Henrique Gomes de Jesus

Precondicionador *multigrid* algébrico para métodos iterativos não estacionários na solução de sistemas lineares de grande porte

Brasil

2021

Henrique Gomes de Jesus

Precondicionador *multigrid* algébrico para métodos iterativos não estacionários na solução de sistemas lineares de grande porte

Dissertação de Mestrado.

Universidade Federal do Espírito Santo - UFES
Departamento de Informática
Programa de Pós-Graduação em Informática

Orientador: Dr.ª Lucia Catabriga
Coorientador: Dr.ª Maria Claudia Silva Boeres

Brasil
2021

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

de Jesus, Henrique Gomes, 1997-
D278p Precondicionador multigrid algébrico para métodos iterativos não estacionários na solução de sistemas lineares de grande porte / Henrique Gomes de Jesus. - 2021.
87 f. : il.

Orientadora: Lucia Catabriga.
Coorientadora: Maria Claudia Silva Boeres.
Dissertação (Mestrado em Informática) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Métodos Multigrid. 2. Multigrid Algébrico. 3. Double Pairwise Aggregation. 4. Precondicionadores. 5. Métodos Iterativos. I. Catabriga, Lucia. II. Boeres, Maria Claudia Silva. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 004

Henrique Gomes de Jesus

Precondicionador *multigrid* algébrico para métodos iterativos não estacionários na solução de sistemas lineares de grande porte

Dissertação de Mestrado.

Trabalho aprovado. Brasil, 5 de março de 2021:

Dr.ª Lucia Catabriga
Orientadora

Dr.ª Maria Claudia Silva Boeres
Coorientadora

Dr. Isaac Pinheiro dos Santos
Membro interno

Dr. Adriano Maurício de Almeida Côrtes
Membro externo

Brasil

2021

Aos meus pais.

Agradecimentos

Os agradecimentos principais são direcionados às minhas orientadoras por toda a atenção, empenho e paciência.

Aos professores avaliadores dessa dissertação, prof. Dr. Adriano Maurício de Almeida Côrtes e prof. Dr. Isaac Pinheiro dos Santos, obrigado por terem aceitado o convite e por se disponibilizarem a avaliar este trabalho.

Agradeço aos colegas de laboratório, o LabOtiM (Laboratório de Otimização e Modelagem Computacional), especialmente Ramoni Zancanela Sedano Azevedo, pelas dúvidas sanadas. Também agradeço ao laboratório pelo café que me manteve acordado.

Agradeço ao Núcleo Avançado de Computação de Alto Desempenho (NACAD) da COPPE/UFRJ pela disponibilização do supercomputador Lobo Carneiro (LoboC) e ao Laboratório de Computação de Alto Desempenho (LCAD) da UFES pela disponibilização de uma máquina que tanto acelerou o processo de execução dos experimentos numéricos.

Agradeço a Marcelo Torres Pereira Carrion pela disponibilização dos códigos do *Multigrid* e auxílio com o uso dos mesmos.

Agradeço a Leonardo Muniz de Lima, Laisa Karoline Muller e Riedson Baptista pela disponibilização de arquivos de código do FEMCodes e malhas essenciais para a realização dos experimentos.

Agradeço também aos desenvolvedores voluntários do projeto abnTEX2, que tanto facilitou a redação deste documento.

Agradecimentos especiais são direcionados aos meus pais, pelo apoio e incentivo, e à minha psicóloga, por me suportar.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

"Men fear thought as they fear nothing else on earth – more than ruin, more even than death. Thought is subversive and revolutionary, destructive and terrible, thought is merciless to privilege, established institutions, and comfortable habits; thought is anarchic and lawless, indifferent to authority, careless of the well-tried wisdom of the ages. Thought looks into the pit of hell and is not afraid. Thought is great and swift and free, the light of the world, and the chief glory of man."

(Bertrand Russell)

Resumo

O objetivo deste trabalho é avaliar o desempenho computacional do *Multigrid* Algébrico (AMG) como precondicionador de métodos baseados em subespaços de Krylov. Foi implementada uma estratégia de engrossamento alternativa conhecida por *Double Pairwise Aggregation* (DPA) que aplica um algoritmo de *matching* em grafos duas vezes em cada nível da hierarquia a fim de produzir os operadores de restrição e interpolação. Neste contexto, matrizes de origens diversas foram utilizadas para comparar as diferentes estratégias de engrossamento do AMG entre si e com precondicionadores derivados da fatoração LU incompleta (ILU) e da fatoração Gauss-Seidel aplicados ao Método do Resíduo Mínimo Generalizado (GMRES). Experimentos computacionais adicionais são realizados com matrizes tipo estêncil e com matrizes oriundas de problemas regidos pelas equações de Euler discretizadas pelo método dos Elementos Finitos, onde a aplicação de um algoritmo de reordenamento de linhas e colunas também é levado em consideração. Por fim, são apontadas vantagens e desvantagens em cada método e algoritmo de engrossamento em cada contexto, com ênfase nos avanços obtidos com a implementação do DPA.

Palavras-chave: Métodos *Multigrid*. *Multigrid* Algébrico. *Double Pairwise Aggregation*. Precondicionadores. Métodos Iterativos.

Abstract

The objective of this work is to evaluate the computational performance of Algebraic Multigrid (AMG) as a preconditioner for methods based on Krylov subspaces. An alternative coarsening strategy known as Double Pairwise Aggregation (DPA) has been implemented which applies a graph matching algorithm twice at each level of the hierarchy in order to produce the coarsening operators. In this context, matrices of different origins were used to compare the different AMG coarsening strategies with each other and with preconditioners derived from the incomplete LU factorization (ILU) and the Gauss-Seidel factorization applied to the Generalized Minimum Residual Method (GMRES). Additional computational experiments were performed with stencil matrices and with matrices originating from problems governed by Euler equations discretized by the Finite Element method, where a row and column reordering algorithm was also taken into account. Finally, the strengths and weaknesses of each method and coarsening algorithms are highlighted in each context, with emphasis on the advantages obtained with the implementation of DPA.

Keywords: Multigrid Methods. Algebraic Multigrid. Double Pairwise Aggregation. Preconditioners. Iterative Methods.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AMG	*Multigrid* Algébrico
GJ	Gauss-Jacobi
GS	Gauss-Seidel
GMG	*Multigrid* Geométrico
GMRES	Método do Resíduo Mínimo Generalizado
ILU	fatoração LU incompleta
DPA	*Double Pairwaise Aggregation*
DPA-G	AMG com engrossamento DPA e GMRES como relaxador
DPA-S	AMG com engrossamento DPA e SOR como relaxador
SOR	*Successive Over Relaxation*
Stb	AMG com esquemas clássicos de engrossamento

Sumário

1 Introdução

O processo de resolução numérica eficiente de sistemas lineares de grande porte se faz necessária na solução de equações diferenciais parciais discretizadas (STUBEN, 1999; YANG, 2006) em aplicações modernas onde, atualmente, não é incomum encontrar sistemas lineares com centenas de milhões ou mesmo bilhões de incógnitas (DAMBRA; FILIPPONE; VASSILEVSKI, 2018). A noção de *"grande"* é qualitativa e se expande em ordens de magnitude ao longo do tempo, e métodos eficientes e escaláveis para a solução de sistemas lineares de grande porte se tornam cada vez mais necessários.

Dentre os métodos de solução de sistemas lineares, a eliminação de Gauss, a eliminação de Gauss-Jordan, etc. compõem uma categoria de estratégias conhecidas como métodos diretos (SAAD, 2003), i.e., encontram a solução exata, exceto por erros de arredondamento, em um número finito de etapas de execução. Estes métodos são os mais robustos, ou seja, solucionam o maior número de sistemas lineares, pois são capazes de resolver sistemas definidos por qualquer matriz não-singular. Porém, para sistemas suficientemente grandes, seus tempos de execução os tornam impraticáveis, pois tais métodos possuem complexidade cúbica, em sua maioria.

Outra categoria de métodos de solução de sistemas lineares é composta por métodos como o Método do Resíduo Mínimo Generalizado (GMRES), o método dos Gradientes Conjugados (CG), a Sobrerelaxação Sucessiva (SOR), etc. Estes métodos são chamados de métodos iterativos (SAAD, 2003), e partem de uma solução arbitrariamente definida para se aproximar mais da solução exata a cada iteração, até que o erro se encontre dentro de uma tolerância especificada, em tempo mais hábil.

Dentre os métodos iterativos, há a sub-classe de métodos iterativos estacionários, ou *relaxações*, composta por métodos como Gauss-Jacobi, Gauss-Seidel, e o SOR. Tais métodos realizam uma sequência de iterações simples e podem evoluir para um processo de baixa qualidade. A sua efetividade, porém, pode ser melhorada através do emprego de uma estratégia conhecida como *Multigrid* (BRIGGS; HENSON; MCCORMICK, 2000).

Os métodos *Multigrid* aceleram a convergência de métodos iterativos estacionários transformando o problema em outro com menos variáveis e são divididos em duas categorias: *Multigrid* Geométrico (GMG) e *Multigrid* Algébrico (AMG).

Nos métodos GMG, o problema é redefinido em um domínio mais grosseiro, ou seja, uma discretização mais grosseira do domínio. Isso pode ser feito várias ve-

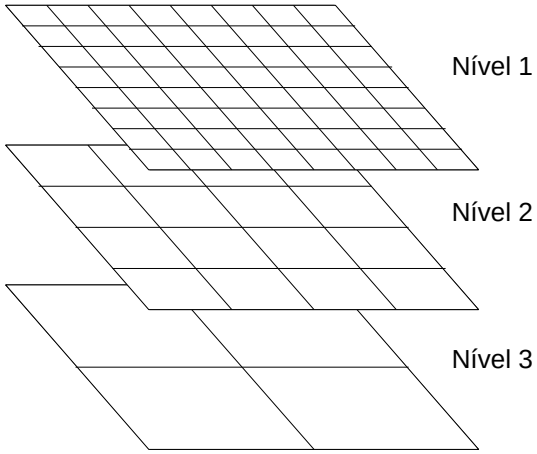


Figura 1 – Hierarquia de discretizações.

zes, assim como ilustrado na Fig. 1. Deste modo, o problema pode ser solucionado no domínio mais grosseiro, onde a execução do método de solução é mais barata, e a solução pode ser trazida de volta ao domínio mais refinado. Tais métodos se tornaram populares por, em tese, possuírem complexidade próxima da linear (BRIGGS; HENSON; MCCORMICK, 2000).

O AMG é uma generalização que busca solucionar sistemas lineares definidos por quaisquer matrizes, sem necessidade de que se conheça a geometria do problema *a priori*. Este método foi introduzido nos anos 80 (BRANDT; MCCORMICK; RUGE, 1982; BRANDT; MCCORUICK; HUGE, 1985; BRANDT, 1986; RUGE; STÜBEN, 1987), quando o princípio de Galerkin e as transferências entre grids baseadas em produtos matriz-vetor foram implementados no GMG com o objetivo de aumentar sua robustez (YANG, 2006). Na metade dos anos 90 o interesse pelo AMG se intensificou devido ao aumento da complexidade geométrica das aplicações e à necessidade de um sistema *Multigrid* comercial (STUBEN, 1999), o que limitou o escopo de aplicação do GMG. Desde então, diversas variantes foram desenvolvidas, como a *smoothed aggregation* (VANĚK; MANDEL; BREZINA, 1996; VAN et al., 2001), a *compatible relaxation* (BRANDT, 1999; STEINBACH; YANG, 2018) e a *double pairwise aggregation* (NOTAY, 2006), apenas para citar algumas. Há livros inteiramente dedicados ao tema (MCCORMICK, 1987; TROTTENBERG; OOSTERLEE; SCHULLER, 2000), incluindo um tutorial (BRIGGS; HENSON; MCCORMICK, 2000). Uma boa introdução ao tema pode ser encontrada em (STUBEN, 2000).

A fim de tornar possível que o *Multigrid* solucione sistemas definidos por matrizes gerais, no AMG é realizada uma fase de pré processamento chamada de *setup*, onde as matrizes de tamanho menor, ditas mais grosseiras, e os operadores de restrição e interpolação são construídos. Mais de um método existe para tal fim, cada um com suas particularidades. As estratégias mais clássicas, que incluem o engros-

samento padrão, o engrossamento agressivo e suas respectivas estratégias de interpolação, foram propostas por Stuben (2000), e se baseiam em aplicar heurísticas de agregação de vértices em grafos definidos pela matriz de coeficientes do sistema linear.

Uma importante aplicação dos métodos *Multigrid*, porém, se concentra no precondicionamento de métodos iterativos não estacionários, dentre os quais se encontram o método dos Gradientes Conjugados (GC), o método das Direções à Esquerda (LCD) e o método do Resíduo Mínimo Generalizado (GMRES), que soluciona sistemas lineares não simétricos aproximando a solução através do vetor de resíduo mínimo em um subespaço de Krylov (SAAD, 2003).

Embora os métodos não estacionários sejam mais robustos que os estacionários, isto é, atinjam a convergência com menos iterações, sua robustez pode ser ampliada por meio do uso de técnicas de precondicionamento. Tais técnicas consistem em modificar o sistema original, transformando-o num sistema equivalente, ou seja, que possui a mesma solução, mas que é mais fácil de resolver, ampliando a taxa de convergência e diminuindo o número de iterações.

Neste trabalho, investigamos o uso do método *Multigrid* como precondicionador do GMRES. Neste contexto são comparadas as estratégias clássicas propostas por (STUBEN, 2000) a um método mais recente conhecido por *Double Pairwise Aggregation* (NOTAY, 2006; NOTAY, 2010; STERCK et al., 2010; DAMBRA; FILIPPONE; VASSILEVSKI, 2018), que aplica duas vezes um algoritmo de *matching* em grafos para decidir quais variáveis criar nos domínios grosseiros e quais variáveis do domínio mais refinado serão respectivamente representadas.

Assim, pretendemos introduzir o método *Multigrid* Algébrico, seu uso como precondicionador, suas diferentes estratégias de *setup* e apresentar resultados experimentais comparativos na solução de matrizes gerais, matrizes estêncil e matrizes oriundas de discretizações do método dos elementos finitos.

