UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

JOÃO PAULO MARTINS DOS SANTOS

**MÉTODO MULTIGRID ALGÉBRICO:
REUTILIZAÇÃO DAS ESTRUTURAS MULTIGRID NO
TRANSPORTE DE CONTAMINANTES**

São Carlos, São Paulo

2015

JOÃO PAULO MARTINS DOS SANTOS

**MÉTODO MULTIGRID ALGÉBRICO:
REUTILIZAÇÃO DAS ESTRUTURAS MULTIGRID NO
TRANSPORTE DE CONTAMINANTES**

Tese apresentada à Escola de Engenharia de São Carlos, da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Doutor em Ciências: Engenharia Hidráulica e Saneamento.

Orientador: Prof. Dr. Edson Cezar Wendland

São Carlos, São Paulo
2015

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

S237m Santos, João Paulo Martins dos
Método multigrid algébrico : reutilização das
estruturas multigrid no transporte de contaminantes /
João Paulo Martins dos Santos; orientador Edson Cezar
Wendland. São Carlos, 2015.

Tese (Doutorado) - Programa de Pós-Graduação e Área
de Concentração em Hidráulica e Saneamento -- Escola de
Engenharia de São Carlos da Universidade de São Paulo,
2015.

1. Método dos elementos finitos. 2. Método
multigrid algébrico. 3. Método iterativo
não-estacionário. 4. Computação científica. 5. Python.
6. Estimador residual. I. Título.

FOLHA DE JULGAMENTO

Candidato: Licenciado **JOÃO PAULO MARTINS DOS SANTOS**.

Título da tese: "Método multigrid algébrico: reutilização das estruturas multigrid no transporte de contaminantes".

Data da defesa: 31/08/2015

Comissão Julgadora:

Resultado:

Prof. Titular **Edson Cezar Wendland (Orientador)**
(Escola de Engenharia de São Carlos/EESC)

Aprovado

Prof. Dr. **Nivaldo Aparecido Corrêa**
(Escola de Engenharia de São Carlos/EESC)

Aprovado

Prof. Titular **José Alberto Cuminato**
(Instituto de Ciências Matemáticas e de Computação/ICMC)

Aprovado

Profa. Dra. **Cristina de Hollanda Cavalcanti Tsuha**
(Escola de Engenharia de São Carlos/EESC)

Aprovado

Prof. Dr. **José Eduardo Alamy Filho**
(Universidade Federal de Uberlândia/UFU)

Aprovado

Coordenadora do Programa de Pós-Graduação em Engenharia Hidráulica e Saneamento:

Profa. Associada **Maria Bernadete A. Varesche Silva**

Presidente da Comissão de Pós-Graduação:
Prof. Associado **Paulo César Lima Segantine**

*Este trabalho é dedicado à todos aqueles que,
direta ou indiretamente, contribuíram para a
sua realização.*

Agradecimentos

Ao Prof. Edson, pela orientação.

À Força Aérea pelo incentivo à titulação de seus docentes.

Ao Prof. Alessandro pelas longas discussões.

Aos professores da Academia da Força Aérea pelo apoio.

Ao pessoal do Departamento de Hidráulica e Saneamento.

Aos amigos do Laboratório de Hidráulica Computacional (LHC).

Ao Senhor José Sidney e Senhora Maria Eugênia.

Ao Senhor Paulo Roberto e Senhora Creuza Alves.

À Rita.

A todos os meus sinceros agradecimentos.

Resumo

SANTOS, J.P.M (2015). Método multigrid algébrico: a reutilização das estruturas multigrid no transporte de contaminantes. São Carlos, 2015. 197p. Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2015.

A necessidade de obter solução de grandes sistemas lineares resultantes de processos de discretização de equações diferenciais parciais provenientes da modelagem de diferentes fenômenos físicos conduz à busca de técnicas numéricas escaláveis. Métodos multigrid são classificados como algoritmos escaláveis. Um estimador de erros deve estar associado à solução numérica do problema discreto de modo a propiciar a adequada avaliação da solução obtida pelo processo de aproximação. Nesse contexto, a presente tese caracteriza-se pela proposta de reutilização das estruturas matriciais hierárquicas de operadores de transferência e restrição dos métodos multigrid algébricos para acelerar o tempo de solução dos sistemas lineares associados à equação do transporte de contaminantes em meio poroso saturado. Adicionalmente, caracteriza-se pela implementação das estimativas residuais para os problemas que envolvem dados constantes ou não constantes, os regimes de pequena ou grande advecção e pela proposta de utilização das estimativas residuais associadas ao termo de fonte e à condição inicial para construir procedimentos adaptativos para os dados do problema. O desenvolvimento dos códigos do método de elementos finitos, do estimador residual e dos procedimentos adaptativos foram baseados no projeto FEniCS, utilizando a linguagem de programação *PYTHON^R* e desenvolvidos na plataforma Eclipse. A implementação dos métodos multigrid algébricos com reutilização considera a biblioteca PyAMG. Baseado na reutilização das estruturas hierárquicas, os métodos multigrid com reutilização com parâmetro fixo e automática são propostos, e esses conceitos são estendidos para os métodos iterativos não-estacionários tais como *GMRES* e *BICGSTAB*. Os resultados numéricos mostraram que o estimador residual captura o comportamento do erro real da solução numérica, e fornece algoritmos adaptativos para os dados cuja malha retornada produz uma solução numérica similar à uma malha uniforme com mais elementos. Adicionalmente, os métodos com reutilização são mais rápidos que os métodos que não empregam o processo de reutilização de estruturas. Além disso, a eficiência dos métodos com reutilização também pode ser observada na solução do problema auxiliar, o qual é necessário para obtenção das estimativas residuais para o regime de grande advecção. Esses resultados englobam tanto os métodos multigrid algébricos do tipo *SA* quanto os métodos pré-condicionados por métodos multigrid algébrico *SA*, e envolvem o transporte de contaminantes em regime de pequena e grande advecção, malhas estruturadas e não estruturadas, problemas bidimensionais, problemas tridimensionais e domínios com diferentes escalas.

Palavras-chave: método de elementos finitos. método multigrid algébrico. método itera-

tivo não-estacionário. estimador residual. computação científica. Python^R.

Abstract

SANTOS, J.P.M (2015). Algebraic multigrid method: the multigrid structures reuse in contaminant transport. São Carlos, 2015. 183p. Thesis (Doctorate) – School of Engineering of São Carlos, University of São Paulo, São Carlos.

The need for solving large linear systems arising from the discretization of partial differential equations modelling physical phenomena motivates the search for scalable numerical techniques. Multigrid algorithms are instances of such techniques. In order to provide a suitable assessment of the solution obtained by such algorithms, an error estimator must be associated to the numerical solution of the discretized problem. In this context, this thesis proposes the reutilization of the hierarchical matrix structures of transfer operators and the restriction to algebraic multigrid methods to speed up the process of solving the linear systems associated with the contaminant transport equation in saturated porous media. In addition, it features the implementation of residual estimates for problems involving constant or non-constant data, the regimes of small- or large-scale advection and the proposal of employing the residual estimates associated to the source term and to the initial condition to build adaptive procedures for the problem data. The development of the computer codes of the finite element method, residual estimator and adaptive procedures were based on the FEniCS project, using the programming language *PYTHON*^R and developed on the Eclipse platform. The implementation of the algebraic methods with reutilization relied upon the library PyAMG. Grounding on the idea of reutilizing the hierarchical structures, fixed and automatic parameters multigrid methods were proposed and extended to non-stationary iterative methods such as *GMRES* and *BICGSTAB*. The numerical results demonstrate that the residual estimator captures the behavior of the real error of the numerical solution, and provide adaptive algorithms for the data whose output mesh yields a numerical solution alike to that obtained from a uniform mesh with more elements. Moreover, the methods with reutilization are faster than those that do not reuse the structures. Besides, the efficiency of such methods can also be observed in the solution of an auxiliary problem, which is necessary for deriving the residual estimates in the regime of large-scale advection. These results encompass both the type *SA* algebraic multigrid method and those pre-conditioned by them. Moreover, they involve the transport of contaminants in regime of small- and large-scale advection, structured and non-structured meshes, bi- and tridimensional problems and domains with different scales.

Keywords:: finite element method. algebraic multigrid method. non-stationary iterative method. residual estimator. scientific computing. Python^R.

Lista de ilustrações

Figura 1 – Malha grossa Ω^H e malha fina Ω^h com $H = 2h$ e esquema multigrid V -ciclo de dois níveis	34
Figura 2 – Esquemas multigrid V -ciclo e F -ciclo de n níveis	35
Figura 3 – Esquemas multigrid V -ciclo de 4 níveis e sistemas lineares associados	35
Figura 4 – Divisão das variáveis da malha fina em agregados: C -variáveis e F -variáveis.	41
Figura 5 – Representação esquemática das condições iniciais e de contorno i.,ii.,iii., iv. e v.	52
Figura 6 – Dois elementos vizinhos que compartilham uma aresta e a representação do salto da aresta de K_+ para K_- (PRAETORIUS; WEINMULLER; WISSGOTT, 2008)	56
Figura 7 – Duas malhas de pontos igualmente espaçados nas direções coordenadas	72
Figura 8 – As soluções numérica e analítica em malha elementos finitos triangulares e funções lagrangeanas lineares	84
Figura 9 – O erro entre as soluções numérica e analítica $E = Erro = \left \frac{C_{Analítica}}{C_0} - \frac{C_{Numérica}}{C_0} \right $ representado em malha de elementos finitos triangulares e funções lagrangeanas lineares	85
Figura 10 – Perfis de concentração comparando as soluções numérica e analítica	86
Figura 11 – O indicador espacial η_K em cada elemento	87
Figura 12 – O indicador do elemento η_{1K} em cada elemento	87
Figura 13 – Os indicadores do salto η_{2E} e temporal η_τ em cada elemento	88
Figura 14 – A solução analítica e isolinhas em malha de elementos finitos e funções lineares	90
Figura 15 – A solução numérica e isolinhas em malha de elementos finitos e funções lineares	90
Figura 16 – O erro entre as soluções numérica e analítica $E = Erro = \left \frac{C_{Analítica}}{C_0} - \frac{C_{Numérica}}{C_0} \right $ representado em malha de elementos finitos triangulares e funções lagrangeanas lineares	91
Figura 17 – O indicador espacial η_K em cada elemento	92
Figura 18 – O indicador do elemento η_{1K} em cada elemento	92
Figura 19 – O indicador do salto η_{2E} em cada elemento	93
Figura 20 – Campo de velocidades não uniforme $\mathbf{v} = (v_x, v_y) = (0.5(x+1.0), -0.5(y+1))$, condição inicial e condições de Dirichlet e Neumann	94
Figura 21 – As soluções numérica e analítica em malha de elementos finitos e funções lineares	96
Figura 22 – O erro entre as soluções numérica e analítica $E = Erro = C_A - C^{1000} $	97

Figura 23 – Perfis de concentração para as soluções numérica e analítica	97
Figura 24 – A expressão analítica avaliada em três malhas diferentes: <i>Malha01</i> possui dois elementos em cada direção coordenada, <i>Malha02</i> foi obtida por refinamento uniforme da <i>Malha01</i> e <i>Malha03</i> foi obtida por 04 refinamentos uniformes e sucessivos da <i>Malha01</i> . A função $f_{Malha03}$ é considerada adequada para a representação da expressão analítica de $f(x) = x^2$	99
Figura 25 – A f_{L^2} e o erro de representação da função f pela função f_{L^2}	100
Figura 26 – Malha T'_0 fornecida pelo usuário antes do processo adaptativo	102
Figura 27 – Malha T_0 adequada aos dados gerada pelo procedimento adaptativo	103
Figura 28 – Solução numérica em malha adequada à representação dos dados	103
Figura 29 – O indicador espacial η_K em cada elemento da malha refinada no 1000-ésimo instante de tempo	106
Figura 30 – O indicador do elemento η_{1K} em cada elemento da malha refinada no 1000-ésimo instante de tempo	106
Figura 31 – O indicador do salto η_{2E} em cada elemento da malha refinada no 1000-ésimo instante de tempo	107
Figura 32 – O indicador temporal $\eta_{\tau_{1000}}$ em cada elemento da malha refinada	107
Figura 33 – Malha inicial T'_0 e malha adaptada adequada aos dados d_{xx} , $d_{xy} = d_{yx}$, d_{yy} , f_{Γ_D} , f	109
Figura 34 – A representação, em malha adaptada, das funções d_{yy} , da condição de Dirichlet f_{Γ_D} e do termo de fonte $f(x, y)$	110
Figura 35 – O domínio computacional, as curvas de nível do termo de fonte, o campo de velocidades e as condições de Dirichlet e Neumann	112
Figura 36 – A solução numérica em malha estruturada com funções lagrangeanas lineares	113
Figura 37 – As curvas de nível da solução numérica apresentada na figura (36)	113
Figura 38 – Medidas dos resíduos associados à solução numérica	115
Figura 39 – O tempo computacional e o tempo total para os sistemas lineares com $Nv = 1282401$ variáveis	116
Figura 40 – Medidas dos resíduos associados à solução numérica	117
Figura 41 – Os tempos computacional e total para sistema linear com $Nv = 1326721$ variáveis	118
Figura 42 – O domínio computacional, as curvas de nível para $\frac{C_0}{C_{C_0}}$, o campo de velocidades com $u_0 = 0.1$ e as condições de Dirichlet e Neumann	119
Figura 43 – O campo de velocidades e as curvas de nível da solução numérica	119
Figura 44 – Resíduos associadas à solução numérica	121
Figura 45 – O tempo computacional e o tempo total para um sistema linear com $Nv = 2253001$ variáveis	122

Figura 46 – Medidas dos resíduos associados à solução numérica	123
Figura 47 – O tempo computacional e o tempo total para um sistema com $N =$ 1024961 variáveis	124
Figura 48 – Malha não-estruturada em domínio não retangular	125
Figura 49 – A condição inicial em malha transladada para origem	125
Figura 50 – Curvas de nível da função de distribuição de contaminantes em malha transladada para origem	126
Figura 51 – Medidas dos resíduos associados a solução numérica	127
Figura 52 – O tempo computacional e o tempo total para sistemas lineares com $Nv = 1617033$ variáveis	128
Figura 53 – Extrapolação do tempo computacional para o número de passos de tempo $n > 100$	129
Figura 54 – Malha inicial e malha adaptada aos dados do problema do transporte de contaminantes	131
Figura 55 – O tempo computacional e o tempo total para os problemas do transporte e auxiliar. Sistema linear com $Nv = 72333$ vértices	133
Figura 56 – O tempo computacional e o tempo total combinados para os problemas do transporte e auxiliar. Sistema linear com $Nv = 72333$ variáveis . . .	134
Figura 57 – Curvas de nível das soluções numérica $C(x, y, z = constante, t_{40})$ e curvas de nível da solução analítica de Wexler (WEXLER, 1992)	135
Figura 58 – O erro entre as soluções numérica e analítica e as respectivas curvas de nível considerando a solução numérica do problema tridimensional $C_{z=constante}^{40} = C(x, y, z = constante, t_{40})$	135
Figura 59 – Medidas dos resíduos associados à solução numérica	137
Figura 60 – O tempo computacional e o tempo total para sistemas lineares com $Nv = 1238361$ variáveis	137

Lista de tabelas

Tabela 1	– Comparação da performance do método multigrid algébrico clássico, proposto por Ruge-Stüben(R-S), com o método do Gradiente Conjugado Pré-condicionado por fatoração de Cholesky incompleta (PCG2) para cinco problemas envolvendo o meio poroso.	40
Tabela 2	– A solução numérica no 200-ésimo instante de tempo com concentração normalizada, $\frac{C^{200}}{C_0} = \frac{C_{Numérica}}{C_0}$, a solução analítica em $t = t_f = 1000.0d$ com concentração normalizada, $\frac{C_A}{C_0} = \frac{C_{Analítica}}{C_0}$, o erro real (<i>Erro</i>), indicador espacial (η_K), o indicador do elemento (η_{1K}) e o indicador do salto (η_{2E}) para um conjunto de pontos	83
Tabela 3	– A solução numérica no 200-ésimo instante de tempo com concentração normalizada, $\frac{C^{200}}{C_0} = \frac{C_{Numérica}}{C_0}$, a solução analítica em $t = t_f = 1000.0d$ com concentração normalizada, $\frac{C_A}{C_0} = \frac{C_{Analítica}}{C_0}$, o erro real (<i>Erro</i>), indicador espacial (η_K), o indicador do elemento (η_{1K}) e o indicador do salto (η_{2E}) para um conjunto de pontos	89
Tabela 4	– As soluções numérica e analítica e o erro real no 1000-ésimo instante de tempo do processo de simulação	95
Tabela 5	– Valores máximos para todos os passos de tempo $n = \{1, 2, 3, \dots, N_I\}$: máximo dos resíduos máximos, máximo dos resíduos médios e o máximo dos resíduos na norma l^2	132

Sumário

Sumário	17
1 Introdução	21
1.1 Método de elementos finitos	22
1.2 Estimativas residuais	23
1.3 Adaptatividade dos dados do problema do transporte	24
1.4 Métodos multigrid	25
1.5 Computação Científica	25
1.6 Considerações do Capítulo	26
2 Objetivos	29
2.1 Objetivo principal	29
3 Revisão Bibliográfica	31
3.1 Conceitos importantes	31
3.2 Método Multigrid Geométrico	33
3.2.1 Os esquemas multigrid V-ciclo e F-ciclo	33
3.3 Método Multigrid Algébrico	36
3.4 Método Multigrid Algébrico baseado em agregação	41
3.4.1 AMG baseado em agregação suavizada	42
3.5 Método Multigrid Algébrico Adaptativo	44
4 Metodologia	47
4.1 A equação do transporte de solutos em meio poroso	47
4.2 Método de Elementos Finitos e Estimativas Residuais	48
4.2.1 Condições adicionais para o estimador residual	49
4.2.2 Formulação Variacional do Problema parabólico	50
4.2.3 Solução Analítica para Campo Uniforme	52
4.2.4 Solução analítica para campo com dependência espacial	53
4.3 Estimativas Residuais	54
4.3.1 Classificação dos Regimes de Advecção	54
4.3.2 Contribuições residuais	55
4.3.3 Estimativas Residuais <i>a posteriori</i>	57
4.3.4 Estimativas Residuais Robustas	59
4.3.5 Estimativas Residuais Gerais	61
4.3.6 Discussão Adicional sobre as Estimativas Residuais	61
4.3.6.1 Partição do Estimador Residual	62
4.3.7 Contribuições Residuais de cada elemento da malha	64
4.4 Adaptatividade	66
4.4.1 Limitação do Erro $\hat{\eta}_I$	66

4.4.2	Adaptatividade da Condição Inicial	68
4.4.3	Adaptatividade da função de fonte	69
4.5	Projeção L^2 e cálculo das constantes ε e β	70
4.5.1	Projeção L^2	70
4.5.2	Cálculo das constante ε e β	71
4.5.2.1	ε através do cálculo de autovalores	71
4.5.2.2	Cálculo da Constante β	72
4.6	AMG com Reutilização de estruturas	73
4.6.1	SA com Reutilização- $SAReu(\omega)$	75
4.6.2	SA com Reutilização e automático	75
4.6.3	Método Iterativo Pré-condicionado por $SA - Reu$	76
4.6.4	Método iterativo Pré-condicionado por $SA - ReuAuto$	76
4.6.5	Comparação de resultados	77
4.6.6	Tempo Computacional	78
4.7	Notas sobre os desenvolvimentos	78
5	Resultados	81
5.1	Método de elementos finitos e Estimador Residual	81
5.1.1	Exemplo 01	81
5.1.1.1	Transporte Conservativo em Campo Uniforme: $\lambda = 0$	83
5.1.1.2	Transporte Reativo: $\lambda \neq 0$	89
5.1.2	Exemplo 02	93
5.1.3	Projeção L^2	98
5.2	Malhas adequadas à representação dos dados	101
5.2.1	Exemplo 03: Malha adequada à Condição Inicial	101
5.2.1.1	Discussão sobre as estimativas residuais	104
5.2.2	Exemplo 04: Malha adequada à representação da função de fonte	108
5.3	Métodos Multigrid Acelerados	111
5.3.1	$BICGSTAB$ Pré-Condicionado SA	112
5.3.1.1	Malha Estruturada	114
5.3.1.2	Malha não estruturada	116
5.3.2	Multigrid Algébrico SA e $GMRES$ Pré-Condicionado por SA	118
5.3.2.1	Malha Estruturada	120
5.3.2.2	Malha não estruturada: $GMRES$ Pré-Condicionado por SA	122
5.3.3	Malha não estruturada em domínio não retangular	124
5.3.4	Reutilização para malha adaptada aos dados em regime de grande advecção	129
5.3.4.1	Malha não estruturada adaptada aos dados do transporte	130
5.3.5	A reutilização para o transporte 3D	134
6	Conclusão	139

6.1	Sugestões para Trabalhos Futuros	140
Referências		143
APÊNDICE A Código Python		151
A.1	Códigos para o Estimador Residual	151
A.1.1	Código para a projeção L^2	152
A.1.2	Código para campo gradiente contínuo	152
A.1.3	Código para transformar os dados	153
A.1.4	Código para componentes do erro residual	154
A.1.5	Código para o fator de ponderação α_S	155
A.1.6	Código para as contribuições residuais	155
A.1.7	Código para os indicadores residuais	156
A.1.8	Código para as estimativas residuais espacial e temporal	157
A.2	O regime de grande advecção	158
A.2.1	Código para as formas bilinear e linear do problema auxiliar	158
A.2.2	Código para as contribuições residuais do problema auxiliar	159
A.2.3	Código para obter as estimativas residuais do problema auxiliar	160
A.3	Códigos para as estimativas dos dados	160
A.3.1	Código das funções auxiliares para estimativas dos dados	161
A.3.2	Código para as componentes residuais dos dados	161
A.3.3	Código para as contribuições residuais dos dados	163
A.3.4	Código para os indicadores residuais dos dados	164
A.3.5	Código para as estimativas residuais dos dados	164
A.4	Códigos para as estimativas associadas ao termo de fonte	165
A.4.1	Código para os indicadores associados ao termo de fonte	166
A.5	Código para as estimativas residuais das fronteiras	167
A.6	Código adicional para as estimativas	167
A.7	Código para adaptatividade da condição inicial	169
A.7.1	Código para estimativa do erro da condição inicial	169
A.7.2	Código para refinamento adaptativo da condição inicial	170
A.7.3	Código modificado para refinamento adaptativo da condição inicial	171
A.7.4	Código para refinamento adaptativo da função de fonte	172
A.8	Códigos adicionais	173
A.8.1	Avaliação do erro real	173
A.9	Reutilização das estruturas multigrid	174
A.10	Nota Final	178
APÊNDICE B Exemplos completos		181
B.1	FEM e solução analítica	181
B.1.1	Comparação das soluções numérica e analítica, gráficos	185

B.2	Estratégia adaptativa	187
B.3	Código completo	189

1 Introdução

Devido à importância dada ao entendimento e à necessidade de previsões envolvendo o transporte de contaminantes, muitos modelos implementando a migração de solutos em meio poroso aparecem constantemente em publicações científicas. Estes modelos, formulados como equações diferenciais parciais (edp's), podem incluir uma série de fenômenos tais como difusão molecular, dispersão mecânica, dispersão hidrodinâmica, dependência espacial e temporal dos parâmetros, reações químicas, processos biológicos, heterogeneidade e anisotropia do meio poroso e fluido de densidade variável (SIMMONS; FENSTEMAKER; JR., 2001), (CAO; KITANIDIS, 1999), (BEKHIT; EL-KORDY; HASSAN, 2009), (FITTS, 2013), (LAPWORTH et al., 2012), (TUFENKJI; EMELKO; NRIAGU, 2011).

Soluções computacionais das edp's através de métodos de aproximação tais como método de elementos finitos requerem um adequado método de solução do sistema linear gerado. Por sua vez, essas soluções contêm erros inerentes ao processo de aproximação, os quais precisam ser quantificados. Essa quantificação favorece a avaliação da qualidade da aproximação numérica e fornece condições para o desenvolvimento de uma metodologia de refinamento adaptativo.

A combinação de alguns fatores tais como estratégias de refinamento adaptativo, grandes domínios, elevada precisão, representação adequada dos fenômenos físico-químicos ou biológicos pode gerar sistemas lineares com milhares, milhões ou bilhões de variáveis. Infelizmente, em muitos casos, tanto os métodos estacionários quanto os não-estacionários não são eficientes para resolver esses sistemas em um tempo aceitável. Para esses sistemas, torna-se necessária a utilização de métodos multigrid, os quais são considerados métodos cuja estimativa de tempo varia aproximadamente com o número de variáveis. Embora nem todos os métodos multigrid sejam robustos, eles são rápidos e, em muitos casos, escaláveis (STÜBEN; DELANEY; CHMAKOV, 2010), (STÜBEN, 2001). Esse potencial único é fundamental e tem conduzido as pesquisas para o desenvolvimento de algoritmos multigrid eficientes.

Nos últimos anos, os métodos multigrid algébricos tornaram-se disponíveis por meio de bibliotecas numéricas tais como PyAMG (BELL; OLSON; SCHRODER, 2011), HYPRE (FALGOUT; YANG, 2002), Trilinos (GEE et al., 2006). A possibilidade de integração com outras bibliotecas tais como SciPy (JONES et al., 2001) e FEniCS (FENICS PROJECT, 2014) torna possível propor e testar novos métodos multigrid algébricos.

Basicamente, a presente tese é composta de quatro partes: a formulação de elementos finitos baseada nas formas bilinear e linear, o estimador de erros com características residuais, a técnica adaptativa para os dados da equação e reutilização das estruturas

multigrid para acelerar o processo de solução dos sistemas lineares.

Primeiramente, a formulação variacional para os elementos finitos e respectiva validação do código são apresentadas. Nesta parte são apresentados: o código para implementação da aproximação do método de elementos finitos com formulação θA -estável e a validação do código computacional através da comparação com soluções analíticas disponíveis na literatura.

A disponibilidade da solução analítica possibilita a avaliação do erro real e, por sua vez, possibilita a comparação com as quantidades disponibilizadas pelo estimador residual. A comparação das quantidades reais locais com as quantidades estimadas locais permite a validação do estimador residual. Essas quantidades locais propiciam o desenvolvimento de estratégias adaptativas e o cálculo das quantidades globais associadas¹.

Em seguida, a adaptatividade dos dados do problema do transporte é discutida. Essa adaptatividade é baseada nas estimativas residuais locais e globais para os dados do problema do transporte e tem por objetivo construir malhas adequadas à representação dos dados.

Finalmente, são propostos os métodos multigrid algébricos que reutilizam as estruturas hierárquicas para acelerar o processo de solução dos sistemas lineares gerados pela aproximação de elementos finitos. Essa proposta é baseada na pequena velocidade do fluxo subterrâneo e na existência de uma estrutura hierárquica dos métodos multigrid. Os resultados envolvendo a técnica de reutilização consideram os métodos multigrid algébricos baseados em agregação suavizada e os métodos não-estacionários pré-condicionados por métodos multigrid baseados em agregação suavizada.

A finalidade das seções seguintes é caracterizar o método de elementos finitos, as estimativas residuais, a adaptatividade dos dados do problema do transporte e os métodos multigrid algébricos.

1.1 Método de elementos finitos

A aproximação por elementos finitos da equação de advecção-difusão-reação, que governa o transporte de contaminantes em meio poroso saturado, gera um sistema linear de equações algébricas da forma $A^n C^n = b^n$, em que A é uma matriz que depende da malha de elementos finitos e do grau das funções utilizadas na aproximação da solução desconhecida, C é a solução do sistema linear e representa a solução de elementos finitos nos vértices dos elementos e n é o passo de tempo. Iterativamente, obtém-se uma sequência de soluções $(C^i)_{i=0}^{N_I}$ que compõe as respectivas soluções em cada passo de tempo da aproximação para

1. Refere-se a uma quantidade definida sobre todo o domínio computacional. Em oposição tem-se as quantidades locais definidas em um elemento específico do domínio computacional.

um total de N_I passos. Nesse caso, para cada n , a solução C^n fornece as respectivas concentrações nodais da malha de elementos finitos. Se um problema com solução analítica é considerado, então uma comparação quantitativa da solução numérica com solução analítica respectiva pode ser efetuada para a validação do código computacional.

O desenvolvimento do código computacional que implementa o método de elementos finitos com esquema θA -estável e as soluções analíticas caracterizam a primeira parte deste trabalho.

1.2 Estimativas residuais

As estimativas residuais são consideradas para fornecer medidas quantitativas da qualidade da solução numérica tanto globalmente como localmente. As medidas globais referem-se às quantidades definidas sobre toda a malha de elementos finitos enquanto que as medidas locais referem-se às quantidades definidas em cada entidade da malha de elementos finitos. A utilização conjunta das quantidades locais e globais favorece a implementação de algoritmos adaptativos para a redução local e global dos erros associados à aproximação de elementos finitos.

Neste trabalho, as contribuições locais são: as contribuições dos interiores; as contribuições dos saltos; as contribuições das fronteiras; as contribuições dos dados, das fronteiras de Neumann e do termo de fonte; as contribuições da norma da energia, a qual está relacionada à contribuição temporal. Essas contribuições locais geram as contribuições globais dos elementos, dos saltos, das fronteiras, dos dados, das fronteiras de Neumann, do termo de fonte e da norma da energia. A soma dessas contribuições globais em determinado passo de tempo, conduz às contribuições globais naquele passo específico de tempo. A adequada ponderação dessas quantidades globais gera a estimativa global sobre toda a malha e sobre todos os passos de tempo.

As medidas residuais adotadas dependem do regime de advecção, o qual é caracterizado por constantes que relacionam os termos difusivos e advectivos da equação. Essas constantes, baseadas apenas nas relações entre os termos difusivos e advectivos, possibilitam separar os regimes de dispersão dominante e advecção dominante². As estimativas residuais para os regimes de dispersão dominante caracterizam-se pela estimativa da norma da energia baseada nas quantidades calculadas na solução da equação de advecção-difusão-reação, enquanto que as estimativas residuais para o regime de advecção dominante caracteriza-se pela solução de um problema elíptico auxiliar. Esse problema auxiliar inclui um esforço computacional para obtenção das estimativas residuais, pois é definido sobre a mesma malha de elementos finitos com funções lagrangeanas lineares.

2. Verfürth (2014) separa os regimes em pequena e grande advecção.

O desenvolvimento de uma metodologia para o cálculo das estimativas residuais em regimes de dispersão dominante e advecção dominante caracteriza a segunda parte do trabalho.

1.3 Adaptatividade dos dados do problema do transporte

A adaptatividade configura-se como uma técnica para aprimorar a solução em regiões específicas. Basicamente, são utilizadas as estimativas globais e estimativas locais para controlar o processo adaptativo. A estimativa global controla a execução do processo enquanto as estimativas locais controlam os elementos que devem ser refinados.

A adaptatividade dos dados refere-se à técnica adaptativa para a construção de uma malha inicial T_0 , adequada à representação dos dados no espaço de elementos finitos. Essa construção é necessária, pois os dados podem não estar bem representados no espaço de funções adotado e, além disso, a existência da malha T_0 é uma das condições essenciais para o emprego do algoritmo adaptativo geral apresentado por [Verfürth \(2014\)](#). Por exemplo, a representação de funções com variação exponencial pode não ser adequada em uma malha de elementos finitos com funções lineares.

Neste trabalho, duas metodologias para a construção da malha T_0 são propostas. A primeira é baseada na estimativa residual associada à condição inicial, enquanto que a segunda é baseada na estimativa residual do termo de fonte. Os exemplos apresentam resultados envolvendo as duas estratégias.

Com base nas estimativas residuais e no algoritmo adaptativo geral, a questão da representação adequada dos dados foi discutida. Essa discussão foi motivada pela suposição, presente no algoritmo adaptativo geral, de que uma malha que represente adequadamente os dados e a geometria do problema estão disponíveis. Nesse caso, uma malha uniforme e refinada pode ser construída, mas, no entanto, a quantidade de elementos pode ser grande e afetar os processos de refinamento adaptativo da solução numérica. Nesse contexto, o algoritmo adaptativo para a adequada representação dos dados, baseado nas estimativas residuais, foi proposto. Esse algoritmo considera que funções lagrangeanas de ordens p e $p + k$, em que p, k são inteiros, fornecem representações diferentes dos dados. Logo, baseado na estimativa residual adequada, o processo de refinamento pode ser proposto.

A questão da adaptatividade dos dados do problema do transporte caracteriza a terceira parte deste trabalho.

1.4 Métodos multigrid

De posse de uma malha adequada à representação dos dados do problema³, os métodos multigrid algébricos com reutilização das estruturas hierárquicas são empregados para obter uma solução da equação do transporte. As malhas de elementos finitos são obtidas através de refinamentos uniformes sucessivos de uma malha inicial ou de refinamentos uniformes sucessivos de uma malha inicial adaptada. Ambos os processos geram sistemas lineares com milhares ou milhões de variáveis.

Os métodos multigrid algébricos com reutilização das estruturas são utilizados para obter as soluções tanto do problema de advecção-difusão-reação quanto do problema auxiliar associado às estimativas residuais do regime de advecção dominante. As soluções numéricas, obtidas por diferentes estratégias de reutilização, são comparadas com as soluções obtidas pelos métodos sem reutilização.

Basicamente, nesta parte duas classes de métodos com reutilização são definidas: a classe de métodos em que os cálculos das agregações e operadores de transferência de resultados são executados a cada ω passos de tempo e a classe de métodos que determina, baseada no número de iterações necessárias para a convergência no momento da execução dos cálculos, automaticamente o passo de tempo em que os cálculos das agregações e operadores de transferência de resultados são necessários. As técnicas de reutilização podem ser aplicadas tanto aos métodos multigrid algébricos quanto aos métodos não-estacionários pré-condicionados por multigrid algébricos.

Os métodos multigrid algébricos considerados são implementados na biblioteca PyAMG (BELL; OLSON; SCHRODER, 2011), enquanto que os métodos não-estacionários são aqueles implementados por meio do pacote linalg disponível na biblioteca SciPy (JONES et al., 2001).

1.5 Computação Científica

A possibilidade de integração das diferentes bibliotecas utilizadas neste trabalho facilita o desenvolvimento e teste de diferentes métodos ou metodologias. No caso deste trabalho, a integração dos pacotes e bibliotecas possibilitaram: análise numérica dos resultados obtidos, apresentação gráfica com metodologia similar ao Matlab^R, a integração numérica, a maximização e minimização de funções, as manipulações vetoriais e matriciais dos resultados numéricos e a conversão de dados para os diferentes formatos exigidos pelas diferentes bibliotecas utilizadas.

A análise detalhada das soluções numéricas é possibilitada pela biblioteca

3. Uma malha com refinamento uniforme ou malha gerada pelo algoritmo adaptativo

NumPy ([WALT; COLBERT; VAROQUAUX, 2011](#)). Isso permite uma análise detalhada das concentrações em cada um dos vértices dos elementos finitos além de disponibilizar as respectivas coordenadas. Além disso, a análise *a posteriori* das soluções é facilitada pelas funções que permitem ler e salvar arquivos em formatos diversos para pós-processamento.

A biblioteca SciPy ([JONES et al., 2001](#)) disponibiliza, entre outras possibilidades, o pacote de integração numérica *QUADPACK*, o pacote *linalg* e alguns métodos de minimização local e global. Isso torna possível, através de uma metodologia aqui apresentada, o cálculo das constantes ε , κ , β necessárias às estimativas residuais, a integração numérica para obtenção das soluções analíticas cuja avaliação acurada exigem métodos adaptativos de integração, a utilização de métodos não estacionários já testados e com códigos eficientemente implementados.

A biblioteca gráfica matplotlib ([HUNTER, 2007](#)) fornece um ambiente adequado para as apresentações gráficas tornando possível tanto a apresentação de soluções em malhas estruturadas e não estruturadas. Para grandes conjuntos de dados e apresentações mais elaboradas pode-se usar o Paraview ([AYACHIT, 2015](#)). Por outro lado, uma visualização rápida pode ser obtida por meio das ferramentas disponíveis em FEniCS ([FENICS PROJECT, 2014](#)).

A biblioteca para métodos multigrid é utilizada para obter solução iterativa dos sistemas lineares gerados pela aproximação de elementos finitos da equação do transporte. A conveniente transformação entre os diversos formatos de dados requeridos é proporcionada pela biblioteca SciPy ([JONES et al., 2001](#)).

A integração entre todas as bibliotecas numéricas utilizadas nessa tese foi feita em Linguagem Python^R. A utilização dessa linguagem está estritamente ligada à disponibilidade de documentação para integração das diferentes bibliotecas numéricas citadas.

1.6 Considerações do Capítulo

O desenvolvimento do código computacional para a aproximação da solução numérica pelo método de elementos finitos é o primeiro item considerado nesse trabalho. Em seguida, as estimativas residuais, a metodologia para o cálculo das constantes ε , κ e as estratégias adaptativas dos dados são apresentados e desenvolvidos. Finalmente, os métodos multigrid com reutilização são introduzidos.

Basicamente, a presente tese apresenta as contribuições divididas em três partes:

1. O desenvolvimento do código computacional para o método de elementos finitos com esquema θ A-estável, o código para o estimador residual para os regimes de dispersão dominante e advecção dominante. Adicionalmente, são desenvolvidos os códigos para incorporar as estimativas residuais associadas aos dados do problema do transporte

e uma metodologia para cálculo das constantes associadas às estimativas residuais do regime de advecção dominante.

A partição do código, a qual reflete a partição do estimador residual, permite que o cálculo das estimativas residuais seja efetuado de maneira similar às fórmulas apresentadas.

2. O desenvolvimento de duas estratégias adaptativas para os dados do problema do transporte de contaminantes. A primeira estratégia adaptativa é baseada na estimativa residual para a condição inicial, enquanto que a segunda estratégia é baseada na estimativa residual para o termo de fonte. A estratégia para a condição inicial constrói uma malha adequada à representação da condição inicial enquanto que a segunda estratégia constrói uma malha adequada à representação do termo de fonte. Aqui, ambas as estratégias são aplicadas aos outros termos presentes na equação de advecção-difusão-reação.
3. O desenvolvimento de um esquema de reutilização das estruturas hierárquicas do método multigrid algébrico. Novamente, duas possibilidades são apresentadas: um método em que a reutilização das estruturas multigrid é feita por um número fixo de passos de tempo e um método que determina automaticamente se a estrutura multigrid pode ser usada no próximo passo de tempo. Essas técnicas são estendidas para os métodos iterativos pré-condicionados por métodos multigrid algébricos.

Para conduzir essas discussões, suas respectivas implementações, as contribuições e os resultados apresentados neste trabalho, a tese encontra-se organizada da seguinte forma:

O capítulo 01 apresenta uma breve descrição do problema a ser abordado juntamente com as etapas desenvolvidas e as contribuições do trabalho.

O capítulo 02 apresenta, de forma sucinta, o objetivo principal da tese e as etapas intermediárias necessárias.

O capítulo 03 apresenta uma revisão bibliográfica, descrevendo os métodos multigrid geométricos, os métodos multigrid algébricos clássicos, os métodos multigrid algébricos baseados em agregação suavizada e os métodos multigrid algébricos adaptativos. A relação entre os métodos multigrid algébricos e geométricos é apresentada de forma sucinta bem como as relações entre os métodos multigrid algébricos clássicos e os baseados em agregação suavizada.

O capítulo 04 descreve a metodologia utilizada nesta tese. Nesse capítulo, são apresentados: a formulação de elementos finitos, as estimativas residuais globais e locais, a estratégia adaptativa para a condição inicial, a estratégia adaptativa para o termo de fonte e a técnica de reutilização. Adicionalmente, são apresentados: a metodologia de comparação dos resultados numéricos e analíticos e a metodologia de comparação dos tempos de soluções dos sistemas lineares.

O capítulo 05 apresenta os resultados e discussão. Inicialmente, a validação do código computacional para a aproximação de elementos finitos, implementado em linguagem Python^R, é apresentada. Em seguida, os estimadores e indicadores residuais são apresentados. Após, os resultados das estratégias adaptativas para a condição inicial e o termo de fonte são apresentados. Finalmente, os resultados envolvendo os métodos multigrid algébricos com reutilização são comparados aos resultados provenientes dos métodos multigrid sem reutilização. Os métodos com reutilização são empregados em diversos problemas envolvendo o transporte de contaminantes e englobam problemas reativos e não reativos, malhas estruturadas e não estruturadas, dados com dependência espacial e problemas tridimensionais.

O capítulo 06 apresenta as conclusões acerca das contribuições apresentadas no corpo desta tese e aplicadas em problemas diversos envolvendo o transporte de contaminantes em meio poroso. Na seção [6.1], são descritos os trabalhos futuros, os quais têm por objetivo complementar os trabalhos desenvolvidos nesta tese.

Os apêndices [A] e [B] apresentam, respectivamente, as classes e funções desenvolvidas para este trabalho e alguns exemplos de códigos que podem ser úteis para a continuidade ou desenvolvimento de trabalhos futuros.

2 Objetivos

Neste capítulo são descritos o objetivo principal e as etapas intermediárias deste trabalho científico orientado pela seguinte hipótese: A reutilização das estruturas hierárquicas multigrid produz um algoritmo multigrid eficiente para resolver a aproximação, pelo método de elementos finitos, da equação do transporte de contaminantes em meio poroso saturado.

2.1 Objetivo principal

Formular e implementar um método multigrid algébrico com reutilização das estruturas de agregação e operadores de transferência de resultados para resolver sistemas lineares gerados pela aproximação, pelo método de elementos finitos, da equação de advecção-difusão-reação.

Para que o Objetivo Principal da tese seja alcançado, são delineadas as seguintes etapas intermediárias:

1. Elaborar um código computacional, em linguagem Python, para resolver a equação do transporte de contaminantes, utilizando o método de elementos finitos com esquema θA -estável;
2. Desenvolver uma metodologia para o cálculo das constantes envolvidas nas estimativas residuais, utilizando os processos de minimização;
3. Elaborar um código computacional para a metodologia do item 2;
4. Elaborar um código computacional, em linguagem Python, para calcular as estimativas residuais associadas à equação de advecção-difusão-reação em meio poroso saturado. Esse código deve ser capaz de incluir as estimativas residuais para os regimes de dispersão dominante e advecção dominante, as estimativas residuais dos dados, as estimativas residuais associadas às fronteiras de Neumann e as estimativas residuais associadas aos termos de fonte;
5. Desenvolver: uma estratégia de refinamento adaptativo para a condição inicial e uma estratégia de refinamento adaptativo para o termo de fonte;
6. Elaborar os códigos computacionais, em linguagem Python, para o item 5;
7. Desenvolver um algoritmo multigrid com reutilização das estruturas em determinados passos de tempo do processo de simulação;

8. Desenvolver um algoritmo multigrid com reutilização automática: o método deve incorporar uma estratégia para verificar se a reutilização das estruturas é eficiente para resolver o sistema linear no próximo passo de tempo;
9. Elaborar um código computacional, em linguagem Python, para o item 6 e 7;
10. Expandir os desenvolvimentos dos itens 7., 8. e 9. para os métodos não estacionários pré-condicionados por métodos multigrid algébricos;
11. Verificar que os métodos propostos fornecem soluções compatíveis com aquelas do método original.

3 Revisão Bibliográfica

A necessidade de obter solução de grandes sistemas lineares provenientes de processos de discretização de equações diferenciais parciais que surgem da modelagem de diferentes fenômenos físicos conduz à busca de técnicas computacionais que sejam rápidas (escaláveis). Considerando um problema de Poisson 3D, a solução de um sistema linear da forma $Ax = b$, em que A é $N \times N$, exige $O(N^9)$ operações para Eliminação Gaussiana sem utilização da característica de esparsidade da matriz, $O(N^5)$ para os métodos de Jacobi e Gauss-Seidel e $O(N^4)$ para o método *SOR* para uma escolha ótima do parâmetro (MACLACHLAN, 2004).

A consideração de extensos domínios, a presença de diversos fenômenos físico-químicos, a necessidade de uso de algoritmos adaptativos e simulações 3D tornam esses métodos menos atrativos em relação aos métodos multigrid. Métodos multigrid são, geralmente, classificados na literatura como algoritmos rápidos ou escaláveis e a generalidade de aplicações favorece seu uso em problemas envolvendo o transporte de contaminantes em meio poroso. Segundo Stüben (2001), os métodos multigrid podem ser divididos em duas classes distintas: métodos multigrid geométricos e métodos multigrid algébricos. Esses métodos são estritamente relacionados e, por isso, os conceitos envolvendo métodos multigrid geométricos são apresentados primeiramente. Em seguida, os conceitos relacionados aos métodos multigrid algébricos são apresentados.

3.1 Conceitos importantes

Nesta parte, alguns conceitos relacionados ao desenvolvimento e entendimento dos métodos multigrid são apresentados. Esses conceitos seguem, principalmente, os pontos de vista das referências (RUGE; STÜBEN, 1986), (TROTTEBERG; OOSTERLEE; SCHULLER, 2001), (BRIGGS, 1999) e (DOUGLAS, 1992).

Considere o problema de obter a solução do sistema $Au = f$, onde A é uma matriz $N \times N$ e seja v uma solução aproximada para u . O erro $e = u - v$, medido de acordo com as normas $\|e\|_\infty = \max|e_i|$ ou a norma $\|e\|_2^2 = \sum_{i=1}^N e_i^2$, pode ser relacionado com o resíduo r através da equação $r = f - Au$. Desta forma, o erro satisfaz a equação $Ae = f - Au = r$. Se a solução exata da equação $e = A^{-1}r$ é conhecida, então $u = v + e$ é a solução exata do sistema de equações. No entanto, esse esquema de correção residual não é efetivo, pois o trabalho computacional equivale a resolver o sistema original.

Baseado na observação acima, um método mais efetivo pode ser desenvolvido. Considere, por exemplo, o método iterativo de *Jacobi* para a solução do sistema linear $Au = f$,

ou seja, $A = D - (L + U)$, onde D é a matriz diagonal e L, U são as matrizes com partes inferiores e superiores de A , respectivamente. Logo,

$$Au = f \Rightarrow (D - (L + U))u = f \Rightarrow u = Ru + D^{-1}f \quad (3.1)$$

onde $R = D^{-1}(L + U)$. O método iterativo pode ser escrito como $v^{(i+1)} = Rv^{(i)} + D^{-1}f$ e o erro $e = u - v^{(i+1)}$ pode ser escrito como

$$e^{(i+1)} = u - v^{(i+1)} = Ru + D^{-1}f - (Rv^{(i)} + D^{-1}f) = R(u - v^{(i)}) = Re^{(i)}, \quad (3.2)$$

o qual pode, recursivamente, ser escrito como $e^{(i+1)} = R^{(i+1)}e^{(0)}$, para todo $i \geq 1$.

Baseado nos conceitos de Álgebra Linear, se existe um conjunto de N autovetores linearmente independentes para uma matriz $N \times N$ então este conjunto forma uma base $\{w_k\}_{k=1}^N$. Logo, um vetor v pode ser escrito como combinação linear dos elementos da base, ou seja, $v = \sum_{k=1}^N c_k w_k$, $c_k \in \mathbb{R}$, $k \geq 1$. Consequentemente, o erro inicial $e^{(0)}$ pode ser representado por esta base, ou seja, $e^{(0)} = \sum_{k=1}^N c_k w_k$ e o erro na i -ésima iteração pode ser representado por

$$e^{(i+1)} = R^{(i+1)}e^{(0)} = \sum_{k=1}^N c_k R^{(i+1)}w_k = \sum_{k=1}^N c_k \lambda_k^{(i+1)} w_k \quad (3.3)$$

onde λ_k são os autovalores da matriz R . Logo, para todo erro inicial $e^{(0)}$, então $e^{(i+1)} \rightarrow 0$ quando $i \rightarrow \infty$ se e somente se $\rho(R) = \max\{|\lambda_1|, |\lambda_2|, \dots, |\lambda_N|\} < 1$ ¹ (BRIGGS, 1999).

O maior valor absoluto dentre os autovalores na metade superior do espectro da matriz de iteração R é chamado fator de suavização (BRIGGS, 1999) e é dado por:

$$SmoothingFactor = \max\{|\lambda_k|\}, \quad \text{para} \quad \frac{N}{2} \leq k \leq N \quad (3.4)$$

A primeira observação para a construção dos métodos multigrid é que muitos esquemas de relaxação possuem essa propriedade de suavização, em que os modos oscilatórios do erro são eliminados efetivamente, enquanto modos suaves são reduzidos lentamente. O fator de suavização é dependente da quantidade de elementos na malha, e a utilização de uma malha mais grossa, com menos elementos, transforma os modos suaves da malha fina em modos oscilantes na malha grossa. Na malha grossa, os modos oscilantes podem, novamente, ser efetivamente eliminados (BRIGGS, 1999).

