz de ordem $50.000.000 \times 50.000.000$.

Tipo de Estrutura	Memória Utilizada (bytes)	% do Consumo Total
Matriz Completa	-	-
Matriz Comprimida CSC	-	-
Vetores	2×10^9	100 %

Fonte: O Autor (2023).

Além disso, verificou-se que a Implementação 7 foi a que obteve um menor tempo de execução nos referidos testes. Sendo assim, na Figura 6.18 é apresentada a comparação dos gráficos da concentração de todas as malhas utilizadas, executadas com a Implementação 7.

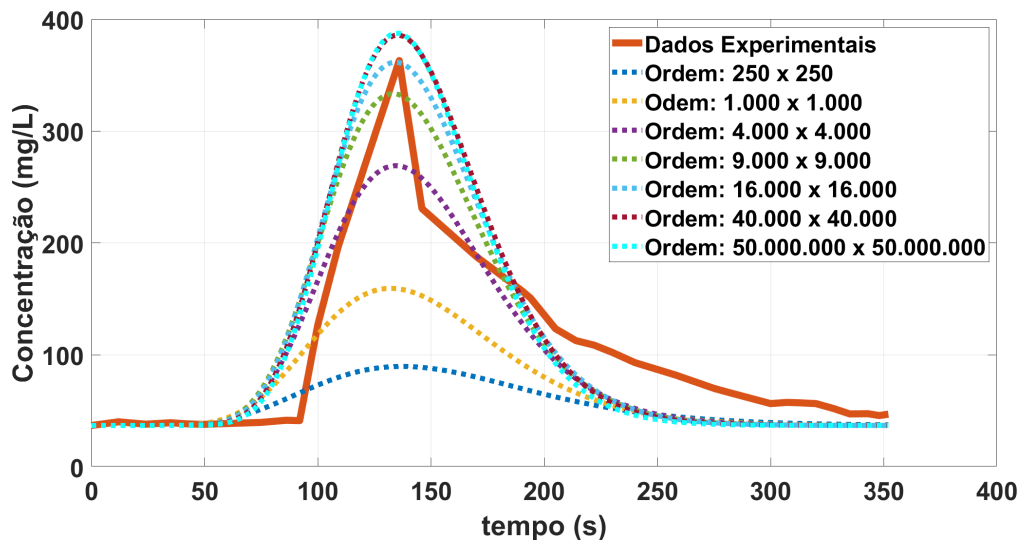


Figura 6.18: Comparação dos perfis de concentração para o caso bidimensional em relação ao aumento da ordem da matriz utilizando a Implementação 7.

Fonte: O Autor (2023).

- **Análise referente ao tempo de criação das estruturas**

Por fim, buscou-se uma relação entre os tempos de execução do método numérico com o tempo de criação da estrutura de cálculo (matrizes ou vetores). Assim como no caso unidimensional, foi feita a comparação entre as Implementações 2 e 6, ambas utilizando o método de Gauss-Seidel, cujo o resultado é mostrado na Figura 6.19.