Os capítulos deste trabalho são organizados como se segue. No Capítulo 2 são abordados os métodos iterativos estacionários, estabelecendo conceitos essenciais para a compreensão do *multigrid* e das técnicas de precondicionamento, e três destes métodos são detalhados em seções próprias. No Capítulo 3 é descrita a ideia geral do *multigrid* e em suas seções são detalhados o *multigrid* geométrico e o algébrico. No Capítulo 4 é descrita a técnica de precondicionamento, abordando o método GMRES precondicionado à esquerda e o uso do *multigrid* algébrico como precondicionador. No Capítulo 5 são exibidos e discutidos os resultados computacionais, comparando diferentes estratégias *multigrid* entre si e com outra opção popular de precondicionamento: o ILU. Enfim, o Capítulo 6 traz as considerações finais.

2 Métodos Iterativos Estacionários

Neste capítulo são apresentados os conceitos relativos aos métodos iterativos estacionários, que são procedimentos empregados na solução de sistemas lineares. É discutida, primeiramente, a formulação geral. Em seguida, são detalhados três métodos desta categoria: os métodos Gauss-Jacobi, Gauss-Seidel e o Successive Over Relaxation (SOR).

Os métodos iterativos solucionam cada sistema linear através da geração de uma sequência de soluções aproximadas que são aprimoradas conforme iterações são executadas. Tais iterações são usualmente chamadas de "relaxações". Em geral, os métodos iterativos são especialmente adequados quando a matriz de coeficientes é esparsa e se pretende evitar a produção drástica de novos coefientes não-nulos (conhecida por *fill-in*) dos métodos diretos.

Suponha um sistema linear do tipo $Ax = b$

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_2 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \\ a_{31}x_3 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n = b_3 \\ \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \ddots \quad\quad \vdots \quad\quad \vdots \\ a_{n1}x_n + a_{n2}x_n + a_{n3}x_n + \cdots + a_{nn}x_n = b_n \end{cases} \tag{2.1}$$

onde $A \in \mathbb{R}^{n\times n}$ é uma matriz quadrada de coeficientes e $b \in \mathbb{R}^n$ é o vetor de termos independentes. Como descrito em (SAAD, 2003), A pode ser escrita como a decomposição aditiva $A = M - N$ onde M e $N \in \mathbb{R}^{n\times n}$. Assim temos

$$\begin{aligned} (M - N)x &= b \\ Mx &= Nx + b \\ x &= M^{-1}Nx + M^{-1}b, \end{aligned}$$

que pode ser escrito como

$$x = Gx + c, \tag{2.2}$$

onde

$$G = M^{-1}N \tag{2.3}$$

$$c = M^{-1}b. \tag{2.4}$$

A Eq. (2.2) define a base dos métodos iterativos. A partir de uma aproximação inicial x^0 um conjunto de aproximações $x^1, x^2, \ldots, x^k$ são calculadas através de

iterações simples chamadas de "relaxações" do tipo

$$x^{k+1} = Gx^k + c, \tag{2.5}$$

onde G é chamada matriz de iteração e c é um vetor constante.

Quando G é fixa, isto é, não muda ao longo do processo, dizemos que o método iterativo é estacionário (FILHO, 2007). Deste modo, a Eq. (2.5) é chamada de expressão geral dos métodos iterativos estacionários (SAAD, 2003). A aplicação da expressão de recorrência (2.5) pode, ou não, levar à convergência para a solução desejada. O conceito de convergência é apresentado na Definição 1.

Definição 1. *Dadas uma sequência de vetores $x^k \in E$ e a norma $||\cdot||$ em E, onde E é um espaço vetorial e $k \in \mathbb{Z} \mid k \geq 0$, dizemos que a sequência x^k converge para $x \in E$ na norma $||\cdot||$ se $\left\|x^k - x\right\| \to 0$, quando $k \to \infty$ (FRANCO, 2006).*

Considerando-se $E = R^n$, as normas mais utilizadas para este propósito, de acordo com (BRIGGS; HENSON; MCCORMICK, 2000), são a norma do máximo – ou norma do infinito – e a norma euclidiana, definidas, respectivamente, pelas Equações (2.6).

$$\|v\|_\infty = \max_{i \leq j \leq n} |v_j| \quad \text{e} \quad \|v\|_2 = \left\{ \sum_{j=1}^{n} v_j^2 \right\}^{\frac{1}{2}}, \text{ para um dado vetor } v \in \mathbb{R}^n \tag{2.6}$$

Como não conhecemos a solução exata x a priori, o passo descrito na Eq. (2.5) pode ser aplicado até que a norma $\|r\|_\infty \in \mathbb{R}$ do resíduo

$$r = b - Ax^k \tag{2.7}$$

seja igual ou menor que uma tolerância ε predefinida, uma vez que o resíduo indica o quanto a solução atual está distante de satisfazer o problema original, ou seja, $Ax - b = 0$.

Deste modo, observando que $N = M - A$, cada escolha de M define um método iterativo estacionário novo. Assim, seguindo (SAAD, 2003) utilizando a decomposição aditiva $A = M - N$ na Eq. (2.3), podemos definir G como

$$\begin{aligned} G &= M^{-1}(M - A) \\ &= M^{-1}M - M^{-1}A \\ &= I - M^{-1}A. \end{aligned} \tag{2.8}$$

Partindo dos conceitos apresentados acima, serão descritos nas subseções seguintes três métodos iterativos estacionários de suma importância: o método de Gauss-Jacobi, o método de Gauss-Seidel, e o método de sobrerelaxação sucessiva (SOR).

2.1 Gauss-Jacobi

No método de Gauss-Jacobi (GJ), apresentado pelo matemático alemão Carl Gustav Jakob Jacobi, a cada iteração todas as componentes da solução x do sistema linear descrito na Eq. (2.1) podem ser relaxadas simultaneamente (LEE et al., 2017; BREZINSKI; WUYTACK, 2012), pois cada valor x_i é atualizado através apenas da i-ésima equação linear do sistema, utilizando as demais componentes com os valores oriundos da iteração anterior. Para tal, de acordo com (SAAD, 2003), é adotado $A = M_{GJ} - N_{GJ}$, sendo

$$M_{GJ} = D, \tag{2.9}$$

onde D é a diagonal principal da matriz A. Como $A = M - N$, temos que

$$N_{GJ} = -(L + U), \tag{2.10}$$

onde L e U representam, respectivamente, a parte triangular estritamente inferior e estritamente superior de A.

Considerando a definição de G nas Eqs. (2.3), (2.9) e (2.10) temos

$$\begin{aligned} G_{GJ} &= M_{GJ}^{-1} N_{GJ} \\ &= -D^{-1}(L + U) \end{aligned}$$

e a partir da definição de c nas Eqs. (2.4) , (2.9) e (2.10) temos

$$\begin{aligned} c_{GJ} &= M_{GJ}^{-1} b \\ &= D^{-1} b. \end{aligned}$$

Utilizando então a expressão geral dos métodos iterativos (2.5), o método de Gauss-Jacobi pode finalmente ser definido como

$$\begin{aligned} x^{k+1} &= G_{GJ} x^k + c_{GJ} \\ &= -D^{-1}(L + U)x^k + D^{-1} b, \end{aligned}$$

podendo ser computado através de

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

ou seja (RUGGIERO, 1996; FRANCO, 2006; FILHO, 2007),

$$
\begin{aligned}
x_1^{k+1} &= \frac{1}{a_{11}}(-a_{12}x_2^k - a_{13}x_3^k - \cdots - a_{1n}x_n^k + b_1),\\
x_2^{k+1} &= \frac{1}{a_{22}}(-a_{21}x_1^k - a_{23}x_3^k - \cdots - a_{2n}x_n^k + b_2),\\
x_3^{k+1} &= \frac{1}{a_{33}}(-a_{31}x_1^k - a_{32}x_2^k - \cdots - a_{3n}x_n^k + b_3),\\
&\vdots\\
x_n^{k+1} &= \frac{1}{a_{nn}}(-a_{n1}x_1^k - a_{n2}x_2^k - \cdots - a_{n,n-1}x_{n-1}^k + b_n)
\end{aligned}
$$

2.2 Gauss-Seidel

No método de Gauss-Seidel (GS), publicado primeiramente pelo matemático alemão Philipp Ludwig von Seidel, ao longo das iterações os novos valores das variáveis são utilizados sempre que possível. Trata-se, portanto, de um melhoramento do método de Gauss-Jacobi, vindo a se tornar o ponto inicial para o desenvolvimento dos métodos de sobrerrelaxação sucessiva, que dominaram a literatura na segunda metade do século XX (BREZINSKI; WUYTACK, 2012; MOHAMED, 2020).

Para tal, de acordo com (SAAD, 2003), adota-se $A = M_{GS} - N_{GS}$, sendo

$$M_{GS} = (L + D), \tag{2.12}$$

onde D é a diagonal principal e L é a parte triangular estritamente inferior da matriz A. Como $A = M - N$, temos que

$$N_{GS} = -U, \tag{2.13}$$

onde U representa estritamente a parte triangular estritamente superior de A.

Considerando a definição de G nas Eqs. (2.8), (2.12) e (2.13) temos

$$
\begin{aligned}
G_{GS} &= I - M_{GS}^{-1}A\\
&= I - (L + D)^{-1}A,
\end{aligned}
$$

e a partir da definição de c nas Eqs. (2.4), (2.12) e (2.13) temos

$$
\begin{aligned}
c_{GS} &= M^{-1}b\\
&= (L + D)^{-1}b.
\end{aligned}
$$

Utilizando então a expressão geral dos métodos iterativos (2.5), o método de Gauss-Seidel pode ser definido como

$$
\begin{aligned}
x^{k+1} &= G_{GS}x^k + c_{GS}\\
&= -(L + D)^{-1}Ux^k + (L + D)^{-1}b,
\end{aligned}
$$

podendo ser computado através de

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{k+1} - \sum_{j>i} a_{ij} x_j^k \right),$$

ou seja (RUGGIERO, 1996; FRANCO, 2006; FILHO, 2007),

$$\begin{aligned}
x_1^{k+1} &= \frac{1}{a_{11}}(-a_{12}x_2^k - a_{13}x_3^k - \cdots - a_{1n}x_n^k + b_1), \\
x_2^{k+1} &= \frac{1}{a_{22}}(-a_{21}x_1^{k+1} - a_{23}x_3^k - \cdots - a_{2n}x_n^k + b_2), \\
x_3^{k+1} &= \frac{1}{a_{33}}(-a_{31}x_1^{k+1} - a_{32}x_2^{k+1} - \cdots - a_{3n}x_n^k + b_3), \\
&\vdots \\
x_n^{k+1} &= \frac{1}{a_{nn}}(-a_{n1}x_1^{k+1} - a_{n2}x_2^{k+1} - \cdots - a_{n,n-1}x_{n-1}^{k+1} + b_n)
\end{aligned}$$

2.3 *Successive Over Relaxation*

O método de Sobrerrelaxação Sucessiva (SOR, do inglês) é uma variante do método Gauss-Seidel, proposta por David M. Young em sua tese de 1950, tornando-se um método fundamental na discussão introdutória a métodos iterativos estacionários (KINCAID, 2004).

No SOR, para cada componente x^{k+1} da solução x, é executada uma média ponderada entre x^k e a solução obtida pelo método Gauss-Seidel na iteração $k+1$, podendo ampliar ou reduzir a alteração do vetor solução em função do parâmetro ω:

$$x_i^{k+1} = \omega \bar{x}_i^{k+1} + (1-\omega)x_i^k,$$

onde $\bar{x}$ denota uma iteração do método Gauss-Seidel e $\omega \in (0,2)$ é um parâmetro de entrada.

Para tal M_{SOR} é definida como (SAAD, 2003)

$$M_{SOR} = \frac{1}{\omega}(D + \omega L), \tag{2.14}$$

onde D é a diagonal principal e L é a parte triangular estritamente inferior da matriz A. Segue-se que

$$N_{SOR} = -U + \left(\frac{1}{\omega} - 1 \right) D, \tag{2.15}$$

onde U representa a parte triangular estritamente superior de A.

Tendo em vista a definição de G expressa nas Eqs. (2.3), (2.14) e (2.15) temos

$$\begin{aligned}
G_{SOR} &= M_{SOR}^{-1} N_{SOR}, \\
&= -(D + \omega L)^{-1}[\omega U + (\omega - 1)D],
\end{aligned}$$

e tendo em vista a definição de c expressa nas Eqs. (2.4), (2.14) e (2.15) temos

$$\begin{aligned} c_{SOR} &= M^{-1}b, \\ &= \omega(D+\omega L)^{-1}b. \end{aligned}$$

Utilizando então a expressão geral dos métodos iterativos (2.5), o método SOR pode ser definido como

$$\begin{aligned} x^{k+1} &= G_{SOR}x^k + c_{SOR} \\ &= -(D+\omega L)^{-1}[\omega U + (\omega-1)D]x^k + \omega(D+\omega L)^{-1}b, \end{aligned}$$

podendo ser computado através de

$$x_i^{k+1} = (1-\omega)x_i^k + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}x_j^{k+1} - \sum_{j>i} a_{ij}x_j^k\right),$$

ou seja (RUGGIERO, 1996; FRANCO, 2006; FILHO, 2007),

$$\begin{aligned} x_1^{k+1} &= \frac{\omega}{a_{11}}(-a_{12}x_2^k - a_{13}x_3^k - \cdots - a_{1n}x_n^k + b_1) + (1-\omega)x_1^k, \\ x_2^{k+1} &= \frac{\omega}{a_{22}}(-a_{21}x_1^{k+1} - a_{23}x_3^k - \cdots - a_{2n}x_n^k + b_2) + (1-\omega)x_2^k, \\ x_3^{k+1} &= \frac{\omega}{a_{33}}(-a_{31}x_1^{k+1} - a_{32}x_2^{k+1} - \cdots - a_{3n}x_n^k + b_3) + (1-\omega)x_3^k, \\ &\vdots \\ x_n^{k+1} &= \frac{\omega}{a_{nn}}(-a_{n1}x_1^{k+1} - a_{n2}x_2^{k+1} - \cdots - a_{n,n-1}x_{n-1}^{k+1} + b_n) + (1-\omega)x_n^k \end{aligned}$$

3 Métodos Multigrid

Os métodos *Multigrid* aceleram a convergência dos métodos iterativos estacionários transformando o problema em outro com menos variáveis. Quando isto é feito, dizemos que o problema foi discretizado em um domínio mais grosseiro, o que pode ser feito várias vezes. Neste capítulo, são apresentados a formulação geral do *Multigrid* e seus subprocedimentos. Em sequência, são abordadas as variações do *Multigrid*, onde os procedimentos internos são detalhados.