A segunda observação consiste em utilizar o esquema de correção residual para a malha grossa. A próxima seção apresenta o esquema de dois níveis.

1. Segundo Briggs (1999) $\rho(R) < 1$ assegura a convergência do processo iterativo definido por R e $\rho(R)$ é chamado o fator de convergência.

3.2 Método Multigrid Geométrico

O primeiro trabalho descrevendo as ideias dos métodos multigrid é datado de 1935, quando Southwell discute um esquema de relaxação para uma malha de dois níveis. Em 1964 Fedorenko formulou o primeiro algoritmo multigrid verdadeiro para um esquema padrão de diferenças finitas aplicado à equação de Poisson, mostrando que o esforço computacional requerido para atingir dada precisão era da ordem do número de variáveis no sistema linear. O reconhecimento da utilidade prática e da generalidade tornou a nova técnica aplicável em diversas áreas do conhecimento (SAAD, 2003).

No contexto do meio subterrâneo, a técnica multigrid encontrou aplicações nas simulações envolvendo reservatórios de petróleo em duas e três dimensões (SCOTT, 1985). A consideração de grandes sistemas envolvendo meio poroso, computação em paralelo, a influência dos avanços computacionais nas técnicas iterativas e solução de grandes sistemas lineares são apresentadas em (HÖLTER; VANDERBERGHE, 1990), (MCBRYAN et al., 1991) e (SAIED; MAHINTHAKUMAR, 1998). A utilização do processamento em paralelo para resolução de problemas envolvendo o meio poroso foi considerada em (MUSTAPHA; GHORAYEB; MUSTAPHA, 2010) e (COUMOU et al., 2008).

Métodos multigrid foram aplicados na resolução numérica das equações de advecção-difusão por (MURATOVA; ANDREEVA, 2009), considerando esquema de diferenças finitas e estado estacionário. Coeficientes variáveis e esquema de diferenças finitas com refinamento local são apresentados em (LAI; WU; TSENG, 2007) e (ZHANG; SUN; ZHAO, 2002). Além disso, a adaptatividade foi considerada por (TRANGENSTEIN, 2002) e (LI; CHENG; YEH, 2000).

3.2.1 Os esquemas multigrid V-ciclo e F-ciclo

Considere duas malhas de elementos finitos Ω^h e Ω^H , sendo que Ω^h é um refinamento uniforme de Ω^H . O processo de aproximação da equação diferencial parcial em cada malha conduz a solução dos sistemas lineares $A^h x_h = b^h$ e $A^H x_H = b^H$. Um esquema multigrid de dois níveis pode ser apresentado como segue (BRIGGS, 1999):

ALGORITMO 01: Método V-ciclo de dois níveis

1. Faça ν_1 iterações em $A^h u^h = f^h$, com solução inicial v^h arbitrária;
2. Calcule resíduo $r^h = f^h - A^h u^h$ e transfira os resultados para a malha grossa usando o operador de transferência de resultados da malha fina para a malha grossa $r^H = I_h^H r^h$;

3. Resolva a equação residual na malha grossa para obter a correção residual, ou seja, $A^H e^H = r^H \Rightarrow e^H = (A^H)^{-1} r^H$;
4. Utilize o operador de transferência de resultados da malha grossa para a malha fina $e^h \approx I_H^h e^H$, para obter um resultado aproximado do erro na malha fina;
5. Atualize o resultado obtido na malha fina, $v^h \leftarrow v^h + e^h = u^h + I_H^h u^h$;
6. Faça ν_2 iterações em $A^h u^h = f^h$, em Ω^h com solução inicial v^h atualizada no passo anterior.

A aplicação de um ciclo do esquema apresentado é denotada por $v^h \leftarrow MG(f^h, v^h, \nu_1, \nu_2)$. Esse esquema é conhecido como esquema de dois níveis, e é essencial no entendimento dos métodos multigrid. Ilustrativamente, a figura (1-a) representa um esquema de duas malhas para o caso de elementos finitos retangulares². Nesse caso, a figura (1 – a) representa as discretizações com malhas de tamanho h e $H = 2h$, enquanto que a figura (1 – b) representa, esquematicamente, o processo de correção.

Figura 1 – Malha grossa Ω^H e malha fina Ω^h com $H = 2h$ e esquema multigrid V-ciclo de dois níveis

Fonte: (FULTON; CIELSKI; SCHUBERT, 1986) e (BRIGGS, 1999), respectivamente.

O esquema de dois níveis possibilita o entendimento de um método multigrid de n níveis, pois basta aplicar o processo recursivamente.

As figuras (2-a) e (2-b) representam, esquematicamente, os métodos multigrid do tipo $V - Ciclo$ e $F - Ciclo$, respectivamente. A diferença básica é o início do processo iterativo, que pode ser dividido em métodos de correção e método de iterações sucessivas. O esquema V-ciclo tem início na malha fina e transfere os resultados para a malha mais grossa enquanto que o esquema F-ciclo tem início na malha grossa e transfere os resultados para a malha mais fina.

2. Originalmente, os esquemas de diferenças finitas foram considerados

A transferência dos resultados da malha fina para a malha grossa é realizada pelo operador de restrição $I_h^H : \Omega^h \longrightarrow \Omega^H$, enquanto que a transferência de resultados da malha grossa para a malha fina é realizada pelo operador de prolongamento $I_H^h : \Omega^H \longrightarrow \Omega^h$. Para o método de elementos finitos, os operadores de restrição e prolongamento são relacionados pelo Princípio de Galerkin, isto é, $A^H = I_h^H A^h I_H^h$ (STÜBEN, 2001).

Figura 2 – Esquemas multigrid V-ciclo e F-ciclo de n níveis

Fonte: (BRIGGS, 1999)

Do ponto de vista matricial, os métodos multigrid geométricos podem ser caracterizados, ilustrativamente, pela figura (3).

Figura 3 – Esquemas multigrid V-ciclo de 4 níveis e sistemas lineares associados

Fonte: (STÜBEN, 2009)

3.3 Método Multigrid Algébrico

Métodos multigrid algébricos, em contraste com os métodos geométricos, não necessitam de conhecimento explícito da geometria do problema. Estes métodos constroem a hierarquia de malhas para o problema baseados no sistema linear e em certas suposições sobre o erro de solução do sistema linear.

Os conceitos dos métodos multigrid algébricos (AMG) foram introduzidos por A. Brandt, S. McCormick e J. Ruge com contribuições de A. Brandt. A apresentação dos principais conceitos envolvendo o método algébrico é fornecida em (STÜBEN, 2001) enquanto um tratamento completo dos métodos multigrid é apresentado em (TROTTERBERG; OOSTERLEE; SCHULLER, 2001).

A dificuldade de utilização dos métodos multigrid geométricos conduziu à busca de métodos algebricamente orientados. Uma das razões é a crescente complexidade das geometrias em aplicações, o que limita o uso dos métodos multigrid baseados em hierarquias de malhas. A segunda razão é que os métodos multigrid podem ser métodos escaláveis, ou seja, o esforço computacional varia, aproximadamente, com o número de variáveis (STÜBEN; DELANEY; CHMAKOV, 2010).

A busca por métodos escaláveis é necessária, pois a quantidade de variáveis envolvidas pode tornar o esforço computacional inviável para problemas com alguns milhões ou bilhões de variáveis, ou seja, problemas em que o número de variáveis são $O(10^6)$ ou $O(10^9)$. Apesar dos avanços computacionais, esses problemas ainda requerem métodos eficientes que possam ser integrados nos softwares existentes, sem necessidade de alteração de todo código desenvolvido e em tempo reduzido (STÜBEN; DELANEY; CHMAKOV, 2010).

Resumidamente, o método multigrid geométrico fixa as malhas e define, convenientemente, os operadores de interpolação e restrição. O método multigrid algébrico tem uma abordagem oposta e substitui os conceitos geométricos por conceitos abstratos de conjuntos. Da mesma forma que no método multigrid geométrico é necessário resolver um sistema linear:

$$A_h u^h = f^h \quad \text{ou} \quad \sum_{i=1}^N a_{ij}^h = f_i^h \quad \text{para} \quad i \in \Omega^h. \quad (3.5)$$

A diferença é que, no contexto dos métodos multigrid algébricos, Ω^h é um conjunto de índices, enquanto no método multigrid geométrico Ω^h corresponde a uma discretização do domínio. Finalmente, uma malha grossa H é um subconjunto, Ω^H , do conjunto de índices Ω^h . Se os operadores de transferência de resultados são conhecidos, então o mapeamento dos vetores de Ω^H em vetores de Ω^h pode ser feito por um operador de interpolação de

posto completo. O operador da malha grossa A^H é definido por

$$A^H := I_h^H A^h I_H^h \quad \text{com} \quad I_h^H = (I_H^h)^T. \quad (3.6)$$

Dessa forma, o método multigrid algébrico de dois níveis pode ser definido de forma similar ao método multigrid geométrico, ou seja, $u^h \leftarrow u^h + I_H^h e^H \approx u^h + e^h$ em que a correção $r^H = I_h^H r^h$ é a solução exata da equação na malha grossa. Esquemáticamente, as figuras (1-b), (2-a), (2-b) e (3) podem ser utilizadas para representar um método multigrid algébrico.

Se $e^h = u^{exact} - u^h$ é o erro entre a solução exata na malha fina e a solução depois de algumas iterações, então

$$e_{i+1}^h = K_h^H e_i^h = (I_h - I_H^h A_H^{-1} I_h^H A_h) e^h, \quad (3.7)$$

onde K_h^H é chamado operador de correção da malha grossa. No AMG clássico, também chamado de Ruge-Stüben (RS), as variáveis da malha fina são divididas em dois conjuntos disjuntos, ou seja, $\Omega^h = C^h \cup F^h$ onde $C^h = \Omega^H$ são variáveis que pertencem à malha grossa (C-variáveis) e $F^h = \Omega^h - C^h$ as variáveis complementares que pertencem estritamente à malha fina (F-variáveis). As variáveis C e F são escolhidas de acordo com as seguintes observações (TROTTERBERG; OOSTERLEE; SCHULLER, 2001), (BRIGGS, 1999):

1. Determine um subconjunto de pontos da malha grossa que possa ser usado com os seguintes objetivos:
 - i. representar os erros suaves;
 - ii. para interpolar estes erros na malha fina;
2. intuitivamente, um ponto u_j é um bom candidato para um C -ponto se o seu valor é importante para determinação do valor de outro ponto u_i na i -ésima equação;
3. Se o coeficiente a_{ij} é "grande" comparado com outros elementos fora da diagonal na i -ésima equação, então u_j influencia u_i , ou u_i depende de u_j . Para matrizes simétricas e positivas definidas, se

$$-a_{ij} > \theta \max_{k \neq i} \{-a_{ik}\}, \quad 0 < \theta \leq 1, \quad (3.8)$$

então o ponto i influencia o ponto j ;

4. O conjunto de dependentes de uma variável u_i , ou seja, as variáveis das quais o valor de u_i depende é definido por $S_i = \{j : a_{ij} > \theta \max_{k \neq i} \{-a_{ik}\}\}$, e o conjunto de pontos que é influenciado por u_i é definido por $S_i^T = \{j : i \in S_j\}$.

Denotando, C o conjunto das variáveis da malha grossa, F o conjunto de variáveis na malha fina e P_i^h o conjunto de variáveis interpoladoras da malha grossa, ou seja, as variáveis da malha grossa usadas para interpolar o valor da variável da malha fina.

5. A escolha dos C-pontos é norteada por dois critérios:
 - i. Para cada $i \in F$, cada $j \in S_i$ deve estar em C ou deve depender de pelo menos um ponto P_i^h
 - ii. C deve ser o maior subconjunto com a propriedade em que nenhum C -ponto depende de outro

Com estes dois critérios, a estratégia para escolher os C-pontos é dada por:

1. para cada ponto da malha k , calcule o número de pontos que dependem de k . Atribua a este ponto o valor calculado
2. escolha o primeiro ponto com máximo valor global como sendo um C-ponto
3. O novo C -ponto pode ser usado para interpolar os valores dos pontos que ele influencia. Marque todos estes pontos como F-pontos
4. outros pontos influenciando estes novos F-pontos podem ser usados na interpolação. Atualize seus valores
5. Repita até que todos os pontos são C -pontos ou F-pontos.

O procedimento apresentado fornece uma partição da malha e é chamada partição C/F . Considerando que é dada uma partição C/F , o operador de interpolação é definido por :

$$e_i^h = (I_H^h e^H)_i = \begin{cases} e_i^H & \text{se } i \in C^h \\ \sum_{k \in P_i^h} w_{ik}^h e_k^H & \text{se } i \in F^h \end{cases} \quad (3.9)$$

Por razões de eficiência P_i precisa ser um conjunto pequeno de variáveis da malha grossa na vizinhança do ponto i (STÜBEN, 2001). Heuristicamente, para o esquema de Gauss-Seidel, um erro que converge lentamente é tal que

$$e^{i+1} = (I - (L + D)^{-1}A)e^i \approx e^i \Rightarrow (L + D)^{-1}Ae^i \approx 0 \Rightarrow r \approx 0 \Rightarrow r_i \ll a_{ii}e_i. \quad (3.10)$$

Logo, se e é um erro com convergência lenta, então localmente e_i pode ser representado pela média de seus vizinhos (STÜBEN, 2001). Para definir os pesos convenientemente, é necessário diferenciar entre os tipos de conexões existentes, ou seja:

- i. C_i -i é dependente destes C -pontos;
- ii. D_i^s - i é dependente destes F -pontos;

- iii. D_i^w - i não é dependente destes pontos fracamente conectados sendo C -pontos ou F -pontos;

O operador de prolongamento é baseado na suavidade do erro e nas dependências definidas acima com os pesos são definidos por

$$\omega_{ij} = \frac{a_{ij} + \sum_{j \in D_i^s} \frac{a_{ik} a_{kj}}{\sum_{m \in C_i} a_{km}}}{a_{ii} + \sum_{n \in D_i^w} a_{in}}. \quad (3.11)$$

Com essas notações, um algoritmo multigrid algébrico pode ser visto como um método multigrid geométrico com as noções de pontos, malha e operadores de transferência substituídas adequadamente.

Considere uma sequência de malhas $\Omega^l \subset \Omega^{l+1}, l = 1, 2, \dots, L$ e um método iterativo $x \leftarrow (I - R_l A_l)x + R_l b_l$ para o sistema $A_l x = b_l$, em cada nível $l \in \{1, 2, \dots, L\}$. Além disso, considere que a hierarquia de malhas é definida pelo operador de Galerkin, $A_{l+1} = (I_{l+1}^l)^T A_l I_{l+1}^l$. Logo, o algoritmo multigrid algébrico é dado por:

ALGORITMO 2: AMG_l

1. *Pré-suavização (Presmoothing):* aplique ν_1 vezes o processo iterativo em $A_l x_l = b_l$ na forma $x_l \leftarrow (I - R_l A_l)x + R_l b_l$;
2. *Correção (Coarse-grid correction):*
 - a. Defina $b_{l+1} = (I_{l+1}^l)^T (b_l - A_l x_l)$
 - b. Se $l + 1 < L$, defina $x_{l+1} = 0$ e resolva o problema da correção da malha grossa $A_{l+1} x_{l+1} = b_{l+1}$, pela aplicação de $x_{l+1} \leftarrow \text{AMG}_{l+1}(x_{l+1}, b_{l+1})$.
Senão, resolva o problema na malha grossa diretamente.
 - c. Faça a correção da solução no nível l : $x_l \leftarrow x_l + I_{l+1}^l x_{l+1}$
3. *Pós-suavização (Postsmoothing):* Aplique ν_2 vezes o processo iterativo em $A_l x_l = b_l$ na forma $x_l \leftarrow (I - R_l A_l)x_l + R_l b_l$.

A aplicação de uma etapa do ALGORITMO 2 é denotada por $\text{AMG}(f^h, v^h, \nu_1, \nu_2)$.

No contexto do meio subterrâneo, o tamanho e a complexidade do domínio computacional são dois fatores-chave para utilização dos métodos multigrid algébricos. As soluções dos sistemas lineares envolvem problemas com milhares ou milhões de variáveis, mesmo sem levar em conta algoritmos adaptativos, heterogeneidades, anisotropia, fluidos com densidade variável e processos químicos e biológicos.

Utilizando esquema de diferenças finitas, [Mehl e Hill \(2001\)](#) comparam a performance

do método algébrico multigrid clássico, proposto por Ruge e Stüben (RUGE; STÜBEN, 1986), com o método do Gradiente Conjugado Pré-Condicionado por Fatoração de Cholesky incompleta (PCG2) para cinco problemas envolvendo o fluxo em meio poroso. Os resultados obtidos, levando-se em conta apenas esforço computacional, são mostrados na tabela [1]:

Tabela 1 – Comparação da performance do método multigrid algébrico clássico, proposto por Ruge-Stüben(R-S), com o método do Gradiente Conjugado Pré-condicionado por fatoração de Cholesky incompleta (PCG2) para cinco problemas envolvendo o meio poroso.

Tipo do problema	Número variáveis	PCG2(s)	AMG(s)	PCG2(s)/AMG(s)
Não linear/estacionário	1 728 000	3082	1781	1.73
Linear/estacionário	1 050 000	5243	205	25.58
Não linear/transiente	147 440	852	631	1.35
Linear/estacionário	465 600	942	50	18.84
Não linear/estacionário	78 817	654	108	6.06

Fonte: (MEHL; HILL, 2001)

As razões entre os tempos gastos pelos métodos PCG2 e AMG mostram que o AMG possui performance superior ao PCG2 para os problemas estudados. Embora os problemas estacionários sejam resolvidos com ganho computacional de até, aproximadamente, 25 vezes, os problemas transientes são resolvidos com ganho computacional de até 1.3 vezes. Segundo Mehl e Hill (2001), a eficiência de ambos os métodos na solução de problemas transientes é devida à iteratividade do processo de simulação, ou seja, a solução no tempo k é utilizada como solução inicial no tempo $k + 1$, e esta solução é uma solução aproximada para o passo $k + 1$.

Mais recentemente, Detwiler et al. (2002) comparam os métodos PCG2 e GMRES pré-condicionado por fatoração ILU, aplicados ao fluxo e transporte em meio poroso. Essa comparação é feita utilizando diferenças finitas e estados estacionários. A análise dos resultados mostrou um ganho computacional de até 140 vezes. No entanto, os problemas envolvendo o transporte de contaminantes o ganho computacional pode atingir 17 vezes.

Stüben, Delaney e Chmakov (2010) comparam um algoritmo AMG modificado (SAMG) com os métodos do Gradiente Conjugado pré-condicionado por fatoração ILU e Gradiente Bi-Conjugado Estabilizado, dependendo do problema ser simétrico ou não. As aplicações incluem os problemas abordados em Mehl e Hill (2001), e os resultados mostram ganho computacional de até duas ordens de grandeza para problemas com até 2.3 milhões de variáveis. Entretanto, mantém-se a preferência por problemas estacionários e a adaptatividade do método multigrid não é considerada.

A próxima seção introduz os métodos multigrid algébricos baseados em agregação e agregação suavizada.

3.4 Método Multigrid Algébrico baseado em agregação

Como no método algébrico multigrid clássico, o método multigrid algébrico baseado em agregação (SA-AMG) baseia seus operadores de transferência em certas hipóteses sobre a natureza dos erros.

Segundo Trottenberg, Oosterlee e Schuller (2001), se a interpolação é definida tal que cada F -variável é interpolada exatamente por uma C -variável, então os métodos baseados em agregação são obtidos. Resumidamente, embora cada F -variável i possa ter mais que uma conexão com o conjunto de C -variáveis, o conjunto de variáveis interpoladoras, P_i^h , é restrito de modo a conter exatamente uma C -variável. Consequentemente, o peso será igual a um se a soma da i -ésima linha da matriz for nula. No entanto, se é assumido que o peso é unitário para qualquer valor da soma das linhas, então o operador de interpolação é simplificado. Consequentemente, o número total de variáveis pode ser dividido em agregados I_k , em que $k \in C$ e I_k contém todos os índices i correspondendo a F -variáveis que são interpoladas pela variável C -variável k .

Geometricamente, um esquema para o método de agregação é apresentado na figura (4). Note que, esquematicamente, os agregados são disjuntos dois a dois e cada F -variável está relacionada com apenas uma C -variável. Portanto, os métodos multigrid algébricos baseados em agregação podem ser vistos como casos limites ou particulares dos métodos multigrid algébricos com partição C/F (TROTTEMBERG; OOSTERLEE; SCHULLER, 2001).

Figura 4 – Divisão das variáveis da malha fina em agregados: C -variáveis e F -variáveis.

Fonte: (TROTTEMBERG; OOSTERLEE; SCHULLER, 2001)

Com essas notações, o operador de Galerkin ou operador de transferência de resultados

para a malha grossa é dado por:

$$I_h^H A_h I_H^h = (a_{kl})^H = \sum_{i \in I_k} \sum_{j \in I_l} a_{ij}^h \quad (k, l \in C). \quad (3.12)$$

O método proposto possui convergência lenta e não é robusto, mas existem duas alternativas para contornar esse problema. A primeira consiste em multiplicar o operador de Galerkin por uma constante $\beta = 1/\alpha$ e a segunda em utilizar o próprio método iterativo para resolver os problemas de convergência (TROTTEMBERG; OOSTERLEE; SCHULLER, 2001). A utilização do método iterativo conduz ao método multigrid algébrico baseado em agregação suavizada (SA-AMG).

3.4.1 AMG baseado em agregação suavizada

No método multigrid algébrico baseado em agregação suavizada (SA-AMG), o operador constante por partes é apenas uma tentativa inicial, a qual é melhorada por algum processo de suavização antes do operador de Galerkin ser computado. Considere, por exemplo, o método de ω -Jacobi para o processo de relaxação e denote $\tilde{I}_H^h$ o operador de interpolação. O operador de transferência do método SA-AMG é dado por (TROTTEMBERG; OOSTERLEE; SCHULLER, 2001):

$$I_H^h = (I_h - \omega D_h^{-1} A_h^f) \tilde{I}_H^h. \quad (3.13)$$

Um algoritmo para o método SA-AMG pode ser formalmente visto como um método multigrid algébrico definido na seção (3.3) com uma escolha especial dos operadores de transferência. O operador de prolongamento é definido como o produto do operador de prolongamento suavizador, S_l , com o operador de prolongamento tentativa P_{l+1}^l , ou seja:

$$I_{l+1}^l = S_l P_{l+1}^l. \quad (3.14)$$

A construção dos operadores P_{l+1}^l está baseada na construção dos agregados. Por sua vez, os agregados A_i^l podem ser formados baseado apenas nas informações de conectividade e força das conexões entre as entradas da matriz A_l , tais que $\cup_{i=1}^N A_i^l = A_l$, $A_i \cap A_j = \{\phi\}$, $i = 1, 2, \dots, N$, conforme definido na partição C/F .

O segundo ingrediente na construção geral de um operador de agregação inicial P_{l+1}^l consiste em fornecer a estrutura de esparsidade derivada da agregação nodal com valores apropriados. Começando com uma matriz inicial, B_1 , cujas colunas representam o espaço nulo aproximado do operador da malha fina, a construção do operador de prolongamento

tentativa é efetuada, e a representação na malha grossa dos componentes do espaço nulo aproximado para satisfazer simultaneamente as equações

$$P_l^{l+1} B^{l+1} = B^l, \quad (P_{l+1}^l)^T P_{l+1}^l = I. \quad (3.15)$$

Dessa forma, o algoritmo para o método de agregação suavizada padrão é dado por:

ALGORITMO 3: Método de Agregação Suavizada Padrão

Dado A_1, B_1, L , para $l = 1, 2, \dots, L - 1$:

- a. Construa os agregados A_i^l , tais que $\cup_{i=1}^l A_i^l = A_l, A_i \cap A_j = \emptyset, i \neq j$ baseado na informação do nível A_l*
- b. Construa os operadores B_{l+1} e P_{l+1} usando as equações acima, baseado na construção dos agregados*
- c. Construa o operador de prolongamento suavizado $:I_{l+1} = S_l P_{l+1}$*
- d. Construa a matriz da malha grossa: $A_{l+1} = (I_{l+1})^T A_l I_{l+1}$*

Segundo [Vaněk et al. \(1995\)](#), este método possui taxa de convergência 10 – 20 vezes menor que o método multigrid algébrico clássico proposto por Ruge e Stüben. Além disso, as características seguintes apontam o método de agregação suavizada como método indicado para problemas envolvendo o transporte de contaminantes em meio poroso:

1. métodos baseados na técnica de agregação suavizada é uma forma de lidar com as dificuldades que surgem do uso de não M-matrizes ([KRAUS; MARGENOV, 2009](#));
2. Segundo [Janka \(2007a\)](#), [Janka \(2007b\)](#)
 - i. esse método é robusto para geometrias alongadas. No caso do transporte de contaminantes em meio poroso, os domínios são grandes e com pouca espessura;
 - ii. pode ser aplicado em problemas com coeficientes com saltos de diversas ordens de grandeza (10 para 10^7);
 - iii. pode ser utilizado em sistemas de equações com $cond(A) \approx 10^9$;
 - iv. é estável com respeito ao tamanho do passo de tempo;

A próxima seção introduz o método multigrid algébrico adaptativo baseado em agregação (α SA-AMG). Esse método conjuga as características do método de agregação suavizada com adaptatividade do próprio método iterativo.

3.5 Método Multigrid Algébrico Adaptativo

Essa seção apresenta os métodos multigrid algébricos adaptativos (α -AMG) seguindo estritamente o ponto de vista das referências (BREZINA et al., 2004), (BREZINA et al., 2005), (BREZINA et al., 2010). Para maiores detalhes envolvendo a construção dos métodos multigrid algébricos adaptativos, as referências devem ser consultadas.

É conveniente ressaltar que o termo adaptativo aqui refere-se ao próprio método multigrid algébrico e diferencia-se do termo adaptativo quando utilizando no contexto da malha de elementos finitos. Em outras palavras, no método de elementos finitos, o termo adaptativo refere-se a um processo executado sobre a malha de elementos finitos e baseado nos estimadores de erros, o qual refina a malha e melhora a solução de acordo com o algoritmo utilizado. Basicamente, um método multigrid algébrico adaptativo refere-se à adaptatividade do método multigrid algébrico, ou seja, um método iterativo que utiliza um algoritmo computacional para construir uma representação adequada dos erros para melhorar a taxa de convergência.

O termo multigrid algébrico significa uma classe de métodos baseados no princípio multigrid que depende pouco ou não depende da informação geométrica do problema, mas em vez disso baseia-se no conceito de suavidade algébrica para determinar um processo efetivo de definição das variáveis da malha grossa. Métodos desse tipo assumem alguma característica da suavidade algébrica, especificando as componentes do erro que não são rapidamente eliminadas pelo processo de relaxação que está sendo usado. No método clássico, por exemplo, todas as componentes variam na direção das conexões fortes da matriz enquanto que no método de agregação suavizado as conexões são representadas localmente por um pequeno conjunto de vetores iniciais fornecidos pelo usuário (BREZINA et al., 2005).

Segundo Brezina et al. (2005), a ideia de empregar o método para melhorá-lo não é nova e foi introduzida por Brandt, McCormick e Ruge. A adaptatividade, como apresentada nas seções prévias, é introduzida por Brezina et al. (2004), por meio do método α SA-AMG. Este método adaptativo considera o esquema de agregação suavizada proposto por Vaněk (VANĚK, 1995). O desenvolvimento e a extensão dos conceitos para os métodos multigrid algébricos adaptativos (α -AMG) foram realizados por Brezina et al. (2005), e a consideração de problemas não-simétricos foi apresentada por Brezina et al. (2010).

As hipóteses da suavidade algébrica e as hipóteses adicionais adotadas possibilitam um tratamento efetivo de muitos problemas, mas também limitam o campo de aplicação. Erros que escapam do processo de relaxação variam drasticamente ao longo das conexões fortes, e muitos outros não oferecem um claro entendimento do que significa conexões fortes (BREZINA et al., 2005).

Enquanto o uso das características de suavidade algébrica parece essencial para

obter solução efetiva dos sistemas lineares, estas hipóteses adicionais limitam as aplicações destes métodos. Em muitos casos importantes, os erros perdidos pelo processo de relaxação podem variar substancialmente ao longo das conexões fortes, e, em muitos casos, até o conceito de conexão forte não é bem entendido. Além disso, o fornecimento de um conjunto representativo completo de protótipos dos componentes suaves não é sempre fácil e nem possível na prática ([BREZINA et al., 2005](#)).

A ideia central do processo adaptativo é a definição do erro algebricamente suave sem suposições adicionais de forma que o método possa determinar as variáveis da malha grossa usando apenas informações do sistema. O método adaptativo está baseado na aplicação do método iterativo linear ao sistema $Ax = 0$ para revelar as componentes do erro que o método não reduz efetivamente. Como as componentes do erro que não são eliminadas pelo processo de suavização são representadas em um espaço de dimensão menor que a dimensão original, então o objetivo da fase inicial do método adaptativo αSA é calcular um conjunto de vetores, B , tal que represente as componentes do erro que a relaxação não é eficiente. Tais componentes, são chamadas de algebricamente suaves ([BREZINA et al., 2004](#)).

4 Metodologia

Este capítulo apresenta a sequência de desenvolvimento deste trabalho. Inicialmente a equação que descreve o transporte de solutos em meio poroso é apresentada. Em seguida, o método de elementos finitos com formulação θ -estável é utilizado para derivar a formulação variacional do problema do transporte de contaminantes em meio poroso saturado. A verificação e validação do código computacional é feita por meio de soluções analíticas conhecidas na literatura, as quais englobam problemas com termos reativos ou não reativos, dados constantes ou não constantes. Finalmente, as estimativas residuais, as técnicas adaptativas para os dados da equação do transporte de contaminantes e os métodos multigrid algébricos com reutilização são introduzidos.

4.1 A equação do transporte de solutos em meio poroso

Água superficial ou subterrânea, em estado natural, nunca é pura. Muitos constituintes estão presentes devido a interação com o ambiente. A qualidade da água depende das diferentes quantidades de substâncias dissolvidas e quantidade de um soluto dissolvido no solo água pode ser expresso através da sua concentração, C , que é a massa de soluto por volume de água, com dimensões $[M/L^3]$, normalmente expressa em mg/l ou g/m^3 equivalente. A adequada quantificação da concentração C em meios subterrâneos é dependente tanto das características do soluto quanto das características do meio subterrâneo (DE SMEDT, 2006).

A água e substâncias dissolvidas se movem lentamente em camadas menos permeáveis tais como argila, mas suas velocidades aumentam significativamente em camadas mais permeáveis tais como areia e cascalho. Além disso, os contaminantes podem ser dispersos por difusão molecular, dispersão mecânica, retardados por troca iônica ou absorvidos pelo material sólido, degradados por reações químicas naturais ou processos bioquímicos (DE SMEDT, 2006). Uma descrição detalhada dos fenômenos envolvidos no transporte de contaminantes em meio poroso subterrâneo pode ser encontrada em (BATU, 2006).

Neste trabalho, a equação de advecção-difusão-reação (ADR) é considerada para descrever o transporte de contaminantes em um meio poroso subterrâneo. Essa equação incorpora os fluxos advectivo e dispersivos associados às transformações químicas de primeira ordem e um termo de fonte ou sorvedouro $f^* \left[\frac{M}{L^3T} \right]$ em aquíferos com fluxo

não-uniforme e é dada por (BEAR, 1979):

$$\partial_t (\theta C + \rho_b K_d C) - \text{div}(D^* \nabla \theta C) + v^* \cdot \nabla C + \lambda(\theta C + \rho_b K_d C) = f^* \quad (4.1)$$

onde C é a concentração de soluto $[M/L^3]$, θ o conteúdo de umidade no solo [adimensional], ∇C o gradiente de concentração $[M/L^4]$, $D^* = D_m + D_d$ o tensor da dispersão hidrodinâmica $[L^2/T]$, v^* vetor de velocidade aparente $[L/T]$, λ o coeficiente de decaimento de 1ª ordem $[T^{-1}]$, ρ_b a densidade volumétrica do meio poroso $[M/L^3]$ e K_d é o coeficiente de distribuição¹.

No caso em que o meio poroso é saturado e o aquífero é confinado, a equação [4.1] pode ser simplificada para

$$\partial_t C - \text{div}(D \nabla C) + v \cdot \nabla C + \lambda C = f \quad (4.2)$$

onde o conteúdo de umidade θ assume o mesmo valor da porosidade efetiva η_{efe} , $D = D^*/R$, $v = v^*/(\eta_{efe}R)$, $f = f^*/(\eta_{efe}R)$.

A adequada modelagem do fenômenos físico requer que condições iniciais e de fronteira sejam impostas às equações diferenciais parciais. A próxima seção considera a equação [4.2] e as condições iniciais e de fronteira, necessárias à obtenção de uma solução numérica ou analítica. Além disso, a formulação do método de elementos finitos e as hipóteses necessárias para obtenção das estimativas residuais são apresentadas.

4.2 Método de Elementos Finitos e Estimativas Residuais

Formalmente, o problema do transporte de contaminantes considera a equação [4.2] e as condições iniciais e de fronteira, e é dado por²:

$$\begin{aligned} \partial_t C - \text{div}(D \nabla C) + v \cdot \nabla C + \lambda C &= f && \text{em } \Omega \times (0, t_{final}] \\ C &= 0 && \text{em } \Gamma_D \times (0, t_{final}] \\ \mathbf{n} \cdot D \nabla C &= g && \text{em } \Gamma_N \times (0, t_{final}] \\ C &= C_0 && \text{em } \Omega \text{ para } t = 0 \end{aligned} \quad (4.3)$$

onde $x = (x_1, x_2, \dots, x_d)$, $d \geq 2$, $\Omega \subset \mathbb{R}^d$, é um domínio poligonal com fronteira de Lipschitz Γ formado de duas partes disjuntas Γ_D , Γ_N , $C = C(x, t)$ é a solução do problema, $D = D(x, t)$ é a matriz de dispersão, $v = v(x, t) = (v_1(x, t), v_2(x, t), \dots, v_d(x, t))$ é o campo de velocidades, $\lambda = \lambda(x, t)$ é uma função que descreve as reações do meio,

1. Ver (BATU, 2006) para uma discussão sobre o coeficiente K_d

2. Esta é a equação adotada por (VERFÜRTH, 2004) para obter as estimativas residuais

$f = f(x, t)$ é a fonte de contaminação, $C = 0$ em $\Gamma_D \times (0, t_{final}]$ é a condição de fronteira de Dirichlet, $n \cdot D \nabla C = g$ em $\Gamma_N \times (0, t_{final}]$ é a condição de fronteira do tipo Neumann e $C_0 = C(x, y) = C(x, y, t = 0)$ é a condição inicial.

4.2.1 Condições adicionais para o estimador residual

A formulação matemática do método de elementos finitos considera a formulação variacional, a qual exige considerações adicionais sobre os dados para garantir a existência e unicidade de solução da formulação variacional (PRAETORIUS; WEINMULLER; WISSGOTT, 2008).

Os dados D , v , λ , f , g e C_0 são funções reais com dependência espacial e temporal. Além disso, as seguintes condições adicionais devem ser satisfeitas, conforme descrito em Verfürth (2004) e Praetorius, Weinmuller e Wissgott (2008):

(P1) A matriz de dispersão³ D é simétrica, uniformemente positiva definida e uniformemente isotrópica. Formalmente,

$$\varepsilon = \inf_{0 < t \leq t_{final}} \inf_{x \in \Omega} \min_{z \in \mathbb{R}^d - \{0\}} \frac{z^T \cdot D(x, t) z}{z^T z} > 0 \quad (4.4)$$

e

$$\kappa = \varepsilon^{-1} \sup_{0 < t \leq t_{final}} \sup_{x \in \Omega} \max_{z \in \mathbb{R}^d - \{0\}} \frac{z^T \cdot D(x, t) z}{z^T z} = O(1) \quad (4.5)$$

A constante ε de (P1) é um limitante inferior para o menor autovalor de D em Ω (PRAETORIUS; WEINMULLER; WISSGOTT, 2008).

(P2) A função de reação $\lambda = \lambda(x, t)$ é uma função escalar e não negativa.

(P3) Existe uma constante β tal que $\lambda - \frac{1}{2} \text{div} v \geq \beta \geq 0$ para quase todo $x \in \Omega$ e $0 \leq t \leq T$. Além disso, existe uma constante $c_\lambda \geq 0$ de tamanho moderado tal que

$$\sup_{0 \leq t \leq T} \sup_{x \in \Omega} |\lambda(x, t)| \leq c_\lambda \beta.$$

(P4) As fronteiras de Dirichlet Γ_D têm medida $(d - 1)$ - dimensional positiva e incluem o fluxo de entrada $\bigcup_{0 < t \leq T} \{x \in \Gamma : v \cdot n(x) < 0\}$.

(P4*) Na fronteira de Neumann Γ_N existe apenas fluxo de saída convectivo, ou seja,

$$\bigcup_{0 < t \leq T} \{x \in \Gamma : v \cdot n(x) > 0\}.$$

3. Verfürth (2014), Verfürth (2004) e Praetorius, Weinmuller e Wissgott (2008) referem-se à difusão. No entanto, no contexto do fluxo subterrâneo o termo mais adequado é dispersão, pois o efeito difusivo é um dos responsáveis pelo transporte de contaminantes (Ver Batu (2006), (BEAR, 1979) para uma discussão detalhada dos termos responsáveis pela dispersão de contaminantes)

4.2.2 Formulação Variacional do Problema parabólico

Para derivar a discretização espaço-temporal da equação (4.3), considere uma função teste $w \in H_D^1(\Omega)$ em que $H_D^1(\Omega)$ denota o subespaço do espaço de Sobolev $H^1(\Omega) = W^{1,2}(\Omega)$, com funções que são nulas nas fronteiras de Dirichlet Γ_D . Multiplicando a equação (4.3) por uma função teste w e usando integração por partes, a forma fraca (4.6) é obtida.

$$\int_{\Omega} (\partial_t C w + \nabla C \cdot D \nabla w + v \cdot \nabla C w + \lambda C w) d\Omega = \int_{\Omega} f w d\Omega + \int_{\Gamma_N} g w dS \quad (4.6)$$

Para a discretização temporal, escolha $N_I > 1$ e defina a partição do tempo por $I = \{t_0 = 0 < t_1 < \dots < t_{N-1} < t_N = t_{final}\}$ com passos de tempo $\tau_n = t_n - t_{n-1}$. Para um tempo arbitrário t_n , denote a solução discreta e os dados por $C^n = C(x, y, t_n)$, $D^n = D(x, y, t_n)$, $v^n = v(x, y, t_n)$, $\lambda^n = \lambda(x, y, t_n)$, $f^n = f(x, y, t_n)$, $g^n = g(x, y, t_n)$. Além disso, defina a triangulação espacial T_n do domínio Ω e o espaço de elementos finitos X_n que consiste de todas as funções polinomiais contínuas por partes com grau no mínimo um⁴.

A derivada temporal é aproximada por diferenças finitas de primeira ordem, ou seja,

$$\partial_t C = \frac{C^n - C^{n-1}}{\tau_n}, \quad (4.7)$$

enquanto que as outras quantidades com dependência temporal são aproximadas pelo θ -esquema, o qual é a combinação convexa dos esquemas de diferenças atrasadas e adiantadas de Euler. Formalmente,

$$\begin{aligned} C^{n,\theta} &:= \theta C^n + (1 - \theta) C^{n-1}, & D^{n,\theta} &:= \theta D^n + (1 - \theta) D^{n-1}, \\ v^{n,\theta} &:= \theta v^n + (1 - \theta) v^{n-1}, & \lambda^{n,\theta} &:= \theta \lambda^n + (1 - \theta) \lambda^{n-1}, \\ f^{n,\theta} &:= \theta f^n + (1 - \theta) f^{n-1}, & g^{n,\theta} &:= \theta g^n + (1 - \theta) g^{n-1}. \end{aligned} \quad (4.8)$$

A equação (4.3), através da formulação variacional, é transformada no problema discreto:

4. No caso em que um algoritmo para a adaptatividade espacial e temporal é empregado, condições adicionais sobre a malha de elementos finitos e o espaço de funções polinomiais devem ser incorporadas. Ver (VERFÜRTH, 2004) para maiores detalhes.

Encontrar $C^n \in X_n, 0 \leq n \leq N_I$, tal que $C^0 = \pi_0 C_0$ e para $n = 1, 2, \dots, N_I$

$$\begin{aligned} \int_{\Omega} \frac{1}{\tau_n} (C^n - C^{n-1}) w_n d\Omega + \int_{\Omega} (\theta \nabla C^n + (1 - \theta) \nabla C^{n-1}) \cdot D_h^{n,\theta} \nabla w_n d\Omega + \\ \int_{\Omega} v_h^{n,\theta} \cdot \nabla (\theta C^n + (1 - \theta) C^{n-1}) w_n d\Omega + \int_{\Omega} \lambda_h^{n,\theta} (\theta C^n + (1 - \theta) C^{n-1}) w_n d\Omega = \\ \int_{\Omega} (f_h^n + (1 - \theta) f_h^{n-1}) w_n d\Omega + \int_{\Gamma_N} (\theta g_h^n + (1 - \theta) g_h^{n-1}) w_n d\Omega, \quad w_n \in X_n \end{aligned} \quad (4.9)$$

onde $D_h^{n,\theta}$, $v_h^{n,\theta}$, $\lambda_h^{n,\theta}$, $f_h^{n,\theta}$ e $g_h^{n,\theta}$ representam os dados no espaço de elementos finitos X_n e malha T_n .

A formulação (4.9) pode ser escrita em uma forma compacta $a(C_n^n, w_n) = l(w_n)$ em que $a(C_n^n, w_n)$ é a forma bilinear (4.10) e $l(w_n)$ é a forma linear (4.11) e dadas por:

$$\begin{aligned} a(C^n, w) = \int_{\Omega} \frac{1}{\tau_n} C^n w_n d\Omega + \int_{\Omega} (\theta \nabla C^n) \cdot D_h^{n,\theta} \nabla w_n d\Omega + \\ \int_{\Omega} v_h^{n,\theta} \cdot \nabla (\theta C^n) w_n d\Omega + \int_{\Omega} \lambda_h^{n,\theta} (\theta C^n) w_n d\Omega - \int_{\Gamma_N} \theta \mathbf{n} \cdot D_h^n \nabla C^n w_n dS \end{aligned} \quad (4.10)$$

$$\begin{aligned} L(w_n) = \int_{\Omega} \frac{1}{\tau_n} C^{n-1} w_n d\Omega + \int_{\Omega} ((\theta - 1) \nabla C^{n-1}) \cdot D_h^{n,\theta} \nabla w_n d\Omega + \\ \int_{\Omega} v_h^{n,\theta} \cdot \nabla ((\theta - 1) C^{n-1}) w_n d\Omega + \int_{\Omega} \lambda_h^{n,\theta} ((\theta - 1) C^{n-1}) w_n d\Omega + \\ \int_{\Omega} (\theta f_h^n + (1 - \theta) f_h^{n-1}) w_n d\Omega + \int_{\Gamma_N} (1 - \theta) \mathbf{n} \cdot D_h^{n-1} \nabla C^{n-1} w_n dS. \end{aligned} \quad (4.11)$$

Essa formulação de elementos finitos possibilita a obtenção das formas bilinear e linear, as quais são essenciais para a implementação do método de elementos finitos seguindo a metodologia FEniCS (FENICS PROJECT, 2014).

A implementação das formas bilinear (4.10) e linear (4.11), geram sistemas lineares da forma $A^n C^n = b^n$, em que n denota tanto o passo de tempo, a malha de elementos finitos e o espaço de elementos finitos no passo n . Desse modo, da solução no $n - 1$ -ésimo passo de tempo, a solução no n -ésimo passo de tempo é calculada. Matematicamente, a solução é computada por $C^n = (A^n)^{-1} b$ mas, computacionalmente, esse procedimento é executado por meio de um método direto ou iterativo.

Um exemplo de código computacional que implementa o transporte de contaminantes, utilizando as formas bilinear e linear, com método direto de solução, é apresentado no apêndice (B.1).

As próximas seções apresentam duas soluções analíticas utilizadas para validação do código computacional para a aproximação de elementos finitos e para o estimador residual.

4.2.3 Solução Analítica para Campo Uniforme

Esta seção considera a solução analítica bidimensional apresentada por Wexler (1992) para campo de velocidades uniforme em domínio infinito e com dados constantes. A solução analítica apresentada requer as seguintes hipóteses adicionais:

1. o aquífero possui extensão infinita, e a fonte não pontual de contaminação tem comprimento finito;
2. a densidade e a viscosidade do fluido são constantes;
3. o contaminante está sujeito a transformação química de primeira ordem. Se o contaminante é conservativo, então $\lambda = 0$;
4. o fluxo uniforme ocorre na direção x com velocidade constante v_x ;
5. os coeficientes de dispersão longitudinal e transversal (D_x, D_y) são constantes.

Além disso, as condições iniciais e de contorno devem ser tais que:

- i. $C = C_0$ para $x = 0$ e $Y_1 < y < Y_2$
- ii. $C = 0$ para $x = 0$ e $y \leq Y_1$ $y \geq Y_2$
- iii. $C = 0, \frac{\partial C}{\partial y} = 0, \quad y = \pm\infty$
- iv. $C = 0, \frac{\partial C}{\partial x} = 0, \quad x = \infty$
- v. $C = 0$ para $t = 0$ e $0 \leq x \leq \pm\infty$

onde Y_1, Y_2 são as coordenadas dos extremos da fonte de contaminantes. Ilustrativamente, a (5-a) representa o campo de velocidades e as condições i., ii. e v., enquanto que a figura (5-b) representa as condições iii. e iv.

(a) O campo de velocidades e as condições i., ii. e v. (b) As condições iii. e iv.

Figura 5 – Representação esquemática das condições iniciais e de contorno i.,ii.,iii., iv. e v.

Neste caso, a solução analítica é dada por:

$$C(x, y, t) = \frac{C_0 x}{\sqrt{\pi D_x}} \exp\left(\frac{x \cdot v_x}{2D_x}\right) \int_0^{t/4} \frac{1}{Z^3} \exp\left[-\left(\frac{V^2}{4D_x} + \lambda\right) Z^4 - \frac{x^2}{4D_x Z^4}\right] \times \\ \left[\operatorname{erfc}\left[\frac{Y_1 - y}{2Z^2 \sqrt{D_y}}\right] - \operatorname{erfc}\left[\frac{Y_2 - y}{2Z^2 \sqrt{D_y}}\right] \right] dZ \quad (4.12)$$

onde $\operatorname{erfc}(z) = 1 - \operatorname{erf}(z)$ é a função erro complementar e $\operatorname{erf}(z)$ é a função erro obtida da distribuição normal padrão.

A solução analítica (4.12) é calculada por intermédio da utilização conjunta das bibliotecas FEniCS (FENICS PROJECT, 2014) e SciPy (JONES et al., 2001). A integral é avaliada pelo pacote de quadratura numérica QUADPACK disponibilizado no SciPy. Para as avaliações envolvendo os valores nodais das concentrações, os resultados analíticos e numéricos são utilizados em forma vetorial ou matricial. As avaliações envolvendo a norma L^2 consideram a interpolação dos resultados analíticos na malha de elementos com funções lagrangeanas da mesma ordem utilizada no métodos dos elementos finitos.

Um exemplo de código computacional que implementa a solução de elementos finitos, a solução analítica, as comparações e interpolação dos resultados analíticos no espaço de elementos finitos é apresentado no apêndice (B.1).

4.2.4 Solução analítica para campo com dependência espacial

Esta seção considera uma solução analítica em que os dados da equação de advecção-difusão-reação são $(v_x, v_y) = (u_0 x, -u_0 y)$, $D_x = D_0 v_0^2 x^2$, $D_y = D_0 v_0^2 y^2$, para $x \geq 0$, $y \geq 0$ e u_0, D_0 constantes.