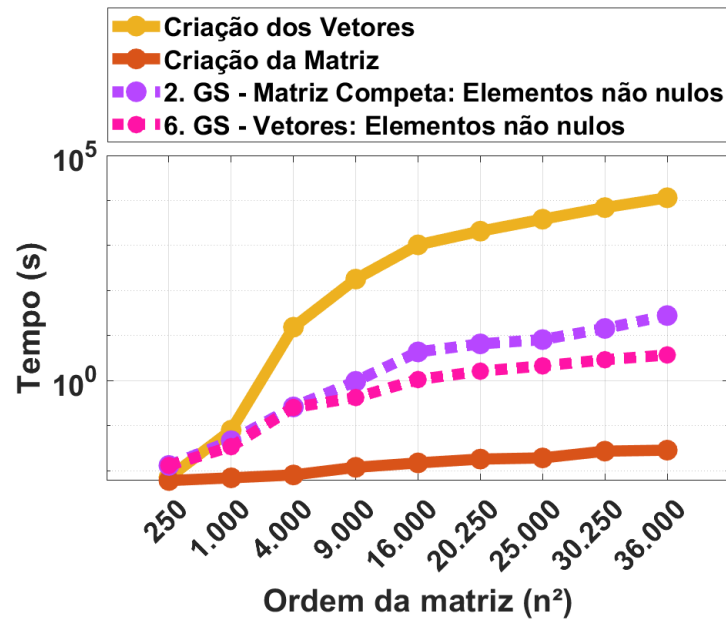


Figura 6.19: Teste Bidimensional - Comparação dos tempos de criação das estruturas (Matriz e Vetores) com a execução das implementações 2 e 6 (gráfico em escala logarítmica).

Fonte: O Autor (2023).

Os resultados apresentados no gráfico da Figura 6.19 seguem o mesmo padrão do caso unidimensional, ou seja, a execução das implementações começa demandando maior tempo. Porém, já na segunda ordem testada ela é ultrapassada pelo tempo de criação da matriz. Outra vez, assim como no caso unidimensional, o tempo de criação dos vetores pouco influenciou no tempo total do programa, mantendo-se bem abaixo do tempo de execução das implementações.

Também foram testadas as Implementações 3 e 7, neste caso, ambas utilizando o método do algoritmo de Thomas. Os resultados são mostrados na Figura 6.20.

Fica evidente, analisando os gráficos das Figuras 6.19 e 6.20, que o tempo de criação da estrutura matricial inviabiliza a utilização das Implementações 2 e 3 para grandes sistemas, uma vez que dependem da criação da matriz.

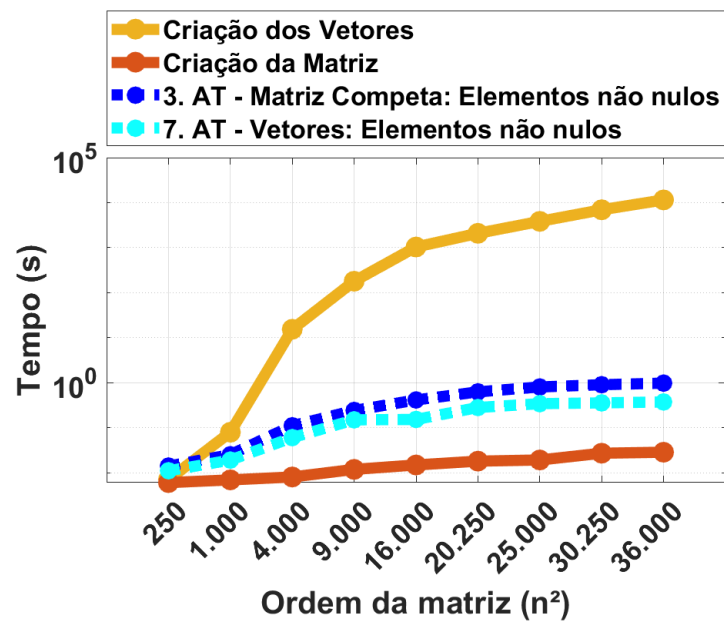


Figura 6.20: Teste Bidimensional - Comparação dos tempos de criação das estruturas (Matriz e Vetores) com a execução das implementações 3 e 7 (gráfico em escala logarítmica).

Fonte: O Autor (2023).

Capítulo 7

CONCLUSÕES E TRABALHOS FUTUROS

Nesse capítulo são apresentadas as conclusões oriundas da corrente pesquisa, bem como desdobramentos para trabalhos futuros que a mesma possibilita.