Todo vetor pode ser decomposto em uma combinação de ondas sinusoidais de diferentes frequências e deslocamentos de fase (BROUGHTON; BRYAN, 2009), tal como representado na Eq. (3.1)

$$e = \sum_{k=1}^{n} c_k E_k, \tag{3.1}$$

onde n é o tamanho da base do espaço vetorial, $c_k \in \mathbb{R}$ e E_k é uma onda sinusoidal. Um caso é ilustrado na Fig. 2.

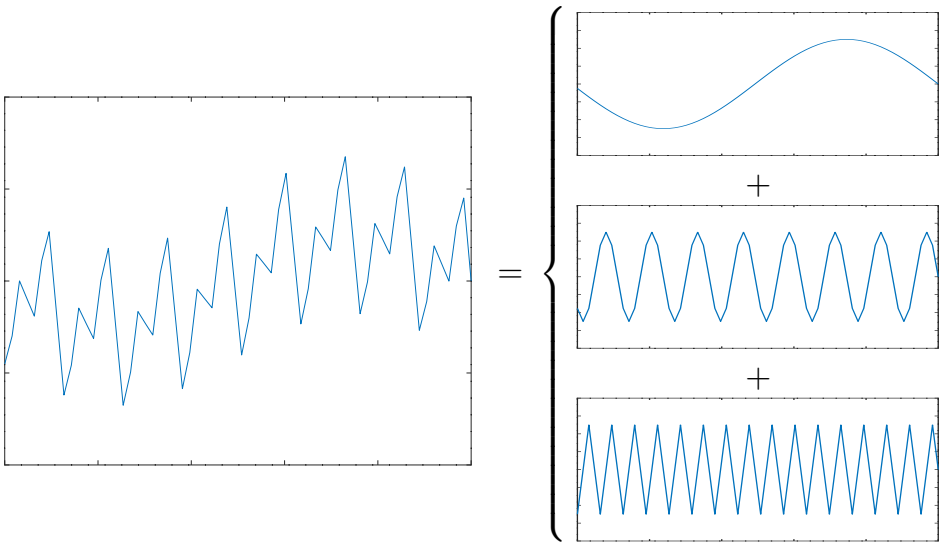
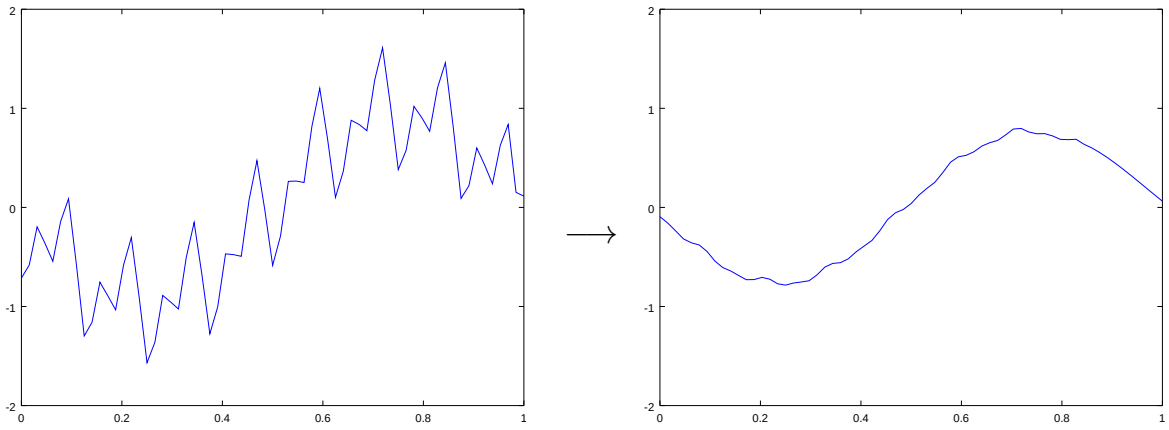


Figura 2 – Decomposição de um vetor para o domínio das frequências.

O método *Multigrid* foi desenvolvido após a descoberta de que os métodos iterativos estacionários são menos eficientes na remoção das componentes de baixa frequência do que na remoção das componentes de alta frequência do vetor erro (CARRION, 2016; BRIGGS; HENSON; MCCORMICK, 2000; YANG, 2006)

$$e = x - \bar{x}, \tag{3.2}$$

onde x representa a solução exata e $\bar{x}$ uma solução aproximada qualquer. Assim, a aplicação dos métodos iterativos estacionários tem como efeito a suavização do vetor erro, tal como ilustrado na Fig. 3.



(a) (b)

Figura 3 – (a) Erro da solução após uma relaxação (b) Erro da solução após dez relaxações. O eixo vertical representa o erro e o eixo horizontal o domínio – Reproduzido de Carrion (2016).

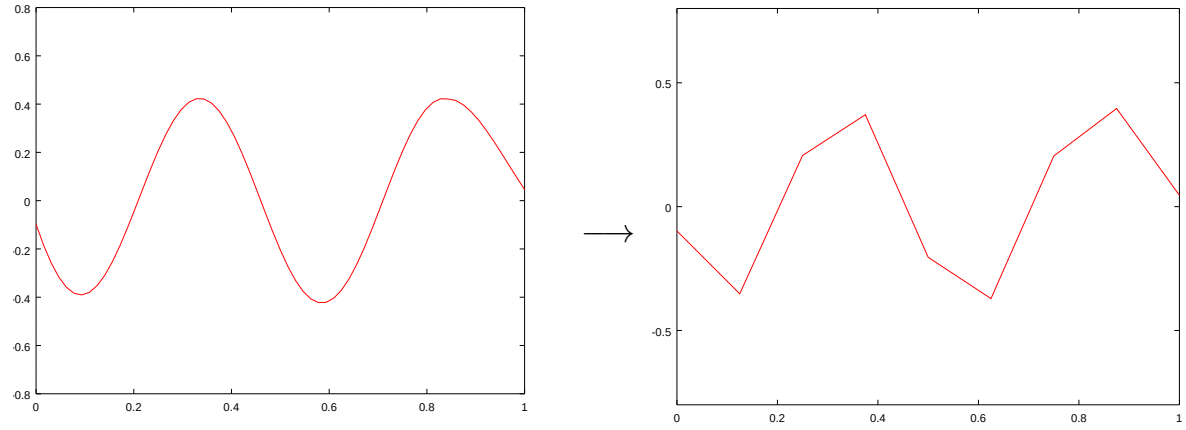
A eficácia dos métodos iterativos estacionários, porém, pode ser significativamente melhorada empregando a estratégia *Multigrid*, que corrige este problema transferindo o sistema para uma discretização mais grosseira antes de o solucionar.

Na discretização grosseira as oscilações de baixa frequência do vetor erro se tornam oscilações de frequência mais alta, tal como ilustrado na Fig. 4, onde a distância maior entre pontos da discretização leva a uma variação maior entre componentes adjacentes do vetor erro. Assim nos é permitido tomar vantagem do fato citado anteriormente de que o método iterativo estacionário é mais eficiente na remoção de componentes de alta frequência (CARRION, 2016; BRIGGS; HENSON; MCCORMICK, 2000).

Após solucionar o problema no domínio grosseiro a solução do sistema é transferida de volta ao domínio original.

Suponha um sistema linear $Ax = b$ resultante de uma discretização através de um método tal como o método das diferenças finitas, onde cada componente do vetor de incógnitas x representa um ponto no domínio do problema, com espaçamento h entre os pontos. Dizemos portanto que este problema pertence ao domínio Ω^h. O domínio ao qual pertence uma variável será indicado através dos sobrescritos $h, 2h, \ldots, H$. O método *Multigrid* representa o sistema

$$A^h x^h = b^h$$



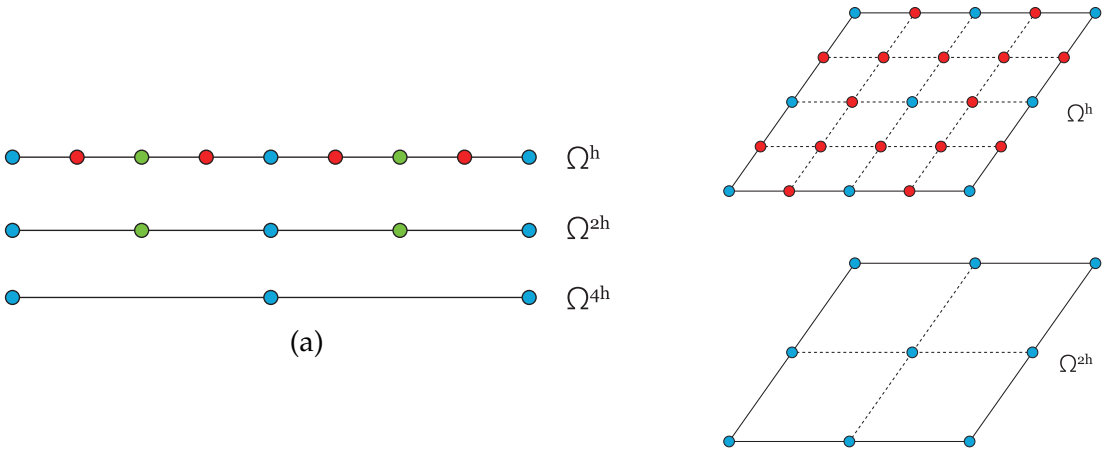
(a) (b)

Figura 4 – (a) Erro no domínio original (b) Erro no domínio grosseiro – Reproduzida de Carrion (2016)

em um domínio mais grosseiro

$$A^{2h}x^{2h} = b^{2h},$$

onde o espaçamento entre os pontos representados é maior, conforme exemplificado na Fig. 5, onde uma malha $1D$ e uma malha $2D$ no domínio Ω^h são engrossados no domínio Ω^{2h}.



(a) (b)

Figura 5 – Espaçamento entre pontos em diferentes domínios de um mesmo problema (a) unidimensional e (b) bidimensional – Reproduzida de Carrion (2016).

No domínio grosseiro, além de serem mais eficientes pelos motivos supracitados, as relaxações são computacionalmente mais leves, visto que o número de incógnitas é menor. A solução encontrada no domínio grosseiro é utilizada para corrigir a solução mais refinada através da correção por erro, que será descrita mais adiante.

Assim, o método *Multigrid* consiste em:

1. Aplicar relaxações no domínio refinado Ω^h;
2. Transferir o problema para o domínio grosseiro Ω^{2h};
3. Resolver o problema em Ω^{2h};
4. Transferir a solução de volta para o domínio refinado Ω^h;
5. Utilizar a solução trazida de Ω^{2h} para corrigir a solução de Ω^h;
6. Aplicar novas relaxações no domínio refinado Ω^h.

A transferência de um vetor de um domínio mais refinado para um mais grosseiro é chamada de *restrição* (passos 2), e a transferência contrária recebe o nome de *interpolação* (passo 4). Estes processos serão descritos em maiores detalhes a seguir.

Restrição e interpolação A transferência de um vetor do domínio mais refinado para o domínio mais grosseiro é chamada de restrição. Em termos algébricos, a operação de restrição pode ser definida por

$$v^{2h} = R^h v^h,$$

onde R^h denota o operador de restrição. A ordem de R^h é $N \times N_c$, onde N é o número de nós em Ω^h e $N_c < N$ é o número de nós em Ω^{2h}. Cada coeficiente r_{ij}^h de R^h contém o peso do valor v_j^h na determinação de v_i^{2h} durante a restrição para Ω^{2h}.

A interpolação, também chamada de prolongação, é a operação inversa da restrição, isto é, a transferência de um vetor do domínio mais grosseiro para o domínio mais refinado. Em termos algébricos, a operação de interpolação pode ser definida por

$$v^h = P^h v^{2h},$$

onde P^h denota o operador de interpolação. A ordem de P^h é $N_c \times N$, onde N é o número de nós em Ω^h e $N_c < N$ é o número de nós em Ω^{2h}. Cada coeficiente p_{ij}^h de P^h contém a contribuição do valor v_j^{2h} na determinação de v_i^h durante a interpolação de volta para Ω^h.

Correção por erro Quando a solução grosseira é interpolada de volta ao domínio original ela é utilizada para corrigir a solução obtida pelas relaxações outrora aplicadas neste domínio. Esta correção é chamada de *correção por erro*.

Dado o erro e definido pela Eq. (3.2), podemos corrigir a solução aproximada $\bar{x}$ e obter a solução exata x através de

$$x = e + \bar{x}.$$

Infelizmente, a obtenção do vetor erro é tão difícil quanto solucionar o próprio sistema original. Porém, utilizando a identidade (2.1) na expressão (2.7)

$$\begin{aligned} r &= \overbrace{b}^{Ax=b} - A\bar{x} \\ &= Ax - A\bar{x} \end{aligned}$$

e observando-se a definição de erro na Eq. (3.2)

$$r = A(\overbrace{x - \bar{x}}^{e=x-\bar{x}}),$$

a obtenção do vetor erro é alcançada através da importante equação residual

$$Ae = r.$$

Deste modo, o procedimento para a eliminação das componentes de baixa frequência do vetor erro através de discretizações grosseiras consiste em:

1. Restringir o vetor resíduo $r^{2h} \leftarrow R^h r^h$ $\Omega^{2h} \leftarrow \Omega^h$;
2. Solucionar o sistema $A^{2h}e^{2h} = r^{2h}$ Ω^{2h};
3. Interpolar o vetor erro $e^h \leftarrow P^h e^{2h}$ $\Omega^h \leftarrow \Omega^{2h}$;
4. Obter a solução corrigida pelo erro através da soma $x^h \leftarrow \bar{x}^h + e^h$ Ω^h,

onde o domínio, em cada passo, é indicado à direita.