Segundo Zoppou e Knight (1999) a equação sem termo reativo, em forma conservativa, é dada por:

$$\frac{\partial C}{\partial t} + \frac{\partial}{\partial x}(v_x C) + \frac{\partial}{\partial y}(v_y C) = \frac{\partial}{\partial x}\left(D_x \frac{\partial C}{\partial x}\right) + \frac{\partial}{\partial y}\left(D_y \frac{\partial C}{\partial y}\right) \quad (4.13)$$

onde $\nabla \cdot (v_x, v_y) = 0$ e a fonte de contaminantes possui formato de segmento de reta unitário cuja massa unitária é liberada instantaneamente em (x_0, y_0) , possui solução analítica dada por:

$$C(x, y, t) = \frac{1}{4\pi D_0 u_0^2 t \sqrt{xyx_0y_0}} \left(\frac{xy_0}{x_0y}\right)^{1/2u_0D_0} \times \exp\left(\frac{-\rho^2 - 2(1 + D_0^2 u_0^2)t^2}{4D_0 t}\right), \quad (4.14)$$

onde $\rho = (1.0/u_0) \cdot \sqrt{\log(x/x_0)^2 + \log(y/y_0)^2}$.

A próxima seção apresenta as estimativas residuais, que incorporam a diferenciação dos regimes de advecção, os termos de fonte, as fronteiras, a condição inicial e a estimativa temporal.

4.3 Estimativas Residuais

Na equação (4.3), $\text{div}(D\nabla C)$ é o termo dispersivo⁵, $v \cdot \nabla C$ é o termo advectivo e λC é o termo reativo. A função f é chamada de termo de fonte e representa a entrada ou saída de solutos do domínio computacional Ω .

O efeito combinado dos termos dispersivo, difusivo e reativo determina a distribuição de contaminantes no domínio computacional. No entanto, um dos termos pode determinar o comportamento qualitativo do transporte, e essa predominância é utilizada para classificar o regime de transporte de contaminantes.

4.3.1 Classificação dos Regimes de Advecção

De acordo com a predominância dos termos, o regime de transporte de contaminantes é classificado em:

1. **Dispersão dominante:** Se $|v|_{L^\infty(0,t_{final};W^{1,\infty}(\Omega))} \leq c_c \varepsilon$ e $\beta \leq c_r \varepsilon$ com constantes de ordem 1;

Neste caso o termo $\text{div}(D\nabla C)$ domina o regime de transporte. Fisicamente, isso significa que o transporte é realizado da região com alta concentração para a região com baixa concentração até que o equilíbrio ocorra. A matriz de coeficientes D é proporcional à dispersividade, ou seja, determina a velocidade e direção do processo (PRAETORIUS; WEINMULLER; WISSGOTT, 2008).

2. **Reação dominante:** O termo λC domina se $|v|_{L^\infty(0,t_{final};W^{1,\infty}(\Omega))} \leq c_c \varepsilon$ e $\beta \gg c_r \varepsilon$ com constantes de ordem 1. O nome deste termo é relacionado com as reações químicas de primeira ordem.⁶

3. **Advecção dominante:** Se $|v|_{L^\infty(0,t_{final};W^{1,\infty}(\Omega))} \gg \varepsilon$, o termo $v \cdot \nabla C$ determina o comportamento qualitativo da solução de (4.3)⁷ (PRAETORIUS; WEINMULLER; WISSGOTT, 2008).

5. D engloba os efeitos de difusão molecular e dispersão hidrodinâmica conforme apresentado em Batu (2006), ou seja, $D = D_0 + D_m$

6. Uma reação de primeira ordem tem a taxa proporcional à concentração de um dos reagentes (PRAETORIUS; WEINMULLER; WISSGOTT, 2008)

7. A convecção ou advecção é a transferência forçada de concentração ou energia. O campo vetorial v define direção, sentido e magnitude do processo

A classificação do regime é baseada apenas nas relações entre os termos advectivos, dispersivos e reativos e, portanto, não é dependente do processo de aproximação da equação diferencial parcial. Por outro lado a classificação baseada no número de Peclet e Courant é dependente da partição do domínio computacional e da partição temporal. [Firmiano \(2010\)](#) propõe um intervalo para a constante característica c_c baseado nos números de Peclet e Courant. Uma generalização dos resultados obtidos por [Firmiano \(2010\)](#) é apresentada na seção (5.1.1).

4.3.2 Contribuições residuais

Esta seção apresenta o estimador residual preliminar em sua forma mais geral apresentada por [Verfürth \(2004\)](#). Isso fornece uma visão geral da construção do estimador residual e possibilita esclarecer algumas considerações adotadas na tese. Além disso, essa apresentação facilita a introdução e justifica a utilização do estimador residual, apresentado por [Praetorius, Weinmuller e Wissgott \(2008\)](#), nos algoritmos adaptativos para os dados do problema do transporte.

Basicamente o estimador residual preliminar engloba os resíduos, medidos em norma conveniente, da aproximação de cada uma das entidades presentes na aproximação da equação ADR.

A quantidade residual que avalia a qualidade da solução numérica em um elemento específico da malha de elementos finitos é chamada residual do elemento. Para todo elemento K , o residual do elemento R_K é dado por:

$$R_K = f_I - \frac{1}{\tau_n} (C^n - C^{n-1}) + \text{div}(D_h^{n,\theta} \nabla(\theta C^n + (1 - \theta)C^{n-1})) - v_h^{n,\theta} \cdot \nabla(\theta C^n + (1 - \theta)C^{n-1}) - \lambda_h^{n,\theta}(\theta C^n + (1 - \theta)C^{n-1}), \quad (4.15)$$

onde as funções f_I, g_I são as projeções L^2 da função f , discretizadas pelo θ -esquema, sobre os espaços de elementos finitos X_n . Matematicamente,

$$f_I(\cdot, t) = \pi_n(\theta f(\cdot, t_n) + (1 - \theta)f(\cdot, t_{n-1})) = \pi_n(\theta f^n + (1 - \theta)f^{n-1}) \quad (4.16)$$

$$g_I(\cdot, t) = \pi_n(\theta g(\cdot, t_n) + (1 - \theta)g(\cdot, t_{n-1})) = \pi_n(\theta g^n + (1 - \theta)g^{n-1}) \quad (4.17)$$

para todo $1 \leq n \leq N_I$.

Outra quantidade importante é o residual lateral, o qual avalia a variação do divergente de um elemento para outro e fornece tanto uma aproximação para conservação do fluxo para elementos vizinhos como elementos com face ou aresta na fronteira de Neumann ([PRAETORIUS; WEINMULLER; WISSGOTT, 2008](#)). Essa quantidade residual deve ser calculada para todas as arestas E da malha de elementos finitos e é dada por:

$$R_E = \begin{cases} \left[\mathbf{n} \cdot D_h^{n,\theta} \nabla (\theta C^n + (1-\theta)C^{n-1}) \right]_E & \text{se } E \not\subset \Gamma \\ g_I - \mathbf{n} \cdot D_h^{n,\theta} \nabla (\theta C^n + (1-\theta)C^{n-1}) & \text{se } E \subset \Gamma_N \\ 0 & \text{se } E \subset \Gamma_D \end{cases} \quad (4.18)$$

onde $[\cdot]_E$ representa o operador de salto.

Operador de Salto: Seja $d = 2$, K_+ e K_- dois elementos vizinhos que compartilham uma mesma aresta $E = K_+ \cap K_-$, como mostrado na figura (6). Considere q em $H^1(K_+) \cap H^1(K_-)$, então existem as restrições $q|_{\partial K_+}$ e $q|_{\partial K_-}$ e o operador de salto de q de K_+ para K_- na direção do vetor normal $n = n_{K_+}$

$$[q]_E = \{q|_{\partial K_+} - q|_{\partial K_-}\}|_E \quad (4.19)$$

Figura 6 – Dois elementos vizinhos que compartilham uma aresta e a representação do salto da aresta de K_+ para K_- (PRAETORIUS; WEINMULLER; WISSGOTT, 2008)

Fonte: (PRAETORIUS; WEINMULLER; WISSGOTT, 2008)

Se $d = 3$, então o operador de salto deve ser definido para as faces e arestas dos elementos.

Da mesma forma que a representação da solução numérica ocasiona as contribuições residuais R_K e R_E , a representação dos dados também ocasiona as contribuições residuais dos dados D_K e D_E . Essas contribuições serão denominadas contribuições dos dados para os elementos e para as laterais, respectivamente, e dados por:

$$D_K = -div \left\{ \left(D_h^{n,\theta} - D^{n,\theta} \right) \nabla \left(\theta C^n + (1-\theta)C^{n-1} \right) \right\} + \left(v_h^{n,\theta} - v^{n,\theta} \right) \cdot \nabla \left(\theta C^n + (1-\theta)C^{n-1} \right) - \left(\lambda_h^{n,\theta} - \lambda^{n,\theta} \right) \left(\theta C^n + (1-\theta)C^{n-1} \right), \quad (4.20)$$

$$D_E = \begin{cases} \left[\mathbf{n} \cdot (D_h^{n,\theta} - D^{n,\theta}) \nabla (\theta C^n + (1 - \theta)C^{n-1}) \right]_E & \text{se } E \notin \Gamma \\ \mathbf{n} \cdot (D_h^{n,\theta} - D^{n,\theta}) \nabla (\theta C^n + (1 - \theta)C^{n-1}) & \text{se } E \subset \Gamma_N \\ 0 & \text{se } E \subset \Gamma_D, \end{cases} \quad (4.21)$$

onde $D_h^{n,\theta}, v_h^{n,\theta}, \lambda_h^{n,\theta}$ denotam a aproximação de elementos finitos, enquanto que $D^{n,\theta}, v^{n,\theta}, \lambda^{n,\theta}$ denotam a discretização temporal.

Nesse momento, algumas observações são necessárias:

1. Considere que $X = D, \lambda$ ou $X = v$. Logo, $X_h^{n,\theta}$ é a representação espacial da discretização temporal $X^{n,\theta}$, enquanto que $X^{n,\theta} = \theta X(\cdot, t_n) + (1 - \theta)X(\cdot, t_{n-1})$ é a representação analítica da discretização temporal.
2. Grosseiramente, a diferença $X_h^{n,\theta} - X^{n,\theta}$ avalia a qualidade da representação espacial do esquema de discretização temporal. Se $X^{n,\theta}$ é representado exatamente, então a diferença é nula e, conseqüentemente, o erro de representação associado é nulo.

4.3.3 Estimativas Residuais a posteriori

Nesta seção o estimador residual preliminar é apresentado. Esse estimador incorpora as quantidades residuais R_K, R_E, D_K, D_E , a norma da energia $||| \cdot |||$ e a norma dual $||| \cdot |||_*$ ⁸, a qual necessita de hipóteses adicionais sobre o regime de advecção para obtenção das estimativas residuais robustas. No entanto, a adequada apresentação do estimador residual requer algumas definições adicionais.

1. A norma da energia $||| \cdot |||$ é dada por:

$$|||C^m - C^{m-1}||| = \left\{ \varepsilon ||\nabla(C^m - C^{m-1})||_{L^2(\Omega)} + \beta ||C^m - C^{m-1}||_{L^2(\Omega)} \right\}^{1/2}. \quad (4.22)$$

2. Fator de ponderação α_S : Para cada elemento na malha de elementos finitos, defina o fator de ponderação α_S por

$$\alpha_S := \min\{h_S \epsilon^{-1/2}, \beta^{-1/2}\} \quad (4.23)$$

onde $S \in \{K, E\}$ é um parâmetro local de um elemento ou uma face ou aresta, respectivamente, ϵ e β são os parâmetros definidos nas condições impostas sobre os dados do problema e h_S é o diâmetro do elemento.

8. Ver [Verfürth \(2004\)](#)

3. Estimador Espacial $((\eta^{(n)})^2)^9$ e Estimador dos Dados $((\Theta^{(n)})^2)$: Para todo $1 \leq n \leq N_I$ e todo elemento N , as quantidades $((\eta^{(n)})^2)$ e $((\Theta^{(n)})^2)$ são dadas por¹⁰:

$$\left(\eta^{(n)}\right)^2 = \sum_K \alpha_K^2 \|R_K\|_{L^2(K)}^2 + \sum_E \varepsilon^{-\frac{1}{2}} \alpha_E \|R_E\|_{L^2(E)}^2, \quad (4.24)$$

$$\left(\Theta^{(n)}\right)^2 = \sum_K \alpha_K^2 \|D_K\|_{L^2(K)}^2 + \sum_E \varepsilon^{-\frac{1}{2}} \alpha_E \|D_E\|_{L^2(E)}^2. \quad (4.25)$$

Segundo [Verfürth \(2004\)](#), considerando que as condições $P1 - P4$ são satisfeitas, o erro da solução do problema variacional (4.6) e do problema discreto (4.9) é limitado superiormente por:

$$\begin{aligned} \|u - u_{h,\tau}\|_{X(0,T)} \leq c^* \left\{ \|C_0 - \pi C_0\|_{L^2(\Omega)}^2 + \right. \\ \left. \sum_{i=1}^{N_I} \tau_n \left[\left(\eta^{(n)}\right)^2 + \left(\Theta^{(n)}\right)^2 + \|C^m - C^{m-1}\|^2 + \|v \cdot \nabla (C^m - C^{m-1})\|_*^2 \right] + \right. \\ \left. \|f - f_{h,\tau}\|_{L^2(0,T,H^{-1}(\Omega))}^2 + \|g - g_{h,\tau}\|_{L^2(0,T,H^{-1}(\Gamma_N))}^2 \right\}^{1/2} \quad (4.26) \end{aligned}$$

e em cada intervalo $(t_{n-1}, t_n]$, $1 \leq n \leq N$, inferiormente por

$$\begin{aligned} \tau_n^{1/2} \left\{ \left(\eta^{(n)}\right)^2 + \|C^m - C^{m-1}\| + \|v \cdot \nabla (C^m - C^{m-1})\|_*^2 \right\}^{1/2} \leq \\ c_* \left\{ \|C - C_{h,\tau}\|_{X(0,T)}^2 + \tau_n \left(\Theta^{(n)}\right)^2 + \|f - f_{h,\tau}\|_{L^2(0,T,H^{-1}(\Omega))}^2 + \|g - g_{h,\tau}\|_{L^2(0,T,H^{-1}(\Gamma_N))}^2 \right\}^{1/2}. \end{aligned}$$

A constante c^* depende das razões h_K/ρ_K e h'_K/ρ_K , enquanto a constante c_* depende das razão h_K/ρ_K , do máximo grau das funções nos elementos finitos e da constante κ . Todas as constantes são independentes do tempo final e dos parâmetros $(\lambda, |c|)$ e r no caso constante).

Segundo [Praetorius, Weinmuller e Wissgott \(2008\)](#), se as condições (P1)-(P4*) são satisfeitas, então o limite superior do estimador residual é dado por:

$$\begin{aligned} \|u - u_{h,\tau}\|_{X(0,T)} \leq c^* \left\{ \|C_0 - \pi C_0\|_{L^2(\Omega)}^2 + \right. \\ \left. \sum_{i=1}^{N_I} \tau_n \left[\left(\eta^{(n)}\right)^2 + \left(\Theta^{(n)}\right)^2 + \left(\eta_f^{(n)}\right)^2 + \left(\eta_N^{(n)}\right)^2 + \left(\eta_\tau^{(n)}\right)^2 \right] \right\}^{1/2} \quad (4.27) \end{aligned}$$

9. [Praetorius, Weinmuller e Wissgott \(2008\)](#) denota essa quantidade por Erro de Verfürth tipo I

10. [Verfürth \(2004\)](#) incorpora condições adicionais devido ao processo adaptativo espacial-temporal

onde $(\eta_f^{(n)})$, $(\eta_N^{(n)})$ e $(\eta_\tau^{(n)})^2$ são as contribuições do termo de fonte, das fronteiras de Neumann e contribuição temporal, respectivamente, e dadas por:

$$(\eta_f^{(n)})^2 = \sum_K \alpha_K \left[\theta^2 \|f(\cdot, t_n) - f_h^{(n)}\|_K^2 + (1 - \theta)^2 \|f(\cdot, t_{n-1}) - f_h^{(n-1)}\|_K^2 \right] \quad (4.28)$$

$$(\eta_N^{(n)})^2 = \varepsilon^{-1/2} \sum_E \alpha_E \left[\theta^2 \|g(\cdot, t_n) - g_h^{(n)}\|_K^2 + (1 - \theta)^2 \|g(\cdot, t_{n-1}) - g_h^{(n-1)}\|_K^2 \right] \quad (4.29)$$

$$\begin{aligned} (\eta_\tau^{(n)})^2 = & |||C^n - C^{n-1}|||^2 + |||v \cdot \nabla(C^n - C^{n-1})|||_* + \\ & (1 + \theta)^2 \tau_n \left[\int_{t_{n-1}}^{t_n} \|\dot{f}(t)\|^2 dt + \int_{t_{n-1}}^{t_n} \|\dot{g}(t)_{\Gamma_N}\|^2 dt + \varepsilon^{-1} \int_{t_{n-1}}^{t_n} \|\dot{D}(t) \cdot \nabla u_h\|^2 dt + \right. \\ & \left. \int_{t_{n-1}}^{t_n} \|\dot{c} \cdot \nabla u_h\|^2 dt + \int_{t_{n-1}}^{t_n} \|\dot{r}(t) u_h(\bar{t})\|^2 dt \right] \end{aligned} \quad (4.30)$$

A quantidade (4.26) diferencia-se de (4.27) pela existência das estimativas dos erros de aproximação das funções f , g e pelas estimativas temporais adicionais. Além disso, o estimador de Praetorius, Weinmuller e Wissgott (2008) não possui um limite inferior para o caso geral. No entanto, se ambas as condições (P4) e (P4*) são satisfeitas, então a estimativa do limite superior de Praetorius, Weinmuller e Wissgott (2008) pode ser usada no estimador de Verfürth (2004)¹¹.

O estimador residual preliminar não é computável, pois envolve o termo $||| \cdot |||_*$. Os próximos resultados limitam o termo $||| \cdot |||_*$ por quantidades computáveis para os casos de dispersão dominante e advecção dominante.

4.3.4 Estimativas Residuais Robustas

A diferenciação dos regimes está ligada à forma de obtenção das estimativas residuais. Se o regime é caracterizado pela dispersão dominante, então estimativas diretas podem ser obtidas. No entanto, se o regime é caracterizado pela advecção dominante, então é necessário resolver um problema auxiliar em cada passo de tempo para obter estimativas residuais robustas da equação de advecção-difusão-reação.

Se o máximo valor do campo de velocidades é relativamente pequeno comparado ao fator $\varepsilon^{1/2} \max\{\epsilon, \beta\}^{1/2}$, então a norma $||| \cdot |||_*$ fornece uma contribuição pequena (PRAETORIUS; WEINMULLER; WISSGOTT, 2008). Formalmente,

11. Análogamente para os limites inferiores

1. Se $\|v(\mathbf{x}, t)\|_{L^\infty(0,T,L^\infty(\Omega)^2)} \lesssim C_c \varepsilon^{\frac{1}{2}} \max\{\varepsilon, \beta\}^{\frac{1}{2}}$ para uma constante C_c de tamanho moderado, o regime é de dispersão dominante. Neste caso, existe uma constante c^* tal que

$$\| \mathbf{v} \nabla (C^n - C^{n-1}) \|_* \leq \max\{1, c_\Omega\} c_c \|C^n - C^{n-1}\|, \quad (4.31)$$

onde c_Ω é a constante da desigualdade de Poincaré (VERFÜRTH, 2004).

2. Se o máximo valor do campo de velocidades é estritamente maior que a dispersão, então o regime é caracterizado pela advecção dominante. Formalmente, o regime é de advecção dominante se

$$\|v(\mathbf{x}, t)\|_{L^\infty(0,T,L^\infty(\Omega)^2)} \gg \varepsilon^{\frac{1}{2}} \max\{\varepsilon, \beta\}^{\frac{1}{2}}. \quad (4.32)$$

Nesse caso, a inequação (4.31) não é indicada para a estimativa da norma dual. O procedimento para limitação da norma dual é baseado na solução de um problema estacionário de reação-difusão, o qual é o caso estacionário da equação do transporte com campo de velocidades nulo (VERFÜRTH, 2004).

Para as estimativas adicionais necessárias ao regime de advecção dominante, considere-se, para todo $1 \leq n \leq N_I$, $\tilde{X}_n^{1,0}(\tilde{T}_n)$ denote o espaço de funções lineares e contínuas por partes correspondentes à partição $\tilde{T}_n$ e tais que são nulas na fronteira de Dirichlet. Adicionalmente, considere que $\tilde{C}^n$ é a única solução do problema discreto de reação-difusão, dado por:

$$\varepsilon \int_{\Omega} \nabla \tilde{C}^n \cdot \nabla w d\Omega + \beta \int_{\Omega} \tilde{C}^n w d\Omega = \int_{\Omega} \mathbf{v}^n \cdot \nabla (C^n - C^{n-1}) w d\Omega \quad (4.33)$$

para todo $\omega \in \tilde{X}_n^{1,0}$.

Com essas notações, o estimador residual do problema auxiliar é dado por:

$$\begin{aligned} \left(\tilde{\eta}^{(n)}\right)^2 = \sum_K \alpha_K^2 \left\| \mathbf{v}^n \cdot \nabla (C^n - C^{n-1}) - \beta \tilde{C}^n \right\|_{L^2(K)}^2 + \\ \sum_E \varepsilon^{-\frac{1}{2}} \alpha_E \left\| [\mathbf{n} \cdot \nabla \tilde{C}^n]_E \right\|_{L^2(E)}^2 \end{aligned} \quad (4.34)$$

Logo, existem constantes $\tilde{c}_*$ e $\tilde{c}^*$, as quais dependem somente da razão h_K/ρ_K tal que:

$$\tilde{c}_* \left\{ \left\| \tilde{C}_{T_n}^n \right\| + \left(\tilde{\eta}^{(n)}\right) \right\} \leq \left\| c \cdot \nabla (C_{T_n}^n - C_{T_{n-1}}^{n-1}) \right\|_* \leq \tilde{c}^* \left\{ \left\| \tilde{C}_{T_n}^n \right\| + \left(\tilde{\eta}^{(n)}\right) \right\} \quad (4.35)$$

4.3.5 Estimativas Residuais Gerais

Conforme apresentado, o estimador residual envolve diferentes termos. No entanto, independentemente do regime de advecção, o estimador residual [Verfürth \(2004\)](#) pode ser escrito como:

$$(\eta_I) = \left\{ \|C_0 - \pi C_0\|_{L^2(\Omega)}^2 + \sum_{i=1}^{N_I} \tau_n \left[\left(\eta^{(n)}\right)^2 + \left(\Theta^{(n)}\right)^2 + \left(\eta_\tau^{(n)}\right)^2 \right] \right\}^{1/2}, \quad (4.36)$$

onde $\left(\eta_\tau^{(n)}\right)^2 = |||C^n - C^{n-1}|||^2$ para o regime de dispersão dominante e $\left(\eta_\tau^{(n)}\right)^2 = |||C^n - C^{n-1}|||^2 + \left(\tilde{\eta}^{(n)}\right)^2 + |||\tilde{C}^n|||^2$ para o regime de advecção dominante.

Da mesma forma o estimador residual de [Praetorius, Weinmuller e Wissgott \(2008\)](#) para os regimes de dispersão dominante ou advecção dominante é dado por:

$$(\eta_I) = \left\{ \|C_0 - \pi C_0\|_{L^2(\Omega)}^2 + \sum_{i=1}^{N_I} \tau_n \left[\left(\eta^{(n)}\right)^2 + \left(\Theta^{(n)}\right)^2 + \left(\eta_f^{(n)}\right)^2 + \left(\eta_N^{(n)}\right)^2 + \left(\eta_\tau^{(n)}\right)^2 \right] \right\}^{1/2} \quad (4.37)$$

4.3.6 Discussão Adicional sobre as Estimativas Residuais

As estimativas residuais para os regimes de dispersão dominante e advecção dominante, que incorporam as estimativas dos erros de representação da condição inicial, do termo de fonte, das fronteiras de Neumann e as estimativas temporais são consideradas para fornecer uma avaliação da qualidade da solução numérica obtida. Essas estimativas consideram as diferentes hipóteses sobre os dados, mas podem ser colocadas em uma única forma fechada dada por:

$$(\eta_I) = \left\{ \|C_0 - \pi C_0\|_{L^2(\Omega)}^2 + \sum_{i=1}^{N_I} \tau_n \left[\left(\eta^{(n)}\right)^2 + \left(\Theta^{(n)}\right)^2 + \left(\eta_f^{(n)}\right)^2 + \left(\eta_N^{(n)}\right)^2 + \left(\eta_\tau^{(n)}\right)^2 \right] \right\}^{1/2}, \quad (4.38)$$

onde $\left(\eta_f^{(n)}\right)^2 = \left(\eta_N^{(n)}\right)^2 = 0$ para o estimador residual de [Verfürth](#). Caso, as hipóteses para o estimador residual de [Praetorius, Weinmuller e Wissgott \(2008\)](#) sejam satisfeitas, então as quantidades $\left(\eta_f^{(n)}\right)^2$ e $\left(\eta_N^{(n)}\right)^2$ devem ser consideradas.

Considerando o estimador residual de [Verfürth \(2004\)](#), dados constantes e regime de dispersão dominante, a equação (4.38) é dada por:

$$(\eta_I) = \left\{ \sum_{i=1}^{N_I} \tau_n \left[(\eta^{(n)})^2 + (\eta_\tau^{(n)})^2 \right] \right\}^{1/2} = \left\{ \sum_{i=1}^{N_I} \tau_n \left[(\eta^{(n)})^2 + |||C^m - C^{m-1}|||^2 \right] \right\}^{1/2}. \quad (4.39)$$

Por sua vez, o estimador dado por (4.39) é mais acurado que o estimador proposto por Firmiano (2010) no sentido que a estimativa espacial incorpora todos os termos. Formalmente, as estimativas aqui implementadas são tais que:

$$\left\{ \sum_K \alpha_K^2 ||R_K||_{L^2(K)}^2 + \sum_E \varepsilon^{-\frac{1}{2}} \alpha_E ||R_E||_{L^2(E)}^2 \right\}^{1/2} > \left\{ \sum_K \alpha_K^2 ||R_K||_{L^2(K)}^2 \right\}^{1/2} \quad (4.40)$$

em que o lado direito é o estimador proposto por Firmiano (2010).

Por sua vez, o estimador de Praetorius, Weinmuller e Wissgott (2008) é mais geral que o estimador de Verfürth (2004) no sentido que incorpora estimativas adicionais para o termo de fonte, a fronteira de Neumann e o tempo. Essas estimativas adicionais são incorporadas por meio de uma conveniente partição das estimativas residuais.

Considere agora que os dados da equação possuem apenas dependência espacial e que a função g seja definida implicitamente pela forma bilinear. Logo, $(\eta_N)^n = 0$ e

$$\eta_\tau = |||C^m - C^{m-1}||| + |||v \cdot \nabla(C^m - C^{m-1})|||_*. \quad (4.41)$$

A primeira igualdade é devida à representação da condição de Neumann no espaço de elementos finitos¹² enquanto que a segunda é devida à dependência apenas espacial dos dados.

A próxima seção considera apenas o estimador residual de Verfürth (2004) para os regimes dispersão dominante e advecção dominante. O estimador residual de Praetorius, Weinmuller e Wissgott (2008) difere apenas pela presença das estimativas η_f , η_N e a estimativa temporal.

4.3.6.1 Partição do Estimador Residual

A metodologia aqui apresentada é baseada em Praetorius, Weinmuller e Wissgott (2008) enquanto que os códigos são baseados em Fenics Project (2014).

Usando as ferramentas disponíveis, a aproximação em elementos finitos para a equação do transporte de contaminantes é implementada diretamente através das formas bilinear e linear. A solução do sistema linear $Ax = b$ é calculada numericamente através de

12. Como a condição de Neumann é definida implicitamente, então é representada no espaço de elementos finitos. Logo, não existem erros associados à representação.

algum método iterativo ou direto. Em seguida, o estimador espacial pode ser calculado¹³ através da somatória de todas as contribuições dos elementos, saltos e fronteiras, ou seja:

$$\left(\eta^{(n)}\right)^2 = \underbrace{\sum_K \alpha_K^2 \|R_K\|_{L^2(K)}^2}_{\eta_K^n} + \underbrace{\sum_{E \notin \Gamma} \varepsilon^{-\frac{1}{2}} \alpha_E \|R_E\|_{L^2(E)}^2}_{\eta_E^n} + \underbrace{\sum_{E \subseteq \Gamma_N} \varepsilon^{-\frac{1}{2}} \alpha_E \|R_E\|_{L^2(E)}^2}_{\eta_B^n} \quad (4.42)$$

onde as componentes η_K^n , η_E^n , η_B^n serão chamadas, respectivamente: contribuições dos elementos, saltos, fronteiras para o n -ésimo passo de tempo. Analogamente, o residual dos dados pode ser calculado por:

$$\left(\Theta^{(n)}\right)^2 = \underbrace{\sum_K \alpha_K^2 \|D_K\|_{L^2(K)}^2}_{D_K^n} + \underbrace{\sum_E \varepsilon^{-\frac{1}{2}} \alpha_E \|D_E\|_{L^2(E)}^2}_{D_E^n}, \quad (4.43)$$

onde D_K^n e D_E^n são as contribuições dos elementos e dos saltos para os dados, respectivamente. Em cada passo de tempo, a contribuição residual total é composta das contribuições espaciais e das contribuições temporais. Logo, a contribuição em cada passo pode ser reescrita como

$$\left(E_{step}^n\right)^2 = \left(\eta^{(n)}\right)^2 + \left(\Theta^{(n)}\right)^2 + \left(\eta_\tau^{(n)}\right)^2. \quad (4.44)$$

Com as notações acima, o resíduo pode ser reescrito como:

$$\hat{\eta}_I = \left\{ \|C_0 - \pi_0 C_0\|_{L(\Omega)}^2 + \sum_{n=1}^{N_I} \tau_n (E_{step}^n)^2 \right\}^{\frac{1}{2}}. \quad (4.45)$$

No caso de advecção dominante, o problema auxiliar é resolvido utilizando a mesma metodologia aplicada ao problema do transporte. Nesse caso, as formas para os cálculos de $(\eta_{\tau_n}^n)^2$ e para o problema auxiliar precisam ser somadas.

A partição das estimativas residuais permite que as contribuições sejam comparadas e esses resultados favorecem o entendimento do comportamento das componentes do estimador residual. As quantidades $R_{Jump/Estep}^n$, $R_{Jump/Element}^n$, $(\eta_\tau^n)^{(2)}$ definidas por:

$$R_{Jump/Element}^n = \frac{\sum_{E \notin \Gamma} \varepsilon^{-\frac{1}{2}} \alpha_E \|R_E\|_{L^2(E)}^2}{\sum_K \alpha_K^2 \|R_K\|_{L^2(K)}^2}, \quad R_{Jump/Estep}^n = \frac{\sum_{E \notin \Gamma} \varepsilon^{-\frac{1}{2}} \alpha_E \|R_E\|_{L^2(E)}^2}{\left(E_{step}^n\right)^2}, \quad (4.46)$$

13. Aqui o autor refere-se à implementação do código computacional

forneem relações entre as componentes residuais. A relação $R_{Jump/Element}^n$ compara a magnitude dos saltos e dos elementos, a relação $R_{Jump/Step}^n$ compara as contribuições dos saltos com as contribuições obtidas em cada passo enquanto que η_τ^n expressa o resíduo temporal. A utilização destas relações para comparar as magnitudes das estimativas residuais são apresentadas em Santos, Firmiano e Wendland (2014), Santos, Firmiano e Wendland (b), Santos, Firmiano e Wendland (a). Outros resultados são apresentados em Firmiano et al. (2013), Firmiano, Santos e Wendland (2013), Firmiano et al. (2012), Firmiano, Wendland e Santos (2012).

4.3.7 Contribuições Residuais de cada elemento da malha

O estimador residual apresentado é composto de quantidades locais que, quando somadas, produzem a quantidade global(sobre toda a malha). No entanto, as quantidades locais podem ser definidas e calculadas da mesma forma que as quantidades globais. Essas quantidades locais são importantes para a formulação de estratégias adaptativas.

Da mesma forma que

$$\left(\eta^{(n)}\right)^2 = \underbrace{\sum_K \alpha_K^2 \|R_K\|_{L^2(K)}^2}_{\eta_K^n} + \underbrace{\sum_{E \notin \Gamma} \varepsilon^{-\frac{1}{2}} \alpha_E \|R_E\|_{L^2(E)}^2}_{\eta_E^n} + \underbrace{\sum_{E \subseteq \Gamma_N} \varepsilon^{-\frac{1}{2}} \alpha_E \|R_E\|_{L^2(E)}^2}_{\eta_B^n} \quad (4.47)$$

define-se a contribuição local para o elemento K , η_K , como a soma das contribuições do elemento, do salto e fronteira em cada elemento. Formalmente, para um dado elemento K , a contribuição local é dada por:

$$\begin{aligned} \eta_K^2 &= \underbrace{\alpha_K^2 \|R_K\|_{L^2(K)}^2}_K + \underbrace{\varepsilon^{-\frac{1}{2}} \alpha_E \|R_E\|_{L^2(E)}^2}_{E \notin \Gamma} + \underbrace{\varepsilon^{-\frac{1}{2}} \alpha_E \|R_E\|_{L^2(E)}^2}_{E \subseteq \Gamma_N} \\ &:= \eta_{1K}^2 + \eta_{2E}^2 + \eta_{3B}^2 \end{aligned} \quad (4.48)$$

onde $\eta_{1K}^2 = \alpha_K^2 \|R_K\|_{L^2(K)}^2$, $\eta_{2E}^2 = \varepsilon^{-\frac{1}{2}} \alpha_E \|R_E\|_{L^2(E)}^2$ e $\eta_{3B}^2 = \varepsilon^{-\frac{1}{2}} \alpha_E \|R_E\|_{L^2(E)}^2$. Os valores η_K^2 , η_{1K}^2 , η_{2E}^2 , η_{3B}^2 serão chamados contribuições do elemento, do salto e da fronteira, respectivamente, para o elemento K . Essas quantidades possibilitam o cálculo das contribuições máximas:

i. A contribuição máxima dos elementos $\eta_{1K,max}^2$:

$$\eta_{1K,max}^2 = \max_K \eta_{1K}^2 \quad (4.49)$$

ii. A contribuição máxima dos saltos $\eta_{2E,max}^2$:

$$\eta_{2E,max}^2 = \max_{E \notin \Gamma_N} \eta_{2E}^2 \quad (4.50)$$

iii. A contribuição máxima das fronteiras $\eta_{3B,max}^2$:

$$\eta_{3B,max}^2 = \max_{E \subset \Gamma_N} \eta_{3B}^2 \quad (4.51)$$

iv. A contribuição máxima espacial $\eta_{N,max}^2$:

$$\eta_{K,max}^2 = \max_K \eta_K^2 \quad (4.52)$$

Analogamente, considere

$$\left(\Theta^{(n)}\right)^2 = \underbrace{\sum_K \alpha_K^2 \|D_K\|_{L^2(K)}^2}_{D_K^n} + \underbrace{\sum_E \varepsilon^{-\frac{1}{2}} \alpha_E \|D_E\|_{L^2(E)}^2}_{D_E^n}. \quad (4.53)$$

Logo, as quantidades locais referentes aos residuais dos dados podem ser calculados por:

$$\Theta_K^2 = \alpha_K^2 \|D_K\|_{L^2(K)}^2 + \varepsilon^{-\frac{1}{2}} \alpha_E \|D_E\|_{L^2(E)}^2 = \Theta_{1K}^2 + \Theta_{2E}^2. \quad (4.54)$$

Desta forma, as quantidades máximas para os dados podem ser definidas por:

v. A contribuição máxima dos dados para os elementos $\Theta_{1K,max}^2$:

$$\Theta_{1K,max}^2 = \max_K \Theta_{1K}^2 \in \quad (4.55)$$

vi. A contribuição máxima dos dados para os elementos $\Theta_{2E,max}^2$:

$$\Theta_{2E,max}^2 = \max_{E \notin \Gamma_N} \Theta_{2E}^2 \quad (4.56)$$

vii. A contribuição máxima dos dados para os elementos $\Theta_{K,max}^2$:

$$\Theta_{K,max}^2 = \max_K \Theta_K^2 \quad (4.57)$$

Outras quantidades de interesse são os indicadores locais, os quais são utilizados nas estratégias adaptativas. Os indicadores locais são obtidos das quantidades locais i-vii, e dados pela expressão $IndicadorLocal = (QuantidadeLocal)^{1/2}$ (PRAETORIUS; WEINMULLER; WISSGOTT, 2008).

4.4 Adaptatividade

Aqui, a adaptatividade refere-se à utilização do estimador residual para controlar o erro espacial e temporal da malha de elementos finitos, enquanto que a adaptatividade dos dados refere-se ao adequado controle dos erros associados aos dados. Nesse caso, a adaptatividade pode ser dividida entre: a adaptatividade da condição inicial, das dispersões, do campo de velocidades e da função de fonte.

A metodologia apresentada engloba esquemas adaptativos para a condição inicial e o termo de fonte. A proposta é baseada nas componentes do estimador residual e no fato de que a geometria e os dados devem ser adequadamente representados.

4.4.1 Limitação do Erro $\hat{\eta}_I$

Considere o estimador residual $\hat{\eta}_I$, o termo $\tau_n^{1/2}\eta^n$ pode ser considerado como indicador do erro espacial local e o termo $\tau_n(\eta_\tau^{(n)})^{1/2}$ pode ser considerado o indicador de erro temporal local (VERFÜRTH, 2014).

Considere que

$$Estep_i^2 = \tau_n \left[(\eta^{(n)})^2 + (\Theta^{(n)})^2 + (\eta_\tau^n)^2 \right] \quad (4.58)$$

é o erro em cada passo de tempo τ_n . Logo, para uma dada tolerância ϵ_1 , se $\tau_n \left[(\eta^n)^2 + (\Theta^{(n)})^2 \right] < \epsilon_1^2/2$ e $\tau_n(\eta_\tau^{(n)})^2 < \epsilon_1^2/2$, então $Estep_i^2 = \tau_n \left[(\eta^n)^2 + (\Theta^{(n)})^2 + (\eta_\tau^n)^2 \right] < \epsilon_1^2$. Portanto, o erro global será dado por

$$\hat{\eta}_I < \left(\sum_{i=1}^{N_I} \epsilon_1^2 \right)^{1/2} = \epsilon_1 \sqrt{N_I} \quad (4.59)$$

considerando que $\|C_0 - \pi_0 C_0\|_{L^2(\Omega)}^2 = 0$. Caso $\|C_0 - \pi_0 C_0\|_{L^2(\Omega)}^2 \neq 0$, mas a representação da condição inicial é tal que $\|C_0 - \pi_0 C_0\|_{L^2(\Omega)}^2 < \epsilon^2$, então o erro global será dado por

$$\hat{\eta}_I < \left(\epsilon^2 + \sum_{i=1}^{N_I} \epsilon_1^2 \right)^{1/2} \leq \max(\epsilon, \epsilon_1) \cdot \sqrt{N_I + 1} \approx \max(\epsilon, \epsilon_1) \cdot \sqrt{N_I}, \text{ para } N_I \gg 1 \quad (4.60)$$

Analogamente, suponha que as condições P1-P4* sejam satisfeitas. Logo, as estimativas residuais incorporam as estimativas para o termo de fonte¹⁴. Nesse caso, se $\eta_f^2 < \epsilon_f^2$,

14. O termo referente à fronteira de Neumann é nulo devido às hipóteses adotadas para o método de elementos finitos.

então

$$Estep_i^2 = \tau_n \left[(\eta^{(n)})^2 + (\Theta^{(n)})^2 + (\eta_f^{(n)})^2 + (\eta_\tau^n)^2 \right] \leq \epsilon_1^2 + \epsilon_f^2. \quad (4.61)$$

Como consequência,

$$\hat{\eta}_I < \left(\epsilon^2 + \sum_{i=1}^{N_I} (\epsilon_1^2 + \epsilon_f^2) \right)^{1/2} \leq \max(\epsilon, \epsilon_1, \epsilon_f) \cdot \sqrt{N_I + 1} \quad (4.62)$$

Essas estimativas sugerem que o algoritmo adaptativo geral, proposto em [Verfürth \(2014\)](#), pode ser modificado para incorporar um procedimento adaptativo para a construção de uma malha de elementos finitos que represente adequadamente a condição inicial e o termo de fonte ¹⁵.

O algoritmo adaptativo geral, proposto por [Verfürth \(2014\)](#), possui a seguinte forma:

ALGORITMO ADAPTATIVO GERAL

1. Considere que são fornecidos os dados iniciais de uma equação diferencial parcial e uma tolerância, deseja-se encontrar uma solução numérica com um erro menor que a tolerância dada;
2. Construa uma malha inicial T_0 representando suficientemente bem a geometria e os dados do problema;
3. Resolva o problema discreto para a malha T_n , para $n = 0$;
4. Para todo elemento $K \in T_k$ calcule a estimativa a posteriori;
5. Se o erro estimado é menor que a tolerância, então pare. Caso contrário, decida quais elementos devem ser refinados e construa a próxima malha T_{k+1} . Faça $k \leftarrow k + 1$ e retorne ao passo (2).

Neste trabalho, o processo de refinamento adaptativo ¹⁶ é governado pela estratégia do máximo ([VERFÜRTH, 2014](#)):

ESTRATÉGIA DO MÁXIMO

1. Dado: uma partição T , um indicador do error η_K para um elemento na malha de elementos finitos, e uma fração $\theta_1 \in (0, 1)$. Encontrar um subconjunto $\tilde{T}$ de elementos que precisam ser refinados;
2. Calcule $\eta_{K,max} = \max_K \eta_K$;
3. Se $\eta_K > \theta_1 \eta_{T,max}$ marque o elemento K para refinamento e adicione-o ao conjunto $\tilde{T}$;
4. Refine todos os elementos na malha de elementos finitos.

15. A representação adequada da geometria não é discutida aqui.

16. referente à construção da malha adequada aos dados

O ponto de partida para as próximas seções é o item 2. do algoritmo adaptativo geral apresentado, pois a construção da malha inicial T_0 não é descrita. No entanto, existe a premissa de que os dados e a geometria são adequadamente representados, ou seja, a condições iniciais e de fronteira, o termo de fonte, o campo de velocidades, as dispersões e a função que descreve as reações do meio são representadas adequadamente.

4.4.2 Adaptatividade da Condição Inicial

Suponha que o erro de representação da condição inicial, dado por $\|C_0 - \pi_0 C_0\|_{L^2(\Omega)}^2$ é não nulo e que a condição inicial é não constante. Neste caso, é necessário que a condição inicial seja adequadamente representada no espaço de elementos finitos, ou seja, é necessário obter um malha tal que $\|C_0 - \pi_0 C_0\|_{L^2(\Omega)}^2 < \epsilon_1^2$, em que ϵ_1 é uma tolerância e πC_0 é a projeção L^2 no espaço de elementos finitos.

Globalmente, a estimativa do erro para a condição inicial é definida por:

$$\|C_0 - \pi_0 C_0\|_{L^2(\Omega)}^2 = \sum_K \|C_0 - \pi C_0\|_{L^2(K)}^2, \quad (4.63)$$

a qual avalia a qualidade da representação da expressão C_0 na malha de elementos finitos¹⁷. Localmente, a contribuição de um elemento K e o respectivo indicador do erro para a condição inicial são dados por:

$$\eta_{K,C_0}^2 = \|C_0 - \pi_0 C_0\|_{L^2(K)}^2, \quad \eta_{K,C_0} = \|C_0 - \pi_0 C_0\|_{L^2(K)}, \quad (4.64)$$

Seja N_T o número de elementos na malha inicial T'_0 , fornecida pelo usuário antes do processo de simulação, e considere que o elemento N possui o maior erro local de representação, ou seja, $\eta_{C_0,max} = \max_K \{\eta_{K,C_0}\}$. Logo, dada uma malha inicial T'_0 , uma tolerância ϵ_{C_0} e uma fração $frac_{C_0}$, o elemento K será marcado para refinamento, de acordo com a estratégia do máximo, se $\eta_{K,C_0} \geq frac_{C_0} \cdot \eta_{Kf,max}$. Com essas notações, o algoritmo adaptativo para a condição inicial é dado por:

ALGORITMO ADAPTATIVO PARA A CONDIÇÃO INICIAL

1. Dado uma malha inicial T'_0 fornecida pelo usuário, uma fração $frac_{C_0}$, tolerância ϵ_{C_0} , a expressão analítica para a condição inicial, obter uma representação adequada da função C_0 ;
2. Se $\|C_0 - \pi_0 C_0\|_{L^2(\Omega)}^2 < \epsilon_{C_0}$, então pare;
3. Senão, calcule a estimativa $\|C_0^p - C_0^{p+k}\|_{L^2(\Omega)}^2$, em que C_0^p e C_0^{p+k} são

17. O problema da representação analítica de uma expressão é discutida na seção [4.5]

- representações da condição inicial que usam funções lagrangeanas de ordens p e $p + k$, respectivamente;
4. Se $\|C_0^p - C_0^{p+k}\|_{L^2(\Omega)}^2 < \epsilon_{C0}$, então pare. Senão, calcule os indicadores locais η_{K,C_0} ;
 5. Se $\eta_{K,C_0} \geq \text{frac}_{C0} \eta_{K_{C_0}, \max}$, então marque o elemento K para refinamento
 5. Refine todos os elementos marcados. Volte ao item 4.

Esse procedimento é similar ao algoritmo adaptativo geral e pode ser utilizado para os outros dados do problema tais como as componentes do campos de velocidades, a condição inicial, as dispersões e o termo reativo.

4.4.3 Adaptatividade da função de fonte

Suponha que o erro de representação do termo de fonte η_f é não nulo. A representação adequada deve ser utilizada para a solução da equação do transporte de contaminantes, pois os erros de representação da função f são inseridos nas estimativas residuais e propagados durante o processo de simulação.

Nesta parte, o estimador residual referente às estimativas de [Praetorius, Weinmuller e Wissgott \(2008\)](#) é utilizado para obter uma malha inicial, adequada ao processo de simulação, a partir de uma malha inicial fornecida pelo usuário.

Globalmente, a estimativa residual associada ao termo de fonte, denotado por $(\eta_f^{(n)})$, é dada por:

$$\eta_f^{(n)} = \left\{ \sum_K \alpha_K^2 \left[\theta^2 \|f(t_n) - f_h^{(n)}\|_K^2 + (1 - \theta)^2 \|f(t_{n-1}) - f_h^{(n-1)}\|_K^2 \right] \right\}^{1/2}. \quad (4.65)$$

onde $f_h := \Pi f^{(n)}$ denota a projeção L^2 no espaço de elementos finitos, $f_h = \theta f^{(n)} + (1 - \theta)f^{(n-1)}$ e $n = \{1, 2, \dots, N_T\}$.

Essa quantidade é utilizada para obter uma malha que representa adequadamente a função f para um determinado passo de tempo τ_n . Localmente, o indicador dado por

$$\eta_{K_f} = \left\{ \alpha_K^2 \left[\theta^2 \|f(t_n) - f_h^{(n)}\|_K^2 + (1 - \theta)^2 \|f(t_{n-1}) - f_h^{(n-1)}\|_E^2 \right] \right\}^{1/2}, \quad (4.66)$$

avalia a qualidade da representação da função f na malha de elementos finitos dada.

Considere que N_T é o número de elementos na malha de elementos finitos, que o elemento N possui o maior erro local de representação da função f , ou seja, $\eta_{K_f, \max} = \max\{\eta_{N_f}\}$. Logo, dado uma malha inicial T_0 , o passo de tempo τ_n , uma tolerância ϵ_f e um

parâmetro $frac_{fonte} \in (0, 1)$, o elemento K será marcado para refinamento, de acordo com a estratégia do máximo, se $\eta_{Kf} \geq frac_{fonte} \cdot \eta_{Kf,max}$. Dessa forma o algoritmo adaptativo para o termo de fonte é dado por:

ALGORITMO ADAPTATIVO PARA A FUNÇÃO DE FONTE f

1. Dado uma malha inicial T_0 fornecida pelo usuário, uma fração $frac_{fonte}$, tolerância ϵ_f , a expressão analítica da função f e os valores das constantes ϵ, β envolvidas no estimador de erros, obter uma representação adequada de f ;
2. Calcule a estimativa global η_f^n , utilizando funções lagrangeanas de ordens p e $p+k$;
3. Se $\eta_f^n \leq \epsilon_f$, então pare. Senão, calcule os indicadores locais η_{Kf} ;
4. Se $\eta_{Kf} \geq frac_{fonte} \eta_{Kf,max}$, então marque o elemento K para refinamento;
5. Refine todos os elementos marcados. Volte ao item 2.