7.1 Conclusões

Diante de todos os dados apresentados, foi possível concluir que:

- **Em relação à convergência:**

1. Todas as implementações descritas na Tabela 6.5 apresentaram um bom desempenho na resolução do problema proposto. Não houve divergência de nenhuma implementação nos testes realizados.
2. As Implementações 3, 5 e 7 (utilizaram o algoritmo de Thomas) foram as que apresentaram uma melhor estabilidade nos resultados, independentemente do aumento da ordem do sistema.
3. Para as Implementações 1, 2, 4 e 6 (utilizaram o método de Gauss-Seidel), com o aumento da ordem da matriz foram necessários refinamentos na malha para se chegar à solução desejada.

- **Em relação ao tempo de execução:**

1. Nas primeiras ordens testadas, tanto no caso unidimensional quanto no caso bidimensional, a Implementação 1 (referência) não apresentou o pior desempenho. Porém, de acordo com o aumento da ordem do sistema, houve uma piora gradativa no tempo de execução da implementação. Pelo fato de trabalhar com a matriz completa e realizar cálculos em todos os seus elementos a cada iteração, sua utilização se mostrou inviável para sistemas extensos, apresentando o pior desempenho em relação ao tempo de execução.
2. As Implementações 2 e 3 apresentaram os melhores tempos de execução entre aquelas que utilizaram a matriz como estrutura de cálculo. Embora trabalhe com a estrutura matricial completa, como os cálculos são realizados apenas nos elementos não nulos, houve uma redução significativa no tempo de execução dos métodos quando comparados com a implementação de referência. Porém, no momento em que se buscou uma alta exigência de refinamento da malha, criando sistemas extensos, as Implementações 2 e 3 representaram uma desvantagem em relação às implementações que utilizaram vetores, pelo fato de depender da criação da matriz.
3. Nas primeiras ordens testadas, tanto para o caso unidimensional quanto para o caso bidimensional, as Implementações 4 e 5 obtiveram os piores desempenhos. Por usar o sistema de compressão *Compressed Sparse Column* (CSC), o qual armazena os elementos não nulos da matriz e os índices com sua posição na mesma. Para cada iteração realizada, o programa precisa percorrer toda estrutura até chegar aos índices dos elementos armazenados e realizar o cálculo, comprometendo o tempo de execução do método.
4. Os melhores desempenhos ficaram com as Implementações 6 e 7, ambas utilizando os vetores em substituição à matriz dos coeficientes. A Implementação 6 apresentou o segundo melhor resultado. Quando testada em sistemas de grandes dimensões, por utilizar vetores em substituição à matriz dos coeficientes, o tempo total do programa chegou a ser 98% menor do que as implementações que dependiam da criação da matriz (Implementação 2 e 3). A Implementação 7 manteve-se estável em todos os testes, mesmo com o aumento da ordem da matriz. A economia no tempo de execução se deu por dois fatores: a utilização de vetores, em substituição à matriz dos coeficientes; a solução não foi obtida por aproximações, mas sim, por substituição regressiva, através eliminação direta.

- **Em relação ao consumo de memória:**

Nos testes em que todas as implementações foram executadas no mesmo programa, a diferença em relação ao consumo de memória foi bastante significativa.

1. As implementações que utilizaram vetores, em substituição à matriz dos coeficientes, apresentaram o melhor desempenho. Por armazenar apenas os elementos não nulos da estrutura matricial foi possível uma economia de mais de 97% de memória do programa em todos os testes. Para grandes sistemas, com alta exigência de refinamento da malha, a economia fica mais evidente, chegando a valores maiores que 99,9%.
2. As implementações que utilizaram o sistema de compressão *Compressed Sparse Column* (CSC), através da função *sparse* do MATLAB, se mostraram um boa opção para armazenamento da estrutura matricial durante a execução do método. Sua economia na memória do programa foi superior a 96% em todos os testes realizados. A desvantagem, em relação às implementações que utilizaram vetores, é que, além do armazenamento dos vetores, a função armazena, também, os índices de seu posicionamento na matriz.
3. As implementações que utilizaram a matriz completa consumiram mais de 90% da memória disponível do programa em todos os testes. Quando houve exigência de refinamento da malha, o consumo ultrapassou 99,98%. Outra desvantagem na utilização deste tipo de estrutura, foi o limite de armazenamento da matriz em 36.000×36.000 volumes, ou seja, para uma malha acima deste valor, devido a insuficiência de memória, o computador não conseguiu armazenar a estrutura matricial.