Hierarquia de Discretizações O *Multigrid* pode utilizar várias camadas de discretizações no lugar de apenas duas. Este conjunto de discretizações é chamado de *hierarquia de discretizações*.

Uma vez no domínio Ω^{2h} precisamos solucionar um novo sistema linear no qual as relaxações serão ineficientes na remoção das componentes de baixa frequência do vetor erro. Um modo de lidar com isso é aplicar a estratégia *Multigrid* novamente para resolver o sistema grosseiro, ou seja, utilizá-lo recursivamente.

Ao restringir novamente o sistema em Ω^{2h}, obteremos o sistema em Ω^{4h} e executaremos iterações da relaxação na equação residual de Ω^{2h} que foi restringida para o domínio Ω^{4h}. Este processo é executado recursivamente em cada nível, gerando assim a hierarquia de discretizações, onde a profundidade é um valor parametrizado. Assim, o método "desce" até a discretização mais grosseira e retorna até o sistema original corrigido. Tal esquema é chamado de ciclo V e está ilustrado na Fig. 6.

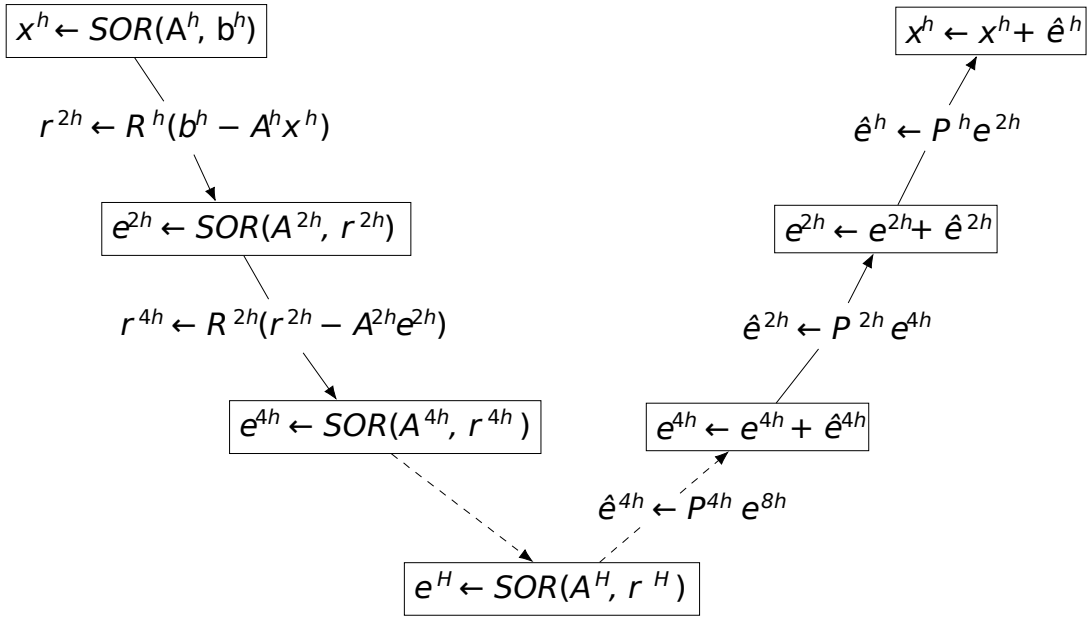


Figura 6 – Ciclo-V utilizando SOR. Adaptado de (CARRION, 2016).

Outros esquemas existem, tais como o ciclo W e o ciclo K (BRIGGS; HENSON; MCCORMICK, 2000), mas estes não serão discutidos neste trabalho.

O método *Multigrid* possui como característica central a construção de uma hierarquia de discretizações do problema original. O número de nível da hierarquia é um parâmetro de entrada que será chamado aqui de NCL.

Algoritmo 1: Ciclo V (NCL, nr, ω, A, x, b)

Dados: Número de níveis NCL; Número de relaxações por nível nr; Parâmetro ω do SOR; Matriz A; Solução inicial x; Vetor de termos independentes b
Resultado: Vetor solução x

```
para i de 1 a nr faça
    Aplicar relaxações SOR com parâmetro ω em Ax = b
fim
se NCL > 0 então
    Calcular resíduo r
    Restringir r
    Ciclo_V(NCL − 1, nr, ω, A, e, r)
    Interpolar e
    x ← x + e
fim
para i de 1 a nr faça
    Aplicar relaxações SOR com parâmetro ω em Ax = b
fim
```

Em cada nível do ciclo V o método *Multigrid* realiza a suavização, que é uma

sequência de relaxações. Em seguida, caso não se tenha atingido a profundidade máxima NCL do ciclo V, uma nova descida é realizada na hierarquia de discretizações, o que pode ser feito aplicando-se o método recursivamente. Por fim, é realizada uma nova suavização. O algoritmo 1 apresenta o pseudocódigo do ciclo V.

As etapas a seguir são essenciais para cada ciclo do método *Multigrid*:

- Suavização - Aplicação de um número parametrizado nr de relaxações no sistema mais refinado $A^h x^h = b^h$, eliminando as componentes de alta frequência do vetor erro $e^h = x^h - \overline{x}^h$ (linhas 1-3).
- Cálculo do resíduo - Calcula-se o resíduo $r^h \leftarrow b^h - A^h x^h$, que indica o quanto a solução atual se adequa ao problema original (linha 5).
- Restrição - Transferência do vetor resíduo do domínio mais refinado para o domínio mais grosseiro $r^{2h} \leftarrow R^h r^h$ (linha 6).
- Cálculo do erro - Solução do sistema mais grosseiro $A^{2h} e^{2h} = r^{2h}$, eliminando as componentes de baixa frequência do vetor erro original. Isto pode ser feito pela aplicação do Ciclo V, através de $2nr$ relaxações ou por um método exato, se for adequado para o tamanho do problema (linha 7).
- Interpolação - Transferência do vetor erro do domínio mais grosseiro para um domínio mais refinado $e^h \leftarrow P^h e^{2h}$ (linha 8).
- Correção por erro - Correção da solução atual utilizando o vetor erro recém interpolado do domínio mais grosseiro $x^h \leftarrow x^h + e^h$ (linha 9).
- Suavização - Aplicação, novamente, de nr relaxações no sistema mais refinado $A^h x^h = b^h$ (linhas 11-13).

O método *Multigrid* executa uma sequência de ciclos V até que a solução atinja uma tolerância ε especificada.

Tipos de *Multigrid* Antes de detalhar os métodos de restrição e interpolação, é necessário observar que os métodos *Multigrid* podem ser divididos em duas categorias bem distintas:

- *Multigrid* Geométrico (GMG, do inglês), que se baseia no pressuposto de que o sistema é oriundo de uma discretização em malha, o que restringe o escopo de sua aplicação, apesar da eficiência, visto que informações do problema são necessárias em componentes do *Multigrid*. Seu uso, via de regra, é empregado em malhas estruturadas. Não é facilmente utilizável como um solucionador

de sistemas lineares gerais, ou seja, como popularmente denominado, um sistema caixa-preta (BRIGGS; HENSON; MCCORMICK, 2000; STUBEN, 2000; FALGOUT, 2006; CARRION, 2016).

- *Multigrid* Algébrico (AMG, do inglês), que é uma generalização do GMG onde o conjunto de variáveis das malhas mais grosseiras é definido de forma exclusivamente algébrica através da força das conexões entre as variáveis refletida nas entradas da matriz do sistema em cada nível, em um processo conhecido como *setup*, permitindo o uso do método *Multigrid* para matrizes genéricas, e portanto, como um sistema caixa-preta (BRIGGS; HENSON; MCCORMICK, 2000; STUBEN, 2000; FALGOUT, 2006; CARRION, 2016; YANG, 2006).

Ambos serão discutidos em maior profundidade nas Seções 3.1 e 3.2, onde diferentes métodos de restrição e interpolação serão detalhados.

O AMG é considerado um dos métodos mais rápidos para várias classes de equações (SAAD, 2003). As técnicas *Multigrid* tornaram-se bastante populares porque, em teoria, têm complexidade computacional linear, além de excelente escalabilidade e possibilidade de paralelização. Como consequência, têm sido empregadas em sistemas de grande porte (PARK et al., 2015).

3.1 *Multigrid* Geométrico

Os métodos *Multigrid* Geométricos (GMG) emergiram nos anos 60 (FEDORENKO, 1962). Nesta estratégia, pressupõe-se que o sistema é oriundo de uma discretização em malha, de modo que se possa inferir diretamente os vizinhos de cada nó representado por uma variável. Deste modo, não é necessária a criação de novas matrizes, tornando o método mais rápido, porém, restrito a menos aplicações e inviável ao uso como um método caixa preta. A seguir apresentamos em detalhes cada operação do GMG.

Restrição

A restrição é a operação para produzir um vetor em Ω^{2h} análogo a um vetor em Ω^{h}, podendo ser definida de dois modos:

- por injeção, onde o vetor é preenchido com os valores diretamente correspondentes no vetor original, como ilustrado na Fig. 7;
- ponderação total, onde o vetor é preenchido com uma média ponderada entre os valores diretamente correspondentes e seus vizinhos, como ilustrado na Fig. 8.

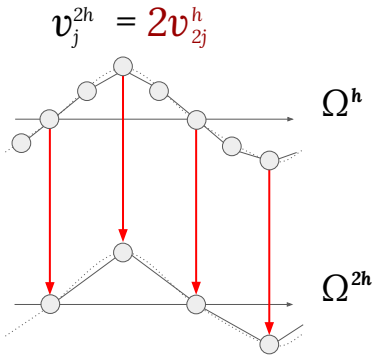


Figura 7 – Restrição por injeção no caso 1D. Cada ponto em Ω^h representa um componente do vetor v^h, cada ponto em Ω^{2h} representa um componente do vetor v^{2h} e cada seta representa uma atribuição.

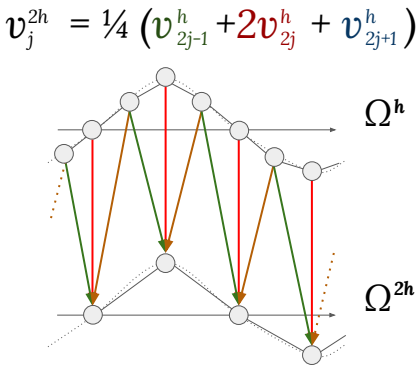


Figura 8 – Restrição por ponderação total no caso 1D. Cada ponto em Ω^h representa um componente do vetor v^h. Um ponto em Ω^{2h} representa um componente do vetor v^{2h}. Uma seta representa uma atribuição.

A matriz R^h, que define o operador de restrição, é preenchida de modo que cada coeficiente r_{ij}^h possua o peso do valor v_j^h na determinação de v_i^{2h}, o que deriva naturalmente da geometria da malha. Por exemplo, suponha o caso 2D ilustrado na Fig. 9. A restrição do elemento central (apenas) se converteria na matriz R^h exibida na Eq. (3.3). O número de linhas é igual ao tamanho do vetor mais grosseiro (número de pontos azuis na Fig. 9), o número de colunas é igual ao tamanho do vetor mais refinado (número total de pontos na Fig. 9) e cada coeficiente r_{ij} é igual ao peso do valor v_j^h na determinação do valor v_i^{2h} dividido pela soma de todos os pesos da mesma

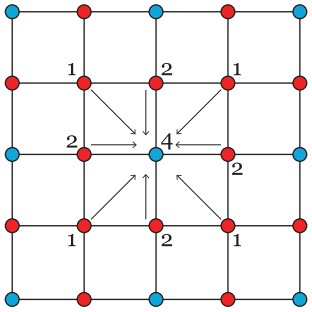


Figura 9 – Restrição por ponderação total no caso 2D – Reproduzida de Carrion (2016). Cada ponto representa um componente do vetor v^h. Um ponto azul representa um componente que também existe no vetor v^{2h}. Uma seta representa uma atribuição. Um número representa o peso de um valor na ponderação total.

ponderação (dividendos da mesma linha).

$$R^h = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1/16 & 2/16 & 1/16 & 0 & 0 & 2/16 & 4/16 & 2/16 & 0 & 0 & 1/16 & 2/16 & 1/16 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \tag{3.3}$$

Interpolação

Outra importante operação do método *Multigrid* é a interpolação, utilizada para produzir um vetor em Ω^h associado a um vetor em Ω^{2h}. O mais tradicional e mais efetivo é o operador de interpolação linear, onde o vetor em Ω^h é preenchido com os valores diretamente correspondentes em Ω^{2h} quando estes existirem, ou com a média dos valores vizinhos em caso contrário, como ilustrado na Fig. 10.

A matriz P^h, que define o operador de interpolação, é preenchida de modo que cada coeficiente p^h_{ij} possua o peso do valor p^{2h}_j na determinação de p^h_i, o que deriva naturalmente da geometria da malha.

Por exemplo, suponha o caso 2D ilustrado na Fig. 11. A interpolação do elemento em verde (apenas) se converteria na matriz P^h exibida na Eq. (3.4). O número de linhas é igual ao tamanho do vetor mais refinado (número total de pontos na Fig. 11), o número de colunas é igual ao tamanho do vetor mais grosseiro (número de pontos vermelhos na Fig. 11) e cada coeficiente p_{ij} é igual ao peso do valor v^{2h}_j na determinação do valor v^h_i dividido pela soma de todos os pesos da mesma ponderação

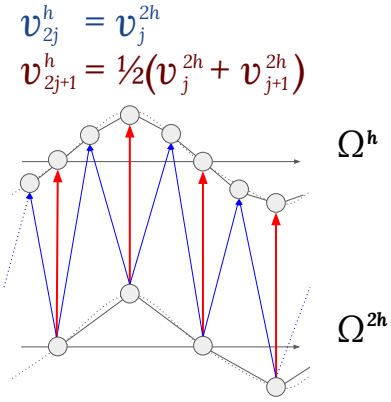


Figura 10 – Interpolação linear no caso 1D. Cada ponto em Ω^h representa um componente do vetor v^h, Um ponto em Ω^{2h} representa um componente do vetor v^{2h}. Uma seta representa uma atribuição.