A utilização desta função exige a avaliação de f em um espaço de ordem superior ou com malha refinada. Ambos os procedimentos incluem um significativo esforço computacional, pois elevam o número de variáveis no sistema linear associado. No entanto, a utilização de malha adequada pode garantir a qualidade do resultado e, ainda, reduzir o esforço computacional.

4.5 Projeção L^2 e cálculo das constantes ϵ e β

A projeção L^2 de uma função f e as constantes ϵ e β estão estritamente relacionadas à obtenção das estimativas residuais. A metodologia de cálculo da projeção L^2 é descrita em [Larson \(2010\)](#), enquanto que a metodologia de cálculo das constantes ϵ e β é baseada no SciPy ([JONES et al., 2001](#)).

4.5.1 Projeção L^2

Segundo [Larson \(2010\)](#), a interpolação é a maneira mais simples de aproximar funções contínuas, mas existem outras formas. Nesta parte a projeção L^2 ou projeção ortogonal é apresentada seguindo estritamente o ponto de vista de [Larson \(2010\)](#).

Definição: Para uma dada função $f \in L^2$, a projeção L^2 de f , denotada por $\pi_h f$ é definida por

$$\int_{\Omega} (f - \pi_h f) v dx = 0, \forall v \in V_h. \quad (4.67)$$

onde V_h é o espaço de elementos finitos.

Para calcular a projeção L^2 , $\pi_h f$, deve-se notar que a definição acima é equivalente a

$$\int_{\Omega} (f - \pi_h f) \phi_i d\Omega = 0, i = 0, 1, 2, 3, \dots, n \quad (4.68)$$

onde ϕ_i é uma base para o espaço V_h . Isso é consequência do fato de que a equação (4.67) é satisfeita para todo $v \in V_h$. Logo, deve ser satisfeita para uma combinação linear qualquer destas funções, e reciprocamente, desde que qualquer função $v \in V_h$ é uma combinação linear das funções ϕ_i , então (4.67) implica (4.68) (LARSON, 2010).

A estimativa do erro de aproximação utiliza a representação da função aproximada em uma malha refinada com espaço de elementos finitos com funções de ordem mais elevada. Essa metodologia de representação e cálculos de erros é apresentada em Fenics Project (2014).

4.5.2 Cálculo das constante ε e β

A metodologia de cálculo dessas constantes baseia-se em conceitos de Álgebra Linear e em um procedimento de minimização de funções em domínios fechados. É importante ressaltar que esse também é um ponto importante do trabalho e pode servir de base para cálculos mais elaborados e problemas com dependência temporal¹⁸.

As constantes a serem determinadas são aquelas relacionadas na hipóteses do estimador residual (P1) e (P3). Para o cálculo da constantes ε e κ o cálculo de autovalores da matriz de dispersão é utilizado. Nesse caso, o problema é reduzido à mínimos de funções de duas variáveis por meio de algumas considerações. Por outro lado, a constante β é avaliada diretamente por meio de um procedimento numérico.

4.5.2.1 ε através do cálculo de autovalores

Considere que as componentes da matriz de dispersão D possuem dependência espacial e são dadas por $d_{xx}(x, y)$, $d_{xy}(x, y)$, $d_{yx}(x, y)$ e $d_{yy}(x, y)$ em um domínio $\Omega \subset \mathbb{R}^2$. Logo, para um ponto $(x_0, y_0) \in \Omega$, os autovalores associados $D(x_0, y_0, t) = D(x_0, y_0)$ são calculados através da equação característica¹⁹:

$$\lambda^2 - (d_{xx}(x_0, y_0) + d_{yy}(x_0, y_0))\lambda + (d_{xx}(x_0, y_0)d_{yy}(x_0, y_0) - d_{xy}(x_0, y_0)d_{yx}(x_0, y_0)) = 0. \quad (4.69)$$

Resumidamente, a equação (4.69) pode ser dada por:

$$\lambda^2 - Tr(D)\lambda + \det(D) = 0.0, \quad (4.70)$$

onde $Tr(D) = Tr(D(x_0, y_0))$ e $\det(D) = \det(D(x_0, y_0))$ são, respectivamente, o traço e o determinante da matriz D no ponto (x_0, y_0) .

18. Uma sugestão para os cálculos envolvendo problemas com dependência temporal é apresentada na seção [6.1]

19. $\det(\lambda I - D) = 0$

As raízes da equação (4.69) são:

$$\lambda_1 = \frac{Tr(D) - \sqrt{Tr(D)^2 - 4Det(D)}}{2}, \quad \lambda_2 = \frac{Tr(D) + \sqrt{Tr(D)^2 - 4Det(D)}}{2}. \quad (4.71)$$

onde λ_1, λ_2 são, respectivamente, o menor e maior autovalores de D em $(x_0, y_0) \in \Omega$.

Como $(x_0, y_0) \in \Omega$ é um ponto genérico no domínio, λ_1 e λ_2 podem ser consideradas funções de duas variáveis em um domínio Ω . O valor ε é o ponto de mínimo de λ_1 , enquanto que κ é o valor máximo de λ_2 .

Computacionalmente, esses cálculos requerem uma discretização do domínio computacional a qual pode não fornecer o valor adequado de ε e κ . Nesse caso, o autor considera o seguinte procedimento:

1. Considere duas discretizações Ω^h e $\Omega^{h/2}$ do domínio computacional Ω . Essas discretizações são constituídas de pontos igualmente espaçados nas direções coordenadas²⁰. As figuras (7-a, b) ilustram o procedimento de construção das malhas de pontos Ω^h e $\Omega^{h/2}$;

(a) Malha de pontos Ω^h

(b) Malha de pontos $\Omega^{h/2}$

Figura 7 – Duas malhas de pontos igualmente espaçados nas direções coordenadas

2. Calcule os valores ε_h, κ_h e $\varepsilon_{h/2}, \kappa_{h/2}$;
3. Se $\varepsilon_h = \varepsilon_{h/2}$ e $\kappa_h = \kappa_{h/2}$ então $\varepsilon = \varepsilon_h$ e $\kappa = \kappa_h$;
4. Senão, considere a discretização $\Omega^{h/4}$ e repita o processo.

4.5.2.2 Cálculo da Constante β

Suponha que $\lambda = \lambda(x, y)$ e v_x e v_y são funções contínuas para o campo de velocidades. Definindo $\beta := \inf_{(x,y) \in \Omega, 0 < t \leq T} (\lambda - (1/2)div(v))$, então β é tal que $\lambda - (1/2)div(v) \geq \beta$.

20. Aqui apenas pontos são considerados para os cálculos. Esse procedimento produz uma malha de pontos similar ao esquema de diferenças finitas

Aqui, algumas observações são necessárias:

1. Se o campo de velocidades é constante nas direções, então $\text{div}(v) = 0$. Logo, o cálculo de β é simplificado;
2. Se o campo de velocidades não é constante nas direções e foi obtido da solução da equação do fluxo, então é necessário transformá-lo em um campo contínuo antes de efetuar o cálculo do divergente;
3. O problema do item [2] é resolvido seguindo a metodologia descrita em [Fenics Project \(2014\)](#) ou pode ser executado por `projectGradient(kargs**)`²¹.

As próximas seções tratam do processo de reutilização das estruturas hierárquicas dos métodos multigrid algébricos para acelerar o processo de solução de sistemas lineares.

4.6 AMG com Reutilização de estruturas

Esta parte considera a solução da equação do transporte de contaminantes em meio poroso saturado através de um método multigrid algébrico com reutilização das estruturas de agregação e operadores de transferência de resultados. Essa proposta é baseada nas seguintes observações e considerações:

1. Como apontado em [Notay \(2012\)](#), para problemas simétricos, provenientes de equações elípticas, existem várias abordagens que conduzem a métodos eficientes de solução do sistema linear $Ax = b$. Entretanto, para problemas não simétricos, obtidos da equação de advecção-difusão $\nu \Delta u + \bar{v} \cdot \nabla u = f$, apenas os resultados de [Brezina et al. \(2010\)](#) e [Wu, Howard e Elman \(2004\)](#) são encontrados.
2. Problemas não simétricos estão associados a solução de equações diferenciais parciais de advecção-difusão (estado estacionário), com dependência temporal, ou não lineares ([BREZINA et al., 2010](#));
3. Problemas envolvendo o transporte de contaminantes em meio poroso saturado, em geral, envolvem:
 - i. termos advectivos, difusivos e reativos;
 - ii. presença de comportamentos transientes - dependência temporal;
 - iii. coeficientes variáveis, ou seja, os dados da equação do transporte podem possuir dependência espacial ou temporal;
 - iv. domínios irregulares e geometrias alongadas;
 - v. processos adaptativos.

21. Esta função é fornecida no apêndice.

4. É necessário que o método seja robusto, ou seja, possa ser usado em aplicações diversas envolvendo o fluxo e o transporte em meio poroso saturado;
5. as estimativas residuais para problemas com advecção dominante aumentam o trabalho computacional devido à necessidade de resolver um problema auxiliar.

Suponha, primeiramente, que o método multigrid gerado no passo n juntamente com a estrutura hierárquica de matrizes e os operadores de transferência de resultados estejam disponíveis para o passo $n + 1$. Com isso, dois tipos de métodos multigrid com reutilização foram elaborados: o método com reutilização com número fixo de reutilizações e o método automático.

O primeiro método consiste em reutilizar as estruturas multigrid em um número fixo de passos de tempo, enquanto o segundo método determina automaticamente o número de passos que o processo deve ser repetido. Como consequência da reutilização, obtém-se os métodos pré-condicionados por métodos multigrid algébricos com reutilização.

Sem perda de generalidade, suponha que o método multigrid algébrico SA é utilizado. Adotam-se as seguintes notações para descrever os métodos multigrid algébricos que empregam a reutilização:

1. SA : o método multigrid algébrico SA a ser adotado para as comparações envolvendo a reutilização.
2. $SAReu(\omega)$: o método multigrid algébrico SA com parâmetro de reutilização ω , em que ω denota o número de passos de tempo em que o processo de reutilização será executado.
3. $SAReuAuto$: o método multigrid algébrico SA com reutilização, cujo número de passos de tempo é determinado automaticamente durante o processo de solução.

Os esquemas apresentados nas subseções (4.6.1) e (4.6.2) podem ser usados tanto para o método multigrid quanto para um método iterativo pré-condicionado por multigrid algébrico. Basta que a adequada manipulação seja executada.

Sem perda de generalidade, suponha que o método iterativo $GMRES$ é utilizado. Adota-se as seguintes notações para descrever os métodos iterativos pré-condicionados por métodos multigrid algébricos que empregam a reutilização:

4. $GMRES_{SA}$: o método $GMRES$ pré-condicionado pelo método multigrid algébrico SA .
5. $GMRES_{SAReu}$: o método $GMRES$ pré-condicionado pelo método multigrid algébrico $SAReu$.
6. $GMRES_{SAReuUTO}$: o método $GMRES$ pré-condicionado pelo método multigrid algébrico $SAReuAuto$.

Segundo Stüben (2001), os métodos multigrid são bons pré-condicionantes e devem ser considerados para a solução iterativa de sistemas lineares.

Os diferentes métodos propostos são apresentados nas subseções (4.6.1), (4.6.2), (4.6.3) e (4.6.4).

4.6.1 SA com Reutilização- $SAReu(\omega)$

Esse método consiste na reutilização das estruturas em um número fixo ω de passos de tempo de simulação da equação do transporte de contaminantes. O parâmetro ω será chamado parâmetro de reutilização.

SA COM REUTILIZAÇÃO: $SAReu(\omega)$

1. Defina o parâmetro de reutilização ω , o número total de passos $N_{step} = N_I$ do processo de simulação e o contador n para o número de passos de tempo.
2. Se $n \equiv 0 \pmod{\omega}$, defina o método SA, guarde as estruturas geradas para os próximos passos de tempo.
3. Se $n \equiv 0 \pmod{\omega}$, então resolva o sistema linear. Senão, reuse o método gerado em (2) para resolver os $\omega - 1$ passos seguintes.

4.6.2 SA com Reutilização e automático

O método $SAReu$ executa os cálculos das estruturas no passo $n = 1$ e repete esses cálculos a cada ω passos, em que o parâmetro ω é escolhido de modo arbitrário. Esta seção propõe que o método determine o passo de tempo em que o cálculo das estruturas é necessário. Essa determinação automática é baseada no número de iterações necessárias à convergência no passo em que o cálculo das estruturas é executado.

O algoritmo abaixo apresenta o método SA com reutilização e automático($SAReuAuto$).

SA COM REUTILIZAÇÃO E AUTOMÁTICO: $SAReuAuto$

1. Defina a fração parâmetro de reutilização $0 < n_{frac} \leq 1$, o contador i e o número total de passos $N_{step} = N_I$ do processo de simulação.
2. Defina o método SA e guarde as estruturas geradas para os próximos passos. Resolva o sistema linear e determine o número de iterações para a convergência($AmgIter1$). Guarde $AmgIter1$ e o método gerado para os próximos passos.
3. Reuse o método gerado no próximo passo e calcule o número de iterações necessárias para a convergência ($AmgIter2$).
4. Se $AmgIter2 > AmgIter1 \cdot (1 + n_{frac})$, siga para o passo 2. Senão, enquanto

$AmgIter2 \leq AmgIter1 \cdot (1 + n_{frac})$ siga para o passo (3).

4.6.3 Método Iterativo Pré-condicionado por $SA - Reu$

Para obter o método $GMRES_{SAReu}$ basta substituir adequadamente o método pré-condicionante do método $GMRES_{SA}$. Dessa forma, a única diferença é a construção do método multigrid $SA - Reu$, o qual foi apresentado na seção (4.6.1).

GMRES PRÉ-CONDICIONADO POR $SA - Reu$: $GMRES_{SA-Reu(\omega)}$

1. Defina o parâmetro de reutilização ω , o número total de passos $N_{step} = N_I$ e o contador n para o número de passos de tempo.
2. Se $n \equiv 0 \pmod{\omega}$, defina o método SA e guarde as estruturas geradas para os próximos passos de tempo. Construa o método $GMRES_{SA}$
3. Se $n \equiv 0 \pmod{\omega}$, então resolva o sistema linear utilizando $GMRES$ pré-condicionado por SA . Senão, reuse o método gerado em (2) nos $\omega - 1$ passos seguintes.

4.6.4 Método iterativo Pré-condicionado por $SA - ReuAuto$

Para obter o método $GMRES_{SAReuAuto}$ basta substituir adequadamente o método pré-condicionante do método $GMRES_{SA}$. Esse método é idêntico àquele apresentado na seção (4.6.2) com as convenientes alterações.

O algoritmo para o método pré-condicionado por SA com reutilização e automático é dado por:

GMRES PRÉ-CONDICIONADO POR $SA - ReuAuto$: $GMRES_{SA-ReuAuto}$

1. Defina a fração parâmetro de reutilização $0 < n_{frac} \leq 1$, o contador i e o número de passos de tempo n_{step} do problema.
2. Defina o método SA e guarde as estruturas geradas para os próximos passos de tempo. Construa o método $GMRES$ pré-condicionado por SA . Resolva o sistema linear e determine o número de iterações necessárias para a convergência ($GmresIter1$). Guarde $GmresIter1$ e o método gerado para os próximos passos.
3. Reuse o método gerado no próximo passo e calcule o número de iterações necessárias para a convergência ($GmresIter2$).
4. Se $GmresIter2 > GmresIter1 \cdot (1 + n_{frac})$, siga para o passo 2. Senão, enquanto $GmresIter2 \leq GmresIter1 \cdot (1 + n_{frac})$ siga para o passo (3)

4.6.5 Comparação de resultados

As soluções obtidas pelas diferentes estratégias $SA-Reu$ ou $SA-ReuAuto$ necessitam ser comparadas com a solução obtida pelo método original SA .

Sem perda de generalidade, considere que o sistema linear, originado da aproximação de elementos finitos em malha triangular, no passo de tempo $n \in \{1, 2, 3 \dots, N_I\}$, possui a forma característica $A^n C^n = b^n$, em que $C^n = (A^n)^{-1} b^n$ é a solução obtida numericamente pelos diferentes métodos adotados.

Considere que $C_{SAReu}^n = C^n(x, y, t_n)$, $C_{SAReuAuto}^n = C_{SAReuAuto}^n(x, y, t_n)$ e $C_{SA}^n = C_{SA}^n(x, y, t_n)$ sejam as soluções obtidas pelos métodos $SAReu$, $SAReuAuto$ e SA , respectivamente. Adicionalmente, considere que os resíduos das soluções são dados por $Res_{SA}^n = Res_{SA}^n(x, y, t_n) = b^n - C_{SA}^n$, $Res_{SAReu}^n = Res^n(x, y, t_n) = b^n - C_{SAReu}^n$ e $Res_{SAReuAuto}^n = Res^n(x, y, t_n) = b^n - C_{SAReuAuto}^n$.

As soluções C_{SA}^n , C_{SAReu}^n e $C_{SAReuAuto}^n$ fornecem as concentrações para os vértices (x, y) dos triângulos no n -ésimo passo de tempo. Da mesma forma, Res_{SA}^n , Res_{SAReu}^n e $Res_{SAReuAuto}^n$ são os resíduos nos vértices (x, y) dos triângulos no n -ésimo passo de tempo.

A comparação dessas sequências de soluções e resíduos é baseada nas seguintes diferenças:

- i. $Diff_{SAReu/SA}^n = Diff_{SAReu/SA}^n(x, y, t_n) = C_{SA}^n - C_{SAReu}^n$
- ii. $Diff_{SAReuAuto/SA}^n = Diff_{SAReuAuto/SA}^n(x, y, t_n) = C_{SA}^n - C_{SAReuAuto}^n$
- iii. $ResDiff_{SAReu/SA}^n = ResDiff_{SAReu/SA}^n(x, y, t_n) = Res_{SA}^n - Res_{SAReu}^n$
- iv. $ResDiff_{SAReuAuto/SA}^n = ResDiff_{SAReuAuto/SA}^n(x, y, t_n) = Res_{SA}^n - Res_{SAReuAuto}^n$

A verificação da igualdade das soluções ou dos resíduos considera a função `count_non_zero`, disponível na biblioteca NumPy (WALT; COLBERT; VAROQUAUX, 2011). Caso valores não nulos sejam retornados, então os valores máximos e mínimos das respectivas diferenças são calculados.

O resíduo da solução numérica no n -ésimo passo de tempo, denotado por Res^n , é aquele obtido após a convergência do método multigrid utilizado. No entanto, as análises numéricas apresentadas neste trabalho também consideram os resíduos após k iterações do método multigrid. Nesse caso, para evitar sobrecarga de notação, a notação $Ax = b$ é utilizada para representar o sistema linear $A^n C^n = b^n$ no passo n , sendo que o passo de tempo e o método ficam indicados implicitamente pelo contexto.

O resíduo da solução aproximada, em cada iteração de um determinado método iterativo, é dado por :

$$r_n^i := b - Ax^i, i = 1, 2, 3 \dots, k \quad (4.72)$$

onde x^i é a solução obtida na iteração i , $i = \{1, 2, \dots, k\}$ é o número de iterações do método, k é o número total de iterações, n é o passo de tempo²². O resíduo relativo para cada iteração do método é definido por res^i/res^1 , $i = \{1, 2, \dots, k\}$. Além do resíduo relativo, este trabalho considera a norma l^2 do resíduo, o máximo resíduo e o resíduo médio.

As apresentações gráficas e análises envolvendo as quantidades numéricas definidas nesta parte são apresentadas em detalhes no capítulo de resultados [5].

4.6.6 Tempo Computacional

Uma das questões principais da proposta dos métodos com reutilização é o tempo gasto na solução dos sistemas lineares. Aqui, o tempo gasto em cada passo do processo de simulação é chamado tempo computacional e leva em conta apenas o tempo requerido pelo método numérico para obter a solução. Formalmente é dado por $\Delta t_n = t_{n_1} - t_{n_0}$, em que t_{n_0} é o tempo imediatamente antes da aplicação do método de solução e t_{n_1} é o tempo imediatamente após a obtenção da solução. Consequentemente, define-se o tempo total gasto para resolver todos os sistemas lineares do processo de simulação pela soma dos tempos computacionais. Matematicamente, o tempo total é dado por:

$$Ta_n := \sum_{i=1}^n \Delta t_i, 1 \leq n \leq N_I. \quad (4.73)$$

4.7 Notas sobre os desenvolvimentos

O desenvolvimento dos códigos utilizados nesta tese foi feito em linguagem Python^R, utilizando a plataforma de desenvolvimento Eclipse²³ que é uma plataforma de código livre desenvolvida para a construção de ambientes integrados e que podem ser usados para o desenvolvimento de aplicações de diversos tipos (ECLIPSE FOUNDATION, 2015). A implementação do código computacional de elementos finitos foi norteadado pela metodologia descrita Fenics Project (2014)²⁴ enquanto que a implementação do método multigrid algébrico foi norteadado pela metodologia descrita na biblioteca Bell, Olson e Schroder (2011). A implementação do código para o cálculo das constantes envolvidas no estimador de erros foi norteadado pela metodologia descrita em Jones et al. (2001), a qual fornece, dentre outros, os métodos de minimização de funções tais como gradiente conjugado. Além disso,

22. O resíduo na iteração k , r_n^k , é tal que $Res^n = r_n^k$. Essa dupla definição serve apenas para evitar sobrecarregar a notação

23. Aqui IDE Eclipse foi utilizada (ECLIPSE FOUNDATION, 2015). A linguagem Python^R é disponibilizada através do plugin PyDev.

24. O site disponibiliza uma série de referências, exemplos e outras facilidades para o leitor.

a implementação e utilização dos métodos iterativos²⁵ utilizam o pacote *Sparse* de Jones et al. (2001).

As *classes* e *funções* que implementam o método de elementos finitos e as estimativas residuais são construídas separadamente, de forma que é possível alterá-las sem comprometimento das partes desenvolvidas. A implementação do método de elementos finitos é facilitado pela introdução das formas bilinear e linear, as quais utilizam linguagens específicas e facilitam a introdução e desenvolvimento das formulações variacionais. Adicionalmente, as ferramentas disponíveis possibilitam avaliar qualquer uma das partes componentes do estimador residual conforme apresentado na seção (4.3.6).

A partição do estimador residual permite que os regimes de dispersão dominante e advecção dominante sejam tratados da mesma forma, ou seja, o estimador residual é dado pela soma das estimativas espacial, dos dados e temporal. A incorporação de estimativas adicionais é realizada pela introdução de classes e funções específicas no código computacional. Dessa forma, o procedimento básico de cálculo das estimativas e indicadores é semelhante aos cálculos efetuados para os problemas caracterizados pela dispersão dominante e por dados constantes. Adicionalmente, a adequada obtenção das estimativas residuais está associada à obtenção das constantes ε , κ e β , cuja metodologia de cálculo é baseada nos métodos de minimização de funções de várias variáveis, disponíveis no em Jones et al. (2001).

O desenvolvimento dos métodos multigrid algébricos com reutilização foi baseado na biblioteca numérica PyAMG (BELL; OLSON; SCHRODER, 2011), a qual disponibiliza os códigos computacionais em *Python*^R. Essa biblioteca requer uma estrutura específica, diferente daquela utilizada pela biblioteca de elementos finitos dolfin (FENICS PROJECT, 2014). Essa estrutura específica é disponibilizada através do pacote Scipy Sparse (JONES et al., 2001), o qual fornece ferramentas para manipulação de matrizes esparsas. As manipulações matriciais adicionais são executadas pela biblioteca numérica NumPy (WALT; COLBERT; VAROQUAUX, 2011).

A integração entre as bibliotecas do FEniCS e a biblioteca PyAMG foi baseada no exemplo disponível da documentação da biblioteca PyAMG (BELL; OLSON; SCHRODER, 2011). A forma como os métodos multigrid são implementados possibilita a implementação e teste de outros métodos multigrid através das classes e estruturas já desenvolvidas.

Baseado no módulo *blackbox* do PyAMG, o código para reutilização foi construído através de algumas modificações. Essa abordagem, além de disponibilizar as estruturas multigrid, número de iterações necessárias à convergência e residuais em cada iteração do método, permite a construção tanto dos métodos com reutilização com parâmetro fixo ω quanto dos métodos com reutilização automática. Além disso, se nenhuma especialização

25. Os métodos iterativos não estacionários utilizados nesta tese tais como *GMRES*, *BICGSTAB*.

é passada via dicionário de parâmetros, então o módulo com modificações utiliza àquela especialização definida pelo módulo `blackbox`.

De posse das soluções numéricas e resíduos, as comparações numéricas entre os valores obtidos pelos diferentes métodos são realizadas por meio da função `count_non_zero`, disponível na biblioteca NumPy (WALT; COLBERT; VAROQUAUX, 2011). Essa função, aplicada às matrizes $Diff_{SAReu/SA}^n$, $Diff_{SAReuAuto/SA}^n$, $ResDiff_{SAReu/SA}^n$ e $ResDiff_{SAReu/SA}^n$, retorna o número de elementos não nulos em uma dada matriz. Como consequência, permite avaliar se as soluções são idênticas ou diferentes. A magnitude das diferenças, caso existam, pode ser avaliada através do cálculo dos valores máximos, os quais podem ser fornecidos pelas ferramentas computacionais disponíveis em NumPy. Adicionalmente, os resultados numéricos envolvendo cálculos da norma l^2 , valores máximos, valores mínimos e médias foram baseados nas funções disponíveis na biblioteca NumPy (WALT; COLBERT; VAROQUAUX, 2011).

As apresentações gráficas são baseadas nas bibliotecas `matplotlib` (HUNTER, 2007) e no software `ParaView` (AYACHIT, 2015). Eventualmente, as ferramentas de visualização disponibilizadas no Projeto FEniCS são utilizadas (FENICS PROJECT, 2014).

É importante salientar que a integração entre as diferentes bibliotecas utilizadas é um dos aspectos fundamentais desse trabalho. A escolha da plataforma de desenvolvimento, da biblioteca de elementos finitos juntamente com a linguagem de programação e as ferramentas disponíveis, a integração da biblioteca para métodos multigrid e a disponibilidade dos métodos iterativos não-estacionários por meio do pacote **linalg** formaram a base essencial para o desenvolvimento dessa tese.

5 Resultados

Neste capítulo os resultados da implementação do código de elementos finitos, as estratégias de refinamento adaptativo para os dados da equação do transporte e os métodos multigrid algébricos com reutilização são descritos. As primeiras seções descrevem os resultados envolvendo a validação do código de elementos finitos juntamente com a validação dos estimadores residuais. Em seguida, as estratégias adaptativas para obter malhas adequadas às representações dos dados são discutidas. Finalmente, os resultados envolvendo os métodos multigrid com reutilização das estruturas hierárquicas são descritos.

5.1 Método de elementos finitos e Estimador Residual

Esta seção apresenta os resultados numéricos da solução da equação do transporte de contaminantes, as respectivas soluções analíticas e os estimadores de erros residuais. As soluções numéricas e analíticas são comparadas de acordo com a metodologia exposta.

5.1.1 Exemplo 01

Este problema considera a equação do transporte de contaminantes em domínio retangular $\Omega = [x_0, x_1] \times [y_0, y_1] = [0.0, 3000.0] \times [0.0, 1500.0]$ e é uma adaptação espacial do problema apresentado em Sorek (SOREK, 1988). Os parâmetros físicos são $D_x = 100.0$, $D_y = 20.0 \frac{m^2}{d}$ para a dispersão longitudinal e transversal, $\mathbf{v}_x = 0.2 \frac{m}{d}$ e $\mathbf{v}_y = 0.0 \frac{m}{d}$ para as componentes da velocidade e reação $\lambda = Constante \in \mathbb{R}$. A fronteira de Dirichlet é definida em $\Gamma_D = \{x_0\} \times [y_0, y_1] \cup \{y_0\} \times [x_0, x_1] \cup \{y_1\} \times [x_0, x_1]$ por

$$\frac{C(x, y, t)}{C_0} = \begin{cases} 1.0, & \text{para } x = 0, |y - 750.0| < 115.0, t > 0 \\ 0, & \text{caso contrário} \end{cases} \quad (5.1)$$

e a fronteira de Neumann é definida em $\Gamma_N = \Omega \setminus \Gamma_D = \{x_1\} \times [y_0, y_1]$ por $g = \mathbf{n} \cdot \nabla(C)$. A condição inicial é $C_0 = C(x, y, t_0 = 0.0) = 0.0$ para todo $x \in \Omega$.

Nesse caso, a solução analítica apresentada na seção (4.2.3), é utilizada para validação do código. Os dois próximos exemplos apresentam os resultados numérico e analítico da solução da equação ADR, os indicadores espaciais, dos elementos, dos saltos e das fronteiras.

Nesse caso, o estimador residual dado pela equação (4.37) é o estimador de Verfürth (2014), pois satisfaz as condições (P1)-(P4). Formalmente ¹,

$$(\eta_I) = \left\{ \|C_0 - \pi C_0\|_{L^2(\Omega)}^2 + \sum_{i=1}^{N_I} \tau_n \left[\left(\eta^{(n)}\right)^2 + \left(\Theta^{(n)}\right)^2 + \left(\eta_\tau^{(n)}\right)^2 \right] \right\}^{1/2}. \quad (5.2)$$

O estimador (5.11) pode ser simplificado, pois a condição inicial e os dados podem ser representados exatamente pelo método de elementos finitos com funções lagrangeanas lineares. Além disso, o regime é caracterizado pela dispersão dominante, ou seja, existe uma constante C_c definida por

$$C_c := \frac{|v|}{\varepsilon^{1/2} \cdot \max(\varepsilon, \beta)^{1/2}} \quad (5.3)$$

tal que $0.2 \leq C_c \cdot \varepsilon^{1/2} \cdot \max(\varepsilon, \beta)^{1/2}$ (FIRMIANO, 2010). Em ambos os casos considerados, reativo ou conservativo, o regime de dispersão dominante caracteriza o estimador de erros e possui constante C_c dada por:

$$C_c := \frac{0.2}{20^{1/2} \cdot \max(20, 0)^{1/2}} = \frac{0.2}{20^{1/2} \cdot \max(20, \beta)^{1/3}} = \frac{0.2}{20} = 0.01, \text{ para } \beta < 20. \quad (5.4)$$

Dessa forma, o estimador residual (5.11) pode ser simplificado para

$$(\eta_I) = \left\{ \sum_{i=1}^{N_I} \tau_n \left[\left(\eta^{(n)}\right)^2 + \left(\eta_\tau^{(n)}\right)^2 \right] \right\}^{1/2} = \left\{ \sum_{i=1}^{N_I} \tau_n \left[\left(\eta^{(n)}\right)^2 + |||C^m - C^{m-1}|||^2 \right] \right\}^{1/2}. \quad (5.5)$$

O estimador residual (5.5) é mais acurado que o estimador proposto por Firmiano (2010) no sentido que a estimativa espacial associada à equação (5.5) é tal que :

$$\left(\eta^{(n)}\right) = \left\{ \sum_K \alpha_K^2 ||R_K||_{L^2(K)}^2 \right\}^{1/2} \leq \left(\eta^{(n)}\right), \quad (5.6)$$

em que o lado esquerdo é o estimador espacial apresentado por Firmiano (2010) e o lado direito é estimador espacial dado pela expressão (4.24).

1. Aqui os termos $(\eta_f)^{(n)}$ e $(\eta_N)^{(n)}$ devem ser desconsiderados pois são válidas as hipóteses (P1)-(P4).

5.1.1.1 Transporte Conservativo em Campo Uniforme: $\lambda = 0$

Este caso considera uma malha com $Ne = 90.000$ elementos finitos triangulares, $Nv = 45.451$ vértices e orientação *left/right*. Os parâmetros para implementação são $n_x = 2n_y = 300$ elementos finitos triangulares em cada direção coordenada, funções lagrangeanas lineares, $\theta = 1/2$, $\tau_n = 5.0d$, $t_{final} = 1000.0d$, $n = \{1, 2, \dots, N_I = 200\}$.

A comparação das soluções fornece um erro máximo $E_{max} = \max(Erro) = 0.0289$ e um erro médio $E_{medio} = 5.39265 \cdot 10^{-05}$. A tabela [2] fornece as soluções numérica e analítica com concentrações normalizadas por C_0 , o erro real e os indicadores residuais para um conjunto de pontos no 200-ésimo instante de tempo.

Tabela 2 – A solução numérica no 200-ésimo instante de tempo com concentração normalizada, $\frac{C^{200}}{C_0} = \frac{C_{Numérica}}{C_0}$, a solução analítica em $t = t_f = 1000.0d$ com concentração normalizada, $\frac{C_A}{C_0} = \frac{C_{Analítica}}{C_0}$, o erro real (*Erro*), indicador espacial (η_K), o indicador do elemento (η_{1K}) e o indicador do salto (η_{2E}) para um conjunto de pontos

(x, y)	$\frac{C^{200}}{C_0}$	$\frac{C_A}{C_0}$	<i>Erro</i>	η_K	η_{1K}	η_{2E}
(10.0,630.0)	0.1955	0.2240	0.0286	7.3409	0.0499	7.3407
(20.0,630.0)	0.2953	0.3233	0.0279	3.9684	0.0237	3.9683
(30.0,630.0)	0.3479	0.3660	0.0181	3.0211	0.0178	3.0211
(40.0,630.0)	0.3760	0.3868	0.0108	1.7680	0.0071	1.7680
(10.0,640.0)	0.7934	0.7644	0.0290	6.9242	0.0499	6.9240
(20.0,640.0)	0.6816	0.6535	0.0281	4.0881	0.0286	4.0880
(30.0,640.0)	0.6171	0.5988	0.0183	2.5756	0.0178	2.5755
(40.0,640.0)	0.5767	0.5658	0.0109	2.6781	0.0179	2.6781
(10.0,860.0)	0.7934	0.7644	0.0290	7.3409	0.0499	7.3407
(20.0,860.0)	0.6816	0.6535	0.0281	3.9684	0.0237	3.9683
(30.0,860.0)	0.6171	0.5988	0.0183	3.0211	0.0178	3.0211
(40.0,860.0)	0.5767	0.5658	0.0109	1.7680	0.0071	1.7680
(10.0,870.0)	0.1955	0.2240	0.0286	4.5142	0.0305	4.5141
(20.0,870.0)	0.2953	0.3233	0.0279	3.5722	0.0237	3.5721
(30.0,870.0)	0.3479	0.3660	0.0181	2.7178	0.0176	2.7178
(40.0,870.0)	0.3760	0.3868	0.0108	1.5031	0.0071	1.5031

As figuras (8-a,b) apresentam, respectivamente, as soluções numérica e analítica com as respectivas isolinhas. O erro real (*Erro*), em cada ponto do domínio computacional, é apresentado na figura (9). Esse erro evidencia a existência de uma região, em torno dos pontos de descontinuidade da função de fonte ($y = 635$ e $y = 865$), em que a solução analítica não é representada adequadamente pela solução numérica. Esses erros englobam

os erros de representação da função de fonte, a qual é descontínua, e os erros devidos ao avanço da frente de contaminação.

(a) Solução analítica e isolinhas nas direções coordenadas

(b) Solução numérica e isolinhas nas direções coordenadas

Figura 8 – As soluções numérica e analítica em malha elementos finitos triangulares e funções lagrangeanas lineares

Figura 9 – O erro entre as soluções numérica e analítica $E = \text{Erro} = \left| \frac{C_{\text{Analítica}}}{C_0} - \frac{C_{\text{Numérica}}}{C_0} \right|$ representado em malha de elementos finitos triangulares e funções lagrangeanas lineares

A representação inadequada é devida às funções contínuas do método de elementos finitos. Uma representação adequada da função de fonte pode ser obtida através da utilização do método de Galerkin descontínuo. Os erros relativos ao avanço da frente de contaminantes são referentes à discretização adotada e podem ser corrigidos através da utilização de uma malha mais refinada. No entanto, aqui apenas os métodos contínuos são utilizados, pois os objetivos são a validação do código computacional de elementos finitos e dos estimadores residuais. Essa validação permite utilizar esses códigos para gerar sistemas lineares, os quais serão resolvidos pelos métodos multigrid com reutilização.

A adequação entre as soluções numérica e analítica também pode ser analisada através dos perfis de concentração, apresentados nas figuras (10-a,b). Esses perfis mostram as soluções numérica e analítica sobre as retas $y = y_0 = 250, 500, 620, 630, 640$. Novamente, as soluções analíticas e numéricas discordam apenas nas regiões de descontinuidade da função de fonte.

A análise mostra que o código computacional está adequadamente implementado, pois as soluções analítica e numérica possuem comportamento qualitativo semelhante. Além disso, as diferenças observadas são devidas à representação inadequada da fonte de contaminantes e ao número de elementos finitos na malha.

(a) Comparação das soluções numérica e analítica para um conjunto de pontos

(b) Comparação das soluções numérica e analítica para um conjunto de pontos

Figura 10 – Perfis de concentração comparando as soluções numérica e analítica

Os indicadores residuais, representados na malha de elementos finitos, são apresentados nas figuras (11, 12 e 13).

Figura 11 – O indicador espacial η_K em cada elemento

Figura 12 – O indicador do elemento η_{1K} em cada elemento

(a) O indicador do salto η_{2E} em cada elemento

(b) O indicador temporal η_{τ_n} em cada elemento

Figura 13 – Os indicadores do salto η_{2E} e temporal η_τ em cada elemento

A análise desses indicadores fornece valores máximos $\eta_{K,max} = 7.3411$, $\eta_{1K,max} = 0.1311$, $\eta_{2E,max} = 7.3407$ e $\eta_{3B,max} = 0.0$, $\max(\eta_\tau) = 3.5 \cdot 10^{-5}$. Os indicadores espaciais

capturam os erros relacionados a representação da descontinuidade da função de fonte e os erros relacionados ao avanço da frente de contaminação. Similarmente, os indicadores dos elementos e dos saltos ainda capturam o comportamento do erro e podem ser usados em uma estratégia adaptativa que contemplem a utilização das componentes do estimador residual. Por sua vez, o indicador temporal não possui uma interpretação relacionada à solução numérica, mas é apresentado para ilustrar a contribuição que cada elemento fornece à estimativa temporal.

O próximo caso analisa o transporte de contaminantes com termo reativo $\lambda \neq 0$.

5.1.1.2 Transporte Reativo: $\lambda \neq 0$

Este caso considera os mesmos dados do problema anterior exceto $\tau_n = 5.0d$, $t_{final} = 1000.0d$, $n = \{1, 2, \dots, N_I = 200\}$ e o fator de reação constante $\lambda = 0.01$. Analogamente ao caso anterior, a tabela [3] fornece um conjunto de pontos e as respectivas soluções numérica e analítica com concentrações normalizadas no 200-ésimo instante de tempo.

Tabela 3 – A solução numérica no 200-ésimo instante de tempo com concentração normalizada, $\frac{C^{200}}{C_0} = \frac{C_{Numérica}}{C_0}$, a solução analítica em $t = t_f = 1000.0d$ com concentração normalizada, $\frac{C_A}{C_0} = \frac{C_{Analítica}}{C_0}$, o erro real (*Erro*), indicador espacial (η_K), o indicador do elemento (η_{1K}) e o indicador do salto (η_{2E}) para um conjunto de pontos

(x, y)	$\frac{C^{200}}{C_0}$	$\frac{C_A}{C_0}$	<i>Erro</i>	η_K	η_{1K}	η_{2E}
(10.0,630.0)	0.1623	0.1913	0.0290	9.0060	0.0667	9.0058
(20.0,630.0)	0.2318	0.2597	0.0279	2.7122	0.0108	2.7122
(30.0,630.0)	0.2559	0.2742	0.0183	4.1785	0.0309	4.1784
(40.0,630.0)	0.2588	0.2695	0.0106	1.1058	0.0036	1.1058
(10.0,640.0)	0.7518	0.7221	0.0297	8.6658	0.0667	8.6656
(20.0,640.0)	0.6026	0.5747	0.0279	5.7929	0.0453	5.7927
(30.0,640.0)	0.5065	0.4879	0.0186	3.7845	0.0309	3.7844
(10.0,860.0)	0.7518	0.7221	0.0297	9.0060	0.0667	9.0058
(20.0,860.0)	0.6026	0.5747	0.0279	2.7122	0.0108	2.7122
(30.0,860.0)	0.5065	0.4879	0.0186	4.1785	0.0309	4.1784
(40.0,860.0)	0.4373	0.4267	0.0107	1.1058	0.0036	1.1058
(10.0,870.0)	0.1623	0.1913	0.0290	3.1874	0.0192	3.1874
(20.0,870.0)	0.2318	0.2597	0.0279	2.5833	0.0108	2.5832
(30.0,870.0)	0.2559	0.2742	0.0183	1.5341	0.0073	1.5341

As figuras (14 e 15) fornecem as soluções numérica e analítica e respectivas curvas de nível.

Figura 14 – A solução analítica e isolinhas em malha de elementos finitos e funções lineares

Figura 15 – A solução numérica e isolinhas em malha de elementos finitos e funções lineares

A comparação das soluções fornece um erro máximo $E_{max} = \max(Erro^{200}) = 0.0296$ e um erro médio $E_{medio} = 2.0982 \cdot 10^{-05}$, mostrando que o código computacional resolve adequadamente o transporte de contaminantes com reação. O mapa de erros é apresentado na figura (20).

Novamente, verifica-se que os maiores erros estão sobre as retas $y = Y_1$ e $y = Y_2$ devido à descontinuidade da fonte de contaminantes nos pontos $(0, Y_1)$ e $(0, Y_2)$ conforme pode ser observado na tabela [3] e nas figuras (20).

Figura 16 – O erro entre as soluções numérica e analítica $E = Erro = \left| \frac{C_{Analítica}}{C_0} - \frac{C_{Numérica}}{C_0} \right|$ representado em malha de elementos finitos triangulares e funções lagrangeanas lineares

As figuras (17), (18) e (19-a,b) apresentam, respectivamente, os indicadores espaciais, dos elementos, dos saltos e das fronteiras de Neumann. Novamente, o comportamento qualitativo do mapa de erros é capturado pelos indicadores de erros. A análise quantitativa fornece valores máximos $\eta_{K,max} = 9.0063$, $\eta_{1K,max} = 0.1088$, $\eta_{2E,max} = 9.0057$ e $\eta_{3B,max} = 0.0$. A diferença entre os valores máximos deste problema e do caso anterior é a presença do fator de reação, o qual afeta o avanço da frente de contaminação e as estimativas residuais.

Figura 17 – O indicador espacial η_K em cada elemento

Figura 18 – O indicador do elemento η_{1K} em cada elemento

Figura 19 – O indicador do salto η_{2E} em cada elemento

A próxima seção considera o transporte de contaminantes em um domínio Ω em que os dados possuem dependência espacial. A representação desses dados em um espaço de elementos finitos pode não ser exata e depende da função que está sendo representada e do grau do polinômio utilizado no método de elementos finitos. Por exemplo, a representação de um campo de velocidades com componentes lineares é exata em um espaço de funções lineares, mas a representação de um campo com componentes com variação quadrática não é exata para espaço de funções lineares. A discussão sobre representação adequada dos dados do problema do transporte é inserida na seção (5.1.3), e é apresentada juntamente com as estimativas residuais.

5.1.2 Exemplo 02

Nesta seção uma adaptação do problema apresentado em Zoppou e Knight (1999) é considerada. O problema do transporte de contaminantes é definido em um domínio bidimensional $\Omega = [0, 20] \times [0, 20]$ com campo variável de velocidades $\mathbf{v}$ e matriz de dispersão $D = D(x, y)$ com elementos com dependência espacial. O campo de velocidades é dado por $\mathbf{v} = (v_x, v_y) = (u_0 \cdot x, -u_0 \cdot y)$ enquanto que as componentes da matriz de dispersão são funções com dependência espacial e dadas por $d_{11}(x, y) = D_0 \cdot u_0^2 \cdot x^2$, $d_{12}(x, y) = d_{21}(x, y) = 0$ e $d_{22}(x, y) = D_0 \cdot u_0^2 \cdot y^2$, em que D_0 é uma constante. Adicionalmente, a fronteira de Neumann é definida em $\Gamma_N = \{x_1\} \times (y_0, y_1) \cup \{y_1\} \times (x_0, x_1)$ por $g = \mathbf{n} \cdot \mathbf{D} \nabla C$, a função

de fonte é dada por $f(x, y, t) = 0$ e a condição inicial é

$$C_0(x, y, t_0 = 0.0) = \begin{cases} \frac{1}{\pi \cdot R^2}, & \text{se } (x - x_0)^2 + (y - y_0)^2 \leq R^2 \\ 0, & \text{caso contrário} \end{cases} \quad (5.7)$$

onde (x_0, y_0) é ponto no domínio Ω e R uma constante.

Como a fonte de contaminantes é pequena em relação ao domínio computacional, então a solução analítica, apresentada em [5.1.2], é considerada adequada para esse problema.

A aproximação de elementos finitos considera uma malha com $Ne = 80000$ elementos finitos triangulares, $Nv = 40401$ vértices e orientação *left/right*. Os parâmetros para implementação são $n_x = n_y = 200$ elementos finitos triangulares em cada direção coordenada, funções lagrangeanas de ordem 1, $\theta = 1/2$, $\tau_n = 0.0001d$, $t_{final} = 0.1d$, $n = \{1, 2, \dots, 1000\}$ e constantes $u_0 = 0.5$, $D_0 = 2.0$.

A figura (20) representa as componentes do campo de velocidades, a condição inicial C_0 e as fronteiras de Dirichlet e Neumann. Nesse caso, o campo de velocidades é representado exatamente pelas funções lineares. No entanto, a variação quadrática das dispersões pode não ser adequadamente capturada.

Figura 20 – Campo de velocidades não uniforme $\mathbf{v} = (v_x, v_y) = (0.5(x + 1.0), -0.5(y + 1.0))$, condição inicial e condições de Dirichlet e Neumann

Qualitativamente, os resultados numéricos são semelhantes aos resultados obtidos por Zoppou e Knight (1999). A comparação das soluções numérica e analítica fornece um erro máximo $E_{max} = \max(Erro^{1000}) = 0.0009288$ e um erro médio $E_{medio} = 4.6092 \cdot 10^{-05}$. Esse erro máximo representa, aproximadamente, $1.79\%(0.0009288/0.05181 = 0.0179)$ do valor da concentração naquela coordenada.

A tabela [4] fornece as soluções numérica e analítica e o erro real em um conjunto de pontos no 1000-ésimo instante de tempo do processo de simulação.

Tabela 4 – As soluções numérica e analítica e o erro real no 1000-ésimo instante de tempo do processo de simulação

(x, y)	C^{1000}	C_A	$Erro^{1000}$
(4.4,3.6)	0.0461	0.0452	0.0009
(4.6,3.6)	0.0483	0.0474	0.0009
(4.3,3.7)	0.0473	0.0464	0.0009
(4.5,3.7)	0.0501	0.0492	0.0009
(4.7,3.7)	0.0519	0.0510	0.0009
(4.2,3.8)	0.0479	0.0470	0.0009
(4.4,3.8)	0.0513	0.0504	0.0009
(4.6,3.8)	0.0537	0.0528	0.0009
(4.8,3.8)	0.0551	0.0542	0.0009
(4.1,3.9)	0.0477	0.0468	0.0009
(4.3,3.9)	0.0518	0.0509	0.0009
(4.5,3.9)	0.0549	0.0539	0.0009
(4.7,3.9)	0.0569	0.0559	0.0009
(4.0,4.0)	0.0469	0.0460	0.0009
(4.2,4.0)	0.0516	0.0507	0.0009
(4.4,4.0)	0.0553	0.0544	0.0009
(4.6,4.0)	0.0579	0.0570	0.0009
(4.1,4.1)	0.0507	0.0498	0.0009
(4.3,4.1)	0.0551	0.0541	0.0009
(4.5,4.1)	0.0583	0.0574	0.0009
(4.0,4.2)	0.0492	0.0483	0.0009
(4.2,4.2)	0.0541	0.0532	0.0009
(4.4,4.2)	0.0580	0.0571	0.0009
(4.1,4.3)	0.0525	0.0516	0.0009
(4.3,4.3)	0.0570	0.0561	0.0009

As figuras (21-a,b) apresentam as soluções analítica e numérica juntamente com as

respectivas curvas de nível.

(a) Solução analítica e isolinhas nas direções coordenadas

(b) Solução numérica e isolinhas nas direções coordenadas

Figura 21 – As soluções numérica e analítica em malha de elementos finitos e funções lineares

Qualitativamente, a solução numérica é adequada ao problema proposto. Uma análise

quantitativa, por meio do mapa de erros (figura 22), mostra que a solução numérica captura adequadamente o comportamento da solução analítica.