Em resumo, as melhores opções para solução do problema proposto foram: Implementação 7, seguida pela Implementação 6. O fato de substituir a matriz dos coeficientes por vetores dos elementos não nulos, de forma a não armazenar e também não computar no cálculo valores desnecessários, foi determinante para a diminuição no tempo de execução dos métodos e economia no consumo de memória do programa.

7.2 Trabalhos Futuros

Com objetivo de dar continuidade aos estudos em relação ao armazenamento da estrutura matricial para problemas que recaem em sistemas esparsos, algumas sugestões são

importantes:

1. Estender as implementações computacionais do problema proposto, buscando aproveitar a esparsidade da matriz dos coeficientes, para o caso tridimensional.
2. A discretização através do Método dos Volumes Finitos resulta em uma diferença de concentrações entre as faces dos volumes de controle. Para determinar uma aproximação do valor dessas concentrações foi utilizado um esquema de interpolação conhecido como *upwind*. Porém, segundo a literatura, existem outras formas de se determinar esses valores, com maior ou menor grau de precisão. Neste sentido, é interessante testar esquemas que alterem o grau de precisão do modelo com intuito de verificar o comportamento da resolução, principalmente em relação ao algoritmo de Thomas, uma vez que irá alterar a exigência de arredondamento da máquina.
3. Nos testes bidimensionais, as implementações que utilizaram o método de Gauss-Seidel convergiram mais rápido para a solução quando comparado com os testes unidimensionais. Esta diferença de tempo, entre os dois estudos de caso, ocorreu porque nos testes bidimensionais, o método Gauss-Seidel necessitou de menos iterações para se chegar à solução. Diante disso, seria interessante uma análise do modelo matemático e da discretização, através do Método dos Volumes Finitos, para entender o motivo desta diferença, uma vez que as condições de contorno e as aproximações por interpolação foram as mesmas para os dois casos.
4. Para problemas cuja discretização resulte em uma matriz esparsa aleatória, ou seja, os elementos não nulos não seguem um padrão específico, impossibilitando sua substituição por vetores, a estratégia de compressão *Compressed Sparse Column* (CSC) pode se mostrar uma boa opção. No entanto, para execução do método, percorre-se toda a estrutura armazenada a cada passo de cálculo, o que compromete o tempo do programa. Dessa forma, seria importante uma investigação em relação à disposição de armazenamento dos valores e de mudanças na estrutura de cálculo do método, de forma a evitar esse desperdício de tempo.