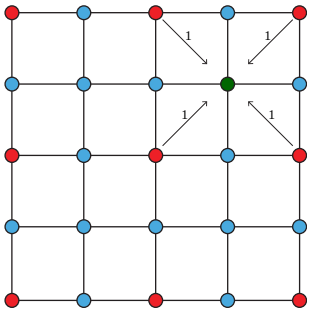


Figura 11 – Interpolação linear no caso 2D – Reproduzido de Carrion (2016). Cada ponto representa um componente do vetor v^h. Um ponto vermelho representa um componente que também existe no vetor v^{2h}. Uma seta representa uma atribuição e cada número representa o peso de um valor na ponderação total.

(dividendos da mesma linha).

$$P^h = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1/4 & 1/4 & 0 & 1/4 & 1/4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \tag{3.4}$$

3.2 *Multigrid* Algébrico

O *Multigrid* Algébrico (AMG) pode ser considerado uma generalização do *Multigrid* Geométrico, estendendo a aplicação do *Multigrid* para sistemas lineares em geral. O AMG gera operadores de restrição e interpolação baseado apenas nas informações contidas na matriz de uma forma puramente algébrica, de modo que nenhuma informação sobre o problema físico original e a discretização do seu domínio seja necessária. Assim, o algoritmo não é construído para lidar com um tipo de malha específico e nem está restrito aos problemas oriundos de discretizações em malha. Deste modo, o *Multigrid* passa a poder ser usado como um sistema caixa preta (BRANDT; MCCORUICK; HUGE, 1985).

Neste método se faz necessária uma fase de preprocessamento chamada de *setup*, onde as matrizes associadas aos problemas grosseiros serão construídas, uma vez que, ao contrário do *Multigrid* Geométrico, no AMG a geometria do problema não é conhecida e as matrizes mais grosseiras não podem ser definidas apenas implicitamente, visto que as operações de restrição e interpolação do *Multigrid* Geométrico se baseiam na geometria do problema. Para tal, é necessário um esquema de engrossamento, onde o conjunto de variáveis mais grosseiras é definido, bem como suas respectivas variáveis representadas da matriz fina, a fim de construir os operadores de restrição e interpolação P^h e R^h, respectivamente, e criar as discretizações grosseiras (STÜBEN, 2001b).

A partir de então, uma matriz mais grosseira é dada, em termos algébricos, pelo operador de Galerkin

$$A^{2h} = R^h A^h P^h, \tag{3.5}$$

onde R^h e P^h são, respectivamente, os operadores de restrição e interpolação, sendo que

$$R^h = (P^h)^T.$$

P também é geralmente referenciada na literatura como matriz de prolongação (HEMKER, 1982).

Deste modo, é necessário definir dois componentes do método AMG a fim de criarmos uma hierarquia de discretizações[1]:

- um esquema de engrossamento, para determinar o conjunto de variáveis do sistema mais grosseiro;

[1] Usaremos o termo *hierarquia de discretizações* no AMG para designar o engrossamento e refinamento dos sistemas lineares no processo *Multigrid*.

- um operador de interpolação, para mapear os vetores dos domínios mais grosseiros nos vetores dos domínios mais refinados.

O método *Multigrid* Algébrico é composto de duas fases: a fase de *setup*, onde os operadores de restrição e interpolação são definidos, bem como a hierarquia de discretizações, e a fase de execução, onde é realizado o ciclo V.

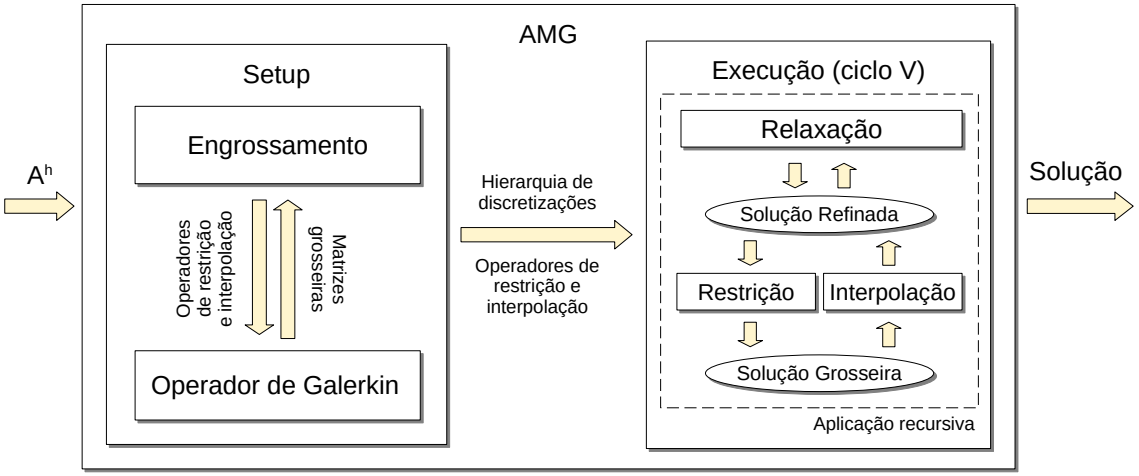


Figura 12 – Esquema do AMG.

O processo é ilustrado na Fig. 12: no *setup* o processo de engrossamento produz os operadores de restrição e interpolação R^h e P^h a partir de uma matriz mais refinada A^h. O operador de Galerkin produz a matriz mais grosseira a partir destes operadores.

Este conjunto de matrizes, que é a própria hierarquia de discretizações, juntamente com os operadores de restrição de interpolação são utilizados na fase de execução, onde ocorre o ciclo V. Nele, a relaxação é executada na matriz mais refinada e os operadores são utilizados na transferência entre níveis para a invocação recursiva do método.

Estes processos serão detalhados nas subseções a seguir.

Setup

Antes do início do ciclo-V, chamado de fase de execução, é necessário que ocorra uma fase de pré-processamento, chamado de *setup*. Nela são geradas as matrizes mais grosseiras. Para tal, é utilizado um esquema de engrossamento, um esquema de interpolação e uma operação de Galerkin. O esquema de engrossamento define o conjunto de variáveis do domínio mais grosseiro, o esquema de interpolação define os operadores R e P, e a operação de Galerkin descrita na Eq. (3.5) gera a matriz grosseira, isto é, a matriz associada ao domínio grosseiro.

Um esquema de engrossamento adequado deve oferecer um balanço entre uma boa taxa de redução do tamanho do problema e uma boa representação do domínio mais refinado pelo domínio mais grosseiro, uma vez que a redução do tamanho do problema influencia no custo do *setup* e no custo do ciclo-V e a qualidade da representação influencia na taxa de convergência.

Uma boa redução do tamanho do problema significa Ω^{2h} ser consideravelmente menor que Ω^h, enquanto a qualidade da representação é definida por quão fortes são as conexões entre as variáveis do domínio mais refinado representadas por uma mesma variável do domínio mais grosseiro e quão fracas são as conexões entre as variáveis do domínio mais refinado representadas por variáveis diferentes do domínio mais grosseiro. As definições algébricas de força de conexão entre variáveis variam em função do esquema de engrossamento.

Esquemas clássicos de engrossamento

Os esquemas de engrossamento particionam o conjunto Ω de variáveis do problema em dois subconjuntos disjuntos C e F, onde as variáveis em C terão variáveis diretamente correspondentes no domínio mais grosseiro e as em F, chamadas de variáveis complementares, serão representadas pelas variáveis em C.

No processo de construção de C e F duas variáveis x_i e x_j são ditas diretamente conectadas se

$$a_{ij} \neq 0,$$

onde a_{ij} é o coeficiente da matriz A presente na linha i e na coluna j. Deste modo, a vizinhança de uma variável x_i será dada pelo conjunto

$$N_i = \{j | j \neq i, a_{ij} \neq 0\}.$$

Adicionalmente,

$$C_i = C \cap N_i, \quad F_i = F \cap N_i$$
$$N_i^- = \{j \in N_i : a_{ij}^h < 0\} \quad N_i^+ = \{j \in N_i : a_{ij}^h > 0\}.$$

É esperado que as variáveis em C tenham um número mínimo de conexões fortes com variáveis em F, de modo a garantir que o domínio mais grosseiro represente bem o domínio mais refinado. Para tal, uma conexão entre duas variáveis x_i e x_j é definida como negativamente forte se

$$-a_{ij} \geq \varepsilon \max_{k \neq i}(|a_{ik}|),$$

onde conexões positivas acabam por ser ignoradas, visto que em muitas aplicações a maioria das ligações fortes são negativas (STUBEN, 2000).

Engrossamentos clássicos	**Interpolações adequadas**
Padrão	Direta Padrão
Agressivo	Multi-passo
Por conexões fortes positivas	Estendida+i

Tabela 1 – Esquemas clássicos de engrossamento e suas respectivas formas interpolação.

Analogamente a N_i, a vizinhança fortemente conectada a uma variável x_i é representada pelo conjunto mais restrito

$$S_i = \{j | i \text{ é fortemente conectado a } j\}.$$

Adicionalmente,

$$P_i = C_i^S = C \cap S_i, \quad F_i^S = F \cap S_i.$$

Uma vez que o domínio Ω^{2h} possui as variáveis em C^h mas não aquelas em F^h, e uma vez que

$$e^h = P^h e^{2h},$$

toda matriz de interpolação $P = R^T$ deve ser definida de modo que

$$(Pe)_i = \begin{cases} e_i & \text{se } i \in C, \\ \sum_{j \in C_i} w_{ij} e_j & \text{se } i \in F, \end{cases} \tag{3.6}$$

onde w_{ij} são os pesos de interpolação utilizados para estimar as variáveis em F através de uma combinação linear dos valores em C e devem ser gerados em função do esquema de engrossamento. Os esquemas de interpolação serão apresentados adiante, junto de seus respectivos esquemas de engrossamento.

Esquemas clássicos de engrossamento

Os esquemas de engrossamento mais antigos foram definidos por Ruge e Stüben (1987). Tratam-se de três opções de estratégias com aplicações distintas, cada uma com sua(s) próprias(s) estratégias de interpolação mais apropriadas, das quais podem ser visualizadas na Tab. 1 aquelas que serão discutidas neste trabalho.

Engrossamento padrão

No engrossamento padrão, cada variável i possui uma medida de importância λ_i associada, usada para indicar a ordem em que as variáveis devem ser inseridas em C. Esta medida é inicialmente igual ao número de conexões fortes de i na matriz

transposta S_i^T, que indica quantas variáveis j possuem i em sua vizinhança fortemente conectada S_j:

$$S_i^T = \{j \in \Omega : i \in S_j\}$$
$$\lambda_i = |S_i^T|.$$

A cada iteração do algoritmo, a variável i de maior λ_i no conjunto $U = \Omega \setminus \{C \cup F\}$ de variáveis ainda não decididas é inserida em C, todos os seus vizinhos fortemente conectados são inseridos em F e o peso λ_k de cada vizinho fortemente conectado k a estas novas variáveis em F é incrementado. Tal procedimento é ilustrado na Fig. 13 e descrito no Algoritmo 2.

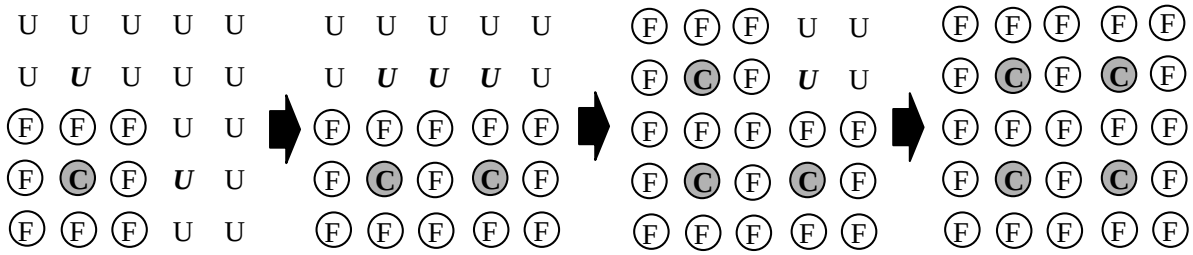


Figura 13 – Primeiros passos do processo de engrossamento padrão. Em cada estágio, os pontos não decididos com maior λ são mostrados em negrito-itálico. Reproduzido de Stuben (2000)

Na inicialização (linhas 1 a 4), é definido o conjunto U de variáveis ainda indefinidas, que contém inicialmente todas as variáveis a serem, uma a uma, removidas de U e inseridas em F ou C, que são conjuntos que iniciam vazios. Assim, o algoritmo consiste em um *loop* iterativo (linhas 6 a 16) que termina quando U se torna vazio, indicando que todo o domínio já foi dividido.

Cada iteração inicia com o cálculo dos pesos de cada variável que ainda se encontra indecidida (linhas 7 e 8). O peso λ_m de uma variável m é definido como o número de variáveis fortemente conectadas a m que ainda estão indecididas somado a 2 vezes o número de variáveis fortemente conectadas a m que já estão em F.

É selecionada a variável ainda indecidida com maior peso para ser inserida em C e removida de U (linha 10 a 12). Assim, vemos que a contabilização de vizinhos indecididos serviu ao propósito de encontrar as variáveis mais representativas, reduzindo o tamanho de C e consequentemente a ordem do sistema grosseiro, e a contabilização de vizinhos em F ao dobro serviu ao propósito de priorizar as variáveis com mais risco de não serem representadas.