Figura 22 – O erro entre as soluções numérica e analítica $E = \text{Erro} = |C_A - C^{1000}|$

Figura 23 – Perfis de concentração para as soluções numérica e analítica

O mapa de erros e as respectivas curvas de nível mostram que a solução numérica e analítica possuem mesmos pontos de máximos e, além disso, o avanço da frente de contaminantes é capturado pela solução numérica. Essa adequação pode ser observada através dos perfis de concentração, apresentados nas figuras (23).

A análise dos perfis de concentração, apresentados nas figuras (23-a,b), mostram que a forma da solução analítica é capturada adequadamente pela solução numérica. Além disso, a presença do campo de velocidades não uniforme e não constante influencia na forma da frente de contaminação e na velocidade da dispersão em cada direção e ambos os efeitos são capturados pelo código numérico.

A próxima seção discute a projeção L^2 . Esse tópico é importante para as estimativas residuais e, além disso, introduz a questão da representação adequada das expressões analíticas.

5.1.3 Projeção L^2

A obtenção das estimativas residuais e, conseqüentemente, dos indicadores residuais exige projeção L^2 de funções sobre o espaço de elementos finitos. Este exemplo discute a projeção L^2 , a representação das funções no espaço de elementos finitos e a representação analítica da função. O termo analítica refere-se a função que é representada em uma malha refinada ou com funções lagrangeanas de ordem superior àquela utilizada na representação L^2 . Para esta discussão considera-se os seguintes problemas:

Problema 01: Ilustrar a representação da função $f(x) = x^2$ utilizando diferentes malhas de elementos finitos.

Ilustrativamente, a figura [24] apresenta a representação da função $f(x) = x^2$ em diferentes malhas de elementos finitos com funções lagrangeanas lineares. A função $f_{Malha01}$ é a aproximação da função $f(x) = x^2$ em uma malha, *Malha01*, com $n_x = n_y = 2$ elementos em cada direção coordenada, a função $f_{Malha02}$ é a aproximação da função f em uma malha, *Malha02*, obtida por meio de um refinamento uniforme de *Malha01* e a função $f_{Malha03}$ é a aproximação obtida por meio de 04 refinamentos uniformes e sucessivos da *Malha01*. Nesse caso, a aproximação $f_{Malha03}$ é considerada uma aproximação adequada para representação da expressão analítica $f(x) = x^2$.

Problema 02: Obter a projeção L^2 da função $f(x, y) = 1 + x^2 + y^2$ em um domínio $\Omega = [0, 32] \times [0, 32]$ com $n_e = n_x = n_y = 8$ elementos finitos triangulares com orientação *left/right* e funções lagrangeanas lineares. Fornecer uma representação gráfica do erro associado.

Neste caso, primeiramente é necessário resolver um sistema linear da forma $Ax = b$.

2. Este sistema é gerado pela adequada definição das formas bilinear a e linear L .

Figura 24 – A expressão analítica avaliada em três malhas diferentes: *Malha01* possui dois elementos em cada direção coordenada, *Malha02* foi obtida por refinamento uniforme da *Malha01* e *Malha03* foi obtida por 04 refinamentos uniformes e sucessivos da *Malha01*. A função $f_{Malha03}$ é considerada adequada para a representação da expressão analítica de $f(x) = x^2$

A solução³ (FENICS PROJECT, 2014) fornece os valores nodais, os quais representam a projeção L^2 . Esses valores são interpolados e o resultado é apresentado na figura (25-a). Essa figura é a projeção L^2 da função $f(x, y) = 1 + x^2 + y^2$ conforme especificado pelo problema.

A comparação dos resultados é feita na malha para representação analítica da função f , ou seja, a malha mais refinada ou com funções de ordem superior.

A figura (25-b) apresenta o erro da aproximação da função f utilizando malha refinada.

A malha refinada foi obtida da malha inicial através de quatro refinamentos uniformes sucessivos. A malha inicial possui $n_e = 128$ elementos com $n_v = 81$ vértices, enquanto a malha refinada possui $n_e = 32768$ elementos com $n_v = 16641$. Os sistemas lineares respectivos possuem 81 e 16641 variáveis.

O erro de representação é calculado através da diferença entre as funções, ou seja, $E = Erro = f(x, y) - f_{L^2}(x, y)$, em que f é a representação da função analítica em malha refinada e f_{L^2} é a projeção L^2 .

3. O método adotado é a fatoração **LU**, o qual está disponível na biblioteca numérica **dolfin/FEniCS**

(a) f_{L^2} : a projeção L^2 da função $f(x,y) = 1 + x^2 + y^2$

(b) Erro(E), $E = |f(x,y) - f_{L^2}|$

Figura 25 – A f_{L^2} e o erro de representação da função f pela função f_{L^2}

Esse exemplo fornece a projeção L^2 da função f , obtida por meio do método da

fatoração LU ⁴. Nesse caso, o sistema linear inicial possui $n = 81$ variáveis enquanto o sistema associado à malha refinada possui 16641 variáveis. Para domínios grandes ou processos de refinamento adaptativo, o crescimento do número de variáveis pode inviabilizar a utilização de alguns métodos numéricos diretos ou iterativos. Para tornar possível a utilização de outros métodos numéricos, é necessário disponibilizar a matriz A e o vetor b .

Considerando a utilização da biblioteca PyAMG (BELL; OLSON; SCHRODER, 2011), a função Python `projectionFunction(kargs**)`, descrita no apêndice [A.1], retorna o sistema linear definido pelas formas bilineares a e L ⁵. Isso facilita a utilização de outros métodos numéricos para a solução do sistema linear associado a projeção L^2 .

A próxima seção considera a construção de malhas adequadas à representação da condição inicial $C_0 = C(x, y, 0)$ e do termo de fonte f .

5.2 Malhas adequadas à representação dos dados

Esta seção descreve duas aplicações da técnica adaptativa, baseada nas componentes η_f e $\|C_0 - \pi C_0\|_{L^2(\Omega)}$ do estimador residual, para os dados do problema do transporte de contaminantes. O primeiro exemplo considera a utilização da estimativa residual da condição inicial, ou seja, a malha retornada é adequada à representação da condição inicial do problema. O segundo exemplo considera a estimativa η_f para gerar uma malha adequada à representação do termo de fonte.

5.2.1 Exemplo 03: Malha adequada à Condição Inicial

Este exemplo considera uma modificação do problema apresentado na seção [5.1.2] de modo que as condição (P1) seja satisfeita. Consequentemente, as condições (P1)-(P4*) são satisfeitas e o estimador residual de Praetorius, Weinmuller e Wissgott (2008) pode ser utilizado.

Considere um domínio retangular $\Omega = [x_0, x_1] \times [y_0, y_1] = [0, 20] \times [0, 20]$ com campo de velocidade variável v e matriz de dispersão $D = D(x, y)$. Sejam $u_0 = 0.5$, $D_0 = 2.0$ constantes e defina o campo de velocidades por $v = (v_x, v_y) = (u_0(x + 1.0), -u_0(y + 1))$. Além disso, considere que os coeficientes da matriz de dispersão são dados por $d_{11} = d_{11}(x, y) = D_0 u_0^2 (x + 1)^2$, $d_{12}(x, y) = d_{21}(x, y) = 0$ e $d_{22} = d_{22}(x, y) = D_0 u_0^2 (y + 1)^2$.

4. Disponível em FEniCS (FENICS PROJECT, 2014)

5. Associadas à projeção L^2

Adicionalmente, considere que a condição inicial é

$$\frac{C_0(x, y, t_0 = 0)}{C_{C_0}} = \exp \left(-\frac{(x - \mu_1)^2}{2\sigma_1^2} - \frac{(y - \mu_2)^2}{2\sigma_2^2} \right), \quad (5.8)$$

onde $\mu_1 = 5.0, \mu_2 = 5.0, \sigma_1^2 = \sigma_2^2 = 0.5$ e C_{C_0} é uma constante.

Neste caso, os dados não são constantes, e uma representação adequada deve ser construída. Aqui, em vez de adotar uma malha uniforme conforme adotado na seção [5.1.2], a função `RefineMesh_for_InitCond(kargs**)`, baseada na estimativa residual para a condição inicial $\|C_0 - \pi C_0\|_{L^2(\Omega)}^2$, é utilizada para a construção de uma malha adequada aos dados do problema do transporte. Adicionalmente, essa função é utilizada para d_{xx} , d_{yy} , carga hidráulica $H := u_0/2 \cdot (x + 1.0)^2 - u_0/2 \cdot (y + 1.0)^2$, v_x, v_y e C_0 .

As figuras (26), (27) e (28) mostram a malha grossa T'_0 fornecida pelo usuário, a malha gerada pela aplicação do processo adaptativo e a solução numérica em malha adaptada.

Figura 26 – Malha T'_0 fornecida pelo usuário antes do processo adaptativo

Figura 27 – Malha T_0 adequada aos dados gerada pelo procedimento adaptativo

Figura 28 – Solução numérica em malha adequada à representação dos dados

A malha adequada T_0 aos dados, de acordo com o critério de parada estabelecido, é composta de $Nv = 4381$ vértices com $Nv = 8504$ elementos. Essa malha é

tal que $\|X - \pi X\|_{L^2} < \text{toler} = 0.25$, em que X refere-se à $d_{xx}, d_{yy}, f, v_x, v_y$, carga hidráulica H e $\|C_0 - \pi C_0\| < 0.01$, respectivamente. Os demais parâmetros para a função `RefineMesh_for_InitCond(kargs**)` são $\text{fraction} = 0.5$, $\text{Degree} = 1$ e $\text{RaiseDegree} = 1$. A solução numérica, apresentada na figura (28), considera uma malha T_2 , a qual é obtida através de um refinamento uniforme da malha adaptada aos dados T_1 , funções lagrangeanas lineares, $\theta = 1.0$, $\tau_n = 0.001d$ e $t_{final} = 0.1d$.

Os resultados em malha T_2 são comparados àqueles obtidos em uma malha uniforme definida por $n_x = n_y = 200$ elementos em cada direção coordenada com orientação `left/right` e parâmetros idênticos. A malha uniforme possui $Nv = 40401$ vértices e $Ne = 80000$ elementos, enquanto que a malha T_2 possui $Nv = 17265$ vértices e $Ne = 34016$ elementos. Essa comparação é feita através da projeção dos resultados em malha T_2 sobre a malha uniforme.

Denotando C_{T_2} a solução numérica na malha T_2 , $C_{Uniforme}$ a solução numérica em malha uniforme e $C_{Projetada}$ a projeção da solução C_{T_2} sobre a malha uniforme. A análise dos resultados mostra que a máxima diferença entre as soluções é $|C_{T_1} - C_{Uniforme}| = 0.0009352$. Além disso, os valores máximos das concentrações nas malhas T_1 e uniforme são produzidos no ponto (5.0, 4.4).

A análise dos resultados mostra que a malha adaptada fornece resultados similares àqueles fornecidos pela malha uniforme. No entanto, a quantidade de elementos e, conseqüentemente, o tamanho do sistema linear é reduzido.

5.2.1.1 Discussão sobre as estimativas residuais

Nesse caso, as condições (P1)-(P4) e (P1)-(P4*) são satisfeitas. Logo, o estimador residual pode incorporar todos os termos apresentados na expressão (4.37). Essa estimativa pode ser simplificada, pois:

- i. a representação da condição de Neumann, dada numericamente por $g = \mathbf{n} \cdot \mathbf{D}\nabla(C)$, possui representação exata e, portanto, $\eta_N = 0$.
- ii. a equação do transporte não possui termo de fonte e, portanto, $\eta_f = 0$
- iii. o regime é de advecção dominante pois $\varepsilon = 0.5$, $\kappa = 220.5/0.5 = 441$, $\beta = 0$ e $C_c = \max(|v|)/\varepsilon = ((u_0(20+1))^2 + (-u_0(20+1))^2)^{1/2}/\varepsilon \approx 29.7$.
- iv. os dados não possuem dependência espacial, ou seja, $\dot{f}(t) = 0$, $\dot{D}(t) = 0$, $\dot{\lambda}(t) = 0$ e $\dot{\mathbf{v}} = 0$.
- v. devido às considerações [iii.] e [iv.], a estimativa temporal é dada por:

$$\begin{aligned}
\left(\eta_\tau^{(n)}\right)^2 &= |||C^n - C^{n-1}|||^2 + |||v \cdot \nabla(C^n - C^{n-1})|||_*^2 + \\
&= |||C^n - C^{n-1}|||^2 + |||C^n - C^{n-1}|||^2 + \left(\tilde{\eta}^{(n)}\right)^2 + |||\tilde{C}^n|||^2 + \\
&\quad (1 + \theta)^2 \tau_n \left[\int_{t_{n-1}}^{t_n} \|\dot{g}(t)_{\Gamma_N}\|^2 dt \right] \approx |||C^n - C^{n-1}|||^2 + \left(\tilde{\eta}^{(n)}\right)^2 + |||\tilde{C}^n|||^2 \quad (5.9)
\end{aligned}$$

em que a última simplificação considera que $\dot{g}(t) \approx 0$. Essa consideração é baseada no seguinte argumento:

$$\begin{aligned}
\dot{g}(t) \approx \frac{g(t_n) - g(t_{n-1})}{\tau_n} &= \frac{\mathbf{n} \cdot D^n \nabla(C^n) - \mathbf{n} \cdot D^{n-1} \nabla(C^{n-1})}{\tau_n} = \\
&\quad \frac{\mathbf{n} \cdot D \nabla(C^n - C^{n-1})}{\tau_n} \approx 0 \quad (5.10)
\end{aligned}$$

Dessa forma, a expressão (4.37) pode ser reescrita como:

$$(\eta_I) = \left\{ \|C_0 - \pi C_0\|_{L^2(\Omega)}^2 + \sum_{i=1}^{N_I} \tau_n \left[\left(\eta^{(n)}\right)^2 + \left(\Theta^{(n)}\right)^2 + \left(\eta_\tau^{(n)}\right)^2 \right] \right\}^{1/2}. \quad (5.11)$$

Esse estimador é mais geral do que o estimador de [Verfürth \(2004\)](#) no sentido que pode englobá-lo. Esses resultados são obtidos por meio de convenientes modificações das classes e funções provenientes das estimativas em dispersão dominante.

Os avanços computacionais em relação aos resultados apresentados em [Praetorius, Weinmuller e Wissgott \(2008\)](#) residem na facilidade de implementação das estimativas, na consideração de problemas envolvendo dados não constantes, na utilização de funções de ordem superior e no procedimento computacional para a estimativa das constantes ε e β . No entanto, o processo de refinamento não é considerado aqui.

As figuras (29, 30) e (31, 32) apresentam os indicadores espacial, do elemento, do salto e da fronteiras de Neumann em cada elemento no domínio computacional na malha adaptada. Essas quantidades fornecem valores que possibilitam avaliar a região que ainda necessita refinamento.

Os indicadores espaciais capturam o comportamento da função de distribuição de concentrações. Devido às magnitudes envolvidas nos indicadores residuais, um procedimento adaptativo pode ser utilizado para melhorar a qualidade da solução numérica nesse passo de tempo. Esse procedimento faz parte de um algoritmo adaptativo geral, o qual não é discutido nessa tese. No entanto, a utilização de uma malha inicial adequada à representação dos dados favorece a aplicação de um algoritmo adaptativo geral, pois reduz a quantidade de elementos necessários à adequada representação dos dados.

Figura 29 – O indicador espacial η_K em cada elemento da malha refinada no 1000-ésimo instante de tempo

Figura 30 – O indicador do elemento η_{1K} em cada elemento da malha refinada no 1000-ésimo instante de tempo

Figura 31 – O indicador do salto η_{2E} em cada elemento da malha refinada no 1000-ésimo instante de tempo

Figura 32 – O indicador temporal $\eta_{\tau_{1000}}$ em cada elemento da malha refinada

5.2.2 Exemplo 04: Malha adequada à representação da função de fonte

O objetivo desta parte é exemplificar a utilização do algoritmo adaptativo baseado na estimativa residual da função f .

Considere um domínio retangular $\Omega = [x_0, x_1] \times [y_0, y_1] = [0, 1000] \times [0, 1000]$ com campo de velocidade variável v e matriz de dispersão $D = D(x, y)$. Sejam u_0, D_0 constantes e defina o campo de velocidades por $v = (v_x, v_y) = (ax + b_1, -ay + b_2)$. Considere que os coeficientes da matriz de dispersão são dependentes do espaço e definidos por

$$\begin{aligned} d_{xx} &= d_{11}(x, y) = D_0 + E_{pL} \frac{v_x^2}{|v|} + E_{pTh} \frac{v_y^2}{|v|} \\ d_{yy} &= d_{11}(x, y) = D_0 + E_{pL} \frac{v_y^2}{|v|} + E_{pTh} \frac{v_x^2}{|v|} \\ d_{xy}(x, y) &= d_{yx}(x, y) = D_0 + (E_{pL} - E_{pTh}) \frac{v_x v_y}{|v|} \end{aligned} \quad (5.12)$$

em que D_0, E_{pL}, E_{pT} são constantes (BEAR, 1979).

A condição de Dirichlet é definida em $\Gamma_D = \{x_0\} \cup \{y_0, y_1\}$, por $f_{\Gamma_D}(x_0, y) = C_{\Gamma_D} = 0.0$, enquanto que a condição de Neumann é definida nas demais fronteiras. Adicionalmente, considere que o termo de fonte é dado por uma função exponencial bidimensional definida por

$$f(x, y) = C_{f_0} \exp \left(-\frac{(x - \mu_1)^2}{2\sigma_1^2} - \frac{(y - \mu_2)^2}{2\sigma_2^2} \right) \quad (5.13)$$

onde $C_{f_0} = 1.0$,

Nesse problema, é considerado que a malha inicial possui $n_x = n_y = 2$ elementos triangulares nas direções coordenadas com orientação `left/right`, $E_{pL} = 10.0$, $E_{pTh} = 1.40$, $D_0 = 0.0$, $\varepsilon = 0.280$, $\kappa = 5.83$, $\beta = 0.0^6$, $a = 0.003$, $b_1 = 0.2$ e $b_2 = 0.0$.

Analogamente ao caso anterior, a representação dos dados não é adequada e uma malha refinada precisa ser obtida. Na realidade, a malha inicial é inadequada para todos os propósitos de aproximação pelo método de elementos finitos. No entanto, o esquema adaptativo consegue produzir uma malha refinada adequada aos dados baseado na estimativa residual da função f .

A função `RefineMeshForFfunction(kargs**)` implementa o esquema adaptativo baseado na estimativa residual para o termo de fonte. Os resultados das figuras (33-a,b) mostram a malha inicial T'_0 e a malha refinada, obtida por aplicações sucessivas procedimento adaptativo aos dados do problema.

6. O cálculo dos valores ε , κ e β é discutido em detalhes em [4.5.2]

(a) Malha inicial T'_0 fornecida pelo usuário

(b) Malha gerada pelo procedimento adaptativo

Figura 33 – Malha inicial T'_0 e malha adaptada adequada aos dados d_{xx} , $d_{xy} = d_{yx}$, d_{yy} , f_{Γ_D} , f .

As figuras (34-a,b) mostram a dispersão d_{yy} , a função que define a fronteira de Dirichlet f_{Γ_D} e o termo de fonte f em malha adaptada.

(a) A dispersão d_{yy} em malha adaptada

(b) A condição de Dirichlet e termo de fonte em malha adaptada

Figura 34 – A representação, em malha adaptada, das funções d_{yy} , da condição de Dirichlet f_{Γ_D} e do termo de fonte $f(x, y)$.

A malha inicial T'_0 possui $n_e = 08$ elementos triangulares com $n_v = 09$ vértices e a malha adaptada possui $n_e =$ elementos triangulares com $n_v =$ vértices. A malha adaptada é tal que $\eta_f < toler = 1.0$, em que X refere-se à d_{xx} , d_{yy} , $d_{xy} = d_{yx}$, f_{Γ_D} e f . Os demais parâmetros para a função `RefineMeshForFfunction` são $maxIterF = 1000$, $toleranceF = 1.0$, $fractionF = 0.5$, $degree = 1$, $raiseDegree = 1$ e $dt1 = 1.0$.

Analogamente ao caso anterior, para comprovar a adequação da malha adaptada, uma malha uniforme com $n_x = n_y = 400$ elementos em cada direção coordenada e mesma orientação da malha inicial é utilizada. A malha uniforme possui $n_e = 320000$ elementos triangulares com $n_v = 160801$ vértices e é considerada adequada para a representação analítica de cada um dos dados da equação. Os dados, na malha inicial e adaptada, são então projetados na malha uniforme e as diferenças entre esses resultados fornecem os erros reais de representação $E_{d_{xx}}$, $E_{d_{yy}}$, $E_{d_{xy}} = E_{d_{yx}}$, $E_{f_{\Gamma_D}}$ e E_f em cada ponto do domínio computacional. Matematicamente, essas diferenças são definidas por:

$$E_{XX}^0 = |XX^{inicial} - XX^{analítica}|, \quad E_{XX}^1 = |XX^{adaptada} - XX^{analítica}|, \quad (5.14)$$

onde XX denota d_{xx} , d_{yy} , d_{xy} , d_{yx} , f ou f_{Γ_D} , $XX^{analítica}$ é a representação analítica de XX e $XX^{adaptada}$ denota a utilização da malha adaptada.

A análise dos resultados fornece $\max(E_{d_{xx}}^0) = 0.4439$, $\max(E_{d_{xx}}^1) = 0.0022$, $\max(E_{d_{xy}}^0) = \max(E_{d_{yx}}^0) = 1.2332$, $\max(E_{d_{xy}}^1) = \max(E_{d_{yx}}^1) = 0.0020$, $\max(E_{d_{yy}}^0) = 1.2190$, $\max(E_{d_{yy}}^1) = 0.0068$, $\max(E_{f_{\Gamma_D}}^0) = 0.9949$, $\max(E_{f_{\Gamma_D}}^1) = 0.0030$, $\max(E_f^0) = 0.7829$ e $\max(E_f^1) = 0.0035$.

Novamente, o esquema adaptativo produz uma malha adequada à representação dos dados. Em todos os casos, o erro produzido pelo procedimento adaptativo é menor que o erro associado à malha inicial.

5.3 Métodos Multigrid Acelerados

Esta seção considera o tempo gasto na solução do sistema linear associado à aproximação da equação do transporte de contaminantes. Cada exemplo considera três métodos iterativos para a solução do sistema linear associado. O primeiro é o método padrão⁷, enquanto que o segundo e terceiro são os métodos otimizados, os quais reutilizam as estruturas multigrid para acelerar o processo de solução. A comparação das soluções numéricas, o cálculo do tempo computacional e o cálculo do tempo total seguem a metodologia descrita no capítulo de metodologia [4].

Os exemplos apresentados consideram tanto os métodos multigrid algébricos quanto os métodos não estacionários pré-condicionados por multigrid.

7. Método padrão refere-se ao método cuja estrutura hierárquica será reutilizada

5.3.1 BICGSTAB Pré-Condicionado SA

Considere o problema do transporte de contaminantes em um domínio retangular $\Omega = [x_0, x_1] \times [y_0, y_1]$ com fonte de contaminante no interior do domínio computacional. Esse problema é uma adaptação de Hofinger e Judex (2005) e considera o domínio computacional $\Omega = [x_0, x_1] \times [y_0, y_1] = [0, 80] \times [0, 40]m^2$, e a fonte de contaminantes

$$\frac{f(x, y)}{f_0} = \exp\left(-\frac{1}{2} \frac{(x - \mu_1)^2}{\sigma_1^2} - \frac{1}{2} \frac{(y - \mu_2)^2}{\sigma_2^2}\right), \quad (5.15)$$

onde $\mu_1 = 10.0, \mu_2 = 2.0, \sigma_1^2 = \sigma_2^2 = 2.0$ e f_0 é constante para normalização; (iii) a fronteira de Neumann definida em $\Gamma_N = \{x_1\} \times (y_0, y_1)$ por $g = \mathbf{n} \cdot \mathbf{D} \nabla C$; (iv) a fronteira de Dirichlet definida em $\Gamma_D = \{x_0\} \times (y_0, y_1) \cup \{y_0\} \times (x_0, x_1) \cup \{y_1\} \times (x_0, x_1)$ por $C = 0$; (v) a condição inicial é dada por $C_0 = C(x, y, t_0) = 0.0$. A difusividade, o campo de velocidades e matriz de dispersão são, respectivamente, $\alpha_D = 0.05m$, $v = (v_x = 0.864, v_y = 0.0) \frac{m}{d}$ e $D = \alpha_D v_x I_{2 \times 2}$, em que $I_{2 \times 2}$ é a matriz identidade.

A figura (35) ilustra o domínio computacional, as curvas de nível para o termo de fonte, o campo de velocidades e as condições de Dirichlet e Neumann.

Figura 35 – O domínio computacional, as curvas de nível do termo de fonte, o campo de velocidades e as condições de Dirichlet e Neumann

A figura (36) apresenta a solução numérica, obtida pelo método *BICGSTAB* pré-condicionado pelo método *SA* sem reutilização, para uma malha com $n_x = 2n_y = 200$

elementos triangulares, $\theta = 1/2$, funções lagrangeanas lineares, $\tau_n = 0.5d$, $t_{final} = 60d$. Esse resultado mostra a distribuição de concentrações, $C^{120} = C(x, y, t_{120})$, que fornece a concentração em cada ponto do domínio computacional no instante de tempo $n = 120$. A avaliação mais detalhada do avanço da frente de contaminação pode ser visualizada através das curvas de nível, apresentadas na figura (37).

Figura 36 – A solução numérica em malha estruturada com funções lagrangeanas lineares

Figura 37 – As curvas de nível da solução numérica apresentada na figura (36)

Nos próximos resultados, a solução numérica é calculada pelo método do Gradiente Bi-Conjugado Estabilizado (*BICGSTAB*) pré-condicionado por *SA*, *SA - Reu* e *SA - ReuAuto*. De forma simplificada, denota-se esses métodos por *BICGSTAB_{SA}*, *BICGSTAB_{SAReu}* e *BICGSTAB_{SAReuAuto}*.

5.3.1.1 Malha Estruturada

O exemplo considera uma malha triangular uniforme com $2n_y = n_x = 1600$ elementos em cada direção coordenada e orientação *left / right* (FENICS PROJECT, 2014). A malha tem $Nv = 1282401$ vértices e $Ne = 2560000$ elementos triangulares. A aproximação de elementos finitos considera: i. passo de tempo fixo $\tau_n = \tau = 0.500d$; ii. tempo final de simulação $t_{final} = 20.0d$; iii. parâmetro de discretização temporal $\theta = 1.0/2.0$; iv. Funções lagrangeanas lineares.

Nesse caso, os seguintes parâmetros para o método multigrid algébrico *SA*, passados via dicionário de parâmetros, foram utilizados.

```

1 config=solver_configuration(Asp,verb=True)
  config['B'] = np.ones((Asp.shape[0],1), dtype=Asp.dtype)
3 config['BH'] = config['B'].copy()
  config['smooth']=(None,{'gauss_seidel'},{'iterations':2})
5 config['strength'] = [( 'classical' ,{'theta':0.1}),('classical' ,{'theta':0.1})]
  config['max_levels']=(max_levels)
7 config['max_coarse']=(10000)
  config['coarse_solver']='gauss_seidel'
9 config['aggregate']='standard'
  config['presmoothen'] = ('gauss_seidel', {'sweep':'symmetric', 'iterations':1})
11 config['postsmoothen'] = ('gauss_seidel', {'sweep':'symmetric', 'iterations':3})
  config['SA_solve_args']=(({'cycle':'V'},{'tol':1.0000000001e-8}))
13 config['improve_candidates']=[('block_gauss_seidel',{'sweep':'symmetric','iterations':0}),None]
  config['diagonal_dominance']=(False)
15 config['keep'] =(False)

```

A comparação da soluções fornece que $Diff_{SAReu/SA}^n = 0$ e $Diff_{SAReuAuto/SA}^n = 0$, para todos os passos de tempo $n \in \{1, 2, \dots, N_I = 40\}$. Como consequência direta, os métodos modificados produzem as mesmas soluções que o método padrão.

Os resultados envolvendo os resíduos da solução e os resíduos associados a cada iteração do método multigrid são apresentados nas figuras (38-a,b,c,d). A figura (38-a) apresenta a norma l^2 do resíduo, o resíduo máximo e o resíduo médio da solução numérica em cada passo de tempo e mostra que o método iterativo fornece um resíduo adequado. A figura (38-b) apresenta a norma l^2 do resíduo res_n^k em cada passo de tempo n e em cada iteração k do método multigrid. A redução do resíduo a cada iteração do método multigrid, ou seja, a sequência de valores $res_n^0, res_n^1, \dots$ pode ser analisada para todos os

passos de tempo. Nesse caso, verifica-se que tanto o método padrão quanto os métodos otimizados produzem a mesma redução do resíduo.

A figura (38-c) mostra a redução relativa do resíduo, ou seja, as sequências $res_n^1/res_n^1, res_n^2/res_n^1, res_n^3/res_n^1, \dots$. Nesse caso, a redução do resíduo é aproximadamente constante em todos os passos da simulação. Esse fato fica evidenciado pela análise de algumas sequências de resíduos relativos, apresentadas na figura (38-d). Ambos os resultados fornecem uma justificativa para eficiência dos métodos acelerados.

(a) Norma l^2 do resíduo, resíduo máximo e média do resíduo da solução para $n \in \{1, 2, \dots, N_I = 40\}$ (b) Resíduo em cada iteração do método multigrid para todo passo de tempo n

(c) Resíduo relativo em cada iteração do método multigrid para todo passo de tempo $n \in \{1, 2, \dots, N_I = 40\}$ (d) res_n^k/res_n^0 para $n = 1, 10, 20, 30, 40$

Figura 38 – Medidas dos resíduos associados à solução numérica

O tempo computacional Δt_n e o tempo total Ta_n para os métodos $BICGSTAB_{SA}$, $BICGSTAB_{SAReu}$ e $BICGSTAB_{SAReuAuto}$ são apresentados nas figuras (39-a) e (39-b), respectivamente. A análise dos resultados mostra que a reutilização das estruturas é uma estratégia efetiva para reduzir o tempo total, o qual é uma consequência da redução do tempo computacional.

Figura 39 – O tempo computacional e o tempo total para os sistemas lineares com $Nv = 1282401$ variáveis

5.3.1.2 Malha não estruturada

Esse exemplo considera uma malha não estruturada com $Nv = 1326721$ vértices e $Ne = 2649088$ elementos triangulares, os quais foram obtidos de uma malha inicial com $l = 1.25$ e quatro refinamentos uniformes e sucessivos. A aproximação de elementos finitos considera: i. passo de tempo fixo $\tau_n = \tau = 0.25d$; ii. tempo final $t_{stop} = 10.0d$; iii. parâmetro de discretização $\theta = 1.0/2.0$; iv. Funções lagrangeanas lineares.

Os seguintes parâmetros para o método multigrid foram utilizados.

```

1  config=solver_configuration(Asp,verb=True)
  config['B'] = np.ones((Asp.shape[0],1), dtype=Asp.dtype)
3  config['BH'] = config['B'].copy()
  config['smooth']=(None,{'krylov':'gmres','maxiter':3,'degree':1,\
5      'weighting':'local'},{'iterations':2})
  config['strength']=[('classical',{'theta':0.2}),('classical',\
7      {'theta':0.2})]
  config['max_levels']=(max_levels); config['max_coarse']=(10000)
9  config['coarse_solver']='gauss_seidel'
  config['aggregate']='standard'
11 config['presmoothing']=(('gauss_seidel',{'sweep':'symmetric','iterations':2}))
  config['postsmoothing']=(('gauss_seidel',{'sweep':'symmetric','iterations':2}))
13 config['SA_solve_args']=(({'cycle':'V'},{'tol':1.0000000001e-8}))
  config['improve_candidates']=[('block_gauss_seidel',{'sweep':'symmetric',\
15      'iterations':0}),None]
  config['diagonal_dominance']=(True)
17 config['keep']=(False)

```

A comparação da soluções fornece que $Diff_{SAReu/SA}^n = 0$ e $Diff_{SAReuAuto/SA}^n = 0$, para todos os passos de tempo $n \in \{1, 2, \dots, N_I\}$. Os resultados envolvendo os resíduos da solução e os resíduos de cada iteração do método multigrid são apresentados nas figuras (40-a,b,c,d).

Figura 40 – Medidas dos resíduos associados à solução numérica

O tempo computacional Δt_n e o tempo total Ta_n são apresentados nas figuras (41-a,b), respectivamente. Analogamente ao caso anterior, a análise dos resultados mostra que a reutilização das estruturas é uma estratégia efetiva para reduzir o tempo computacional e o tempo total. O método automático é mais efetivo, pois realiza menos agregações e cálculos dos operadores de transferência de resultados. Nesse caso, o tempo computacional de ambas as estratégias, $SAReu$ e $SAReuAuto$, representam $385/496 = 0.7762$ e $382.6/496 = 0.7714$ do tempo total do método padrão SA . Novamente, essa é uma diferença significativa para uma classe de métodos que é considerado escalável.

Figura 41 – Os tempos computacional e total para sistema linear com $Nv = 1326721$ variáveis

5.3.2 Multigrid Algébrico SA e $GMRES$ Pré-Condicionado por SA

Nesta seção, os métodos SA , $SAReu$ e $SAReuAuto$ são usados para calcular a solução numérica em malha estruturada, enquanto que o método $GMRES$ pré-condicionado por SA , $SAReu$ e $SAReuAuto$ são usados para o problema em malha não estruturada. Simplicadamente, denota-se os métodos pré-condicionados por $GMRES_{SA}$, $GMRES_{SAReu}$ e $GMRES_{SAReuAuto}$.

Esta seção considera uma adaptação do problema apresentado em Zoppou e Knight (1999). O transporte de contaminantes é definido em um domínio bidimensional retangular $\Omega = [x_0, x_1] \times [y_0, y_1] = [0, 20.0] \times [0, 20.0]$ com um campo variável de velocidades $\mathbf{v}$ e uma matriz de dispersão com dependência espacial $D = D(x, y)$.

Considere que u_0 , D_0 são constantes e defina o campo de velocidades por $\mathbf{v} = (v_x, v_y) = (u_0(x + 1.0), -u_0(y + 1))$, ou seja, a velocidade é uma função linear da variável naquela direção. Considere que as componentes da matriz de dispersão são funções com dependência espacial e definidas por $d_{11}(x, y) = D_0 u_0^2 (x + 1)^2$, $d_{12}(x, y) = d_{21}(x, y) = 0$ e $d_{22}(x, y) = D_0 u_0^2 (y + 1)^2$. Adicionalmente, considere que a fronteira de Neumann é definida em $\Gamma_N = \{x_1\} \times (y_0, y_1) \cup \{y_1\} \times (x_0, x_1)$ por $g = \mathbf{n} \cdot \mathbf{D} \nabla C$, a fonte de contaminantes é dada pela função $f(x, y, t) = 0$ e que a condição inicial é

$$\frac{C_0(x, y, t_0 = 0)}{C_{C_0}} = \exp \left(-\frac{1}{2} \frac{(x - \mu_1)^2}{\sigma_1^2} - \frac{1}{2} \frac{(y - \mu_2)^2}{\sigma_2^2} \right). \quad (5.16)$$

onde C_{C_0} é uma constante, $\mu_1, \mu_2, \sigma_1, \sigma_2$ são parâmetros da função C_0 .

A figura (42) ilustra o domínio computacional, o campo de velocidades, as curvas de nível da condição inicial e as condições de fronteira de Dirichlet e Neumann para

$\mu_1 = \mu_2 = y_1/4$ e $\sigma = 1/2$, $u_0 = 0.1$ e $D_0 = 1.0$.

Figura 42 – O domínio computacional, as curvas de nível para $\frac{C_0}{C_{C_0}}$, o campo de velocidades com $u_0 = 0.1$ e as condições de Dirichlet e Neumann

Figura 43 – O campo de velocidades e as curvas de nível da solução numérica

A figura (43) apresenta a curvas de nível da solução numérica, obtida pelo método *GMRES* pré-condicionado pelo método *SA* sem reutilização, para uma malha não estru-

turada com elementos triangulares com lado $l = 0.25$, $\theta = 1$, funções lagrangeanas lineares, $\tau_n = 0.025d$, $t_{final} = 1.5d$.

5.3.2.1 Malha Estruturada

Este exemplo considera as constantes $u_0 = 0.1$ e $D_0 = 1.0$, uma malha triangular uniforme com $n_y = n_x = 1500$ elementos em cada direção coordenada e orientação *left /right* (FENICS PROJECT, 2014) com $Nv = 2253001$ vértices, $Ne = 4500000$ elementos triangulares, passo de tempo fixo $\tau_n = \tau = 0.005d$, tempo final $t_{final} = 0.25d$, parâmetro de discretização temporal $\theta = 1.0/2.0$, funções lagrangeanas lineares, $C_{C_0} = 1.0$, $\mu_1 = \mu_2 = y_1/4 = 5.0$ e $\sigma_1 = \sigma_2 = 0.50$.

Neste caso, os seguintes parâmetros foram adotados.

```

1 config=solver_configuration(Asp,verb=True)
3 config['B'] = np.ones((Asp.shape[0],1), dtype=Asp.dtype)
  config['BH'] = config['B'].copy()
5 config['smooth']=(None,{'krylov':'gmres'},{'iterations':2})
  config['strength'] = [( 'classical ',{'theta':0.10}),('classical ',
7                      {'theta':0.10})]
  config['max_levels']=(max_levels);config['max_coarse']=(10000)
9 config['coarse_solver']='gauss_seidel'
  config['symmetry']='symmetric'
11 config['aggregate']='standard'
  config['presmoothing'] = ('block_gauss_seidel',{'sweep':
13                      'symmetric', 'iterations':2})
  config['postsmoothing'] = ('block_gauss_seidel',{'sweep':
15                      'symmetric', 'iterations':2})
  config['SA_solve_args']=(({'cycle':'V'},{'tol':1.0000000001e-8}))
17 config['improve_candidates']=[('block_gauss_seidel',{'sweep':
                      'symmetric', 'iterations':0}),None]
19 config['diagonal_dominance']=(False)
  config['keep'] = (False)
21

```

As comparações das soluções mostram que os métodos calculam as mesmas soluções numéricas, ou seja, $Diff_{SAReu/SA}^n = 0$, $Diff_{SAReuAuto/SA}^n = 0$, $ResDiff_{SAReu/SA}^n = 0$ e $ResDiff_{SAReu/SA}^n = 0$ para todos os passos de tempo $n \in \{1, 2, \dots, N_I\}$. Os resultados envolvendo os resíduos das soluções e os resíduos de cada iteração do método são apresentados nas figuras (44-a,b,c,d).

Figura 44 – Resíduos associadas à solução numérica

O tempo computacional Δt_n e o tempo total Ta_n requerido método multigrid algébrico baseado em agregação são apresentados nas figuras (45-a) e (45-b), respectivamente. A análise dos resultados mostra que o método *SA – ReuAuto* é mais rápido devido ao número de vezes que o processo de agregações e cálculos dos operadores de transferência são realizados. No entanto, ambas as estratégias, *SA – Reu* e *SA – ReuAuto*, fornecem um avanço significativo em relação do método padrão *SA*, pois representam $378.85/484.03 = 0.7826$ e $372.05/484.03 = 0.7686$ do tempo.

A análise dos resultados apresentados nas figuras (44) e (45) mostra que a redução no tempo computacional dos métodos com reutilização não ocasiona mudanças na performance dos respectivos métodos.

(a) Tempo Δt_n para resolver o sistema linear em cada passo de tempo $n \in \{1, 2, \dots, N_I\}$ (b) Tempo total $Ta_n = \sum_{i=1}^n \Delta t_i$ até o n -ésimo passo de tempo $n \in \{1, 2, \dots, N_I\}$

Figura 45 – O tempo computacional e o tempo total para um sistema linear com $Nv = 2253001$ variáveis

5.3.2.2 Malha não estruturada: *GMRES* Pré-Condicionado por *SA*

Este exemplo considera constantes $u_0 = 0.1$, $D_0 = 1.0$ e uma malha não estruturada com $N = 1024961$ vértices e $Ne = 2045824$ elementos triangulares, a qual foi obtida de uma malha inicial com $l = 0.25$ e três refinamentos sucessivos. Além disso, passo de tempo fixo $\tau_n = \tau = 0.025d$, tempo final $t_{final} = 1.50d$, parâmetro de discretização $\theta = 1.0$, funções lagrangeanas lineares.

Neste caso, os seguintes parâmetros foram adotados.

```

1 config['B'] = np.ones((Asp.shape[0],1), dtype=Asp.dtype)
2 config['BH'] = config['B'].copy()
3 config['smooth'] = ('energy',{ 'krylov': 'gmres' },{' iterations ':1})
4 config['strength'] = [( ' classical ',{' theta ':0.10} ),
5                       ( ' classical ',{' theta ':0.10} )]
6 config['max_levels']=(max_levels)
7 config['max_coarse']=(10000)
8 config['coarse_solver']='gauss_seidel'
9 config['symmetry']='symmetric'
10 config['aggregate']='standard'
11 config['presmoothing'] = ('block_gauss_seidel',{ 'sweep': 'symmetric', ' iterations ':2})
12 config['postsolving'] = ('block_gauss_seidel',{ 'sweep': 'symmetric', ' iterations ':3})
13 config['SA_solve_args']=(({' cycle ':'V' },{' tol ':1.0000000001e-8}))
14 config['improve_candidates']=[('block_gauss_seidel',{ 'sweep':
15                               'symmetric', ' iterations ':0}),None]
16 config['diagonal_dominance']=(True)
17 config['keep'] = (False)
18

```


Nesse caso, comparação da soluções também fornecem que $Diff_{SAReu/SA}^n = 0$, $Diff_{SAReuAuto/SA}^n = 0$, $ResDiff_{SAReu/SA}^n = 0$ e $ResDiff_{SAReuAuto/SA}^n = 0$ para todos os passos de tempo $n \in \{1, 2, \dots, N_I\}$. Os resultados envolvendo os resíduos das soluções e os resíduos de cada iteração do método são apresentados nas figuras (46-a,b,c,d).

(a) Norma l^2 do resíduo, máximo e média do resíduo da solução para $n \in \{1, 2, \dots, N_I = 60\}$

(b) Resíduos em cada iteração do método multigrid para todo passo de tempo $n \in \{1, 2, \dots, N_I = 60\}$

(c) Resíduos relativos em cada iteração do método multigrid para todo passo de tempo $n \in \{1, 2, \dots, N_I = 60\}$

(d) res_n^k / res_n^0 para $n = 1, 20, 30, 40, 50, 60$

Figura 46 – Medidas dos resíduos associados à solução numérica

O tempo computacional Δt_n e o tempo total Ta_n para os métodos $GMRES_{SA}$, $GMRES_{SAReu}$ e $GMRES_{SAReuAuto}$ são apresentados nas figuras (47-a,b), respectivamente.

Neste caso, o método $GMRES_{SAReuAuto}$ realiza as agregações e cálculos dos operadores de transferência de resultados em três passos de tempo, enquanto que o método $GMRES_{SAReu}$ executa as operações em seis passos de tempo. O tempo computacional de ambas as estratégias, com reutilização ou reutilização automática, representam $760.88/1184.7 = 0.6422$ e $744.34/1184.7 = 0.6282$ do tempo total do método que não executa a reutilização.

Figura 47 – O tempo computacional e o tempo total para um sistema com $N = 1024961$ variáveis

5.3.3 Malha não estruturada em domínio não retangular

Esta seção considera o transporte de contaminantes em um domínio bidimensional Ω definido pela malha T_0 apresentada na figura (48). Assuma que i. o contaminante é conservativo, ii. condição de Dirichlet é definida sobre toda a fronteira por $C = C_0 = 0.0$ e iii. a condição inicial é dada por

$$C_0(x, y, t = 0.0) = C_{01} \exp\left(-\frac{(x - \mu_{x1})^2}{2\sigma_{x1}^2} - \frac{(x - \mu_{y1})^2}{2\sigma_{y1}^2}\right) + C_{02} \exp\left(-\frac{(x - \mu_{x2})^2}{2\sigma_{x2}^2} - \frac{(x - \mu_{y2})^2}{2\sigma_{y2}^2}\right) + C_{03} \exp\left(-\frac{(x - \mu_{x3})^2}{2\sigma_{x3}^2} - \frac{(x - \mu_{y3})^2}{2\sigma_{y3}^2}\right), \quad (5.17)$$

onde $\mu_{x1} = 400000.00$, $\mu_{y1} = 9000000.00$, $\sigma_{x1} = \sigma_{y1} = 10000.0$, $\mu_{x2} = 400000.00$, $\mu_{y2} = 8700000.00$, $\sigma_{x2} = \sigma_{y2} = 10000.0$, $\mu_{x3} = 500000.00$, $\mu_{y3} = 9200000.00$, $\sigma_{x3} = \sigma_{y3} = 10000.0$, $C_{01} = C_{02} = C_{03} = 1.0$. Considere que a matriz de dispersão possui componentes constantes $d_{11}(x, y) = 50.0m^2/d$, $d_{12}(x, y) = d_{21}(x, y) = 0$, $d_{22}(x, y) = 5.0m^2/d$ e que o campo de velocidades é variável na direção do eixo X e dado por:

$$v_x = \begin{cases} 0.05 & \text{se } x \leq \mu_{x1} \\ -0.04 \frac{x - \mu_{x1}}{\mu_{x3} - \mu_{x1}} + 0.05 & \text{se } \mu_{x1} < x \leq \mu_{x3} \\ 0.01 & \text{se } x > \mu_{x3} \end{cases} \quad (5.18)$$

$$v_y = \begin{cases} 0.05 & \text{se } x \leq \mu_{x1} \\ -0.04 \frac{x - \mu_{x1}}{\mu_{x3} - \mu_{x1}} + 0.05 & \text{se } \mu_{x1} < x \leq \mu_{x3} \\ 0.0 & \text{se } x > \mu_{x3} \end{cases} \quad (5.19)$$

Os resultados apresentados nas figuras (48) e (49) consideram um refinamento uniforme da malha T_0 , $\theta = 1.0$, $\tau_n = 500.0d$ funções lagrangeanas de ordem dois. Adicionalmente, uma translação da malha foi realizada de modo que o ponto com menores coordenadas em T_0 coincida com a origem do sistema de coordenadas cartesianas⁸.

Figura 48 – Malha não-estruturada em domínio não retangular

Figura 49 – A condição inicial em malha tranladada para origem

8. Esse artifício é utilizado para facilitar a apresentação dos resultados gráficos

Os seguintes parâmetros para os métodos multigrid foram adotados.

```

1 config['B'] = np.ones((Asp.shape[0],1), dtype=Asp.dtype)
  config['BH'] = config['B'].copy()
3
  config['smooth'] = ('energy', {'krylov': 'gmres'}, {'filter' : True}, {'sweep': 'symmetric', 'iterations': 2})
5 config['strength'] = ('classical', {'theta': 0.0})
  config['max_levels'] = (max_levels)
7
  config['max_coarse'] = (1000)
9 config['coarse_solver'] = ('gauss_seidel')
  config['symmetry'] = ('symmetric')
11 config['aggregate'] = ('standard')
  config['presmoothing'] = ('gauss_seidel', {'sweep': 'symmetric', 'iterations': 2})
13 config['postsmoothing'] = ('gauss_seidel', {'sweep': 'symmetric', 'iterations': 3})
  config['SA_solve_args'] = ({'cycle': 'V'}, {'tol': 1.000001e-12})
15 config['improve_candidates'] = [(('gauss_seidel', {'sweep': 'symmetric', 'iterations': 2})]
17
  config['diagonal_dominance'] = (True)
  config['keep'] = (False)
19

```

Para a adequada visualização do avanço da frente de contaminantes, os resultados foram projetados sobre uma malhas retangulares e uniformes. A figura (50) mostra as curvas de nível da solução numérica projetada sobre malha uniforme.

Figura 50 – Curvas de nível da função de distribuição de contaminantes em malha transladada para origem

Para o cálculo do tempo computacional, os mesmos parâmetros são adotados, exceto

a malha. Essa, por sua vez, foi obtida por meio de uma sequência de três refinamentos uniformes e sucessivos da malha inicial T_0 , o qual produz uma malha T_1 com $N_e = 792640$ elementos triangulares com $N_v = 412197$ vértices. Nesse caso, os métodos SA , $SA - Reu$ e $SA - ReuAuto$, sem preconditionantes, foram considerados para a solução do sistema linear, o qual possui $N_v = 1617033$ variáveis.