Referências

- [1] ARANTES, R. D. *Codificações estruturais para a resolução escalar de sistemas lineares esparsos simétricos definidos positivos*. Tese de doutorado, Universidade Federal do Rio de Janeiro, Rio de Janeiro-RJ, 1996.
- [2] BASSANEZI, R. C. *Ensino-aprendizagem com modelagem matemática*, 1 ed. Editora Contexto, 2002.
- [3] BRAGA, B., HESPANHOL, I., CONEJO, J. L., BARROS, M. T. *Introdução à Engenharia Ambiental: O desafio do desenvolvimento sustentável*, 2 ed. Pearson, 2010.
- [4] BURDEN, R., FAIRES, J. D. *Análise numérica*, 3 ed. Cengage Learning, 2016.
- [5] CARAPETO, C. *Poluição das águas: causa e efeito*, 1 ed. Universidade Aberta, 1999.
- [6] COELHO, M. A. O. Métodos iterativos para resolver sistemas de equações algébricas lineares em estruturas esparsas. Dissertação de mestrado, Universidade Federal Fluminense, Volta Redonda-RJ, 2014.
- [7] DYMINSKI, A. S. Contaminação de Solos e Águas Subterrâneas. *Curitiba: UFPR* (2006), 2–3.
- [8] FERREIRA, J. *Matrizes e Sistemas de Equações Lineares Espaços Vetoriais*, 1 ed. EDUFRRN, 2011.
- [9] FERZIGER, J. H., PERIC, M. *Computational methods for fluid dynamics*, 3 ed. Springer, 2002.
- [10] FORTUNA, A. O. *Técnicas Computacionais Para Dinâmica dos Fluidos*. Edusp, 2000.
- [11] GILAT, A., SUBRAMANIAM, V. *Métodos Numéricos Para Engenheiros E Cientistas: uma introdução com aplicações usando o MATLAB*, 1 ed. Bookman Editora, 2008.
- [12] HARTFIEL, D. J. *Matrix Theory and Applications with MATLAB*, 1 ed. CRC Press, 2017.
- [13] INSTITUTION, W. H. O. New study finds oceans arrived early to earth, 2014.
- [14] JÚNIOR, A. F. S. Método dos volumes finitos para equação de convecção e difusão em uma dimensão espacial. Dissertação de mestrado, Universidade Federal Fluminense, Volta Redonda-RJ, 2012.
- [15] LAMIN, M. R. N. Resolução de problemas modelados com sistemas de equações lineares. Monografia, Universidade Federal de Santa Catarina, Florianópolis-SC, 2000.

- [16] MALISKA, C. *Transferência de Calor e Mecânica dos Fluidos Computacional*, 2 ed. LTC, 2004.
- [17] MILLER, G. T. *Living in the environment: An introduction to environmental*, 1 ed. Wadsworth Pub. Co, 1985.
- [18] OLIVEIRA, T. R. M. Avaliação de formatos de armazenamento com compressão para resolução de sistemas de equações lineares esparsos. Monografia, Universidade Tecnológico Federal do Paraná, Apucarana-PR, 2019.
- [19] POTTER, M., WIGGERT, D., RAMADAN, B. *Mecânica dos Fluidos*, 2 ed. Cengage Learning, 2014.
- [20] ROSER, M., RODÉS-GUIRAO, L. Future population growth, 2014.
- [21] SANTOSO, V. População mundial deve ultrapassar marca de 8 bilhões ainda este ano, 2022.
- [22] SOUZA, E. P. Avaliação de mecanismos dispersivos em rios através de problemas inversos. Dissertação de mestrado, Universidade do Estado do Rio de Janeiro, Nova Friburgo-RJ, 2009.
- [23] SOUZA, M. J. F. Cálculo numérico - sistemas lineares, 2018.
- [24] TELLES, W. R. Simulação do transporte horizontal bidimensional de substância conservativa. Dissertação de mestrado, Universidade do Estado do Rio de Janeiro, Nova Friburgo-RJ, 2009.
- [25] TOSCANI, L. V., VELOSO, P. A. S. *Complexidade de Algoritmos*, 3 ed. Bookman, 2012.
- [26] VERSTEEG, H. K., MALALASEKERA, W. *An introduction to computational fluid dynamics: the finite volume method*. Pearson education, 2007.
- [27] VIEIRA, C., MORAIS, V. *MATLAB. Curso Completo*, 1 ed. FCA, 2013.
- [28] WALBERT, A. Onu aponta carência e má distribuição de água para uso, 2013.
- [29] WORTH, K. Onu news - perspectiva global reportagens humanas. disponível em: <https://news.un.org/pt/story/2022/11/1805267>. acesso em: 23 de jan. 2023.
- [30] ZILL, D. G., CULLEN, M. R. *Equações Diferenciais: Volume 1*, 3 ed. Pearson Universidades, 2000.