Algoritmo 2: Engrossamento padrão (Ω, S)

Dados: Ω e S
Resultado: C e F
Inicialização:
  $F \leftarrow \varnothing$,
  $C \leftarrow \varnothing$,
  $U \leftarrow \Omega$
fim
repita
  para cada $m \in U$ **faça**
    $\lambda_m = |S_m^T \cap U| + 2|S_m^T \cap F|$
  fim
  escolha $i \in U$ com max. λ_i
  $C \leftarrow C \cup \{i\}$
  $U \leftarrow U \setminus \{i\}$
  para cada $j \in S_i^T \cup U$ **faça**
    $F \leftarrow F \cup \{j\}$
    $U \leftarrow U \setminus \{j\}$
  fim
até que $U = \varnothing$;

Por fim, as variáveis fortemente conectadas à variável recentemente inserida em C são removidas de U e inseridas em F, pois serão representadas por ela.

Quando empregado este esquema, deve-se utilizar a interpolação direta ou a interpolação padrão.

Interpolação direta

Representaremos as conexões estritamente negativas e estritamente positivas entre as variáveis, respectivamente, por

$$a_{ij}^- = \begin{cases} a_{ij} & \text{se } a_{ij} < 0 \\ 0 & \text{se } a_{ij} \geq 0 \end{cases} \quad \text{e} \quad a_{ij}^+ = \begin{cases} 0 & \text{se } a_{ij} \leq 0 \\ a_{ij} & \text{se } a_{ij} > 0 \end{cases}$$

Na interpolação direta a definição de interpolação descrita em Stuben (2000)

$$e_i = \sum_{j \in C_i} w_{ij} e_j$$

é diretamente aplicada no cálculo de w, que deve ser proporcionalmente ajustado para manter fixa a somatória do lado esquerdo de cada equação

$$a_{ii}e_i + \sum_{j \in N_i} a_{ij}e_j = a_{ii}e_i + \alpha_i \sum_{k \in P_i} a_{ik}^- e_k + \beta_i \sum_{k \in P_i} a_{ik}^+ e_k \tag{3.7}$$

onde

$$\alpha_i = \frac{\sum_{j \in N_i} a_{ij}^-}{\sum_{k \in P_i} a_{ik}^-} \quad \text{e} \quad \beta_i = \frac{\sum_{j \in N_i} a_{ij}^+}{\sum_{k \in P_i} a_{ik}^+}$$

são as proporções entre a soma de todos os vizinhos e a soma dos vizinhos em C para os casos negativos e positivos, respectivamente, levando-nos diretamente à formula

$$w_{ik} = \begin{cases} -\alpha_i a_{ik}/a_{ii} & (k \in P_i^-) \\ -\beta_i a_{ik}/a_{ii} & (k \in P_i^+) \end{cases}.$$

Definindo, assim, as matrizes de restrição e interpolação conforme a equação (3.6).

Esta interpolação é muito simples e frequentemente não muito precisa, mas pode ser apropriada quando o número de entradas positivas fora da diagonal na matriz for relativamente pequeno.

Como a separação entre coeficientes positivos e negativos não leva a um benefício em certos problemas, Sterck et al. (2008) decidiu por não fazer tal separação. Esta abordagem unificada também foi adotada no código *multigrid* implementado em Carrion (2016), utilizado nos experimentos computacionais deste trabalho. Para tal, os pesos de interpolação são calculados como

$$w_{ij} = -\frac{a_{ij}}{a_{ii}} \frac{\sum_{k \in N_i} a_{ik}}{\sum_{k \in C_i^S} a_{ik}}, \quad j \in C_i^S.$$

Interpolação padrão

O engrossamento padrão garante que exista uma forte conectividade entre as variáveis de F e C. Porém, não garante que *toda* variável em F tenha uma porcentagem fixa do seu total de conectividade refletida em C.

Embora, geralmente, este fato não seja um problema na prática, já que na maioria dos casos o algoritmo de engrossamento por si só garante uma conectividade entre F e C suficiente, esta situação pode ser corrigida modificando a interpolação direta de modo que, para cada $i \in F$, suas conexões fortes em F também sejam indiretamente incluídas na interpolação. Este novo esquema é chamado de **interpolação padrão**.

Para tal, cada e_j $(j \in F_i^S)$ é substituído em (3.7) por uma aproximação obtida de seus respectivos vizinhos

$$e_j \to -\sum_{k \in N_j} a_{jk} e_k / a_{jj},$$

resultando assim em uma nova equação para e_i:

$$\widehat{a}_{ii} e_i + \sum_{j \in \widehat{N}_i} \widehat{a}_{ij} e_j = 0 \quad \text{com} \quad \widehat{N}_i = \{j \neq i : \widehat{a}_{ij} \neq 0\}.$$

Essa modificação melhora a qualidade da interpolação substancialmente (STUBEN, 2000), introduzindo menos erro. Ela também contribui com o objetivo de cada variável em F estar "envolvida"por variáveis interpolatórias.

A Fig. 14 ilustra a interpolação direta e a interpolação padrão de um nível H para o nível h sob dois pontos de vista: acima, temos a variação do vetor interpolado e, abaixo, as relações de dependência, onde são indicadas quais variáveis em Ω^{2h} representam cada variável em Ω^h. Na metade superior, observe que na interpolação direta (linha pontilhada) as variáveis em F não se afetam mutuamente. Já a interpolação padrão (linha contínua), porém, pode ser vista como correspondente à interpolação linear.

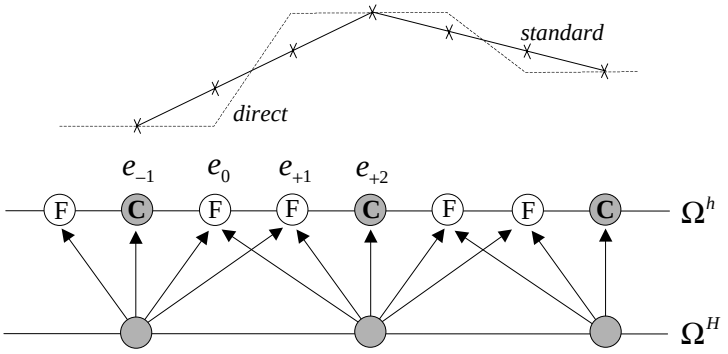


Figura 14 – Interpolação direta e interpolação padrão - Reproduzida de Stuben (2000).

Engrossamento agressivo

Muitas aplicações envolvendo equações diferenciais parciais discretizadas pelo método das diferenças finitas resultam em sistemas lineares com matrizes do tipo stencil com poucos elementos não nulos em cada linha. Nestes casos o engrossamento padrão, baseado em conexões fortes diretas, pode levar a uma complexidade alta no primeiro nível, pois a matriz interpoladora será mais densa que a original.

No engrossamento agressivo, a definição de conexão forte é estendida de modo a incluir variáveis que não estão diretamente conectadas: Uma variável i é dita fortemente l-conectada a uma variável j se existe um caminho de conexões fortes entre i e j de tamanho menor ou igual a um valor l.

Dados os valores $p \geq 1$ e $l \geq 1$. Uma variável i está fortemente l-conectada a uma variável j com respeito a (p, l) se existem *pelo menos* p caminhos de conexões fortes entre i e j de tamanho menor ou igual a l. Portanto, define-se a vizinhança como

$$S_i^{p,l} = \{j \in \Omega : i \text{ fortemente l-conectado a } j \text{ com respeito a} (p, l)\}.$$

Neste trabalho optou-se por considerar $l = 2$ e $1 \leq p \leq 2$ por este ser o caso mais utilizado na literatura, visto que quanto maior o valor de l mais agressiva a estratégia e pior a convergência, conforme destacado por Stüben (2001a).

Assim, tal procedimento pode ser implementando simplesmente aplicando-se duas vezes o algoritmo de engrossamento padrão, como descrito anteriormente. Neste caso, a segunda aplicação estará restrita apenas ao conjunto de variáveis em C da primeira aplicação. Esta vizinhança é definida por:

$$\widehat{S}_i^{p,l} = \{j \in C : i \text{ fortemente l-conectado a } j \text{ com respeito a}(p,l)\}.$$

Apesar do prejuízo na convergência, a complexidade pode ser significantemente reduzida por essa abordagem, visto que Ω^{2h} será um subconjunto menor de Ω^h que no engrossamento padrão. Porém, mesmo utilizando o engrossamento padrão, a complexidade seria reduzida nos níveis subsequentes, pois as matrizes estêncil seriam maiores. Portanto, o emprego do engrossamento agressivo vale a pena apenas no primeiro nível do ciclo V.

Interpolação multi-passo

Quando aplicado o engrossamento agressivo, deve ser utilizada a interpolação multi-passo.

Ela inicia utilizando interpolação direta sempre que possível, isto é, em variáveis F direta e fortemente conectadas a variáveis em C. Nem toda variável em F obedece a esta imposição, devido ao uso de conexões indiretas. Para estas variáveis, utilizam-se as fórmulas de interpolação padrão entre variáveis F vizinhas, conforme exibido no Algoritmo 3.

Engrossamento por conexões positivas fortes

As abordagens anteriores para a divisão C/F foram baseadas apenas nas conexões negativas, considerando-se que as conexões positivas que possam vir a existir seriam relativamente pequenas, podendo ser ignoradas. Porém, existem aplicações onde é necessário permitir que matrizes também contenham conexões positivas (STERCK et al., 2008).

Uma alternativa consiste em empregar alguma das estratégias anteriormente apresentadas e, a *posteriori*, verificar se existem ou não conexões fortes positivas entre as variáveis em F.

Para cada variável $i \in F$, primeiramente, é verificado se

$$a_{ij} \geq \varepsilon_{str}^{+} \max_{k \neq i}(|a_{ik}|) \tag{3.8}$$

Algoritmo 3: Interpolação Multipasso(C,F)

Dados: C e F
Resultado: P
Inicialização:
  $F^* \leftarrow \varnothing$
fim
para cada $i \in F \cup C$ **faça**
  se $C_i^S \neq \varnothing$ **então**
    $F^* \leftarrow F^* \cup$ interpolação direta
  fim
fim
repita
  para cada $i \in F \setminus F^*$ **faça**
    se $S_i \cap F^* \neq \varnothing$ **então**
      $F^* \leftarrow F^* \cup$ interpolação padrão
    fim
  fim
até que $F^* = F$;

é verdade para algum $j \neq i$, onde ε_{str}^{+} é um parâmetro de tolerância empiricamente definido, por exemplo, $\varepsilon_{str}^{+} = 0.5$. Se existir, todos os valores de j que satisfizerem a condição (3.8) serão adicionados ao conjunto S_i e a variável com a conexão positiva mais forte será transferida para o conjunto C.

O engrossamento por conexões positivas é uma generalização da primeira estratégia de engrossamento e, quando empregado, deve ser utilizada a interpolação estendida+i.

Interpolação estendida+i

Esta interpolação utiliza uma vizinhança estendida para determinar os pesos, ou seja, considera não apenas as conexões fortes de cada variável $i \in F$ mas também a vizinhança fortemente conectada destas conexões, como ilustrado na Fig. 15.

A fórmula de interpolação, como apresentada em Yang (2010), é dada por

$$w_{ij} = -\frac{1}{\tilde{a}_{ii}} \left(a_{ij} + \sum_{k \in F_i^S} a_{ik} \frac{\bar{a}_{kj}}{\sum_{l \in \hat{C}_i \cup \{i\}} \bar{a}_{kl}} \right) \quad j \in \hat{C}_i,$$

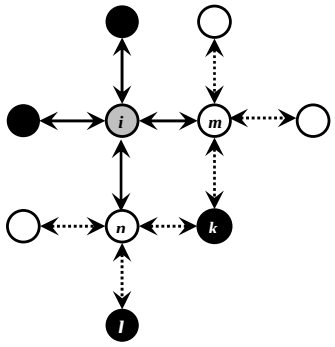


Figura 15 – Exemplo de interpolação estendida. O ponto cinza é a variável a ser interpolada, os pontos pretos são variáveis em C e os pontos brancos são variáveis em F - Reproduzida de (STERCK et al., 2008)

onde

$$
\begin{aligned}
N_i^w &= N^i \backslash (F_i^S \cup C_i^S), \\
\hat{C}_i &= C_i^S \cup \bigcup_{j \in F_i^S} C_j^S, \\
\tilde{a}_{ii} &= a_{ii} + \sum_{n \in N_i^w \backslash \hat{C}_i} a_{in} + \sum_{k \in F_i^S} a_{ik} \frac{\bar{a}_{ki}}{\sum_{l \in \hat{C}_i \cup \{i\}} \bar{a}_{kl}}, \\
\bar{a}_{kl} &= \begin{cases} 0 & \text{se } sinal(a_{kk}) = sinal(a_{kl}) \\ a_{kl} & \text{caso contrário,} \end{cases}.
\end{aligned}
$$

Truncamento de interpolação

Os conjuntos P_i de variáveis interpoladoras podem se tornar grandes, levando a um custo computacional elevado após a aplicação em cada nível do AMG. Porém, os pesos de interpolação entre variáveis distantes usualmente se tornam muito pequenos. Portanto, é possível utilizar truncamento de interpolação, onde para cada variável i são ignorados os pesos de interpolação menores, em valor absoluto, que o maior peso de interpolação de i multiplicado por um fator de ε_{tr}, e os pesos restantes são reescalados de modo a manter a soma total constante. Na prática, um valor de $\varepsilon_{tr} = 0.2$ é usualmente adotado (STUBEN, 2000). Testes empíricos de diferentes valores de ε_{tr} podem ser encontrados em (STERCK et al., 2008).

Double Pairwise Aggregation

O *Double Pairwise Aggregation* (DPA) é um esquema de engrossamento proposto em (NOTAY, 2006), podendo ser usado no lugar dos esquemas clássicos. Neste, é feito um pareamento de variáveis (DIESTEL, 2017), processo este também conhecido por "*matching*".

Em esquemas de engrossamento por agregação, os vértices são separados em conjuntos disjuntos chamados de agregados, tais que cada conjunto é representado por uma variável da matriz grosseira. Estes esquemas são equivalentes a uma divisão C/F onde cada variável em F é representada por uma única variável em C (STUBEN, 2000), de modo que seu valor seja exatamente igual à soma dos valores de suas variáveis representadas, o que simplifica o processo de interpolação. A ideia é que variáveis com ligações fracas estejam em conjuntos diferentes e variáveis com ligações fortes entre si estejam no mesmo conjunto (NOTAY, 2006).