A comparação dos resultados numéricos fornece $Diff_{SAReu/SA}^n = Diff_{SAReuAuto/SA}^n = 0.0$, $ResDiff_{SAReu/SA}^n = ResDiff_{SAReuAuto/SA}^n = 0.0$ para todos os passos de tempo $n \in \{1, 2, \dots, N_I = 100\}$. Os resultados envolvendo os resíduos das soluções e os resíduos de cada iteração do método são apresentados nas figuras (51-a,b,c,d).

(a) Norma l^2 do resíduo, resíduo máximo e média do resíduo da solução para $n \in \{1, 2, \dots, N_I = 100\}$ (b) Resíduos em cada iteração do método multigrid para todo passo de tempo $n \in \{1, 2, \dots, N_I = 100\}$

(c) Resíduos relativos em cada iteração do método multigrid para todo passo de tempo $n \in \{1, 2, \dots, N_I = 100\}$ (d) res_n^k / res_n^0 para $n = 1, 25, 50, 75, 100$

Figura 51 – Medidas dos resíduos associados a solução numérica

O tempo computacional e o tempo total são apresentados nas figuras (52-a,b). Nesse caso, ambos os métodos modificados $SAReu$ e $SAReuAuto$ são efetivos para reduzir o

tempo computacional e representam $2836.9552/4993.2425 = 0.57$ e $2639.7683/4993.2425 = 0.53$ do tempo total gasto pelo método *SA*. Para todos os efeitos, essa redução é significativa.

(a) Tempo computacional Δt_n para $n \in \{1, 2, \dots, N_I\}$ (b) Tempo total $Ta_n = \sum_{i=1}^n \Delta t_i$ para $n \in \{1, 2, \dots, N_I\}$

Figura 52 – O tempo computacional e o tempo total para sistemas lineares com $Nv = 1617033$ variáveis

Devido à relação aproximadamente linear entre o tempo total e o passo de tempo, as retas de regressão de mínimos quadrados r_{SA} , r_{SAReu} e $r_{SAReuAuto}$ foram calculadas por meio do módulo stats do Scipy Jones et al. (2001). A reta de mínimos quadrados, o valor r , o valor p e o erro padrão são apresentados em (5.20), (5.21) e (5.22):

$$r_{SA}(n) = 49.830n - 13.058$$

$$r = 1.0, p_{value} = 0.000, std_{error} = 0.00520, r^2 = 1.0 \quad (5.20)$$

$$r_{SAReu}(n) = 28.284n + 16.689$$

$$r = 1.0, p_{value} = 0.000, std_{error} = 0.0152, r^2 = 1.0 \quad (5.21)$$

$$r_{SAReuAuto}(n) = 26.1040n + 29.76$$

$$r = 1.0, p_{value} = 0.000, std_{value} = 0.009865, r^2 = 1.0 \quad (5.22)$$

A figura (53) ilustra a extrapolação do tempo computacional para um número $n > 100$ de passos de tempo. A análise desses resultados mostra o avanço computacional do método multigrid com reutilização em relação ao método multigrid sem reutilização.

Figura 53 – Extrapolação do tempo computacional para o número de passos de tempo $n > 100$

Até o momento, os exemplos considerados mostram que as soluções numéricas e os resíduos são idênticos e, portanto, tanto o método *SAReu* quanto o método *SAReuAuto* podem ser utilizados. No entanto, o método automático é mais eficiente pois fornece a solução do problema sem, no entanto, escolher o parâmetro ω para a reutilização.

Os próximos resultados conjugam a solução do problema auxiliar, o qual é associado ao problema transiente do transporte nos casos em que o regime é de advecção dominante. Além disso, malhas adaptadas aos dados são consideradas.

5.3.4 Reutilização para malha adaptada aos dados em regime de grande advecção

O exemplo desta seção considera malha adaptada aos dados do transporte de contaminantes. Além disso, considera-se a solução do problema auxiliar, o qual é necessário para a obtenção das estimativas residuais em regime de advecção dominante. A obtenção das estimativas residuais para o regime de advecção dominante foi discutida na seção [5.2.1] e não é discutida nesta parte, pois o objetivo é avaliar a performance do método multigrid para os diferentes problemas envolvidos.

5.3.4.1 Malha não estruturada adaptada aos dados do transporte

Este exemplo considera os dados do problema apresentado na seção (5.3.2) com método *GMRES* pré-condicionado por multigrid algébrico *SA*. A malha utilizada foi obtida por meio dois refinamentos uniformes e sucessivos da malha (54-b) que, por sua vez, foi obtida pelo procedimento adaptativo `RefineMesh_for_InitCond(kargs**)` aplicado aos dados do problema com a malha inicial (54-a). Os resultados numéricos para a solução e as curvas de nível são similares aos apresentados na seção (5.3.2) e não são apresentados nesta seção.

O procedimento adaptativo foi utilizado sequencialmente aos dados $d_{xx} = D_0 \cdot u_0^2(x+1)^2$, $d_{yy} = D_0 \cdot u_0^2(y+1)^2$, à condição inicial $C_0(x, y, t=0)$, ao potencial hidráulico H e às componentes do campo de velocidades $v_x = u_0(x+1)$, $v_y = -u_0(y+1)$, respectivamente. A condição inicial e o potencial hidráulico são dados por:

$$C_0(x, y, t=0) = C_{C_0} \exp \left(-\frac{1}{2} \frac{x - \mu_1}{\sigma_1^2} - \frac{1}{2} \frac{y - \mu_1}{\sigma_2^2} \right), \quad (5.23)$$

e

$$H = \frac{u_0}{2} \cdot (x + 1.0)^2 - \frac{u_0}{2} \cdot (y + 1.0)^2. \quad (5.24)$$

Nesse caso, as constantes $\mu_1 = y_1/4$, $\sigma = 0.5$, $u_0 = 0.1$ e $C_{C_0} = 1$ foram considerados. Os parâmetros abaixo forma considerados para função `RefineMesh_for_InitCond(kargs**)`.

```
2 toldxx=0.005;fraction=0.5;degree=1;RaiseDegree=2
u_0=0.1;D_0=1.0;kappa=4.41;epsilon=0.01;beta=0.0
mu_1=y_1/4; sigma_1=0.5
```

Um exemplo de código computacional para a utilização procedimento adaptativo é fornecido no apêndice [B].

A malha inicial possui $Ne = 4065$ elementos triangulares com $Nv = 2105$ vértices, enquanto que a malha adaptada aos dados possui $Ne = 8978$ elementos triangulares e $Nv = 4617$ vértices. A malha final, utilizada para aproximação do transporte de contaminantes, obtida por dois refinamentos sucessivos da malha refinada, possui $Nv = 72333$ vértices e $Ne = 143648$ elementos triangulares. Os demais parâmetros da aproximação de elementos finitos são $\tau_n = 0.005$, $\theta = 1/2$, $t_{final} = 0.25$ e funções lagrangeanas lineares⁹.

9. O problema auxiliar utiliza os mesmos dados que o problema do transporte

Figura 54 – Malha inicial e malha adaptada aos dados do problema do transporte de contaminantes

Nesse caso, o método *GMRES* pré-condicionado por *SA*, *SA-Reu* e *SA-ReuAuto* foi utilizado tanto para o problema do transporte quanto para o problema auxiliar.

Os seguintes parâmetros para os métodos multigrid foram adotados.

```

1 def DictSA(Asp,max_levels,solver_configuration):
    config=solver_configuration(Asp,verb=True)
3     config['symmetry']='nonsymmetric'
    config['B'] = np.ones((Asp.shape[0],1), dtype=Asp.dtype)
5     config['BH'] = config['B'].copy()
    config['smooth']=(None,{ 'krylov':'big' },{'sweep':'symmetric', 'iterations':2})
7     config['strength'] =('classical ',{'theta':0.1})
    config['max_levels']=(max_levels)
9     config['max_coarse']=(1000)
    config['coarse_solver']='lu'
11    config['symmetry']='symmetric'
    config['aggregate']='standard'
13    config['presmoothing'] = ('block_gauss_seidel', {'sweep':'symmetric', 'iterations':1})
    config['postsmoothing'] = ('block_gauss_seidel', {'sweep':'symmetric', 'iterations':2})
15    config['SA_solve_args']=({'cycle':'V'},{'tol':1.000001e-10})
    config['improve_candidates']=[('block_gauss_seidel', {'sweep':'symmetric', 'iterations':2})]
17    config['diagonal_dominance']=(True)
    config['keep'] =(False)
19    return config

21 def DictSAAuxiliar(AspLarge,max_levels,solver_configuration):
    config2=solver_configuration(AspLarge,verb=True)
23    return config2

```

Diferentemente dos outros casos considerados, as soluções obtidas não são idênticas para todos os passos de tempo. Nesse caso, apenas as soluções para o problema do transporte são idênticas, enquanto que as soluções para o problema auxiliar associado não são idênticas. A tabela [5] os valores máximos sobre todos os passos de tempo tanto para o problema do transporte quanto para o problema auxiliar. Especificamente, o máximo dos resíduos máximos, o máximo dos resíduos médios e o máximo dos resíduos da norma l^2 são apresentados.

A análise dos resultados apresentados na tabela [5] mostra que os métodos modificados fornecem praticamente a mesma solução tanto para problema do transporte quanto para o problema auxiliar. Essa conclusão é baseada nos resultados para os máximos das diferenças para as diversas medidas consideradas. Logo, considera-se que os métodos produzem soluções equivalentes e, portanto, os tempos computacionais podem ser comparados.

Tabela 5 – Valores máximos para todos os passos de tempo $n = \{1, 2, 3, \dots, N_I\}$: máximo dos resíduos máximos, máximo dos resíduos médios e o máximo dos resíduos na norma l^2

Transporte $n = 1, 2, 3, \dots, N_I$	$\max \max()$	$\max_{média}()$	$\max_{norma-l^2}()$
Res_{SA}^n	1.83e-13	2.97e-15	2.86e-12
Res_{SAReu}^n	1.83e-13	2.97e-15	2.86e-12
$Res_{SAReuAuto}^n$	1.83e-13	2.97e-15	2.86e-12
$Diff_{SAReu/SA}^n$	0.0	0.0	0.0
$Diff_{SAReuAuto/SA}^n$	0.0	0.0	0.0
$ResDiff_{SAReu/SA}^n$	0.0	0.0	0.0
$ResDiff_{SAReuAuto/SA}^n$	0.0	0.0	0.0
Auxiliar $n = 1, 2, 3, \dots, N_I$	$\max \max()$	$\max_{média}()$	$\max_{norma-l^2}()$
Res_{SA}^n	4.31e-13	9.16e-15	8.22e-12
Res_{SAReu}^n	4.60e-13	9.71e-15	8.72e-12
$Res_{SAReuAuto}^n$	4.33e-13	9.13e-15	8.20e-12
$Diff_{SAReu/SA}^n$	1.23e-11	3.39e-13	2.57e-10
$Diff_{SAReuAuto/SA}^n$	1.23e-11	3.39e-13	2.57e-10
$ResDiff_{SAReu/SA}^n$	1.29e-13	7.72e-16	7.88e-13
$ResDiff_{SAReuAuto/SA}^n$	1.31e-13	3.24e-16	5.06e-13

Os tempos computacional e total para o problema do transporte e problema auxiliar são apresentados nas figuras (55-a,b) e (55-c,d), respectivamente.

As figuras (56-a,b) apresentam os tempos computacionais combinados, ou seja, a soma dos tempos computacionais provenientes dos problemas do transporte e auxiliar. A análise dos resultados mostra que a reutilização produz um tempo computacional combinado menor que o tempo combinado dos métodos sem reutilização. Esse resultado reforça a tese de reutilizar as estruturas multigrid como forma de acelerar os métodos multigrid algébricos em problemas envolvendo o transporte de contaminantes.

(a) Tempo Δt_n para o sistema linear da equação do transporte $n \in \{1, 2, \dots, N_I\}$ (b) Tempo Δt_n para o sistema linear do problema auxiliar $n \in \{1, 2, \dots, N_I\}$

(c) Tempo total $Ta_n = \sum_{i=1}^n \Delta t_i$ do problema do transporte para $n \in \{1, 2, \dots, N_I\}$ (d) Tempo total $Ta_n = \sum_{i=1}^n \Delta t_i$ do problema auxiliar para $n \in \{1, 2, \dots, N_I\}$

Figura 55 – O tempo computacional e o tempo total para os problemas do transporte e auxiliar. Sistema linear com $Nv = 72333$ vértices

(a) Soma dos tempo de (55-a) e (55-b)

(b) O tempo total obtido de (56-a)

Figura 56 – O tempo computacional e o tempo total combinados para os problemas do transporte e auxiliar. Sistema linear com $Nv = 72333$ variáveis

A próxima seção ilustra a aplicação do procedimento de reutilização para problemas tridimensionais.

5.3.5 A reutilização para o transporte 3D

Esta seção considera uma adaptação do transporte de contaminantes tridimensional apresentado em Parkhurst (PARKHURST D.L.; CHARLTON, 2004), o qual é um problema tridimensional. Nesse caso, fluxo unidimensional na direção x em um domínio computacional Ω com $100m \times 40m \times 24m$ é considerado. Os parâmetros incluem transporte conservativo, velocidade $\mathbf{v} = 0,1m/d$, um coeficiente de dispersividade longitudinal $\alpha_L = 1,5m$, dispersividade horizontal transversal $\alpha_{T_H} = 0,3m$, dispersividade vertical transversal $\alpha_{T_v} = 0,02m$.

A condição inicial é dada por $C(x, y, z, t = 0.0) = 0.0$ e as condições de contorno são dadas por

$$\frac{C(x, y, t)}{C_0} = \begin{cases} 1 & \text{se } |y - 20| \leq \frac{ss}{2} \\ 0 & \text{caso contrário} \end{cases} \quad (5.25)$$

onde ss é uma constante.

Nesse caso, a validação do código numérico considera a solução analítica bidimensional de Wexler (WEXLER, 1992) pois as hipóteses apresentadas em [4.2.3] são satisfeitas. Além disso, utilizar uma solução analítica bidimensional facilita os cálculos.

Os resultados apresentados nas figuras (57-a,b) apresentam as curvas de nível da solução numérica $C(x, y, z = constante, t_{40})$ projetada em uma malha bidimensional com

$n_x = 2n_y = 100$ elementos triangulares nas direções coordenadas. A aproximação de elementos finitos do problema 3D considera uma malha com $Ne = 900000$ tetraedros e $Nv = 159681$ vértices obtida por meio de $n_x = 100$, $n_y = 50$ e $n_z = 30$ elementos em cada direção coordenada, $ss = 5.0$, $\tau_n = 5.00$, $t_{final} = 200.0$, $\theta = 1.0$ e funções lagrangeanas lineares.

(a) Curvas de nível da solução numérica em $z = constante$ (b) Curvas de nível da solução analítica de Wexler

Figura 57 – Curvas de nível das soluções numérica $C(x, y, z = constante, t_{40})$ e curvas de nível da solução analítica de Wexler (WEXLER, 1992)

(a) O erro, E , entre as soluções numérica e analítica (b) Curvas de nível do

Figura 58 – O erro entre as soluções numérica e analítica e as respectivas curvas de nível considerando a solução numérica do problema tridimensional $C_{z=constante}^{40} = C(x, y, z = constante, t_{40})$

Apesar da reduzida quantidade de elementos em cada direção coordenada, a malha de elementos finitos possui uma quantidade significativa de variáveis mostrando a influência do problema tridimensional. As diferenças entre as soluções numérica e analítica são

devidas à representação do termo de fonte no domínio tridimensional. O mapa de erros e respectivas curvas de nível ilustram essa discussão e são apresentados nas figuras (58-a, b).

A avaliação do tempo computacional considera uma malha com $N_e = 7200000$ tetraedros e $N_v = 1238361$ vértices, obtida por meio de dois refinamentos uniformes e sucessivos de uma malha inicial com $n_x = 50, n_y = 25, n_z = 15$ elementos em cada direção coordenada, $\tau_n = 10.0$, $t_{final} = 500.0$, $\theta = 1.0$ e funções lagrangeanas lineares. A comparação dos resultados numéricos fornece $Diff_{SAReu/SA}^n = 0.0$, $Diff_{SAReuAuto/SA}^n = 0.0$, $ResDiff_{SAReu/SA}^n = 0.0$ e $ResDiff_{SAReuAuto/SA}^n = 0.0$ para todos os passos de tempo $n \in \{1, 2, \dots, N_I = 70\}$.

Os seguintes parâmetros para os métodos multigrid foram adotados.

```

1 config=solver_configuration(Asp,verb=False)
3 config['B'] = np.ones((Asp.shape[0],1), dtype=Asp.dtype)
  config['BH'] = config['B'].copy()
5 config['smooth']=(None,{ 'gauss_seidel'},{'iterations':2})
  config['strength'] =('classical ',{'theta':0.1})
7 config['max_levels']=(max_levels)
  config['max_coarse']=(100)
9 config['coarse_solver']='lu'
  config['symmetry']='symmetric'
11 config['aggregate']='standard'
  config['presmoothen'] = ('block_gauss_seidel',
13      {'sweep':'symmetric', 'iterations':2})
  config['postsmoothen'] = ('block_gauss_seidel',
15      {'sweep':'symmetric', 'iterations':3})
  config['SA_solve_args']=(({'cycle':'V'},{'tol':1.000001e-8}))
17 config['improve_candidates']=[('block_gauss_seidel',
      {'sweep': 'symmetric', 'iterations':0}),None]
19 config['diagonal_dominance']=(True)
  config['keep'] =(False)
21

```

O tempo computacional e o tempo total são apresentados, respectivamente, nas figuras (59-a,b), enquanto que as figuras (60-a,b) apresentam as medidas dos resíduos das soluções e a evolução dos resíduos em alguns passos de tempo, respectivamente.

Aqui, o ganho computacional não é tão expressivo, mas isso pode ser consequência dos parâmetros adotados para o método multigrid. No entanto, a análise dos resultados mostra a influência do processo de agregação no tempo computacional e, portanto, ainda reforça a tese do trabalho.

Figura 59 – Medidas dos resíduos associados à solução numérica

Figura 60 – O tempo computacional e o tempo total para sistemas lineares com $Nv = 1238361$ variáveis

6 Conclusão

De posse das estimativas residuais, os indicadores residuais foram calculados e os resultados mostram que o comportamento qualitativo dos erros reais são capturados pelos indicadores das componentes residuais espaciais, dos elementos e dos saltos. No entanto, as magnitudes envolvidas variam e há a predominância, nos casos considerados, das estimativas dos saltos em relação às outras quantidades. O avanço computacional em relação aos resultados apresentados em [Praetorius, Weinmuller e Wissgott \(2008\)](#) residem na facilidade de implementação das estimativas, na consideração de problemas envolvendo dados não constantes, na utilização de funções lagrangeanas de ordem superior e, principalmente, a consideração de problemas diversos envolvendo o transporte de contaminantes em meio poroso saturado.

Estreitamente ligados às estimativas residuais estão os cálculos das constantes ε e β . O desenvolvimento e implementação da metodologia de cálculo das constantes ε e β possibilitou a adequada implementação das estimativas residuais.

A solução da ADR em malha adaptada aos dados produziu uma solução similar àquela dada por uma malha uniforme. Além disso, a malha adequada aos dados possuía, aproximadamente, duas vezes menos elementos. Esse resultado reforça a tese da influência da adequada representação dos dados no processo de solução.

Finalmente, os métodos multigrid algébricos com reutilização das estruturas hierárquicas foram desenvolvidos. Esses métodos incluem:

- i. os métodos multigrid algébricos com reutilização automática ou com parâmetro ω ;
- ii. os métodos iterativos não-estacionários pré-condicionados por multigrid com reutilização automática ou com parâmetro ω .

Ambos os casos i. e ii. foram analisados e os resultados mostraram um ganho computacional significativo tanto para métodos com reutilização automática quanto para os métodos com parâmetro fixo ω . Os métodos automáticos são mais eficientes que os métodos com parâmetro ω no sentido que o critério de decisão é tomado baseado no número de iterações necessárias à convergência. No entanto, independente do esquema adotado, todos resultados apontam que a reutilização é uma ferramenta eficaz para acelerar o cálculo das soluções dos sistemas lineares. Essa otimização é dependente do método adotado e do tipo de problema considerados.

A próxima seção propõe alguns trabalhos futuros.

6.1 Sugestões para Trabalhos Futuros

As propostas de trabalhos futuros aqui apresentadas refletem as ideias ou trabalhos não desenvolvidos nesta tese.

1. A incorporação da dependência temporal dos dados do problema do transporte. Nesse caso, além da adequação dos códigos desenvolvidos, a metodologia de cálculo das constantes ε e β deve ser modificada. Uma proposta para essa implementação é a seguinte:
 - i. considerar duas partições temporais $I_M = \{t_0, t_1, \dots, t_M\}$ e $I_N = \{t_0, t_1, \dots, t_N\}$ com $N > M$, desvinculadas do método de elementos finitos;
 - ii. Calcular as sequências de mínimos e máximos autovalores para cada partição temporal, ou seja,

$$Seq_1 = (\varepsilon_{I_{M_k}})_{k=0}^M, Seq_2 = (\varepsilon_{I_{N_j}})_{j=0}^N \quad (6.1)$$

$$Seq_3 = (\kappa_{I_{M_k}})_{k=0}^M, Seq_4 = (\kappa_{I_{N_j}})_{j=0}^N \quad (6.2)$$

$$(6.3)$$

- ii. Se $\varepsilon_0 = \min Seq_1 = \min Seq_2 \Rightarrow \varepsilon = \varepsilon_0$. Caso contrário, construa uma nova partição mais refinada e repita o processo. Analogamente, o valor de κ pode ser calculado.
2. Considere que as constantes ε e κ para dados com dependência temporal estão disponíveis. Os códigos apresentados nesse trabalho devem ser adequadamente modificados para incorporar a dependência temporal dos dados. A questão adaptativa dos dados deve ser repensada, pois a adequação da malha passa a possuir dependência temporal. Além disso, é interessante considerar a implementação de uma classe para refinamento adaptativo. Uma sugestão é conjugar os códigos aqui desenvolvidos com o código desenvolvido em [Praetorius, Weinmuller e Wissgott \(2008\)](#). A dificuldade inicial é a transferência das informações entre as bibliotecas utilizadas nesse trabalho e o software *Matlab*^R 1.
3. A inadequação do método de elementos finitos contínuos em capturar descon-tinuidades dos dados conduz ao métodos dos elementos finitos descontínuos. Nesse caso, analogamente aos trabalhos desenvolvidos, adequadas estimativas residuais devem ser obtidas. Em princípio, os métodos multigrid com reutilização podem ser utilizados neste tipo de problema.
4. Aqui, apenas o caso linear da equação do transporte de contaminantes foi tratado. Consequentemente, sugere-se que as equações não-lineares sejam abordadas. Nesse caso, as estimativas residuais propostas em [Verfürth \(2014\)](#), [Verfürth \(2004\)](#) e

1. O projeto FEniCS traz documentação e sugestões de como conjugar o *Matlab*^R e as bibliotecas do Projeto FEniCS.

[Praetorius, Weinmuller e Wissgott \(2008\)](#) precisam ser revisadas para adequação às estimativas propostas em [Verfürth \(2004b\)](#).

5. A integração dessas ferramentas e códigos desenvolvidos em um software de referenciamento geográfico é outro aspecto importante a ser considerado. O trabalho desenvolvido por [Firmiano \(2015\)](#) leva em conta essa integração.
-

Referências

- AYACHIT, U. *The ParaView Guide: A Parallel Visualization Application*. Kitware, 2015. ISBN 978-1930934306.
- BATU, V. *Applied Flow and Solute Transport Modeling in Aquifers: Fundamental Principles and Analytical and Numerical Methods*.: CRC Press Taylor, 2006. ISBN 10: 0-8493-3574-4 (Hardcover).
- BEAR, J. *Hydrogeology*.: CRC Press Taylor, 1979. 698 p.
- BEKHIT, H. M.; EL-KORDY, M. A.; HASSAN, A. E. Contaminant transport in groundwater in the presence of colloids and bacteria: Model development and verification. *Journal of Contaminant Hydrology*, v. 108, n. 3-4, p. 152–167, set. 2009. ISSN 01697722.
- BELL, W. N.; OLSON, L. N.; SCHRODER, J. B. *PyAMG: Algebraic Multigrid Solvers in Python v2.0*. 2011. Release 2.0. Disponível em: <<http://www.pyamg.org>>.
- BREZINA, M. et al. Adaptive smoothed aggregation (α SA). *SIAM J. SCI. COMP*, v. 25, n. 6, p. 2004, 2004.
- BREZINA, M. et al. Adaptive algebraic multigrid. *SIAM J. Sci. Comput.*, v. 27, n. 4, p. 1261–1286, nov. 2005. ISSN 1064-8275. Disponível em: <<http://dx.doi.org/10.1137/040614402>>.
- BREZINA, M. et al. Towards adaptive smoothed aggregation (α sa) for nonsymmetric problems. *SIAM Journal on Scientific Computing*, v. 32, n. 1, p. 14–39, jan. 2010. ISSN 1064-8275, 1095-7197. Disponível em: <<http://epubs.siam.org/doi/abs/10.1137/080727336>>.
- BRIGGS, W. L. *A multigrid tutorial*. 1999. Disponível em: <<http://www.mgnet.org>>. Acesso em: 24 set. 2012.
- CAO, J.; KITANIDIS, P. K. Adaptive-grid simulation of groundwater flow in heterogeneous aquifers. *Advances in Water Resources*, v. 22, n. 7, p. 681–696, abr. 1999. ISSN 0309-1708. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0309170898000475>>.
- COUMOU, D. et al. A parallel FE–FV scheme to solve fluid flow in complex geologic media. *Computers & Geosciences*, v. 34, n. 12, p. 1697–1707, dez. 2008. ISSN 0098-3004. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S009830040800112X>>.
- DETWILER, R. L. et al. Comparison of an algebraic multigrid algorithm to two iterative solvers used for modeling ground water flow and transport. *Ground water*, v. 40, n. 3, p. 267–272, jun. 2002. ISSN 0017-467X. PMID: 12019641.

- HÖLTER, B.; VANDERBERGHE, G. A comparison of vectorized methods for solving the two-dimensional diffusion equation: multigrid versus polynomial preconditioned conjugate gradient. *Applied Mathematics and Computation*, v. 40, n. 1, p. 77–103, nov. 1990. ISSN 0096-3003. Disponível em: <<http://www.sciencedirect.com/science/article/pii/0096300390900990>>.
- HOFINGER, G.; JUDEX, F. Pollution in groundwater flow: definition of argesim comparison c19. *Simulation News Europe*, v. 44, n. 45, p. 51–52, 2005.
- HUNTER, J. D. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007.
- JANKA, A. *Smoothed aggregation multigrid for incompressible flows*. 2007. Disponível em <http://perso.unifr.ch/ales.janka/papers/> acessado em 23/11/2012.
- JANKA, A. Smoothed aggregation multigrid for incompressible flows. *PAMM*, v. 7, n. 1, p. 1025901–1025902, 2007. ISSN 1617-7061. Disponível em: <<http://onlinelibrary.wiley.com/doi/10.1002/pamm.200700457/abstract>>.
- JONES, E. et al. *SciPy: Open source scientific tools for Python*. 2001. Disponível em: <<http://www.scipy.org/>>.
- KRAUS, J.; MARGENOV, S. *Robust Algebraic Multilevel Methods and Algorithms*. [S.l.]: Walter de Gruyter GmbH & Co., 2009. ISBN 978-3-11-019365-7.
- LAI, M.-C.; WU, C.-T.; TSENG, Y.-H. An efficient semi-coarsening multigrid method for variable diffusion problems in cylindrical coordinates. *Applied Numerical Mathematics*, v. 57, n. 5–7, p. 801–810, maio 2007. ISSN 0168-9274. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0168927406001577>>.
- LAPWORTH, D. et al. Emerging organic contaminants in groundwater: A review of sources, fate and occurrence. *Environmental Pollution*, v. 163, n. 0, p. 287–303, abr. 2012. ISSN 0269-7491. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0269749111007044>>.
- LARSON, F. B. M. G. *The Finite Element Method: Theory, Implementation, and Practice*. Springer, 2010. 270 p.
- LI, M.; CHENG, H.; YEH, G. Solving 3D subsurface flow and transport with adaptive multigrid. *Journal of Hydrologic Engineering*, v. 5, n. 1, p. 74–81, 2000. ISSN 1084-0699. Disponível em: <<http://ascelibrary.org/doi/abs/10.1061/%28ASCE%291084-0699%282000%295%3A1%2874%29>>.
- LOGG, A. et al. *Automated Solution of Differential Equations by the Finite Element Method*. [S.l.]: Springer, 2012. ISBN 978-3-642-23098-1.

- MACLACHLAN, S. P. *Improving Robustness in Multiscale Methods*. Tese (Doutorado) — Department of Applied Mathematics, Faculty of the Graduate School of the University of Colorado, 2004. Disponível em <<http://neumann.math.tufts.edu/~scott/research/SMthesis.pdf>>.
- MCBRYAN, O. A. et al. Multigrid methods on parallel computers—A survey of recent developments. *IMPACT of Computing in Science and Engineering*, v. 3, n. 1, p. 1–75, mar. 1991. ISSN 0899-8248. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S089982489190015M>>.
- MEHL, S. W.; HILL, M. C. *MODFLOW-2000, the U.S. Geological Survey Modular Ground-Water Model – User Guide to the Link-AMG (LMG) Package for Solving Matrix Equations Using an Algebraic Multigrid Solver*. 2001. Disponível em: <<http://water.usgs.gov/acessadoem21/11/2012>>. Acesso em: 21 nov. 2012.
- MURATOVA, G. V.; ANDREEVA, E. M. Multigrid method for solving convection-diffusion problems with dominant convection. *Journal of Computational and Applied Mathematics*, v. 226, n. 1, p. 77–83, abr. 2009. ISSN 0377-0427. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0377042708002239>>.
- MUSTAPHA, H.; GHORAYEB, A.; MUSTAPHA, K. Underground flow simulations using parallel finite element method. *Computers & Geosciences*, v. 36, n. 2, p. 161–166, fev. 2010. ISSN 0098-3004. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0098300409002933>>.
- NOTAY, Y. Aggregation-based algebraic multigrid for convection-diffusion equations. *SIAM Journal on Scientific Computing*, v. 34, n. 4, p. A2288–A2316, jan. 2012. ISSN 1064-8275, 1095-7197. Disponível em: <<http://epubs.siam.org/doi/abs/10.1137/110835347>>.
- PARKHURST D.L., K. K. E. P.; CHARLTON, S. Phast – a program for simulating groundwater flow, solute transport and multicomponent geochemical reactions. usgs, denver, colorado, 2004. In: *Techniques and Methods 6–A8, 154 p.* U.S. Geological Survey, 2004. Disponível em: <<http://pubs.usgs.gov/tm/2005/tm6A8/pdf/tm6a8.pdf>>.
- PRAETORIUS, D.; WEINMULLER, E.; WISSGOTT, P. *A Space-Time Adaptive Algorithm for Linear Parabolic Problems*. [S.l.], 2008. Available at www.asc.tuwien.ac.at ISBN 978-3-902627-00-1.
- RUGE, J.; STÜBEN, K. *Algebraic Multigrid (AMG)*, in “*Multigrid Methods*” (S. McCormick, ed.), *Frontiers in Applied Mathematics, Vol 5*. [S.l.]: SIAM- Society for Industrial and Applied Mathematics, 1986. Philadelphia.
- SAAD, Y. *Iterative Methods for Sparse Linear Systems*. SIAM- Society for Industrial and Applied Mathematics, 2003.

- SAIED, F.; MAHINTHAKUMAR, G. Efficient parallel multigrid based solvers for large scale groundwater flow simulations. *Computers & Mathematics with Applications*, v. 35, n. 7, p. 45–54, abr. 1998. ISSN 0898-1221. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0898122198000315>.
- SANTOS, J.P. M. ;FIRMIANO; A.; WENDLAND, E. Jump dominance on the contaminant transport residual error estimator. *Tendências em Matemática Aplicada e Computacional*, v. 15, p. 037–046, 2014.
- SANTOS, J. P. M.; FIRMIANO, A.; WENDLAND, E. C. In: . [S.l.: s.n.].
- SANTOS, J. P. M.; FIRMIANO, A.; WENDLAND, E. C. Dominância das contribuições laterais no estimador residual da equação do transporte de contaminantes. In: *Congresso de Matemática Aplicada e Computacional CMAC Sudeste 2013, 2013, Bauru. Anais do Congresso de Matemática Aplicada e Computacional CMAC Sudeste 2013*. São Carlos, São Paulo: [s.n.].
- SCOTT, T. Multi-grid methods for oil reservoir simulation in two and three dimensions. *Journal of Computational Physics*, v. 59, n. 2, p. 290–307, jun. 1985. ISSN 0021-9991. Disponível em: <http://www.sciencedirect.com/science/article/pii/0021999185901470>.
- SIMMONS, C. T.; FENSTEMAKER, T. R.; JR., J. M. S. Variable-density groundwater flow and solute transport in heterogeneous porous media: approaches, resolutions and future challenges. *Journal of Contaminant Hydrology*, v. 52, n. 1–4, p. 245–275, nov. 2001. ISSN 0169-7722. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0169772201001607>.
- DE SMEDT, F. Groundwater Modelling *Course Notes, 2006* Disponível em: [Home-page:http://homepages.vub.ac.be/~fdesmedt/](http://homepages.vub.ac.be/~fdesmedt/);
- SOREK, S. Two-dimensional adaptive Eulerian-Lagrangian method for mass transport with spatial velocity distribution. *Transport in Porous Media*, v. 3, n. 5, p. 473–489, out. 1988. ISSN 0169-3913, 1573-1634. Disponível em: <http://rd.springer.com/article/10.1007/BF00138612>.
- STÜBEN, K. A review of algebraic multigrid. *Journal of Computational and Applied Mathematics*, v. 128, n. 1–2, p. 281–309, mar. 2001. ISSN 0377-0427. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377042700005161>.
- STÜBEN, K.; DELANEY, P.; CHMAKOV, S. Algebraic multigrid (amg) for ground water flow and oil reservoir simulation. 2010. Disponível em acessado em 10/2012. Disponível em: https://www.scai.fraunhofer.de/fileadmin/download/samg/paper/Modflow_Paper.pdf. Acesso em: 15/05/2015.
- STÜBEN, K. *Multigrid Methods and Parallel Computing*. 2009. Disponível em: <http://fefflow.info/uploads/media/Stueben.pdf>. Acesso em: 26 set. 2015.

- TRANGENSTEIN, J. A. Multi-scale iterative techniques and adaptive mesh refinement for flow in porous media. *Advances in Water Resources*, v. 25, n. 8–12, p. 1175–1213, ago. 2002. ISSN 0309-1708. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0309170802000532>>.
- TROTTERBERG, U.; OOSTERLEE, C.; SCHULLER, A. *Multigrid*. [S.l.]: Academic Press, 2001.
- TUFENKJI, N.; EMELKO, M.; NRIAGU, E. in C. O. Fate and transport of microbial contaminants in groundwater. In: *Encyclopedia of Environmental Health*. Burlington: Elsevier, 2011. p. 715–726. ISBN 978-0-444-52272-6. Disponível em: <<http://www.sciencedirect.com/science/article/pii/B9780444522726000404>>.
- VANĚK, P. Fast multigrid solver. *Applications of Mathematics*, v. 40, n. 1, p. 1–20, 1995. ISSN 0862-7940. Disponível em: <<http://dml.cz/dmlcz/134274>>.
- VANĚK, P. et al. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, v. 56, p. 179–196, 1995.
- VERFÜRTH, R. *A posteriori error estimates for non-linear parabolic equations*. Disponível em: <<http://www.ruhr-uni-bochum.de/num1/files/reports/APNLPE.pdf>>.
- VERFÜRTH, R. *A posteriori error estimates for linear parabolic equations*. p. 26, 2004. Disponível em: <<http://www.ruhr-uni-bochum.de/num1/files/reports/APEELPE.pdf>>.
- VERFÜRTH, R. *Adaptive Finite Element Methods: Lecture Notes Winter Term 2013/14*. [s.n.], 2014. Disponível em: <<http://www.ruhr-uni-bochum.de/num1/files/lectures/AdaptiveFEM.pdf>>.
- WALT, S. van der; COLBERT, S.; VAROQUAUX, G. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science Engineering*, v. 13, n. 2, p. 22–30, mar. 2011. ISSN 1521-9615.
- WEXLER, E. Chapter b7: Applications of hydraulics analytical solutions for one-, two- and three-dimensional solute transport in groundwater systems with uniform flow. In: *Techniques of Water Resources Investigations of the United State Geological Survey, Book 3: Applications to Hydraulics*. Denver, USA: U.S. Geological Survey, 1992.
- WU, C.-t.; HOWARD; ELMAN, C. *Analysis and comparison of geometric and algebraic multigrid for convection-diffusion equations*. [S.l.], 2004.
- ZHANG, J.; SUN, H.; ZHAO, J. J. High order compact scheme with multigrid local mesh refinement procedure for convection diffusion problems. *Computer Methods in Applied Mechanics and Engineering*, v. 191, n. 41–42, p. 4661–4674, set. 2002. ISSN 0045-7825. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0045782502003985>>.

ZOPPOU, C.; KNIGHT, J. H. Analytical solution of a spatially variable coefficient advection–diffusion equation in up to three dimensions. *Applied Mathematical Modelling*, v. 23, n. 9, p. 667–685, set. 1999. ISSN 0307-904X. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0307904X99000050>>.

APÊNDICE A – Código Python

Nesta parte, a implementação em linguagem de programação Python, desenvolvida na plataforma Eclipse ([ECLIPSE FOUNDATION, 2015](#)), é descrita em detalhe. As funções, classes e resultados gráficos apresentados utilizam as bibliotecas Numpy ([WALT; COLBERT; VAROQUAUX, 2011](#)), Scipy ([JONES et al., 2001](#)), Matplotlib ([HUNTER, 2007](#)), ParaView ([AYACHIT, 2015](#)) juntamente com aquelas disponibilizadas pela biblioteca FEniCS ([FENICS PROJECT, 2014](#)). Uma descrição completa do Projecto FEniCS é o livro ([LOGG et al., 2012](#)).

O capítulo [B] fornece alguns exemplos de códigos completos para a solução pelo método dos elementos finitos, a implementação da solução analítica de [Wexler \(1992\)](#), o método multigrid com reutilização e a metodologia de comparação de soluções.

A.1 Códigos para o Estimador Residual

Nesta seção, as classes e funções necessárias à implementação do estimador residual são apresentadas. Esses códigos incluem as classes e funções necessárias para a obtenção das estimativas residuais para o regime de pequena advecção. Além disso, são fornecidos os indicadores de erros e algumas classes e funções adicionais desenvolvidas, mas que não foram utilizadas na tese.

Primeiramente, o desenvolvimento dos códigos é baseado nas bibliotecas Numpy e dolfin, a qual é disponibilizada através do Projeto FEniCS([FENICS PROJECT, 2014](#)). A adequada manipulação das estruturas matriciais para posterior disponibilização para os métodos iterativos é feita através do pacote scipy.sparse. Essas ferramentas são disponibilizadas através do código abaixo:

```

1 import numpy as np
2 from numpy import *
3 from dolfin import *
4 dolfin.parameters.reorder_dofs_serial = False
5 #https://answers.launchpad.net/dolfin/+question/219220
6 #-----
7 # Part I: Setup problem with Dolfin
8 try:
9     from dolfin import *
10 except ImportError:
11     raise ImportError('Problem with Dolfin Installation')
12 parameters.linear_algebra_backend = "uBLAS"
13 #-----
14 from scipy.sparse import csr_matrix
15 from numpy import intc

```

A.1.1 Código para a projeção L^2

A função `projectionFunction(**kargs)` disponibiliza o sistema linear associado ao cálculo da projeção L^2 de uma função f . Essa ferramenta pode ser útil para os métodos disponíveis na biblioteca numérica PyAMG, a qual permite a utilização dos métodos multigrid algébricos.

1. **Uso:** `Aspf,bf,nfsize=projectionFunction(mesh,C,f)`
2. **mesh:** malha de elementos finitos;
3. **C:** espaço de funções lagrangeanas definido sobre a malha de elementos finitos;
4. **f:** a expressão analítica para a função f ;
5. **Aspf, bf,nfsize:** a matriz do sistema linear $Asp * f_{projection} = bf$, em que $f_{projection}$ é a projeção L^2 a ser calculada, **bf** é o lado direito do sistema linear e **nfsize** é o tamanho do sistema linear associado.

```

1 def projectionFunction(mesh,C,f):
2     # Define variational problem for projection
3     wf = TestFunction(C)
4     vf = TrialFunction(C)
5     af = inner(wf,vf)*dx
6     Lf = inner(wf,f)*dx
7     # Assemble linear system with no boundary conditions because defining a function
8     Af, rhsf = assemble_system(af,Lf)
9
10    (rowf,colf,dataf) = Af.data() # get sparse data
11    colf = intc(colf)
12    rowf = intc(rowf)
13    nfsize = Af.size(0) # the size of the linear system
14    Aspf = csr_matrix((dataf,colf,rowf),shape=(nfsize,nfsize))
15    bf = rhsf.data()
16    return Aspf,bf,nfsize

```

A.1.2 Código para campo gradiente contínuo

A função `ProjectionGradient(**kargs)` disponibiliza o campo gradiente contínuo, o qual é obtido da solução de elementos finitos do problema do transporte. Este cálculo pode ser útil, quando existe a exigência por campos gradientes contínuos e é baseado nos exemplos disponíveis no endereço eletrônico do Projeto FEniCS. Da mesma forma que

no item anterior, o sistema linear associado pode ser disponibilizado para a solução com outros métodos iterativos.

1.Uso:

```
gradFieldContx,gradFieldConty,\
gradFieldCont=ProjectionGradient(mesh,gradField,degree)
```

2.mesh: malha de elementos finitos;

3.gradField: o campo gradiente calculado pelo método de elementos finitos;

4.degree: grau do espaço de funções;

5.gradFieldContx,gradFieldConty,gradFieldCont: campo gradiente contínuo na direção x , campo gradiente contínuo na direção y e campo gradiente, respectivamente.

```
def ProjectionGradient(mesh,gradField,degree):
2   Vg = VectorFunctionSpace(mesh,'Lagrange',degree)
   w = TrialFunction(Vg)
4   v1 = TestFunction(Vg)
   a = inner(w,v1)*dx(mesh)
6   L = inner(gradField, v1)*dx
   gradFieldCont= Function(Vg)
8   solve(a == L,gradFieldCont)
   parameters['allow_extrapolation'] = True
10  gradFieldContx,gradFieldConty = gradFieldCont.split(deepcopy=True)
   return gradFieldContx,gradFieldConty,gradFieldCont
```

A.1.3 Código para transformar os dados

A função `MakeSystem_and_TransformData(**kargs)` disponibiliza o sistema linear associado ao problema do transporte de contaminantes. Esta ferramenta, possibilita a utilização de outros métodos iterativos para resolver o sistema linear. No entanto, neste trabalho, este processo é pouco utilizado, pois o autor prefere a transformação direta dentro do código utilizado.

1.Uso: `Asp,b,nsize=MakeSystem_and_TransformData(a,L,boundary_parts,bcs)`

2.a, L: formas bilinear e linear associadas ao problema do transporte ;

3.boundary_parts: classes definindo a fronteira de Neumann;

4.bcs: lista de classes definindo a fronteira de Dirichlet;

5.Asp,b,nsize: o sistema linear, o lado direito e o tamanho do sistema linear, respectivamente.

```

1 def MakeSystem_and_TransformData(a,L,boundary_parts,bcs):
    A = assemble(a, exterior_facet_domains=boundary_parts)
3   rhs = assemble(L,exterior_facet_domains=boundary_parts)
    #Setting the Dirichlet boundary conditions
5   for bc in bcs:
        bc.apply(A, rhs)
7   (row,col,data) = A.data()
        col = intc(col)
9   row = intc(row)
        nsize = A.size(0)
11  Asp = csr_matrix((data,col,row),shape=(nsize,nsize))
        b = rhs.data()
13  return Asp,b,nsize

```

A.1.4 Código para componentes do erro residual

A função `ErrorComponents(**kargs)` disponibiliza as componentes do estimador residual relacionado à estimativa espacial sem considerar os residuais referentes aos dados do problema.

1. Uso:

```

n,gx,rk,gradot,salt,rneumann,\
rdirichlet=ErrorComponents(mesh,D,c,c1,VField,theta,Lambda,fx,dt)

```

2. `mesh,n`: malha de elementos finitos e vetor normal as arestas ou faces dos elementos;

3. `D,c,c1,VField,theta,Lambda,fx,dt`: a matriz de dispersão, a solução no tempo k , a solução no tempo $k - 1$, o campo de velocidades, o parâmetro de discretização temporal, a função de reação, a projeção L^2 da função f e o valor para o passo de tempo, respectivamente.

3. `gx`: a projeção na direção normal ao gradiente;

4. `rk,gradot`: o erro residual do elemento e função auxiliar, respectivamente;

5. `salt,rneumann,rdirichlet`: o erro do salto, o erro na fronteira de Neumann e o erro na fronteira de Dirichlet.

```

1 def ErrorComponents(mesh,D,c,c1,VField,theta,Lambda,fx,dt):
    n = FacetNormal(mesh);
3   gx=dot((D*grad(c)),n);
    rk=fx-(1.0/dt)*(c-c1)+div(D*grad(theta*c+(1.0-theta)*c1))-
5   theta*dot(VField,grad(c))-(1.0-theta)*dot(VField,grad(c1))-Lambda*(theta*c-(1.0-theta)*c1);
    function=D*grad(theta*c+(1.0-theta)*c1);
7   gradot=dot(function,n);
    salt=-jump(gradot);
9   rneumann=gx-gradot;

```



```

11 rdirichlet = 0.0;
    return n, gx, rk, gradot, salt, rneumann, rdirichlet

```

A.1.5 Código para o fator de ponderação α_S

A função `AlphaS(**kargs)` disponibiliza o fator de ponderação $\alpha_S := \min(\epsilon^{-1/2}, \beta^{-1/2})$ para a malha de elementos finitos. Essa função é essencial para o cálculo do estimador residual e é definida sobre todas as entidades da malha de elementos finitos.

1.Uso: `alphaS=AlphaS(mesh,epsilon,beta,C)`

2.`mesh,epsilon,beta,C`: malha de elementos finitos, `epsilon`= ϵ , `beta`= β ;

3.`alphaS`: o fator de ponderação α_S definido em cada entidade da malha de elementos finitos.

```

1 def AlphaS(mesh,epsilon,beta,C):
    tolerance=1.0E-5
3     class my_func(Expression):
        def __init__(self,mesh,epsilon,beta):
5             self.mesh = mesh
            self.epsilon = epsilon
7             self.beta = beta
        def eval_cell( self , values , x , ufc_cell ):
9             dolfin_cell = Cell(self.mesh, ufc_cell.index)
            #print dolfin_cell.diameter()
11            if beta<=0.0+tolerance:
                values[0]=(1.0/np.sqrt(epsilon))*dolfin_cell.diameter()
13            else :
                intermed=(1.0/np.sqrt(epsilon))*dolfin_cell.diameter()
15                values[0] =min(intermed,1.0/np.sqrt(beta))
    alphaS=my_func(mesh,epsilon,beta);
17    alphaS=interpolate(alphaS,C)
    return alphaS

```

A.1.6 Código para as contribuições residuais

A função `FormErrorsDef(**kargs)` disponibiliza as matrizes que armazenam as contribuições residuais de cada entidade da malha de elementos finitos. Essa função incorpora todas as contribuições e é um dos elementos principais do código do estimador residual.

1.Uso:

```

indicators1, indicators, indicator1, indicator2, indicator3, \
    time_indicator=FormErrorsDef(alpha_S, mesh, gx, rk, gradot, \
    salt, rneuman, rdirichlet, epsilon, beta)

```

2.alphaS: o fator de ponderação α_S definido em cada entidade da malha de elementos finitos.

3.mesh, gx, rk, gradot, salt, rneuman, rdirichlet, epsilon, beta: malha de elementos finitos, componentes do estimador residual e constantes ϵ , β ;

4.indicators1, indicators, indicator1, indicator2, indicator3, time_indicator: as contribuições residuais para cada entidade na malha de elementos finitos: as contribuições espaciais, dos elementos, dos saltos, das fronteiras e temporal, respectivamente.

```

def FormErrorsDef(alpha_S, mesh, gx, rk, gradot, salt, rneuman, rdirichlet, epsilon, beta):
2   Constants = FunctionSpace(mesh, "DG", 0)
    w = TestFunction(Constants)
4   # Define forms for assembling error indicators and error estimator
    form1 = alpha_S**2*rk**2*w*dx()
6   #---COULD USE: form2=alpha_S('+')*salt**2*w('+')*dS
    form2=(1.0/np.sqrt(epsilon))*avg(alpha_S)*salt**2*avg(w)*dS
8   form3=(1.0/np.sqrt(epsilon))*alpha_S*rneuman**2*w*ds(0)+(1.0/np.sqrt(epsilon))*alpha_S*rneuman**2
        *w*ds(1)
    form_time=epsilon*(grad(c-c1))**2*w*dx()+beta*(c-c1)**2*w*dx()
10   form= form1+form2+form3; form_space=form1+form2+form3
    indicators1=assemble(form_space); indicators=assemble(form); indicator1=assemble(form1)
12   indicator2=assemble(form2); indicator3=assemble(form3); time_indicator=assemble(form_time)
    return indicators1, indicators, indicator1, indicator2, indicator3, time_indicator

```

A.1.7 Código para os indicadores residuais

A função `Indicators(**kargs)` disponibiliza as matrizes que armazenam os indicadores residuais de cada entidade da malha de elementos finitos. Essa função possibilita a construção de estratégias de refinamento para o problema do transporte de contaminantes.

Nos casos em que as estimativas para os dados e para o regime de grande advecção são necessárias, a manipulação adequada dessas classes deve ser efetuada.

1.Uso:

```

indicator_spatial, indicator_form1, indicator_form2,
indicator_form3, indicator_time=Indicators(indicators1, indicators, \
indicator1, indicator2, indicator3, time_indicator)

```

2.indicators1,indicators,indicator1,indicator2,indicator3 e

time_indicator: as contribuições residuais para cada entidade na malha de elementos finitos: as contribuições espaciais, dos elementos, dos saltos, das fronteiras e temporal, respectivamente.

3.indicator_spatial,indicator_form1,indicator_form2,indicator_form3 e

indicator_time: os indicadores residuais locais para cada entidade na malha de elementos finitos: os indicadores espaciais, dos elementos, dos saltos, das fronteiras de Neumann e temporal.

```

1 def Indicators(indicators1, indicators, indicator1, indicator2, indicator3, time_indicator):
2     """ The various indicators that can be formed using the forms presented in AssembleForm_Estimates"""
3     indicator_spatial = array([sqrt(i) for i in indicators])
4     indicator_form1 = array([sqrt(i) for i in indicator1])
5     indicator_form2 = array([sqrt(i) for i in indicator2])
6     indicator_form3 = array([sqrt(i) for i in indicator3])
7     indicator_time = array([sqrt(i) for i in time_indicator])
8     return indicator_spatial, indicator_form1, indicator_form2, indicator_form3, indicator_time

```

A.1.8 Código para as estimativas residuais espacial e temporal

A função `ErrorEstimatesSpaceTime(**kargs)` fornece as contribuições espacial, dos elementos, dos saltos, das fronteiras e temporal.

1.Uso:

```

error_estimate_space,error_estimate1,error_estimate2,\
error_estimate3,error_time=ErrorEstimatesSpaceTime(indicators1,indicators,
indicator1,indicator2,indicator3,time_indicator,dt)

```

2.error_estimate_space: a contribuição espacial, a qual é composta da soma das contribuições dos elementos, dos saltos e das fronteiras, respectivamente;

3.error_estimate1: a contribuição dos elementos, a qual é composta da soma das contribuições residuais dos elementos R_K ;

4.error_estimate2: a contribuição dos saltos, a qual é composta da soma das contribuições residuais dos saltos R_E ;

5.error_estimate3: a contribuição das fronteiras de Neumann, a qual é composta da soma das contribuições residuais dos elementos, definidos sobre a fronteira de Neumann, R_E .

6.error_time: a contribuição temporal, a qual é composta da soma das contribuições residuais, sobre toda a malha de elementos finitos.

```

def ErrorEstimatesSpaceTime(indicators1,indicators,indicator1,indicator2,indicator3,time_indicator,dt):
2   error_estimate_space = dt*(sum(i for i in indicators1.array()))#the spatial error contribution
   error_estimate1 = dt*(sum(i for i in indicator1.array()))# local interior contribution to global error
4   error_estimate2 = dt*(sum(i for i in indicator2.array()))# local jump contribution to global error
   error_estimate3 = dt*(sum(i for i in indicator3.array()))# local boundary contribution to global
6   error_time      = dt*(sum(i for i in time_indicator.array()))# time contribuion to global error
   return error_estimate_space,error_estimate1,error_estimate2,error_estimate3,error_time

```

As estimativas residuais para o regime de grande advecção podem ser inseridas por meio de uma matriz de valores similar à `error_time`.

A.2 O regime de grande advecção

Nesta seção, as classes e funções necessárias às estimativas residuais associadas ao regime de grande advecção são apresentadas. Essas estimativas devem ser incorporadas adequadamente às estimativas temporais, obtidas na seção anterior.

A.2.1 Código para as formas bilinear e linear do problema auxiliar

A função `FormLargeAdvection(**kargs)` disponibiliza o sistema linear associado ao problema auxiliar. Esse sistema linear pode ser resolvido por um método multigrid algébrico ou qualquer outro método iterativo disponível.

1. **Uso:** `ALarge,bLarge=FormLargeAdvection(epsilon,beta,c,c1,v,mesh=mesh)`
2. **epsilon**= ε e **beta**= β são as constantes envolvidas no estimador residual;
3. **c,c1,v,mesh**: as soluções numéricas no tempo k e $k-1$, o campo de velocidades e a malha de elementos finitos;
4. **ALarge,bLarge**: os componentes do sistema linear $ALarge * ctil = bLarge$, em que $ctil$ é a solução do problema auxiliar associado ao problema do transporte.

```

1  #-----Form for large advection
   def FormLargeAdvection(epsilon,beta,c,c1,v,mesh=mesh):
3     degtil=1
       Ctil = FunctionSpace(mesh,"Lagrange",degtil)
5     ctil = TrialFunction(Ctil)
       wtil = TestFunction(Ctil)
7     #-----
       c=interpolate(c,Ctil)
9     c1=interpolate(c1,Ctil)
       difftil =c.vector()-c1.vector();
11    diff =Function(Ctil)
       diff .vector() [:]= difftil

```

```

13      #Dirichlet conditions is null as well the Neumann conditions
      c0=Constant(0.0)
15      # the problem has only dirichlet boundary
      class AuxiliarBoundary(SubDomain):
17          def inside( self , x, on_boundary):
              tol = 1E-14
19          return on_boundary
      Gamma_0 = DirichletBC(Ctil, c0, AuxiliarBoundary())
21      bcs=[Gamma_0]
      g=Constant(0.0)# the function for neumann boundaries...
23      aLarge=inner(epsilon*nabla_grad(ctil),nabla_grad(wtil))*dx+beta*ctil*wtil*dx
      LLarge=inner(v,nabla_grad(diff))*wtil*dx-g*wtil*ds
25      #Assemble and apply boundary conditions
      ALarge=assemble(aLarge)
27      bLarge=assemble(LLarge)
      for bc in bcs:
29          bc.apply(ALarge, bLarge)
      return ALarge,bLarge
31

```

A.2.2 Código para as contribuições residuais do problema auxiliar

A função `LargeAdvecEstimates(**kargs)` implementa as contribuições residuais associadas ao problema auxiliar. Analogamente às contribuições residuais para o regime de pequena advecção, essas contribuições são divididas entre as contribuições dos elementos e dos saltos.

1. Uso:

```

formarray, form1array, form2array,
    form3array=LargeAdvectionEstimates(alpha_S, mesh, epsilon, beta, c,
    c1, ctil, VField)

```

2. `alpha_S, mesh, epsilon, beta, c, c1, ctil, VField`: o fator de ponderação α_S , a malha de elementos finitos, as constantes do estimador residual `epsilon`= ε e `beta`= β , as soluções numéricas do problema do transporte nos tempos k e $k-1$, a solução numérica do problema auxiliar `ctil` e o campo de velocidades, respectivamente.

3. `formarray`: as matrizes que contêm as contribuições relativas ao problema auxiliar. É composta das contribuições dos elementos, dos saltos e da norma da energia, respectivamente.

4. `form1array, form2array, form3array`: as contribuições dos elementos, dos saltos e da norma da energia, respectivamente.

5. Observação: O termo $\varepsilon \cdot \text{Div}(\text{ctil})$ que aparece nas fórmulas apresentadas por (VERFÜRTH, 2014) não é necessário, pois funções lineares são consideradas. Isso não é uma conveniência, mas sim uma hipótese do problema auxiliar.

```

def LargeAdvectionEstimates(alpha_S,mesh,epsilon,beta,c,c1,ctil,VField):
2
    n = FacetNormal(mesh);# normal vector to facet elements
4    rk=dot(VField,grad(c-c1))+div(epsilon*grad(ctil))-beta*ctil
    gradot=dot((grad(ctil)),n); salt=-jump(gradot);
6    #-----
    Constants = FunctionSpace(mesh,"DG",0)
8    w = TestFunction(Constants)
    #-----
10   form1 =alpha_S**2*rk**2*w*dx# form to interior contribution
    form2=(1.0/np.sqrt(epsilon))*avg(alpha_S)*salt**2*avg(w)*dS# form for jump contribution
12   form3=epsilon*(grad(ctil)**2*w*dx+beta*(ctil)**2*w*dx# the |||ctil||| energy norm
    #-----
14   form=form1+form2+form3; formarray=assemble(form)
    form1array=assemble(form1);form2array=assemble(form2)
16   form3array=assemble(form3)# matrices of values
    return formarray,form1array,form2array,form3array

```

A.2.3 Código para obter as estimativas residuais do problema auxiliar

A função `AuxiliarErrorEstimatesSpaceTime` fornece as contribuições residuais relacionadas às estimativas

```

1 def AuxiliarErrorEstimatesSpaceTime(formLarge,form1Large,form2Large,form3Large,dt):
    print 'divide by dt to get the space estimates without time step weight'
3    #-----
    LargeGeralEstimate = dt*(sum(i for i in formLarge.array()))#the spatial contribution
5    LargeInteriorEstimate = dt*(sum(i for i in form1Large.array()))# interior contribution
    LargeJumpEstimate = dt*(sum(i for i in form2Large.array()))# jump contribution
7    LargeEnergyCtilEstimate= dt*(sum(i for i in form3Large.array()))#energy contribution
    return LargeGeralEstimate,LargeInteriorEstimate,LargeJumpEstimate,LargeEnergyCtilEstimate

```

A.3 Códigos para as estimativas dos dados

Nesta seção, as classes e funções necessárias às estimativas residuais associadas aos dados do problema do transporte são apresentadas. Essas estimativas devem ser incorporadas adequadamente às estimativas espaciais, obtidas na seção A.1 deste capítulo.

A.3.1 Código das funções auxiliares para estimativas dos dados

Sem perda de generalidade, suponha que $d = d(x, y)$ seja uma expressão matemática que representa a dispersão do problema do transporte. Suponha que a representação na malha de elementos finitos seja dada por d_h . A representação analítica de d considera uma malha refinada em espaço de elementos finitos com funções lagrangeanas de ordem superior. A função auxiliar `DiffFunction(**kargs)` fornece a diferença entre a aproximação de elementos finitos e a representação analítica de d .

1. Uso:

```
Diffd, mesh, mesh2 = DiffFunction(C, AnalyticalExpression,
                                  mesh, degree, raiseDegree)
```

2. `C, AnalyticalExpression, mesh, degree, raiseDegree`: espaço de funções, expressão analítica, malha de elementos finitos, grau do espaço de funções e grau adicional para criar espaço de grau elevado.

3. `Diffd, mesh, mesh2`: a diferença entre as representações e as malhas de elementos finitos, respectivamente.

```
def DiffFunction(C, AnalyticalExpression, mesh, degree, raiseDegree):
2   C1 = C.#FunctionSpace(mesh, 'Lagrange', degree)
   mesh2 = mesh; deg = degree + raiseDegree
4   C2 = FunctionSpace(mesh2, 'Lagrange', deg)
   dxx1 = project(AnalyticalExpression, C);
6   dxx2 = project(AnalyticalExpression, C2);

8   class diffxx(Expression):
       def eval(self, values, x):
10       values[0] = dxx2(x[0], x[1]) - dxx1(x[0], x[1])
       Diffd = diffxx(element=C2.ufl_element());
12   Diffd = interpolate(Diffd, C2)
   return Diffd, mesh, mesh2
```

Na função apresentada, a projeção pode ser substituída por interpolação, a malha `mesh2` pode ser um refinamento da malha `mesh`. Nesse caso, o número de elementos na malha `mesh2` associado ao grau elevado afeta o tempo de cálculo.

A.3.2 Código para as componentes residuais dos dados

A função `DataResidualComponents(kargs**)` fornece as componentes do estimador residual dos dados do problema do transporte.

1. `mesh2, DK, Function, DiffGradot, DiffJump, \`
`DiffNeumann, DiffDirichlet=DataResidualComponents(C, alpha_S, \`
`epsilon, c1, c, dxx, dxy, dyy, AnalyticalExpression, theta, vx, vy, \`
`ReactionFunc, mesh, degree, raiseDegree)`
2. `C, alpha_S, epsilon, c1, c`: o espaço de funções, o parâmetro de ponderação α_S , `epsilon`= ε , a solução no tempo $k - 1$ e a solução no tempo k , respectivamente.
3. `dxx, dxy, dyy, AnalyticalExpression`: as dispersões d_{xx} , d_{xy} e d_{yy} , a expressão analítica para a carga hidráulica;
4. `vx, vy, ReactionFunc`: as componentes da velocidade v_x e v_y , a função que descreve as reações do meio, respectivamente;
5. `theta, mesh, degree, raiseDegree`: o parâmetro de discretização temporal, a malha de elementos finitos, o grau e o número de graus para construção do espaço de grau elevado.
6. `mesh2`: a malha de elementos finitos `mesh2` idêntica a malha inicial. Pode-se considerar malhas refinadas com adequadas modificações do código computacional. Aqui, apenas malhas idênticas foram consideradas.
7. `DK, Function, DiffGradot, DiffJump, DiffNeumann, DiffDirichlet`: as componentes do estimador residual dos dados do problema do transporte.

```

1 def DataResidualComponents(C,alpha_S,epsilon,c1,c,dxx,dxy,dyy,AnalyticalExpression,theta,vx,vy,
    ReactionFunc,mesh,degree,raiseDegree):
    Diffdxx,mesh,mesh2=DiffFunction(C,dxx,mesh,degree,raiseDegree);
3     print 'First calculations'
    Diffdxy,mesh,mesh2=DiffFunction(C,dxy,mesh,degree,raiseDegree);
5     print 'Second calculations'
    Diffdyx=Diffdxy#Diffdyx,mesh,mesh2=DiffFunction(dyx,mesh,degree,raiseDegree);
7     print 'Third calculations'
    Diffdyy,mesh,mesh2=DiffFunction(C,dyy,mesh,degree,raiseDegree);
9     print 'Fourth calculations'
    DiffD=as_matrix([[Diffdxx,Diffdxy],
11                    [Diffdyx,Diffdyy]])
    Diffvx,mesh,mesh2=DiffFunction(C,vx,mesh,degree,raiseDegree);
13     print 'five calculations'
    Diffvy,mesh,mesh2=DiffFunction(C,vy,mesh,degree,raiseDegree)
15     print 'six calculations'
    DiffReactionFunc,mesh,mesh2=DiffFunction(C,ReactionFunc,mesh,degree,raiseDegree);
17     print 'seven calculations'
    Diffd,mesh,mesh2=DiffFunction(C,AnalyticalExpression,mesh,degree,raiseDegree)
19     DiffV=grad(Diffd)
    print 'calculation of the velocity field'
21     DiffSol=theta*c+(1.0-theta)*c1 # Difference of solution with theta-schema
    # Bilinear forms and residual errors definitions
23     n=FacetNormal(mesh)
    print 'calculation of components:DK,Function,DiffJump,DiffGradot,DiffJump...'
25     DK=-div(DiffD*grad(DiffSol))+dot(DiffV,grad(DiffSol))+DiffReactionFunc*DiffSol
    Function=DiffD*grad(DiffSol); DiffGradot=dot(Function,n);
27     DiffJump=jump(DiffGradot); DiffNeumann=DiffGradot;
    DiffDirichlet=DiffGradot;

```



```
29 return mesh2,DK,Function,DiffGradot,DiffJump,DiffNeumann,DiffDirichlet
```

A.3.3 Código para as contribuições residuais dos dados

A função `FormDataOneNeumann(kargs**)` fornece as contribuições residuais dos dados do problema do transporte. Nesse caso, a fronteira de Neumann é definida por uma única classe, a qual pode conter várias partes da fronteira.

1. Uso:

```
DataInterior,DataJump,DataBoundary,
\DataGeral=FormDataOneNeumann(C,epsilon,degree,
raiseDegree,beta,mesh2,DK,function,DiffGradot,
DiffJump,DiffNeumann,DiffDirichlet)
```

2. `C,epsilon,degree,raiseDegree,beta,mesh2`: o espaço de funções, $\epsilon = \epsilon$, o grau do espaço de funções, o número de graus para a construção do espaço de grau elevado, $\beta = \beta$ e a malha de elementos finitos;

3. `DK,Function,DiffGradot,DiffJump,DiffNeumann,DiffDirichlet`: as componentes do estimador residual dos dados do problema do transporte gerado na subseção anterior.

4. `DataInterior, DataJump,DataBoundary,DataGeral`: as contribuições residuais associadas aos dados do problema do transporte.

```
def FormDataOneNeumann(C,epsilon,degree,raiseDegree,beta,mesh2,DK,function,DiffGradot,DiffJump,
    DiffNeumann,DiffDirichlet):
2  C1=C#FunctionSpace(mesh,'Lagrange',degree)
    #mesh2=mesh;#mesh2=refine(mesh);
4  deg=degree+raiseDegree
    C2=FunctionSpace(mesh2,'Lagrange',deg)
6  alphaSAux=AlphaS(mesh2,epsilon,beta,C2)
    #print "The refined mesh has%s" % str(mesh2)
8  ConstantsAux = FunctionSpace(mesh2,"DG",0)
    waux= TestFunction(ConstantsAux)
10 form1 =alphaSAux**2*DK**2*waux*dx(mesh2)
    form2 = (1.0/np.sqrt(epsilon))*avg(alphaSAux)*DiffJump**2*avg(waux)*dS
12 form3=(1.0/np.sqrt(epsilon))*DiffGradot*waux*ds(0)
    form=form1+form2+form3
14 DataInterior = assemble(form1); DataJump=assemble(form2,mesh=mesh2)
    DataBoundary = assemble(form3,mesh=mesh2); DataGeral= assemble(form,mesh=mesh2)
16 return DataInterior,DataJump,DataBoundary,DataGeral
```

A.3.4 Código para os indicadores residuais dos dados

A função `FormOneNeumannIndicator(kargs**)` fornece os indicadores residuais para os dados do problema do transporte. Essa função pode ser útil para procedimentos adaptativos para os dados. Neste caso, os indicadores são independentes das contribuições residuais para facilitar o processo de refinamento. No entanto, a adequada manipulação das quantidades `DataInterior`, `DataJump`, `DataBoundary`, `DataGeral` também fornece os indicadores residuais dos elementos, saltos, fronteiras de Neumann e espaciais.

1. Uso:

```
DataInterior,DataJump,DataBoundary,DataGeral=\
FormOneNeumannIndicator(epsilon,alpha_S,mesh2,DK,
function,DiffGradot,DiffJump,DiffNeumann,DiffDirichlet)
```

2. A notação é idêntica a subseção anterior.

```
def FormOneNeumannIndicator(epsilon,alpha_S,mesh2,DK,function,DiffGradot,DiffJump,DiffNeumann,
    DiffDirichlet):
2   Constants = FunctionSpace(mesh2,"DG", 0)
   w = TestFunction(Constants)
4   form1 =alpha_S**2*DK**2*w*dx
   form2 = (1.0/np.sqrt(epsilon))*avg(alpha_S)*DiffJump**2*avg(w)*dS
6   # The Neumann boundary
   form3=(1.0/np.sqrt(epsilon))*DiffGradot*w*ds(0)
8   form=form1+form2+form3

10   DataInterior = assemble(form1); DataJump    = assemble(form2)
   DataBoundary = assemble(form3); DataGeral    = assemble(form)

12   DataElementIndicator=array([sqrt(i) for i in DataInterior])
14   DataJumpIndicator=array([sqrt(i) for i in DataJump])
   DataBoundaryIndicator=array([sqrt(i) for i in DataBoundary])
16   DataGeralIndicator=array([sqrt(i) for i in DataGeral])
   return DataInterior,DataJump,DataBoundary,DataGeral
```

A.3.5 Código para as estimativas residuais dos dados

A função `DataErrorEstimateStepN(kargs**)` fornece as estimativas residuais associadas aos dados do problema. Em problemas que exigem essa estimativa, as quantidades respectivas devem ser adequadamente somadas para produzir a estimativa residual.

1. Uso:

```

ErEstInt,ErEstJump,ErEstBound,\
ErEstGeral=DataErrorEstimateStepN(DataInterior,
DataJump,DataBoundary,DataGeral,dt)

```

2.DataInterior, DataJump,DataBoundary,DataGeral,dt: as contribuições residuais e o passo de tempo;

3.ErEstInt,ErEstJump,ErEstBound,ErEstGeral: as estimativas residuais para os elementos, salto, fronteiras e a estimativa espacial.

```

1 def DataErrorEstimateStepN(DataInterior,DataJump,DataBoundary,DataGeral,dt):
    print 'Divide by the time step to get estimates without time step weight'
3 ErEstInt  = dt*(sum(i for i in DataInterior.array()))# local interior contribution to global
ErEstJump  = dt*(sum(i for i in DataJump.array()))# local jump contribution global
5 ErEstBound = dt*(sum(i for i in DataBoundary.array()))# local boundaries contributions to global
ErEstGeral = dt*(sum(i for i in DataGeral.array()))#the spatial error contribution
7 return ErEstInt,ErEstJump,ErEstBound,ErEstGeral

```

A.4 Códigos para as estimativas associadas ao termo de fonte

A função `FunctionFDataErrorEstimateStepN(kargs**)` fornece as contribuições residuais, os indicadores residuais e a estimativa residual. As contribuições residuais ou a estimativa residual podem ser incluídas no estimador residual, enquanto que os indicadores residuais podem ser utilizados em processos adaptativos envolvendo o termo de fonte.

1[1.] Uso:

```

DataFunctionEstimate,DataFunctionIndicator,\
DataFunction=FunctionFDataErrorEstimateStepN(alpha_S,epsilon,
theta,Fexpression,mesh,degree,raiseDegree,dt)

```

2.alpha_S,epsilon,theta e mesh,degree,raiseDegree,dt: o fator de ponderação α_S , ϵ , o parâmetro de discretização temporal, a malha de elementos finitos, o grau do espaço de funções, o número de graus adicional para construir o espaço de funções de ordem elevada e o passo de tempo, respectivamente.

3.Fexpression: a expressão analítica do termo de fonte.

4.DataFunctionEstimate,DataFunctionIndicator,DataFunction: as contribuições residuais, os indicadores residuais e a estimativa residual, respectivamente.

5.Observação: esta função considera malhas refinadas e funções lagrangeanas de grau elevado. A seguir é apresentada uma simplificação deste procedimento. O objetivo é simplificar a obtenção dos indicadores de erros relacionados ao termo de fonte.

```

1  #-----THE F DATA ERROR ESTIMATE
def FunctionFDataErrorEstimateStepN(C,alpha_S,epsilon,theta,Fexpression,mesh,degree,raiseDegree,dt):
3      print 'error estimates for functions time independent'
      print 'Time dependent functions needs to be implemented'
5      from Tools.ErrorEstimatorFunctions import DiffFunction
      DiffFunctionStepN,mesh,mesh2=DiffFunction(C,Fexpression,mesh,degree,raiseDegree)
7      DiffFunctionStepN1=DiffFunctionStepN # the N-1 step..but function is not time deperdent
      #-----RESIDUAL ERROR DEFINITIONS FOR F FUNCTION
9      Constants = FunctionSpace(mesh2,"DG",0); w = TestFunction(Constants)
      DFN=DiffFunctionStepN; DFN1=DiffFunctionStepN1
11     FormFunctionF=alpha_S**2*(theta**2*DFN**2+(1.0-theta)**2*DFN1**2)*w*dx(mesh2)
      DataFunction=assemble(FormFunctionF)
13     DataFunctionIndicator=array([sqrt(i) for i in DataFunction])
      DataFunctionEstimate=dt*(sum(i for i in DataFunction.array()))
15     return DataFunctionEstimate,DataFunctionIndicator,DataFunction

```

A.4.1 Código para os indicadores associados ao termo de fonte

A função `FunctionFDataErrorIndicatorsStepN(kargs**)` fornece os indicadores residuais para processos adaptativos da função de fonte. Essa função é uma modificação da função `FunctionFDataErrorEstimateStepN(kargs**)`, mas fornece os mesmos elementos que o item anterior. A função `DiffFunction2(kargs**)` considera malhas produzidas por meio da modificação da função `DiffFunction(kargs**)` com `mesh2=mesh` e espaço de funções de grau elevado.

```

1  def FunctionFDataErrorIndicatorsStepN(C,alpha_S,epsilon,theta,Fexpression,mesh,degree,raiseDegree,dt):
      print 'error estimates for functions time independent.'
      print 'Time dependent functions needs to be implemented'
3      from Tools.ErrorEstimatorFunctions import DiffFunction2
      DiffFunctionStepN,mesh,mesh2=DiffFunction2(C,Fexpression,mesh,degree,raiseDegree)
      DiffFunctionStepN1=DiffFunctionStepN
7      #-----RESIDUAL ERROR DEFINITIONS FOR F FUNCTION
      DiffFunctionStepN=interpolate(DiffFunctionStepN,C)
9      DFN=DiffFunctionStepN
      DFN1=DiffFunctionStepN1
11     #-----
      Constants = FunctionSpace(mesh,"DG",0)
13     w = TestFunction(Constants)
      FormFunctionF=alpha_S**2*(theta**2*DFN**2+(1.0-theta)**2*DFN1**2)*w*dx(mesh)
15     DataFunction=assemble(FormFunctionF)
      DataFunctionIndicator=array([sqrt(i) for i in DataFunction])
17     DataFunctionEstimate=dt*(sum(i for i in DataFunction.array()))
      return DataFunctionEstimate,DataFunctionIndicator,DataFunction

```

A.5 Código para as estimativas residuais das fronteiras

A função `NeumanDataErrorEstimateN(kargs**)` fornece as contribuições residuais, os indicadores residuais e a estimativa residual associadas às fronteiras de Neumann.

1. Uso:

```
DataGfunctionEstimate, DataGfunction, \
DataGfunctionIndicator=NeumanDataErrorEstimateN(alpha_S, epsilon,
theta, Gexpression, mesh, degree, raiseDegree, dt, boundary_parts)
```

2. `alpha_S, epsilon, theta` e `mesh, degree, raiseDegree, dt`: o fator de ponderação α_S , `epsilon`= ε , o parâmetro de discretização temporal, a malha de elementos finitos, o grau do espaço de funções, o número de graus adicional para construir o espaço de funções de ordem elevada e o passo de tempo, respectivamente.

3. `Gexpression`: a expressão analítica do termo de fonte.

4. `DataGfunctionEstimate, DataGfunction, DataGfunctionIndicator`: as contribuições residuais, os indicadores residuais e a estimativa residual, respectivamente.

```

#-----
2 def NeumanDataErrorEstimateN(alpha_S, epsilon, theta, Gexpression, mesh, degree, raiseDegree, dt,
    boundary_parts):
    print 'error estimates for functions time independent. Time dependent functions needs to be
        implemented yet'
4     DiffGfunctionStepN, mesh, mesh2=DiffFunction(Gexpression, mesh, degree, raiseDegree)
    DiffGfunctionStepN_1=DiffGfunctionStepN# the N-1 step..but function is not time dependent
6     #-----RESIDUAL ERROR DEFINITIONS FOR F FUNCTION
    Constants = FunctionSpace(mesh2, "DG", 0)
8     DGN=DiffGfunctionStepN; DGN_1=DiffGfunctionStepN_1
    w = TestFunction(Constants)
10    FormFunctionG=alpha_S*(theta**2*DGN**2+(1.0-theta)**2*DGN_1**2)*w*ds(0)
    DataGfunction=assemble(FormFunctionG);
12    DataGfunctionIndicator=array([sqrt(i) for i in DataGfunction])
    DataGfunctionEstimate=dt*epsilon*(-1.0/4.0)*(sum(i for i in DataGfunction.array()))
14    return DataGfunctionEstimate, DataGfunction, DataGfunctionIndicator

```

A.6 Código adicional para as estimativas

A função `MaxMinIndicators(kargs**)` fornece os máximos e mínimos de cada um dos indicadores residuais. Essa classe pode ser útil para o cálculo direto das quantidades máximas. No entanto, essas quantidades podem ser obtidas diretamente através da biblioteca NumPy.

1. Uso:

```
MaxMinIndicatorArray=MaxMinIndicators(indicator_spatial,
indicator_form1,indicator_form2,indicator_form3,indicator_time)
```

2. `indicator_spatial, indicator_form1, indicator_form2, indicator_form3` e `indicator_time`: os indicadores residuais espacial, dos elementos, dos saltos, das fronteiras de Neumann e temporal;

3. `MaxMinIndicatorArray`: a matriz de máximos e mínimos para os indicadores residuais.

```
def MaxMinIndicators(indicator_spatial,indicator_form1,indicator_form2,indicator_form3,indicator_time):
2   maxindicators = max(indicator_spatial); maxindicator1=max(indicator_form1);
   maxindicator2=max(indicator_form2); maxindicator3=max(indicator_form3);
4   maxindicatorTime=max(indicator_time);minindicators = max(indicator_spatial);
   minindicator1=min(indicator_form1);   minindicator2=min(indicator_form2);
6   minindicator3=min(indicator_form3);   minindicatorTime=min(indicator_time);

8   MaxMinIndicatorArray=np.array([maxindicators,maxindicator1,maxindicator2,maxindicator3,
   maxindicators,maxindicatorTime,\
                                   minindicators,minindicator1,minindicator2,minindicator3,minindicators,
   minindicatorTime])
10  return MaxMinIndicatorArray
```

A função `EachStepError_Indicator_Space_Time` fornece as quantidades residuais globais espaciais e temporais, ou seja, as quantidades que incorporam as contribuições de todas as entidades da malha de elementos finitos. Além disso, os indicadores espaciais e temporais também são fornecidos.

Uso:

```
time_indicator2,error_each_step,error_indicator2,\
indicator_spatial,indicador_temporal=\
EachStepError_Indicator_Space_Time(time_indicator,\
error_estimate_space,error_time)
```

2. `time_indicator, error_estimate_space, error_time`: as contribuições residuais espacial e temporal.

```
# because we nee add this terms in global error estimator
2 def EachStepError_Indicator_Space_Time(time_indicator,error_estimate_space,error_time):
   time_indicator2 = sqrt(sum(i for i in time_indicator.array()))
4   error_each_step=error_estimate_space+error_time
   error_indicator1=error_estimate_space
6   #error_time is the weighted by time time estimate
```

```

8   error_indicator2=error_time
   indicador_spatial=sqrt(error_indicator1)
   indicador_temporal=sqrt(error_indicator2)
10  return time_indicator2,error_each_step,error_indicator2,indicador_spatial,indicador_temporal

```

A.7 Código para adaptatividade da condição inicial

As funções e classes apresentadas nesta seção fornecem as ferramentas necessárias para o refinamento adaptativo da condição inicial. A adaptatividade considera espaços de funções com graus diferentes, enquanto que a estimativa residual considera malha refinada e espaço de funções com grau elevado.

Inicialmente, as bibliotecas numéricas são disponibilizadas conforme o código seguinte:

```

from dolfin import *
2  #----for adequate use of matplotlib
  #https://answers.launchpad.net/dolfin/+question/219220
4  dolfin.parameters.reorder_dofs_serial = False
  #-----
6  import numpy as np
  from scipy import *
8  from time import time

```

A.7.1 Código para estimativa do erro da condição inicial

A função `ErrorEstimate_for_InitialCondition(kargs**)` fornece as quantidades residuais relacionadas à condição inicial: os indicadores, a estimativa residual devido à projeção L^2 da função que determina a condição inicial e o máximo de todos os indicadores.

1.Uso:

```

error_indicator,L2Error_for_ICond,largest_error\
=ErrorEstimate_for_InitialCondition(mesh,\
Cinit,Cinitial,degree,RaiseDegree)

```

2.`mesh,Cinit,Cinitial,degree,RaiseDegree`: a malha de elementos finitos, a aproximação de elementos finitos da condição inicial, a expressão analítica para a condição inicial, o grau do espaço de funções e o número de graus adicional do espaço de funções de grau superior.

3.error_indicator,L2Error_for_ICond,largest_error: os indicadores residuais, a estimativa residual e o máximo indicador residual.

```

def ErrorEstimate_for_InitialCondition(mesh,Cinit,Cinitial,degree,RaiseDegree):
2   #-define space functions
    mesh1=mesh
4   C = FunctionSpace(mesh1,'Lagrange',degree)
    C1 = FunctionSpace(mesh1, 'Lagrange',degree+RaiseDegree)
6   c0linha=Cinitial(element=C.ufl_element());
    c0linha=project(c0linha,C1)
8   Cinit0 = Cinitial(element=C1.ufl_element());
    c1linha=interpolate(Cinit0,C1)#interpolate(Cinit0,C1)
10  #-----
    Constants = FunctionSpace(mesh1, "DG", 0)
12  w = TestFunction(Constants)
    form4=(c1linha-c0linha)**2*w*dx();    indicator4=assemble(form4)
14  error_indicator  = array([sqrt(i) for i in indicator4])
    L2Error_for_ICond = sqrt(sum(i for i in indicator4.array()))#sum(i for i in indicator4.array())
16  largest_error=max(error_indicator)
    print 'error estimates:L2 error(defined in ErrorEstimate_for_InitialCondition) and error indicator ',
        L2Error_for_ICond,largest_error
18  return error_indicator,L2Error_for_ICond,largest_error

```

A.7.2 Código para refinamento adaptativo da condição inicial

A função `RefineMesh_for_InitCond(kargs**)` fornece uma malha adequada à representação da função que descreve a condição inicial. Esse processo adaptativo é baseado na estimativa residual para a condição inicial $\|C_0 - \pi C_0\|_{L^2(\Omega)}^2$.

1.Uso:

```

RefinedMesh=RefineMesh_for_InitCond(mesh,Cinit,
    Cinitial,C, tolerance,fraction,degree,RaiseDegree)

```

2.mesh,Cinit,Cinitial,C,tolerance,fraction,degree,RaiseDegree: a malha de elementos finitos fornecida pelo usuário, a aproximação de elementos finitos da condição inicial, a expressão analítica para a condição inicial, o espaço de funções, a tolerância(critério de parada), a fração do máximo indicador em que um elemento é marcado para refinamento, o grau das funções lagrangeanas, o grau adicional para o espaço de funções com grau elevado.

3.RefinedMesh: a malha refinada

```

def RefineMesh_for_InitCond(mesh,Cinit,Cinitial,C,tolerance,fraction,degree,RaiseDegree):
2   X=ErrorEstimate_for_InitialCondition(mesh,Cinit,Cinitial,degree,RaiseDegree)
    error_indicator=X[0];L2Error_for_ICond=X[1];largest_error=X[2];

```



```

4  print 'the error inside loop of refine function',L2Error_for_ICond
   print 'the largest indicator inside loop of refine function',largest_error
6  if (X[1]>tolerance):
       print ' Initial Condition Error greater than tolerance: error=, tolerance=',X[1],tolerance
8
   while(X[1]>tolerance):
10      #mesh=mesh; Cinit = interpolate(Cinit,C)
       X=ErrorEstimate_for_InitialCondition(mesh,Cinit,Cinitial,degree,RaiseDegree)
12      error_indicator=X[0];L2Error_for_ICond=X[1];largest_error=X[2];
       if X[1]>tolerance:
14          print ' Initial Condition Error greater than tolerance: error=, tolerance=',X[1],tolerance
       else :
16          break
       print 'the error inside loop of refine function',L2Error_for_ICond
18       print 'the largest indicator inside loop of refine function',largest_error

20      cell_markers = MeshFunction("bool",mesh,mesh.topology().dim())
       for mark in cells (mesh):
22          cell_markers[mark] = ((error_indicator[mark.index()] > fraction*largest_error))
       mesh=refine(mesh,cell_markers)
24       plot(mesh,interactive=False)
   return mesh

```

A.7.3 Código modificado para refinamento adaptativo da condição inicial

Esse caso é uma modificação da função `RefineMesh_for_InitCond(kargs**)` com critério de parada baseado no indicador residual máximo. Essencialmente é o mesmo código, mas pode ser útil na geração de uma malha adaptada.

```

1  def RefineMesh_for_InitCond2(mesh,Cinit,Cinitial,C,tolerance,fraction,degree,RaiseDegree):
3
   X=ErrorEstimate_for_InitialCondition(mesh,Cinit,Cinitial,degree,RaiseDegree)
5   error_indicator=X[0];L2Error_for_ICond=X[1];largest_error=X[2];
   print 'the error inside loop of refine function',L2Error_for_ICond
7   print 'the largest indicator inside loop of refine function',largest_error

9   if (X[1]>tolerance):
       print ' Initial Condition Error greater than tolerance: error=, tolerance=',X[1],tolerance
11
   while(X[2]>tolerance):
13      #mesh=mesh; Cinit = interpolate(Cinit,C)

       X=ErrorEstimate_for_InitialCondition(mesh,Cinit,Cinitial,degree,RaiseDegree)
       error_indicator=X[0];L2Error_for_ICond=X[1];largest_error=X[2];
15
17      if X[2]>tolerance:
19          print 'maximum indicator greater than tolerance: error=, tolerance=',X[2],tolerance
       else :
21          break

```

```

23     print 'the error inside loop of refine function',L2Error_for_ICond
24     print 'the largest indicator inside loop of refine function',largest_error
25     cell_markers = MeshFunction("bool",mesh,mesh.topology().dim())
26
27     for mark in cells(mesh):
28         cell_markers[mark] = ((error_indicator[mark.index()] > fraction*largest_error))
29     mesh=refine(mesh,cell_markers)
30
31     return mesh

```

A.7.4 Código para refinamento adaptativo da função de fonte

A função `RefineMeshForFfunction(kargs**)` fornece uma malha adequada à representação da função matemática que descreve o termo de fonte. Esse processo adaptativo é baseado na estimativa residual para o termo de fonte.

1.Uso:

```

RefinedMesh=RefineMeshForFfunction(toleranceF,fractionF,
alpha_S,epsilon,theta,Fexpression,mesh,degree,raiseDegree,dt)

```

2.toleranceF,fractionF,alpha_S,epsilon: a tolerância, a fração do máximo indicador em que um elemento é marcado para refinamento, o fator de ponderação α_S , $\epsilon = \epsilon$;

3.:theta,Fexpression,mesh,degree,raiseDegree,dt: o parâmetro de discretização temporal, a expressão analítica para a função de fonte, a malha de elementos finitos, o grau das funções e o número de graus adicional para o espaço de funções de grau elevado.

4.RefinedMesh: a malha refinada

```

11 def RefineMeshForFfunction(maxIterF,toleranceF,fractionF,epsilon,beta,theta,Fexpression,mesh,degree,
12     raiseDegree,dt):
13     from Tools.ErrorEstimatorFunctions import AlphaS
14     C=FunctionSpace(mesh,'Lagrange',degree)
15     alpha_S=AlphaS(mesh,epsilon,beta,C)
16     from Tools.ErrorEstimatorFunctions import FunctionFDataErrorEstimateStepN,
17     FunctionFDataErrorIndicatorsStepN
18     DataFfunctionEstimate,DataFfunctionIndicator,\
19     DataFfunction=FunctionFDataErrorEstimateStepN(C,alpha_S,epsilon,theta,Fexpression,mesh,degree,
20     raiseDegree,dt)
21     largest_error=max(DataFfunctionIndicator)
22     DataFfunctionErrorEstimate=sqrt(DataFfunctionEstimate)
23     print 'the values of largest_error and DataFfunctionErrorEstimate',largest_error,
24     DataFfunctionErrorEstimate

```

```

13     if (DataFfunctionErrorEstimate>toleranceF):
14         print ' Initial Condition Error greater than tolerance: error=, tolerance=',
            DataFfunctionErrorEstimate,toleranceF
15
16     while(DataFfunctionErrorEstimate>toleranceF):
17         C=FunctionSpace(mesh,'Lagrange',degree)
18         alpha__S=AlphaS(mesh,epsilon,beta,C)
19
20         DataFfunctionEstimate,DataFfunctionIndicator,\
            DataFfunction=FunctionFDataErrorEstimateStepN(C,alpha__S,epsilon,theta,Fexpression,mesh,degree,
            raiseDegree,dt)
21         largest__error=max(DataFfunctionIndicator)
22         DataFfunctionErrorEstimate=sqrt(DataFfunctionEstimate)
23         print 'the values of largest__error and DataFfunctionErrorEstimate',largest__error,
            DataFfunctionErrorEstimate
24         #for i in range(maxIterF):
25             if DataFfunctionErrorEstimate>toleranceF:
26                 print 'maximum indicator greater than tolerance: error=, tolerance=',DataFfunctionErrorEstimate
                ,toleranceF
27             else :
28                 break
29
30         DataFfunctionEstimate2,DataFfunctionIndicator2,\
            DataFfunction2=FunctionFDataErrorIndicatorsStepN(C,alpha__S,epsilon,theta,Fexpression,mesh,degree
            ,raiseDegree,dt)
31
32         largest__error2=max(DataFfunctionIndicator2)
33         #DataFfunctionErrorEstimate2=sqrt(DataFfunctionEstimate2)
34         #print 'largest__error,DataFfunctionEstimate,sqrt(DataFfunctionEstimate)',largest__error,
            DataFfunctionEstimate,sqrt(DataFfunctionEstimate)
35
36         cell__markers = MeshFunction("bool",mesh,mesh.topology().dim())
37
38         for mark in cells (mesh):
39             cell__markers[mark] = ((DataFfunctionIndicator2[mark.index()] > fractionF*largest__error2))
40         mesh=refine(mesh,cell__markers)
41         #plot(mesh,interactive=False)
42
43     return mesh

```

A.8 Códigos adicionais

A.8.1 Avaliação do erro real

A função `MyErrorNorm(kargs**)` foi criada com base nos códigos e manuais disponíveis no endereço eletrônico do Projeto FEniCS e tem objetivo de acelerar o cálculo do erro executado pela função `error_norm` do FEniCS (FENICS PROJECT, 2014). Essa função pode ser usada para calcular o erro real do processo de aproximação.

1. Uso: Suponha que u e u_e são as soluções numérica e exata. O erro real, avaliado na norma L^2 , pode ser obtido através de:

`L2Error=MyErrorNorm(u,u_e,mesh,deg,raise_degree)`

2. `u,u_e,mesh,deg,raise_degree`: a solução numérica, a solução exata, a malha de elementos finitos, o grau das funções polinomiais e o número de graus adicional para construir o espaço de funções de ordem elevada.

3. Observação: solução exata significa uma representação acurada da expressão analítica no domínio computacional.

```

1 def MyErrorNorm(u,u_e,mesh,deg,raise_degree):
    mesh1=refine(mesh);
3    C1=FunctionSpace(mesh1,'Lagrange',deg+raise_degree)
    u_e=u_e(element=C1.ufl_element())
5    u_e_Ve=project(u_e,C1)#interpolate(u_e,C1)
    u_Ve=project(u,C1)
7
    e_Ve = Function(C1)
9    # Subtract degrees of freedom for the error field
    e_Ve.vector()[:] = u_e_Ve.vector().array() - u_Ve.vector().array()
11    error = e_Ve**2*dx
    return sqrt(assemble(error))

```

A.9 Reutilização das estruturas multigrid

Nesta seção, processo de reutilização das estruturas multigrid é apresentado. Este módulo implementa uma modificação no módulo Python blackbox, disponível em PyAMG (BELL; OLSON; SCHRODER, 2011) e no website <https://github.com/pyamg/pyamg>. Esse módulo possibilita a implementação dos dois esquemas de reutilização propostos nesta tese.

1. Uso:

```

    config=solver_configuration(Asp,verb=True)
    from Dictionary import DictSA
    config=DictSA(Asp, max_levels,solver_configuration)
    ml=solver(Asp,config)
    solut,itera,res = solve(Asp,b,x0=c10,tol=1e-10,maxiter=400,\
existing_solver=ml)

```

2. `config=solver_configuration(kargs**)`: fornece um dicionário de parâmetros para o método multigrid algébrico.

3.DictSA(kargs):** retorna um dicionário de parâmetros para especialização do método multigrid algébrico.

```

def DictSA(Asp,max_levels,solver_configuration):
2   config=solver_configuration(Asp,verb=True)
   config['symmetry']='nonsymmetric'
4   config['B'] = np.ones((Asp.shape[0],1), dtype=Asp.dtype)
   config['BH'] = config['B'].copy()
6   config['smooth'] = ('energy', {'krylov': 'bicg'}, \
   {'filter' : True},{ 'sweep': 'symmetric', 'iterations':2})
8   config['strength'] = ('classical', {'theta':0.1})
   config['max_levels']=(max_levels);   config['max_coarse']=(1000)
10  config['coarse_solver']='lu' ;       config['symmetry']='symmetric'
   config['aggregate']='standard'
12  config['presmoothing'] = ('block_gauss_seidel', {'sweep': 'symmetric', 'iterations':2})
   config['postsmoothing'] = ('block_gauss_seidel', {'sweep': 'symmetric', 'iterations':2})
14  config['SA_solve_args']=( {'cycle':'V'}, {'tol':1.000001e-12})
   config['improve_candidates']=[('block_gauss_seidel', {'sweep': 'symmetric', 'iterations':1})]
16  config['diagonal_dominance']=(False)
   config['keep'] = (False)
18  return config

```

4.Asp,verb: matriz do sistema linear e variável booleana.