Muito embora a tendência atual das abordagens de engrossamento leve a matrizes de prolongação mais densas, os esquemas de engrossamento por agregação, não muito populares, tomam a direção contrária, onde a matriz de prolongação é uma matriz booleana tal que cada linha possui no máximo um coeficiente não-nulo, de modo a ser muito mais esparsa que nas estratégias clássicas. Assim, o custo computacional da prolongação é reduzido, tanto em uso de memória quanto em processamento (STÜBEN, 2001a; NOTAY, 2006).

Tal matriz é definida por

$$P_{ij} = \begin{cases} 1 & \text{se } i \in G_j, \\ 0 & \text{caso contrário,} \end{cases} \qquad (1 \leq i \leq n_f, 1 \leq j \leq n_c),$$

onde n_f é a ordem original do sistema, n_c é a ordem do sistema grosseiro, e G_j é o j-ésimo agregado. Note que, uma vez que $\bigcup_{1 \leq i \leq n} G_i = \{1, , 2, \ldots, n\}$, P possuirá exatamente um coeficiente não nulo por linha. Às vezes, pode ser vantajoso deixar algumas variáveis fora do conjunto de agregados por não possuírem conexões fortes disponíveis para agregação.

Por exemplo, considere a matriz 7×7

$$A^h = \begin{bmatrix} & & 7 & 3 & 2 & & \\ & & 3 & 6 & 3 & 1 & 9 \\ 7 & 3 & & 5 & & & \\ 3 & 6 & 5 & & 8 & & \\ 2 & 3 & & 8 & & & \\ & 1 & & & & & \\ & 9 & & & & & \end{bmatrix}$$

Em seu grafo correspondente, o algoritmo de *matching* selecionaria o conjunto de arestas independentes – i.e. que não compartilham vértices (DIESTEL, 2017) – circuladas na Fig. 16. Portanto, os operadores de restrição e interpolação seriam definidos

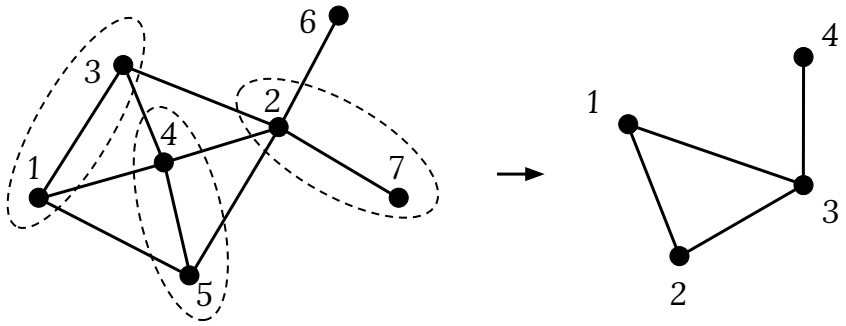


Figura 16 – *Matching* sobre grafo à esquerda. As arestas selecionadas estão circuladas. Na matriz grosseira, cada aresta terá se tornado uma única variável, como representado à direita.

como

$$R^h = (P^h)^T = \begin{bmatrix} 1 & & 1 & & & & \\ & & & 1 & 1 & & \\ & 1 & & & & 1 & \\ & & & & & & 1 \end{bmatrix}.$$

Segue-se que a matriz grosseira seria igual a

$$R^h A^h P^h = \begin{bmatrix} 34 & 10 & 3 & \\ 10 & 36 & 9 & \\ 3 & 9 & 38 & 1 \\ & & 1 & 10 \end{bmatrix}.$$

Portanto, não há necessidade de representar a matriz explicitamente, pois a operação de Galerkin que gera a matriz grosseira A_c, expressa na Eq.(3.5), poderá ser computada através de

$$(A_c)_{ij} = \sum_{k \in G_i} \sum_{l \in G_j} a_{kl} \quad (1 \leq i, j \leq n_c),$$

o que exige menos operações de ponto flutuante que um procedimento de produto matriz vetor completo.

O Algoritmo 4 exibe o algoritmo de *matching* utilizado. A cada iteração do algoritmo é selecionada a variável i com menor número m_i de vizinhos (linha 8). As variáveis com menos vizinhos recebem prioridade a fim de evitar que se tornem *singletons*, isto é, variáveis em C que representam uma única variável do domínio original. Também se seleciona a variável j com menor coeficiente a_{ij} (linha 9). Se j for vizinho fortemente conectado a i, a aresta incidente a i e j – i.e. que conecta i a j (DIESTEL, 2017) – é inserida no *matching* (linha 10). Caso contrário, i se torna um *singleton* (linha

Algoritmo 4: Agregação em pares(A, β)

Dados: Matriz $A = (a_{ij})$ com n linhas; Limite de força de conexão β
Resultado: Número de variáveis grosseiras n_c e agregados $G_i, i = 1 \ldots, n_c$

1 **Inicialização:**
2 $U = [1, n]$;
3 Para todo $i : S_i = \{j \in U \backslash \{i\} | a_{ij} < -\beta \max_{a_{ik}<0} |a_{ik}|\}$;
4 Para todo $i : m_i = |\{j | i \in S_j\}|$;
5 $n_c = 0$.
6 **fim**
7 **enquanto** $U \neq \varnothing$ **faça**
8 Selecione $i \in U$ com menor m_i; $n_c = n_c + 1$;
9 Selecione $j \in U$ tal que $a_{ij} = \min_{k \in U} a_{ik}$;
10 **se** $j \in S_i$ **então**
11 $G_{n_c} = \{i, j\}$,
12 **senão**
13 $G_{n_c} = \{i\}$;
14 **fim**
15 $U = U \backslash G_{n_c}$;
16 **para** $k \in G_{n_c}$ **faça**
17 **para** $l \in S_k$ **faça**
18 $m_l = m_l - 1$;
19 **fim**
20 **fim**
21 **fim**

12). Por fim, as variáveis fortemente conectadas às variáveis selecionadas tem sua vizinhança atualizada (linha 16). Este procedimento é realizado até que todas as variáveis tenham sido selecionadas (linha 7).

Como a utilização de um único *matching* em grafos para gerar os agregados resulta em um processo relativamente lento devido à baixa taxa de redução da ordem do problema em cada nível, o DPA aplica o algoritmo de *matching*, realiza o engrossamento e o aplica uma segunda vez (NOTAY, 2006), de forma que os agregados sejam formados por pares de pares, elevando o potencial de redução da ordem para até quatro vezes.

Deste modo, o número de vértices por agregado é diretamente controlado para um máximo de quatro, enquanto nos esquemas clássicos de engrossamento cada variável em C representa todos os seus vizinhos, sendo que o número de vizinhos de uma variável não é determinado *a priori*. Outra diferença em relação às heurísticas empregadas pelos esquemas clássicos de engrossamento é que o algoritmo de *matching* seleciona primeiro as variáveis com menos vizinhos, para evitar que se tornem *singletons*, i.e., variáveis não agrupadas com nenhuma outra variável. (STERCK et al.,

2010)

Detalhes de Implementação

Enquanto o *Multigrid* Geométrico é capaz de lidar com as matrizes grosseiras apenas implicitamente, isto é, sem efetivamente gerá-las e armazená-las na memória, o *Multigrid* Algébrico necessita de armazenar todas as matrizes. A fim de otimizar o uso de memória e possibilitar o armazenamento de matrizes esparsas de grande ordem foi empregado o formato *Compressed Sparse Row* (CSR) (BARRETT et al., 1994).

O CSR é uma estrutura de dados compacta para matrizes esparsas que economiza elementos nulos armazenando apenas os elementos não-nulos em espaços contíguos através de três vetores para representar uma matriz A : *val*, *col_ind* e *row_ptr*. Os dois primeiros vetores armazenam os coeficientes não-nulos e seus índices de coluna, respectivamente. Já o vetor *row_ptr*, por sua vez, armazena em cada posição i o ponteiro para o primeiro elemento da i-ésima linha, e na posição $n+1$ o ponteiro para o último valor não-nulo acrescido de 1 (CARRION, 2016).

Por exemplo, a representação CSR da matriz

$$A = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & -1 & 4 & -1 & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{bmatrix}$$

será dada por

$$\begin{aligned} \text{val} &= \begin{bmatrix} 4 & -1 & -1 & 4 & -1 & -1 & 4 & -1 & -1 & 4 & -1 & -1 & 4 \end{bmatrix} \\ \text{col\_ind} &= \begin{bmatrix} 1 & 2 & 1 & 2 & 3 & 2 & 3 & 4 & 3 & 4 & 5 & 4 & 5 \end{bmatrix} \\ \text{row\_ptr} &= \begin{bmatrix} 1 & 3 & 6 & 9 & 12 & 14 \end{bmatrix}. \end{aligned}$$

Assim, uma matriz de ordem n e número de elementos não nulos iguais a nnz necessita armazenar apenas $2nnz + n + 1$ valores, em vez de n^2, que é consideravelmente maior quando a matriz é esparsa. Assim, além de poupar memória, a realização de produtos matriz-vetor requer a leitura de menos coeficientes, tornando-se assim mais otimizada.

O algoritmo de *matching* utilizado pelo *Double Pairwise Aggregation* gera sua saída através de um *array* m tal que

$$m_i = \begin{cases} \text{j} & \text{se } i \text{ estiver agrupado a } j \\ -1 & \text{se } i \text{ for agrupado sozinho, i.e., for um } \textit{singleton} \end{cases}.$$

Por exemplo, para a matriz A da Eq. (21), nosso algoritmo agrega o primeiro vértice com o segundo, o último com o penúltimo, e mantém o vértice central como um *singleton*. Assim, é gerado o *array*

$$m = \begin{bmatrix} 1 & 0 & -1 & 4 & 3 \end{bmatrix}.$$

O operador de prolongação P utilizado pelo *Double Pairwise Aggregation* consiste em uma matriz com altura igual à ordem da matriz A e largura igual ao numero de agregados gerados pelo *matching*, onde cada linha i tem exatamente um coeficiente de valor igual à 1 na coluna k, onde k é o índice do agregado ao qual pertence o vértice i. Os demais coeficientes são nulos. Note que, como cada agregado possui no máximo dois vértices, então consequentemente cada coluna possui no máximos dois coeficientes não-nulos. Em nosso exemplo, o operador correspondente ao *matching* representado em (21) seria

$$R^T = P = \begin{bmatrix} 1 & & & \\ 1 & & & \\ & 1 & & \\ & & 1 & \\ & & 1 & \end{bmatrix}.$$

O operador de Galerkin necessário para construir a matriz grosseira consiste no produto matriz-matriz-matriz $P^t \times A \times P$. Porém, uma vez que a matriz P tem exatamente um número 1 por linha e no máximo dois números 1 por coluna, sendo os demais valores nulos, como pode ser visto no exemplo da Eq. (21), não há a necessidade de efetuar nenhum produto matriz-matriz completo. Neste trabalho, para a realização desta operação, foi desenvolvido o método que se segue.

$$R \times A \times P = \begin{bmatrix} 1 & 1 & & & & \\ & & 1 & & & \\ & & & 1 & 1 & \end{bmatrix} \begin{bmatrix} 4 & -1 & & & & \\ -1 & 4 & -1 & & & \\ & -1 & 4 & -1 & & \\ & & -1 & 4 & -1 & \\ & & & -1 & 4 & \end{bmatrix} \begin{bmatrix} 1 & & & \\ 1 & & & \\ & 1 & & \\ & & 1 & \\ & & 1 & \end{bmatrix}$$

Como o início de cada linha pode ser acessado de forma imediata pelo *array* *row_ptr*, para cada linha i, se o vértice i for um *singleton*, copia-se os valores da linha i da matriz original para a linha k da matriz resultante, onde k é uma variável que inicia com o valor 0 e representa o índice do agregado ao qual pertence o vértice i. Caso contrário, e se $i < m_i$, são posicionados ponteiros no início das linhas agrupadas i e m_i que percorrem ambas as linhas em conjunto copiando os valores para a linha k da matriz resultante, somando os coeficientes que possuírem o mesmo índice horizontal.

Após cada processo de cópia, incrementa-se o valor de k. Em nosso exemplo, as linhas da nossa matriz resultante seriam

$$\begin{bmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \end{bmatrix} \longrightarrow \begin{bmatrix} 3 & 3 & -1 & \end{bmatrix}$$

$$\begin{bmatrix} & -1 & 4 & -1 & \end{bmatrix} \longrightarrow \begin{bmatrix} & -1 & 4 & -1 & \end{bmatrix}$$

$$\begin{bmatrix} & & -1 & 4 & -1 \\ & & & -1 & 4 \end{bmatrix} \longrightarrow \begin{bmatrix} & & -1 & 3 & 3 \end{bmatrix}$$

Porém, em cada processo de cópia de linhas, tanto quando i é um *singleton* quanto quando não o é, a mesma estratégia é aplicada internamente para realizar a soma das colunas, isto é, para cada coeficiente da coluna j, se o vértice j for um *singleton*, o valor é diretamente copiado para a posição $[k, l]$ da matriz resultante, onde l é uma variável que inicia com o valor 0 e denota o agregado ao qual pertence o vértice j. Caso contrário, se $j < m_j$, a soma do coeficiente da coluna j com o coeficiente da coluna m_j é armazenada na posição $[k, l]$ da matriz resultante. Após cada processo de cópia, incrementa-se o valor de l. Assim, nosso resultado final seria

$$\begin{bmatrix} 3 & 3 & -1 & & \\ & -1 & 4 & -1 & \\ & & -1 & 3 & 3 \end{bmatrix}$$

$$\downarrow \quad \downarrow \quad \downarrow$$

$$\begin{bmatrix} 6 & -1 & \\ -1 & 4 & -1 \\ & -1 & 6 \end{bmatrix}$$

Deste modo, a matriz é lida apenas uma vez, sequencialmente, enquanto o algoritmo soma os elementos das linhas e colunas. Assim, a função possui ordem $\sim O(nnz)$. Como não é possível prever o número de elementos não nulos da matriz resultante, primeiro a estrutura CSR é armazenada em uma lista de adjacências, e por fim é transferida para *arrays*.