5.max_levels,solver_configuration: número de níveis e configuração padrão. Se nenhuma especialização é fornecida, então a função DictSA retorna a configuração padrão do módulo blackbox disponível em PyAMG (BELL; OLSON; SCHRODER, 2011).

```

"""
2 The MIT License (MIT)

4 Copyright (c) 2008–2015 PyAMG Developers

6 Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
8 in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
10 copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

12 The above copyright notice and this permission notice shall be included in all
14 copies or substantial portions of the Software.

16 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
18 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
20 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
22 SOFTWARE.
"""

```

```

24 __docformat__ = "restructuredtext en"
import numpy
26 import scipy
from scipy.sparse import isspmatrix_csr, isspmatrix_bsr, csr_matrix
28 from pyamg import smoothed_aggregation_solver
from pyamg.util.linalg import ishermitian, norm
30 __all__ = ['solve', 'solver', 'solver_configuration']
def make_csr(A):
32     if not (isspmatrix_csr(A) or isspmatrix_bsr(A)):
        try:
34             A = csr_matrix(A)
            print 'Implicit conversion of A to CSR in pyamg.blackbox.make_csr'
36         except:
            raise TypeError('Argument A must have type csr_matrix or bsr_matrix,\
38                             or be convertible to csr_matrix')
        if A.shape[0] != A.shape[1]:
40             raise TypeError('Argument A must be a square')
        A = A.asfptype()
42     return A

44 def solver_configuration(A, B=None, verb=True):
    A = make_csr(A); config = {}
46     # Detect symmetry
    if ishermitian(A, fast_check=True):
48         config['symmetry'] = 'hermitian'
        if verb:
50             print "Detected a Hermitian matrix"
    else:
52         config['symmetry'] = 'nonsymmetric'
        if verb:
54             print "Detected a non-Hermitian matrix"
    if config['symmetry'] == 'hermitian':
56         config['smooth'] = ('energy', {'krylov': 'cg', 'maxiter': 3, 'degree': 2, 'weighting': 'local'})
        config['presmoothen'] = ('block_gauss_seidel', {'sweep': 'symmetric', 'iterations': 1})
58         config['postsmoothen'] = ('block_gauss_seidel', {'sweep': 'symmetric', 'iterations': 1})
        config['strength'] = ('evolution', {'k': 2, 'proj_type': 'l2', 'epsilon': 3.0})
60     else:
        config['smooth'] = ('energy', {'krylov': 'gmres', 'maxiter': 3, 'degree': 2, 'weighting': 'local'})
62         config['presmoothen'] = ('block_gauss_seidel', {'sweep': 'symmetric', 'iterations': 2})
        config['postsmoothen'] = ('block_gauss_seidel', {'sweep': 'symmetric', 'iterations': 2})
64         config['strength'] = ('evolution', {'k': 2, 'proj_type': 'l2', 'epsilon': 3.0})
    # Determine near null-space modes B
66     if B == None:
        # B is the constant for each variable in a node
68         if isspmatrix_bsr(A) and A.blocksize[0] > 1:
            bsize = A.blocksize[0]
            config['B'] = numpy.kron(numpy.ones((A.shape[0]/bsize, 1), dtype=A.dtype), \
70                                     numpy.eye(bsize))
72         else:
            config['B'] = numpy.ones((A.shape[0], 1), dtype=A.dtype)
74     elif (type(B) == type(numpy.zeros((1,)))) or (type(B) == type(scipy.mat(numpy.zeros((1,))))):
76         if len(B.shape) == 1:
            B = B.reshape(-1, 1)
78         if (B.shape[0] != A.shape[0]) or (B.shape[1] == 0):
            raise TypeError('Invalid dimensions of B, B.shape[0] must equal A.shape[0]')
80     else:

```

```

        config['B'] = numpy.array(B, dtype=A.dtype)
82     else:
        raise TypeError('Invalid B')
84     if config['symmetry'] == 'hermitian':
        config['BH'] = None
86     else:
        config['BH'] = config['B'].copy()
88
        # Set non-symmetry related parameters
90     config['strength'] = ('evolution', {'k':2, 'proj_type':'l2', 'epsilon':3.0})
        config['max_levels'] = 15
92     config['max_coarse'] = 1000
        config['coarse_solver'] = 'lu'
94     config['aggregate'] = 'standard'
        config['keep'] = False
96     return config

98 def solver(A, config):
    # Convert A to acceptable format
100    A = make_csr(A)
    # Generate smoothed aggregation solver
102    try:
        return smoothed_aggregation_solver(A,
104            B=config['B'],
            BH = config['BH'],
106            smooth = config['smooth'],
            strength = config['strength'],
108            max_levels = config['max_levels'],
            max_coarse = config['max_coarse'],
110            coarse_solver = config['coarse_solver'],
            symmetry = config['symmetry'],
112            aggregate = config['aggregate'],
            presmoothing = config['presmoothing'],
114            postsmoothing = config['postsmoothing'],
            keep = config['keep'])
116    except:
        raise TypeError('Failed generating smoothed_aggregation_solver')
118 def solve(A, b, x0=None, tol=1.000000001E-08, maxiter=400, residuals=None,
        return_solver=False, existing_solver=None, verb=True):
120     # Convert A to acceptable CSR/BSR format
        A = make_csr(A)
122     # Generate solver if necessary
        if existing_solver == None:
124         # Parameter dictionary for smoothed_aggregation_solver
            config = solver_configuration(A, B=None, verb=verb)
126         # Generate solver
            existing_solver = solver(A, config)
128     else:
        if existing_solver.levels[0].A.shape[0] != A.shape[0]:
130         raise TypeError('Argument existing_solver must have level 0 matrix of same size as A')
        # Krylov acceleration depends on symmetry of A
132     if existing_solver.levels[0].A.symmetry == 'symmetric':
        accel = 'cg'
134     else:
        accel = None
136     # Initial guess
        if x0 == None:

```

```

138     x0 = numpy.array(scipy.rand(A.shape[0,], dtype=A.dtype)
else :
140     x0=x0
# Callback function to print iteration number
142     if verb:
        iteration = numpy.zeros((1,))
144         print "    maxiter = %d"%maxiter
        def callback(x, iteration ):
146             iteration [0] = iteration [0] + 1
            print "    iteration %d"%iteration[0]
148             callback2 = lambda x : callback(x, iteration )
        else :
150             callback2 = None
# Solve with accelerated Krylov method
152     res=[]
x = existing_solver.solve(b, accel=accel, residuals=res, tol=tol, maxiter=maxiter, callback=callback2)
154     if verb:
        r0 = norm( numpy.ravel(b) - numpy.ravel(A*x0))
156         rk = norm( numpy.ravel(b) - numpy.ravel(A*x) )
        if r0 != 0.0:
158             print "    Residual reduction ||r_k||/||r_0|| = %1.2e"%(rk/r0)
        else :
160             print "Residuals ||r_k||, ||r_0|| = %1.2e, %1.2e"%(rk,r0)

162     if return_solver:
        return (x.reshape(b.shape), existing_solver, config['B'], iteration , res
164     else :
        return x.reshape(b.shape), iteration , res

```

A.10 Nota Final

Para facilitar a utilização, as classes foram separadas nos seguintes módulos. Por sua vez, contém as funções descritas nesse apêndice.

ErrorEstimatorFunctions.py

- projectionFunction(kargs**)
- ProjectionGradient(kargs**)
- MakeSystem_and_TransformData(kargs**)
- ErrorComponents(kargs**)
- AlphaS(kargs**)
- Indicators(kargs**)
- ErrorEstimatesSpaceTime(kargs**)
- AuxiliarErrorEstimatesSpaceTime(kargs**)
- MaxMinIndicators(kargs**)
- EachStepError_Indicator_Space_Time(kargs**)


```

-FormLargeAdvection(kargs**)
-LargeAdvectionEstimates(kargs**)
-DiffFunction(kargs**)
-DiffFunction2(kargs**)
-DataResidualComponents(kargs**)
-DataResidualComponents2(kargs**)
-FormDataOneNeumann(kargs**)
-FormOneNeumannIndicator(kargs**)
-DataErrorEstimateStepN(kargs**)
-FunctionFDataErrorEstimateStepN(kargs**)
-FunctionFDataErrorIndicatorsStepN(kargs**)
-NeumanDataErrorEstimateN(kargss**)

```

`PolyErrorEstimator.py`

```

- ErrorEstimate_for_InitialCondition(kargss**)
-RefineMesh_for_InitCond(kargss**)
- RefineMesh_for_InitCond2(kargss**)
- RefineMeshForFfunction(kargss**)
- RefineMeshForData(kargss**)
- MyErrorNorm(kargss**)

```

Essas funções, por sua vez, foram englobadas em um único módulo chamado *Tools*. Esse procedimento simplifica a quantidade de informação a ser gerenciada dentro de uma mesma pasta da plataforma Eclipse.

As demais funções utilizadas foram definidas localmente no código computacional. O apêndice seguinte apresenta alguns exemplos de códigos utilizados nesta tese.

APÊNDICE B – Exemplos completos

B.1 FEM e solução analítica

Exemplo de implementação da aproximação de elementos finitos com cálculo de solução analítica.

```

1  """Exemplo: equacao do transporte de contaminantes.
   i. aproximacao por elementos finitos.
3
   Objetivo do codigo: servir de base para estudos posteriores .
5  Consultar referencias para maiores detalhes.

7  Referencias basicas:
   1. http://fenicsproject.org/documentation/tutorial/index.html
   2. http://fenicsproject.org/qa/
   3. http://matplotlib.org/
11  4. http://www.numpy.org/
   5. http://www.scipy.org/
13  6. R. Verfurth, Adaptive finite element methods lecture notes winter term 2007/08.
   7. https://code.google.com/p/scitools/wiki/EasyvizDocumentation
15  8. WEXLER, E. Chapter b7: Applications of hydraulics analytical solutions for one-, two-
   and three-dimensional solute transport in groundwater systems with uniform flow. In:
17  Techniques of Water Resources Investigations of the United State Geological Survey, Book
   3: Applications to Hydraulics. Denver, USA: U.S. Geological Survey, 1992.
19  """

21  #----a biblioteca para elementos finitos
   from dolfin import *
23  from numpy import *#a biblioteca para matrizes
   import numpy as np #alternativa para evitar duplicidade
25  import scipy as sp # Python Científico-Scipy
   import matplotlib as mpl #biblioteca para graficos
27  import matplotlib.pyplot as plt #plot

29

   from scipy import integrate# pacote de integracao numerica(ver manual do scipy)
31  import scitools.BoxField# grafico
   import scitools.easyviz as ev#grafico
33  parameters.reorder_dofs_serial = False#evita reordenacao. Necessario para usar BoxField
   #---questao respondida em: https://answers.launchpad.net/dolfin/+question/219220
35  #parametros do metodo de elementos finitos
   x0=0;y0=0;x1=3000;y1=1500;nx =100;ny = 50;
37  #fronteiras da fonte, grau, esquema de discretizacao, passo de tempo tau_n, tempo final
   Y1=635.0;Y2=865.0;degree=1;theta = 0.50;dt=5.0;t_stop = 4000.0;nsteps=floor(t_stop/dt);
39  nsteps=int(nsteps); print 'the number of steps',nsteps
   #-----malha de elementos triangulares orientada
41  mesh = RectangleMesh(x0,y0,x1,y1,nx, ny,'left/right')
   #-----resumo da malha de elementos
43  print "%s"%str(mesh)
   #Funcoes continuas por partes com grau=degree

```

```

45 C = FunctionSpace(mesh, 'Lagrange', degree)
#=====Fronteiras
47 #---Fronteira de Dirichlet 01
u_c = Expression('1.0')#funcao
49 class C0Boundary(SubDomain):
    def inside( self , x, on_boundary):
51         tol = 1E-14 # tolerance for coordinate comparisons
        return on_boundary and (abs(x[0]) < tol and abs(x[1]-750.0)<115.0)
53 Gamma_0 = DirichletBC(C, u_c, C0Boundary())
#---Fronteira de Dirichlet 02
55 u_c1 = Expression('0.0')
class C1Boundary(SubDomain):
    def inside( self , x, on_boundary):
57         tol = 1E-14 # tolerance for coordinate comparisons
        return on_boundary and (abs(x[0]) < tol and abs(x[1]-750.0)>115.0)
59 Gamma_1 = DirichletBC(C, u_c1, C1Boundary())
61 #---lista de fronteiras.
#Ex: bcs=[Gamma0,Gamma1,...,Gamma10]
63 bcs=[Gamma_0,Gamma_1]
#-----Fronteiras de Neumann
65 #---"marcacao do dominio das entidades do dominio"
#para definicao das fronteiras
67 boundary_parts = MeshFunction("size_t", mesh, mesh.topology().dim()-1)
#---fronteira de neumann em y=0 e y=1500. Usar 0.0 e 1500.0
69 class NeumanBoundary(SubDomain):
    def inside( self , x, on_boundary):
71         tol = 1E-14 # tolerance for coordinate comparisons
        return on_boundary and (abs(x[1]) < tol or abs(x[1]-1500.0)<tol)
73 N_0 = NeumanBoundary()
N_0.mark(boundary_parts, 0)
75 #---fronteira de neumann em y=3000.0
class NRightBoundary(SubDomain):
77     def inside( self , x, on_boundary):
        tol = 1E-14 # tolerance for coordinate comparisons
79         return on_boundary and abs(x[0] - 3000.0) < tol
N_1 = NRightBoundary()
81 N_1.mark(boundary_parts, 1)
#=====Dados do problema definidos no espaco
83 #de elementos finitos
#---expressao para a condicao inicial
85 C0=Expression('0.000000');
#-----interpolando para o espaco de elementos finitos
87 c1=interpolate(C0,C)
#---fonte e dispersoes: expressoes
89 f=Constant('0.0')# fonte
dxx=10.0; dyy=2.0;dxy=0.0;dyx=dxy;D=as_matrix([[dxx,dxy],[dyx,dyy]])
91 #---velocidades em x e y. Campo de velocidades: VField
vx=Expression('0.2'); vy=Expression('0.0'); VField=(vx,vy)
93 #expressao que fornece velocidade.Aqui vx e vy sao usados
H=Expression('0.2*x[0]'); H1= interpolate(H, C)
95 #---reacao e vetor normal
Lambda=Constant('0.0'); n=FacetNormal(mesh)
97 #=====Formas bilinear e linear
#funcao tentativa e funcao teste
99 c= TrialFunction(C)
w = TestFunction(C)
101 f1=f#f~n=f~{n-1}(ver verfurth nas referencias basicas)

```

```

#-----forma bilinear e linear: s(0) e s(1) indicam fronteira Neumann-ver [1.]
103 a = (1.0/dt)*c*w*dx + theta*inner(D*nabla_grad(c),nabla_grad(w))*dx+\
      theta*(inner(VField, nabla_grad(c)))*w*dx + theta*Lambda*c*w*dx-\
105      theta*inner(D*grad(c),n)*w*ds(0)-theta*inner(D*grad(c),n)*w*ds(1)
#-----forma linear
107 L = ((1.0/dt)*c1*w+(theta-1.0)*inner(D*nabla_grad(c1), nabla_grad(w))*dx+\
      (theta-1.0)*inner(VField, grad(c1))*w*dx+(theta-1.0)*Lambda*c1*w*dx+(theta*f1+(1.0-theta)*f)*w*
      dx+\
109      (1.0-theta)*inner(D*grad(c1),n)*w*ds(0)+(1.0-theta)*inner(D*grad(c1),n)*w*ds(1)
#-----construindo sistema de equacoes com fronteira de neumann
111 A = assemble(a, exterior_facet_domains=boundary_parts)
# lado esquerdo. Forma indicada no manual para
113 #economizar memoria
b = None
115 #-----funcao
c = Function(C);
117 t = dt;contador=0;counter=0;tol=1.0E-5;
#---marcador do tempo
119 #=====incremento tempo e
#solucao e loop no tempo
121 while t < t_stop+tol:
    #-----construindo o lado b com fronteira de neumann
123     b = assemble(L,exterior_facet_domains=boundary_parts)
    #-----Fronteira de Dirichlet
125     for bc in bcs:
        bc.apply(A, b)
127     #-----resolvendo com metodo padrao do dolfin. Calculando o tempo
    #necessario para a solucao
129     from time import time
    tempo10=time()
131     solve(A,c.vector(), b)
    from time import time
133     tempo11=time()
    # Definindo temo computacional
135     TempoComputcional=tempo11-tempo10
    print 'O tempo computacionla',TempoComputcional
137     contador+=1; print 'contador',contador
    c1.assign(c)
139     #-----salvando solucao especifica
    if contador==nsteps-1:
141         print 'saving a time before final time step',t
        #salvar na mesma pasta da simulacao. Pasta diferente basta
143         # fornecer o caminho
        #-----np.save(): ver manual do numpy. Salva em formato .npv
145         #-----ler com np.read(): ver manual do numpy.
        np.save('cReaction',c1.vector())
147         t+= dt
        plot(c1,mesh=mesh,interactive=False)
149     #-----salva solucao para pos-processamento
    #-vetor de valores nodais da solucao numerica
151     np.save('c1Reaction',c1.vector())
    #-----salva figura solucao em formato .png
153     p=plot(c1,mesh=mesh,interactive=False)
    p.write_png('SolucaoTfinal')
155
#-----
157 #Solucao analitica de Wexler(ver referencia 08.) para aquifero infinito e fonte nao pontual de

```

```

#contaminante
159 print 'Analytical solution calculation'
#Dados
161 C0=1.0; dxx=10.0;dyy=2.0;dxy=0.0;dyx=dxy;
vx=0.2;vy=0.0
163 # expressao fora da integral:Ver referencia
def Expr_out_integral(x,y):
165     sigma=(1./sqrt(np.pi*dxx))
     return C0*x*sigma*exp((vx)*x/(2*dxx))
167 reaction_rate=0.00

169 #expressao dentro da integral:ver referencia
def Expr_to_integrate(z,x,y):
171     # quadpack: biblioteca fortran para integracao[Ver referencia SciPy]
     #usa primeiro argumento para integrar
173     exp1=(1.0/(z**3))*exp(-(((vx)**2)/(4*dxx)+reaction_rate)*(z**4)-(x**2)/(4*dxx*(z**4)))
     #fator dentro da primeira funcao erfc
175     Y11=(Y1-y)/(2*(z**2)*sqrt(dyy))
     #fator dentro da segunda funcao erfc
177     Y12= (Y2-y)/(2*(z**2)*sqrt(dyy))
     #definindo funcao auxiliar erfc1
179     erfc1=1.0-erf(Y11)-(1.0-erf(Y12))

181     return exp1*erfc1

183 # Uma classe que implementa a solucao analitica: impondo
# condicao de fronteira e executando integracao com modulo
185 #integrate do QuadPack...[Ver SciPy]
#http://www.scipy.org/
187
class analytical(Expression):
189     def eval( self , value , x):
         tol=1.0E-15
191         if (abs(x[0])<tol and abs(x[1]-750.0)>115.0):
             value[0]=0.0
193         elif (abs(x[0])<tol and abs(x[1]-750.0)<115.0):
             value[0]=C0
195         else :
             #limite superior de integracao
197             sup_z=t_stop**0.25
             #-----
199             y,err=integrate.quad(Expr_to_integrate,0,sup_z, args=(x[0],x[1],))
             y2=Expr_out_integral(x[0],x[1])
             value[0]=y*y2
201
f2= analytical(element=C.ufl_element());
203 f2 = interpolate(f2,C)

205 #salvando vetor de valores nodais
#solucao analitica
207 f2_exact=f2.vector()
np.save('c_ReactionExact.npy',f2_exact)

```

B.1.1 Comparação das soluções numérica e analítica, gráficos

As soluções podem ser comparadas tanto nos pontos nodais quanto em forma gráfica. O código seguinte apresenta um exemplo de comparação dos valores nodais e apresenta os resultados em forma gráfica.

```

1  """Exemplo: Analise dos resultados obtidos no caso anterior
Comparacao de resultados e graficos com matplotlib
3  Referencias basicas:
1. http://fenicsproject.org/documentation/tutorial/index.html
5  2. http://fenicsproject.org/qa/
3. http://matplotlib.org/
7  4. http://www.numpy.org/
5. http://www.scipy.org/
9  6. R. Verfurth, Adaptive finite element methods lecture notes winter term 2007/08.
7. https://code.google.com/p/scitools/wiki/EasyvizDocumentation
11 """

13 #----a biblioteca para elementos finitos
from dolfin import *
15 #a biblioteca para matrizes
from numpy import *
17 #alternativa para evitar duplicidade
import numpy as np
19 # Python Científico-Scipy
import scipy as sp
21 import matplotlib as mpl #biblioteca para graficos
import matplotlib.pyplot as plt #plot
23
# pacote de integracao numerica(ver manual do scipy)
25 from scipy import integrate
import scitools.BoxField# grafico
27 import scitools.easyviz as ev#grafico
#evita reordenacao. Necessario para usar BoxField
29 parameters.reorder_dofs_serial = False
#---questao respondida em: https://answers.launchpad.net/dolfin/+question/219220
31 #=====Definindo parametros gerais
#-----parametros do metodo de elementos finitos
33 x0=0;y0=0;x1=3000;y1=1500;nx =100;ny = 50;
##-----fronteiras da fonte, grau, esquema de discretizacao, passo de tempo tau_n,
tempo final
35 Y1=635.0;Y2=865.0;degree=1;theta = 0.50;dt=5.0;t_stop = 4000.0
nsteps=floor(t_stop/dt); nsteps=int(nsteps);
37 print 'the number of steps',nsteps
#-----malha de elementos triangulares orientada
39 mesh = RectangleMesh(x0,y0,x1,y1,nx, ny,'left/right')

41 C = FunctionSpace(mesh,'Lagrange',degree)
#-----Carregando solucao numerica obtida
43 c1=np.load('c1Reaction.npy')
solut=Function(C)
45 solut.vector()[:] =c1
c1=Function(C)
47 c1.assign(solut)
#visualizacao rapida da solucao em malha: modo inter

```

```

49 plot(c1,mesh=mesh,interactive=True)
# Carregando solucao analitica calculada. Considerando mesmo espaco
51 c_exact=np.load('c_ReactionExact.npy')
u_e=Function(C)
53 u_e.vector()[:]=c_exact
#classe diff: diferenca entre solucoes em modulo
55 class diff(Expression):
    def eval(self, value, x):
57         tol=1.0E-15
        valor=u_e(x[0],x[1])-c1(x[0],x[1])
59         value[0]=abs(valor)
diff = diff(element=C.ufl_element());
61 diff = interpolate(diff,C)

63 class diff2(Expression):
    def eval(self, value, x):
65         tol=1.0E-15
        valor=u_e(x[0],x[1])-c1(x[0],x[1])
67         value[0]=valor
diff2 = diff2(element=C.ufl_element());
69 diff2 = interpolate(diff2,C)

71 u_nodal_values = c1.vector();u_array = u_nodal_values.array()
ue_nodal_values = u_e.vector();ue_array = ue_nodal_values.array()
73 diffe_array=diff.vector().array()

75 #imprimindo valores na tela
coor = mesh.coordinates()
77 maxvalue=max(diffe_array)
# valores referentes as diferencas de solucoes
79 print 'the maximum value for difference matrix',maxvalue
print 'the mean for differences of errors', mean(diffe_array)
81 print 'the mean for differences of errors',std(diffe_array)

83 if coor.shape[0] == u_array.shape[0]:
    for i in range(len(u_array)):
85         if diffe_array[i]>0.5*maxvalue:
            print 'C(%f,%f) = %f,%f,%f' %(coor[i][0], coor[i][1], u_nodal_values[i],ue_nodal_values
            [i],diffe_array[i])
87
# L2 norma usando FEniCS metodologia
89 error_norm=errornorm(u_e,c1,'L2',1)
print 'norma L2 usando FEniCS funcao=', error_norm
91 #Graficos com a biblioteca MATPLOTLIB
print 'NOTA: Se degree>1 precisa projetar solucao para\n'
93 print 'espaco com degree=1. Ver funcao "project(kargs**)' do FEniCS"'
#=====
95 from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
97 from matplotlib.ticker import LinearLocator
import matplotlib.pyplot as plt
99 import matplotlib.tri as tri
import numpy as np

101 xy=mesh.coordinates()
103 x=xy[:,0]; y=xy[:,1]
triangle=mesh.cells()# a nx3 matrix storing triangles conectivities

```



```

105 triang = tri.Triangulation(x, y, triangle)
    Z1e=u_e.vector()[:];
107 Z1n=c1.vector()[:];
    Z1d=diff.vector()[:];
109 seq=np.linspace(0.0001,1.0,100)
    #-----
111 plt.figure()
    print 'plotting figure 1: curvas da analitica'
113 plt.gca().set_aspect('1.5')
    plt.tricontour(xy[:,0], xy[:,1], Z1e,seq,linewidth=0.6,zdir='z', offset=0.0)
115 plt.colorbar(); plt.grid(True)
    plt.xlabel('x'); plt.ylabel('y')
117 #salva resultado na pasta atual
    plt.savefig('analiticaCurvas.png')
119 #-----
    fig1 = plt.figure()
121 print 'plotting figure: numerica 3D'
    ax = fig1.gca(projection='3d')
123 ax.tick_params(axis='both', which='major', labelsize=20)
    ax.tick_params(axis='both', which='minor', labelsize=20)
125 ax.plot_trisurf(xy[:,0], xy[:,1], Z1n,cmap=cm.gist_heat_r,linewidth=0.2)
    ax.view_init(47,-57)
127 ax.set_xlim(x0,x1)
    ax.set_ylim(y0,y1)
129 t1=ax.set_xlabel(r'$x$', fontsize=20,color='black')
    ax.set_ylabel(r'$y$', rotation=45,fontsize=20,color='black')
131 ax.text(500.0,1400.00,0.7, r"$C=C(x,y,t)$",fontsize=20,color='blue')
    ax.set_zlim3d(0.0, 0.10)
133 ax.set_zlabel(r"$C$",rotation=45,fontsize=20,color='black')
    plt.savefig('numerica3D.png')
135 plt.show()

```

B.2 Estratégia adaptativa

Código para malha adaptada para os dados similar ao exemplo (5.3.4.1). Malha não estruturada em campo de velocidade variável. O procedimento adaptativo é utilizado sequencialmente nos dados da equação do transporte.

```

1  """
    Referencias basicas:
2  1. http://fenicsproject.org/documentation/tutorial/index.html
3  2. http://fenicsproject.org/qa/
4  3. http://matplotlib.org/
5  4. http://www.numpy.org/
6  5. http://www.scipy.org/
7  6. R. Verfurth, Adaptive finite element methods lecture notes winter term 2007/08.
8  7. https://code.google.com/p/scitools/wiki/EasyvizDocumentation
9  8. Zoppou, C. and Knight, J. H., Analytical solution of a spatially variable coefficient
11 advection–diffusion equation in up to three dimensions,v.23,doi = {10.1016/S0307–904X(99)00005–0},

```

```

number = {9}, {Applied Mathematical Modelling}, year = {1999}, pages = {667--685}.
13 9. Dirk Praetorius, Ewa Weinmuller, Philipp Wissgott, A Space-Time Adaptive Algorithm for Linear
    Parabolic Problems,
    Institute for Analysis and Scientific Computing Vienna University of Technology – TU Wien, ASC Report
    07/2008,
15 url={ www.asc.tuwien.ac.at ISBN 978-3-902627-00-1}
    """
17
18 #-----The velocity field for transport problem
19 from scitools.std import *
20 from numpy import *
21 from scitools.easyviz.gnuplot__ import *
22 from dolfin import *
23 dolfin.parameters.reorder_dofs_serial = False
24 #https://answers.launchpad.net/dolfin/+question/219220
25 import numpy as np
26 from scipy import *
27 from time import time
28 if not has_cgal():
29     print "DOLFIN must be compiled with CGAL to run this demo."
30     exit(0)
31
32 x0=0.0;y0=0.0;x1=20.0;y1=20.0;nx = 20;ny = 20;degree=1;theta = 1.0; dt=0.0025;t_stop = 0.10
33
34 # Create empty Mesh
35 mesh = Mesh()
36
37 # Create list of polygonal domain vertices
38 domain_vertices = [Point(0.0, 0.0),Point(20.0, 0.0),Point(20.0, 20.0),Point(0.0, 20.0)]
39
40 # Generate mesh and plot
41 PolygonalMeshGenerator.generate(mesh, domain_vertices, 0.70);
42 C = FunctionSpace(mesh, 'Lagrange',degree)
43
44 #-----Data: Dispersions
45 u0=0.1;D0=1.0;kappa=4.41;epsilon=0.01;beta=0.0
46
47 from Tools.PolyErrorEstimator import ErrorEstimate_for_InitialCondition
48 from Tools.PolyErrorEstimator import RefineMesh_for_InitCond,MyErrorNorm
49
50 C = FunctionSpace(mesh, 'Lagrange',degree) # defining the Function Space on the adapted mesh
51
52 class dxx(Expression):
53     def eval( self , values , x):
54         """ Defining the dxx component as a function of the velocity field that is defined over all
55             elements in mesh...
56             Could use the same space of fuctions for solving the flux and transport equations because the space
57             can be the same..
58             just the coefficients for interpolating functions are differents """
59         values[0]=D0*u0*u0*(x[0]+1.0)*(x[0]+1.0)
60 dxx1 = dxx(element=C.ufl_element());
61 dxx2 = interpolate(dxx1,C)
62 toldxx=0.001;fraction=0.5;degree=1;RaiseDegree=2
63 u0=0.1;D0=1.0;kappa=4.41;epsilon=0.01;beta=0.0
64 mesh=RefineMesh_for_InitCond(mesh,dxx2,dxx,C,toldxx,fraction,degree=degree,RaiseDegree=RaiseDegree);
65
66 #atualizacao

```

```

65 C = FunctionSpace(mesh, 'Lagrange',degree)
    dxx1 = dxx(element=C.ufl_element());
67 dxx2 = interpolate(dxx1,C);
    EdxxRefine = errornorm(dxx1, dxx2,"L2", 3,mesh=refine(mesh));
69 Edxx=errornorm(dxx1,dxx2,"L2",3)
    raise_degree=3
71
    MyErrorordxx=MyErrorNorm(dxx2,dxx,mesh,degree,raise_degree)
73 print 'the L2 norm calculated with function',EdxxRefine,Edxx
    print 'The L2 norm of the representation of dxx dispersion',MyErrorordxx
75

```

B.3 Código completo

Esse código é completo no sentido que incorpora: solução do problema do transporte pelo método multigrid com reutilização, a solução do problema auxiliar utilizando método do gradiente conjugado pré-condicionado por método multigrid algébrico com reutilização, as estimativas residuais para o regime de grande advecção com partição do estimador residual, os indicadores residuais e respectiva transformação para posterior análise e apresentação gráfica.

```

#from dolfin import *
2 from numpy import *
import numpy as np
4 from time import time
from dolfin.cpp.mesh import Mesh
6 #===Part I: Setup problem with Dolfin
try:
8     from dolfin import *
except ImportError:
10     raise ImportError('Problem with Dolfin Installation')
parameters.linear_algebra_backend = "uBLAS"
12 #=== Part II: Solve with PyAMG
from scipy.sparse import csr_matrix
14 from scipy.sparse import csc_matrix
from pyamg import smoothed_aggregation_solver
16 from numpy import intc
from BlackBox2Test import solve,solver_configuration,solver
18 from scipy.linalg import norm
if not has_cgal():
20     print "DOLFIN must be compiled with CGAL to run this demo."
    exit(0)
22
x0=0.0;y0=0.0;x1=20.0;y1=20.0;nx =200;ny = 200;degree=1;
24 theta = 1.0;dt=0.001;t_stop = 0.1; epsilon=0.5;kappa=220.5;beta=0.0;

26 mesh = RectangleMesh(x0,y0,x1,y1,nx, ny,'left/right')
print "%s"%str(mesh)
28

```

```

C = FunctionSpace(mesh, 'Lagrange',degree)
30 u_c=Constant(0.0)
class C0Boundary(SubDomain):
32     def inside( self , x, on_boundary):
        tol = 1E-14
34         return on_boundary and abs(x[0]) < tol or abs(x[1])<tol
Gamma_0 = DirichletBC(C, u_c, C0Boundary())
36 bcs=[Gamma_0]
boundary_parts = MeshFunction("size_t", mesh, mesh.topology().dim()-1)
38 class NeumanBoundary(SubDomain):
    def inside( self , x, on_boundary):
40         tol = 1E-14 # tolerance for coordinate comparisons
        return on_boundary and (abs(x[1]-y1)<tol and abs(x[0] - x1) < tol)
42 N_0 = NeumanBoundary()
N_0.mark(boundary_parts, 0)
44 mu2=5.0;mu1=5.0;sigma=0.50
class Cinitial (Expression):
46     def eval( self , values, x):
        tol=1.0E-5
48         values[0] = 1.0*exp(-0.5*pow(((x[1]-mu2)/sigma),2)-0.5*pow(((x[0]-mu1)/sigma),2))
Cinit = Cinitial(element=C.ufl_element());
50 Cinit = interpolate(Cinit,C)
c1=Cinit
52
54 u0=0.50
56 ''' Defining the velocity field based on a Fenics expression. The potential velocity
definition ... as Zoppou and Knight h(x,y)=u0(x^2+y^2)/2'''
HeadExpression=Expression('(u0/2.0)*(x[0]+1.0)*(x[0]+1.0)-(u0/2.0)*(x[1]+1.0)*(x[1]+1.0)',u0=u0)
58 class HEAD(Expression):
    def eval( self , values, x):
60         values[0]=(u0/2.0)*(x[0]+1.0)*(x[0]+1.0)-(u0/2.0)*(x[1]+1.0)*(x[1]+1.0)
head = HEAD(element=C.ufl_element());
62 head = interpolate(head,C)
vx_e=Expression('u0*(x[0]+1.0)',u0=u0)
64 vy_e=Expression('-u0*(x[1]+1.0)',u0=u0)
h=interpolate(HeadExpression,C)
66 V=grad(h)#campo descontinuo
D0=2.0
68 class dxx(Expression):
    def eval( self , values, x):
70         values[0]=D0*u0*u0*(x[0]+1.0)*(x[0]+1.0)
dxx1 = dxx(element=C.ufl_element());
72 dxx1 = interpolate(dxx1,C)
class dyy(Expression):
74     def eval( self , values, x):
        values[0]=D0*u0*u0*(x[1]+1.0)*(x[1]+1.0)
76 dyy1 = dyy(element=C.ufl_element());
dyy1 = interpolate(dyy1,C)
78
dxy=dyx=Constant(0); dxx=dxx1;dyy=dyy1; D=as_matrix([[dxx,dxy],[dyx,dyy]])
80 fc=Expression('0.0')
v=V;Lambda=Constant(0.0)
82 c= TrialFunction(C)
w = TestFunction(C)
84 n=FacetNormal(mesh)
f=Constant(0.0);f1=f

```

```

86 a = (1.0/dt)*c*w*dx + theta*inner(D*nabla_grad(c),nabla_grad(w))*dx+\
    theta*(inner(v, nabla_grad(c))*w*dx + theta*Lambda*c*w*dx-\
88     theta*inner(D*grad(c),n)*w*ds(0)#-theta*inner(D*grad(c),n)*w*ds(1)
L = ((1.0/dt)*c1*w+(theta-1)*inner(D*nabla_grad(c1), nabla_grad(w))*dx+\
90     (theta-1.0)*inner(v, grad(c1))*w*dx+(theta-1.0)*Lambda*c1*w*dx+(theta*f+(1.0-theta)*f)*w*dx+\
    (1.0-theta)*inner(D*grad(c1),n)*w*ds(0)#+(1.0-theta)*inner(D*grad(c1),n)*w*ds(1)
92 A = assemble(a,exterior_facet_domains=boundary_parts)
rhs = None
94
c= Function(C)
96 t = dt
tol=1.0E-5
98 contador=0

100 hvalue=mesh.hmax()
print 'the size of the mesh',hvalue
102 from ErrorEstimatorFunctions import AlphaS
alpha_S=AlphaS(mesh,epsilon,beta,C)
104 alpha_S=interpolate(alpha_S,C)

106 arrayalphas=alpha_S.vector()
print 'valor maximo alphaS',max(arrayalphas)
108 vx=vx_e;vy=vy_e;ReactionFunc=Lambda
raiseDegree=RaiseDegree
110 from Tools.ErrorEstimatorFunctions import ErrorComponents, AssembleForm_Estimates,Indicators
tol=1.0E-5
112
def FormErrorsDef(alpha_S,mesh,gx,rk,gradot,salt,rneuman,rdirichlet,epsilon,beta):
114
    Constants = FunctionSpace(mesh,"DG", 0)
116    w = TestFunction(Constants)
    form1 =alpha_S**2*rk**2*w*dx()# spatial interior
118    form2=(1.0/np.sqrt(epsilon))*avg(alpha_S)*salt**2*avg(w)*dS# form for jump error indicator
    form3=(1.0/np.sqrt(epsilon))*alpha_S*rneuman**2*w*ds(0)#form for boundary error indicator
120    form_time=epsilon*(grad(c-c1))*2*w*dx()+beta*(c-c1)**2*w*dx()# the form for time error indicators
    form= form1+form2+form3#the form for global error estimates associated to space contributions
122    form_space=form1+form2+form3
    indicators1=assemble(form_space)
124    indicators=assemble(form)
    indicator1=assemble(form1)
126    indicator2=assemble(form2)
    indicator3=assemble(form3)# the form3 matrix of values
128    time_indicator=assemble(form_time)
    return indicators1 , indicators , indicator1 , indicator2 , indicator3 , time_indicator
130 #-----
tempoTotal0=0.0;tempoTotal1=0.0;contador=0; tol=1.0E-5;
132 from time import time
while t < t_stop+tol:
134     tempoTotal0=time()
    #---SOLVE THE TRANSIENT PROBLEM WITH AVAILABLE METHOD
136     rhs = assemble(L,exterior_facet_domains=boundary_parts,tensor=rhs)
    for bc in bcs:
138         bc.apply(A, rhs)
        (row,col,data) = A.data() # get sparse data
140        col = intc(col)
        row = intc(row)
142        n = A.size(0)

```

```

print 'the size of the linear system',n
144 Asp = csr_matrix( (data,col,row), shape=(n,n))
b = rhs.data()
146 c10=c1.vector()
c10=c10.array()
148
from time import time
150 tempo10=time()
max_levels=20
152 if contador%1==0:
    print contador
    config=solver_configuration(Asp,verb=True)
    config['B'] = np.ones((Asp.shape[0],1), dtype=Asp.dtype)
    config['BH'] = config['B'].copy()
    config['smooth'] =(None,{'krylov':'gmres'},{'iterations':2})
    config['strength'] = [( 'classical ',{'theta':0.10}), ( 'classical ',{'theta':0.10})]
    config['max_levels']=(max_levels)
    config['max_coarse']=(10000)
    config['coarse_solver']='gauss_seidel'
    config['symmetry']='symmetric'
    config['aggregate']='standard'
    config['presmoothing'] = ('block_gauss_seidel',{'sweep':'symmetric', 'iterations':2})
    config['postsmoothing'] = ('block_gauss_seidel',{'sweep':'symmetric', 'iterations':2})
    config['SA_solve_args']=( {'cycle':'V'},{'tol':1.0000000001e-12})
    config['improve_candidates']=[('block_gauss_seidel',{'sweep': 'symmetric', 'iterations':0}),None]
    config['diagonal_dominance']=(False)
    config['keep'] =(False)
    ml=solver(Asp,config)

    solut, itera, res= solve(Asp,b,x0=c10,tol=1.0000000001e-12,return_solver=False,maxiter=400,
existing_solver=ml)
else:
174 solut, itera, res = solve(Asp,b,x0=c10,tol=1.0000000001e-12,maxiter=400,return_solver=False,
existing_solver=ml)
from time import time
176 tempo11=time()
ravelizeResidualSA= np.ravel(b)-np.ravel(Asp*solut)
178 residuo=norm(ravelizeResidualSA)
print 'the residual', residuo
180
if residuo>0.0001:
182     print 'breaking the simulation'
    break
184
maxResidual=0.0;minResidual=0.0
186 c=Function(C)
RaResFunc=Function(C)
188
RaResFunc.vector()[:]=abs(ravelizeResidualSA)
190 c.vector()[:]=solut
print 'the residual from problem',residuo
192 if max(c.vector().array())>max(c1.vector().array()):
    print 'BREAKING BECAUSE METHOD INSERTING CONTAMINANT:BAD VALUE'
    break
194
if min(c.vector().array())<-0.00001:
196     print 'BREAKING BECAUSE BAD NEGATIVE VALUES IN CONCENTRATIONS'
    break

```

```

198 #---DEFINE RESIDUAL COMPONENTS FOR STATIONARY DIFUSION-REACTION PROBLEM
199 if contador==nsteps-1:
200     from Tools.ErrorEstimatorFunctions import FormLargeAdvection
201     ALarge,bLarge=FormLargeAdvection(epsilon,beta,c,c1,v,mesh=mesh)
202     (rowLarge,colLarge,dataLarge) = ALarge.data() # get sparse data
203     colLarge = intc(colLarge)
204     rowLarge = intc(rowLarge)
205     nLarge = ALarge.size(0)
206     print 'the size of the linear system',n
207     AspLarge = csr_matrix( (dataLarge,colLarge,rowLarge), shape=(nLarge,nLarge))
208     bLarge = bLarge.data()
209     ml = smoothed_aggregation_solver(AspLarge)
210     res1 = []
211     x = ml.solve(bLarge,accel='cg',tol=1e-12, residuals=res1)
212     ctil=Function(C)
213     ctil.vector()[:]=x
214     plot( ctil ,mesh=mesh,interactive=False,title='auxiliar problem')

215 #---COMPUTE ALL PARTS OF RESIDUAL ERROR ESTIMATOR:
216 #--ETA SPATIAL, DATA ESTIMATE,
217 #--TEMPORAL ESTIMATE FOR SMALL, ETA F, ESTIMATE FOR LARGE ADVECTION
218 #--INTERIOR, JUMP,BOUNDARY,DATA INTERIOR,DATA JUMP, FUNTION INTERIOR,
219 #--FUNCTION BOUNDARY, ENERGY SMALL, LARGE INTERIOR, LARGE JUMP.
220 #--DEFINE RESIDUAL ERROR COMPONENTS

221
222 from Tools.ErrorEstimatorFunctions import ErrorEstimatesSpaceTime
223 VField=V;fx=f#this is because difference is small: that is less than e-09. Could use \PiC0 ou f
224 function
225 n,gx,rk,gradot,salt,rneumann,rdirichlet=ErrorComponents(mesh,D,c,c1,VField,theta,Lambda,fx,dt)
226 print 'defining forms to residuals'
227 #---DEFINE ERROR FORMS(SMALL ADVECTION) FOR SPATIAL AND TIME WITHOUTH
228 FUNCTION, DATA AND
229 #--AUXILIAR ESTIMATES
230 indicators1, indicators, indicator1, indicator2, indicator3, time_indicator=\
231 FormErrorsDef(alpha_S,mesh,gx,rk,gradot,salt,rneumann,rdirichlet,epsilon,beta)

232 indicator_spatial =array([sqrt(i) for i in indicators]); print np.shape(indicator_spatial)
233 indicator_form1 = array([sqrt(i) for i in indicator1])
234 indicator_form2 = array([sqrt(i) for i in indicator2])
235 indicator_form3 = array([sqrt(i) for i in indicator3])
236 indicator_form4 = array([sqrt(i) for i in time_indicator])
237 #DEFINE ERROR ESTIMATES FOR TRANSIENT PROBLEM:
238 ErrorSpace,ErrorInterior,ErrorJump,ErrorBoundary,\
239 ErrorTime=ErrorEstimatesSpaceTime(indicators1,indicators,indicator1,indicator2,indicator3,
240 time_indicator,dt)

241 #-----ESTIMATIVAS DOS DADOS
242 from Tools.ErrorEstimatorFunctions import DataResidualComponents,FormDataOneNeumann,\
243 DataErrorEstimateStepN
244 from Tools.ErrorEstimatorFunctions import FormOneNeumannIndicator
245 #-----Required data for function DataResidualComponents
246 vx=vx_e;vy=vy_e;ReactionFunc=Lambda;raiseDegree=RaiseDegree;H=HeadExpression
247 #-----Residual error components
248 mesh2,DK,function,DiffGradot,DiffJump,DiffNeumann,\
249 DiffDirichlet =DataResidualComponents(C,alpha_S,epsilon,c1,c,dxx,dxy,dyy,HeadExpression,theta,
250 \vx,vy,ReactionFunc,mesh,degree,raiseDegree)
251 #-----Data residual matrix of residual components(indicator^2) and indicator
252 DataInteriorForm,DataJumpForm,DataBoundaryForm,DataGeralForm=\

```

```

252     FormDataOneNeumann(C,epsilon,degree,raiseDegree,beta,mesh2,DK,function,DiffGradot,DiffJump,
DiffNeumann,\
    DiffDirichlet )
254     #-----Indicadores dos dados
    DataInteriorIndicator,DataJumpIndicator,DataBoundaryIndicator,DataGeralIndicator=\
256     FormOneNeumannIndicator(epsilon,alpha__S,mesh2,DK,function,DiffGradot,DiffJump,DiffNeumann,
DiffDirichlet)
    #-----Data residual evaluation :interior, jump, boundary and general
258     DataInterior,DataJump,DataBoundary,DataGeral=\
    DataErrorEstimateStepN(DataInteriorForm,DataJumpForm,DataBoundaryForm,DataGeralForm,dt)
260     #---F FUNCTION ESTIMATES
    from Tools.ErrorEstimatorFunctions import FunctionFDataErrorEstimateStepN
262     Fexpression=fc
    DataFfunctionEstimate,DataFfunctionIndicator,\
264     DataFfunction=FunctionFDataErrorEstimateStepN(C,alpha__S,epsilon,theta,Fexpression,mesh,degree,
raiseDegree,dt)
    #-----LARGE ADVECTION ESTIMATES
266     from Tools.ErrorEstimatorFunctions import LargeAdvectionEstimates,
AuxiliarErrorEstimatesSpaceTime
    #---LARGE ADVECTION MATRIX OF ERROR COMPONENTS
268     print 'the solution for the linear system was calculated... calculations of the estimates'
    print ''
270     formLarge,form1Large,form2Large,form3Large=LargeAdvectionEstimates(alpha__S,mesh,epsilon,beta,c,
c1,ctil,v)

272     LargeGeralEstimate,LargeInteriorEstimate,LargeJumpEstimate,\
    LargeEnergyCtilEstimate=AuxiliarErrorEstimatesSpaceTime(formLarge,form1Large,form2Large,
form3Large,dt)
274     #---ERROR ESTIMATES FOR LARGE ADVECTION USING THE SAME CLASS USED FOR
#SPATIAL ERROR
276     LargeGeralEstimate,LargeInteriorEstimate,LargeJumpEstimate=AuxiliarErrorEstimatesSpaceTime(
formLarge,\
    form1Large,form2Large,form3Large,dt)
278     #--FUNCTION ON THE NEUMANN BOUNDARY: THERE IS NO ERROR
    #---GERAL ESTIMATES
280     #ESPACIAL GERAL: INICIAL+ESPACIAL+TEMPORAL+DADOS+ETAF+ETAAUXILIAR
    #LEMBRAR QUE EXISTE MULTIPLICACAO POR DT
282     #INDICADOR ESPACIAL E TEMPORAL: (ESPACIAL+DADOS)^(1/2) E
    #(TEMPORAL+ETAF+ETAAUXILIAR)^(1/2)
284     ErroGeral+=ErrorSpace+ErrorTime+DataGeral+DataFfunctionEstimate+LargeGeralEstimate
    RaizErroGeral=sqrt(EroGeral)#o estimador ate aquele passo de tempo
286     IndicadorEspacial=sqrt(ErrorSpace+DataGeral)
    IndicadorTemporal=sqrt(ErrorTime+DataFfunctionEstimate+LargeGeralEstimate)
288     ErrorSpace,ErrorInterior,ErrorJump,ErrorBoundary,ErrorTime, DataInterior,DataJump,DataBoundary
,DataGeral,
    DataFfunctionEstimate, LargeGeralEstimate,LargeInteriorEstimate,LargeJumpEstimate
290     #Adjustment for graphical analysis and plot as FEM
    class IndicForm(Expression):
292         def eval_cell( self , value, x, ufc_cell):
            value[0] = np.sqrt(indicators[ufc_cell.index]+DataGeralForm[ufc_cell.index])
294         IndicatorSpatial = IndicForm()
        IndicatorSpatial=interpolate(IndicatorSpatial,C)
296
    class IndicFormData(Expression):
298         def eval_cell( self , value, x, ufc_cell):
            value[0] = np.sqrt(DataGeralForm[ufc_cell.index])
300         IndicatorDataGeral = IndicFormData()

```



```

IndicatorDataGeral=interpolate(IndicatorDataGeral,C)
302 class IndicFormInteriorData(Expression):
    def eval_cell(self, value, x, ufc_cell):
304         #for cell in ufc_cell.index:
            value[0] = np.sqrt(DataInteriorForm[ufc_cell.index])
306 IndicatorDataInterior = IndicFormInteriorData()
IndicatorDataInterior=interpolate(IndicatorDataInterior,C)
308 print 'saving indicatorSpatialInside'
class IndicForm1(Expression):
310     def eval_cell(self, value, x, ufc_cell):
        value[0] = np.sqrt(indicator1[ufc_cell.index]+DataInteriorForm[ufc_cell.index])
312 IndicatorForm1 = IndicForm1()
IndicatorForm1=interpolate(IndicatorForm1,C)
314
class IndicForm2(Expression):
316     def eval_cell(self, value, x, ufc_cell):
        value[0] = np.sqrt(indicator2[ufc_cell.index]+DataJumpForm[ufc_cell.index])
318 IndicatorForm2 = IndicForm2()
IndicatorForm2=interpolate(IndicatorForm2,C)
320 class IndicForm3(Expression):
    def eval_cell(self, value, x, ufc_cell):
322         value[0] = np.sqrt(indicator3[ufc_cell.index]+DataBoundaryForm[ufc_cell.index])
IndicatorForm3 = IndicForm3()
324 IndicatorForm3=interpolate(IndicatorForm3,C)
class IndicFormTime(Expression):
326     def eval_cell(self, value, x, ufc_cell):
        value[0] = np.sqrt(time_indicator[ufc_cell.index]+formLarge[ufc_cell.index])
328 IndicatorForm4 = IndicFormTime()
IndicatorForm4=interpolate(IndicatorForm4,C)
330 contador+=1
#-----
332 print 'contador',contador
c1.assign(c)
334 print 'the maximum and minimum values',max(c.vector().array()), min(c1.vector().array())
if contador%10==0:
336     plot(c1,mesh=mesh,interactive=False,title='time %s'%contador)
    t+= dt
338     print 'time', t
print 'saving solution transport and auxiliary'
340 np.save('c1',c1.vector())
np.save('ctil',ctil.vector())
342

```