As operações matriz-vetor necessárias para transferir vetores para níveis mais grosseiros ou refinados, isto é, restringir e interpolar, seguem a mesma estratégia e possuem complexidade $O(n)$.

Quanto ao algoritmo de *matching*, este requer a cada passo a seleção do vértice com menor vizinhança fortemente conectada a fim de evitar a formação de *singletons*.

Para encontrar os índices destes vértices sem ampliar o custo computacional para acima da complexidade linear, foi utilizado o algoritmo de ordenação *counting sort* no início do algoritmo para dispor os vértices pelo número de vizinhos.

O *counting sort* não é um algoritmo de ordenação por comparação (estes possuem complexidade mínima $O(n \log n)$), e sua complexidade em casos gerais é igual a $O(n + k)$, onde k é a amplitude dos dados. O algoritmo do método é exibido no Algoritmo 5 (CORMEN et al., 2009).

Algoritmo 5: *Counting sort*(A)

```
   Dados: vetor A
   Resultado: vetor B
 1 Inicialização:
 2 |
 3 |  // k é o maior valor do vetor A
 4 |  k ← max A
 5 |
 6 |  // Criar vetor auxiliar com k+1 elementos e inicializar com zeros
 7 |  para i ← 0 até k faça
 8 |  |  C[i] ← 0
 9 |  fim
10 fim
11
12 para j ← 1 até length(A) faça
13 |  C[A[j]] ← C[A[j]] + 1
14 fim
15 // Agora C[i] contem o numero de elementos igual a i.
16
17 para i ← 1 até k faça
18 |  C[i] ← C[i] + C[i − 1]
19 fim
20 // Agora C[i] contem o numero de elementos menor que ou igual a i.
21
22 para j ← length[A] até 1 faça
23 |  B[C[A[j]]] ← A[j]
24 |  C[A[j]] ← C[A[j]] − 1
25 fim
```

Uma vez que o número de vizinhos de um vértice é um valor inteiro que varia entre 0 e $n - 1$, a ordenação dos vértices por número de vizinhos pode ser feita em $O(n)$ excepcionalmente neste caso através do algoritmo *counting sort* antes do início do *loop* iterativo do Algoritmo de *matching* (4), a fim de que, a cada iteração dele, o vértice i com menor m_i possa ser selecionado em tempo $O(1)$, já que os vértices estão ordenados por valor de m.

Como a cada iteração ao menos uma variável é removida do conjunto U e o

algoritmo termina quando $U = \varnothing$, o máximo de iterações é n. Tendo em vista que a cada iteração os vizinhos da variável selecionada i são percorridos, temos que a ordem de complexidade do algoritmo é de $O(n \cdot \overline{|N|})$, onde $\overline{|N|}$ representa a média do número de vizinhos por variável, sendo que $O(n \cdot \overline{|N|}) = O(nnz)$, onde nnz é o número de elementos não nulos.

Assim, a ordem de complexidade do *Double Pairwise Aggregation* se torna linear em função no número de elementos não nulos da matriz.

Em relação ao engrossamento padrão representado no Algoritmo (2), a cada iteração é selecionado o vértice de maior λ_i, e isso também pode ser feito através de uma ordenação prévia de complexidade $O(n)$ por *counting sort* tornando cada seleção de ordem $O(1)$. Como a variável selecionada e todos os seus vizinhos são removidos do conjunto U, o máximo de iterações é igual a $n/\overline{|N|}$, assim como no caso do DPA.

Porém, a cada iteração os vizinhos da variável selecionada i são percorridos a fim de inseri-los em F. Quando uma variável é inserida em F deve-se percorrer as variáveis que a possuem como vizinha fortemente conectada a fim de atualizar seus respectivos λs. Assim, a cada iteração do algoritmo, após selecionar uma variável é necessário também percorrer os vizinhos de seus vizinhos. Deste modo temos que a ordem de complexidade do algoritmo é de $O(n \cdot \overline{|N|} \cdot \overline{|N|}/\overline{|N|})$, sendo que $O(n \cdot \overline{|N|}) = O(nnz)$, onde nnz é o número de elementos não nulos.

Portanto, a ordem de complexidade dos esquemas clássicos de engrossamento é linear em função no número de elementos não nulos da matriz.

4 Precondicionadores

Uma das desvantagens dos métodos iterativos em relação aos diretos é a possibilidade de não convergência, i.e., menor robustez. Porém, métodos iterativos não estacionários podem ter tanto a eficiência quanto a robustez melhoradas com o uso de uma técnica chamada de precondicionamento.

O precondicionamento consiste numa transformação do sistema original num sistema equivalente onde a matriz tenha um número de condição mais favorável. Esta transformação pode ser obtida, por exemplo, premultiplicando e posmultiplicando a matriz A por duas matrizes P e Q:

$$(PAQ)(Q)^{-1}x = Pb.$$

Assim, o sistema precondicionado pode ser representado como

$$\bar{A}\bar{x} = \bar{b} \tag{4.1}$$

onde

$$\begin{aligned}\bar{A} &= PAQ,\\ \bar{x} &= Q^{-1}x,\\ \bar{b} &= Pb.\end{aligned}$$

As matrizes P e Q devem ser escolhidas de modo a proporcionar à matriz $\bar{A}$ melhores propriedades de convergência em relação a matriz original A e são chamadas de matrizes de precondicionamento.

No caso onde $Q = I$, chamamos de precondicionamento à esquerda, e no caso onde $P = I$, chamamos de precondicionamento à direita. Neste trabalho, vamos considerar o precondicionamento à esquerda conforme descrito a seguir.

Uma vez que o sistema original é dado pela Eq. (2.1), podemos transformar a iteração (2.2) da seguinte forma:

$$\begin{aligned}c &= x - Gx\\ &= (I - G)x\\ &= (I - G)A^{-1}b.\end{aligned} \tag{4.2}$$

Portanto, uma vez que G tem a forma expressa na Eq. (2.8) e c pode ser definido pela Eq. (4.2), através da expressão geral dos métodos iterativos estacionários (2.5)

podemos reescrever o sistema original (2.1) por

$$\begin{aligned} x &= (I - M^{-1}A)x + M^{-1}b \\ &= x - M^{-1}Ax + M^{-1}b \end{aligned}$$

ou

$$M^{-1}Ax = M^{-1}b \implies \bar{A}x = \bar{b}. \tag{4.3}$$

Este novo sistema é equivalente ao original, isto é, possui a mesma solução, porém, é melhor condicionado.

A matriz $P = M^{-1}$, assim, se torna a matriz de precondicionamento e deve ser suficientemente simples de se construir. Note que, mesmo que M seja esparsa, não há nenhuma razão pela qual M^{-1} deveria apresentar a mesma esparsidade que M. Deste modo, deve-se evitar a construção explícita da matriz M^{-1}. Além do que, o cálculo da inversa M^{-1} é mais custoso do que resolver o sistema original por um método direto.

A única operação dos métodos iterativos não estacionários que depende da matriz $\bar{A}$ - presente no sistema (4.3) - é o produto matriz vetor, que pode ser representado por $v = \bar{A}z$. Tal operação pode ser calculada em dois passos. No primeiro o produto matriz-vetor original $w = Az$ é calculado e no segundo passo o sistema linear trivial $Mv = w$ é resolvido por algum método de baixa complexidade.

$$\begin{aligned} v &= \bar{A}z \\ &= M^{-1}\overbrace{Az}^{w \leftarrow Az} \\ &= M^{-1}w \end{aligned} \tag{4.4}$$

isto é,

$$Mv = w.$$

Neste trabalho, utilizamos o método do resíduo mínimo generalizado (GMRES, do inglês): método de solução de sistemas lineares não simétricos que aproxima a solução através do vetor de resíduo mínimo em um subespaço de Krylov.

A seguir apresentamos os precondicionadores implementados: precondicionador Gauss-Seidel (SAAD, 2003), a fatoração ILU (p) (SAAD, 2003) e o precondicionador *multigrid* algébrico (YANG, 2006).

4.1 GMRES precondicionado à esquerda

Considere o espaço de Krylov

$$\mathcal{K}_m(r_0, A) = \text{span}\{r_0, Ar_0, \ldots, A^m r_0\}.$$

sendo r_0 o vetor resíduo inicial. O GMRES computa $x_m \in x_0 + \mathcal{K}_m(r_0, A)$ de menor $\|r\|_2$, onde r é definido pela Eq. (2.7).

Uma vez que o sistema precondicionado pode ser descrito pela Eq. (4.3), quando se utiliza o precondicionamento à esquerda temos que no sistema expresso pela Eq. (4.1)

$$\begin{aligned}\bar{A} &= M^{-1}A \\ \bar{b} &= M^{-1}b,\end{aligned}$$

e portanto, o resíduo do sistema precondicionado é calculado através de

$$\bar{r}_0 = M^{-1}r_0 = M^{-1}(b - Ax_0),$$

de modo que

$$\begin{aligned}x_m &= x_0 + p_{m-1}(M^{-1}A)\bar{r}_0 \\ &= x_0 + p_{m-1}(M^{-1}A)M^{-1}r_0 \\ &= x_0 + M^{-1}p_{m-1}(AM^{-1})r_0,\end{aligned}$$

onde p_{m-1} é um polinômio de grau $\leq m-1$.

Assim, no precondicionamento à esquerda do GMRES é utilizado o espaço de Krylov

$$\mathcal{K}_m(\bar{r}_0, M^{-1}A) = \text{span}\{\bar{r}_0, M^{-1}A\bar{r}_0, \ldots, (M^{-1}A)^{m-1}\bar{r}_0\}$$

e computa-se $x_m \in x_0 + \mathcal{K}_m(\bar{r}_0, M^{-1}A)$ tal que

$$\|\bar{r}_m\|_2 = \|(I - M^{-1}Ap_{m-1}(M^1A))\bar{r}_0\|$$

é tão pequeno quanto possível. O Algoritmo 6 mostra os principais passos do método GMRES.

Nota-se que o método GMRES executa, a cada iteração i, 3 operações principais: i produtos escalares, i combinações lineares e 1 produto matriz-vetor. A ação do precondicionamento ocorre em cada produto matriz-vetor (linha 7), e, em cada início do *restart*, na atualização do resíduo (linha 3).

Algoritmo 6: GMRES precondicionado à esquerda(m, A, $\bar{b}$)

Dados: Tamanho da base m de Krylov; Matriz A; Vetor de termos independentes precondicionado $\bar{b} = M^{-1}b$
Resultado: Vetor solução x

$x \leftarrow 0$
repita
  $\bar{r}_0 = M^{-1}(b - Ax_0)$
  $\beta = \|\bar{r}_0\|_2$
  $v_1 = \bar{r}_0/\beta$
  para $j = 1, \ldots, m$ **faça**
    $w \leftarrow M^{-1}Av_j$
    Ortogonalizar w em $v_1, \ldots, v_j$ (Gram-Schmidt)
    $h_{j+1,j} \leftarrow \|w\|_2$
    $v_{j+1} \leftarrow w/h_{j+1,j}$
  fim
  $V_m \leftarrow [v_1, \ldots, v_m]$
  $\bar{H}_m \leftarrow ((h_{i,j}))$
  $y_m \leftarrow \text{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2$
  $x_m \leftarrow x_0 + V_m y_m$
  $x_0 \leftarrow x_m$
até que *convirja*;

4.2 Precondicionador Gauss-Seidel

Os métodos iterativos estacionários são baseados na fatoração simples da matriz dos coeficientes definida por:

$$A = L + D + U$$

onde D é a diagonal de A e L e U representam, respectivamente, as partes estritamente inferior e estritamente superior de A. O precondicionador Gauss-Seidel é definido por:

$$M = (L + D)D^{-1}(D + U).$$

Uma vantagem do precondicionador Gauss-Seidel é que não há necessidade de armazenamento extra, pois os coeficientes de M são os próprios coeficientes da matriz A.

4.3 Precondicionador ILU (p)

A fatoração LU incompleta (ILU), usualmente utilizada como precondicionador, é uma aproximação da fatoração LU, que por sua vez, pode ser vista como a

forma matricial da eliminação de Gauss. Pode ser empregado independentemente da simetria da matriz original e é especialmente útil quando se pretende solucionar mais de um sistema linear com a mesma matriz de coeficientes.

Na eliminação de Gauss, ou método do escalonamento, que pode ser visualizada no Algoritmo 7, dada uma matriz de coeficientes A são aplicadas operações elementares a fim de produzir uma matriz triangular superior U tal que os sistemas $Ax = b$ e $Ux = y$ apresentam a mesma solução x.

Algoritmo 7: Eliminação de Gauss(A)

Dados: Matriz A
Resultado: Matriz U
para $i = 2, \ldots, n$ **faça**
  para $k = 1, \ldots, i-1$ **faça**
    $a_{ik} \leftarrow a_{ik}/a_{kk}$
    para $j = k+1, \ldots, n$ **faça**
      $a_{ij} \leftarrow a_{ij} - a_{ik}a_{kj}$
    fim
  fim
fim
$U \leftarrow A$

A fatoração LU, apresentada no Algoritmo 8, é um dos métodos mais conhecidos para solução direta de sistemas lineares. Nela, além de U é produzida uma matriz triangular inferior L tal que $LU = A$. Assim, o sistema $Ax = b$ se torna equivalente ao sistema $LUx = b$. De tal modo, feita a fatoração, o sistema pode ser solucionado de forma mais barata computacionalmente empregando-se substituição direta para resolver $Ly = b$ e, finalmente, substituição reversa para resolver $Ux = y$.

Algoritmo 8: Fatoração LU completa(A)

Dados: Matriz A
Resultado: Matriz L, matriz U
para $i = 2, \ldots, n$ **faça**
  para $k = 1, \ldots, i-1$ **faça**
    $l_{ik} \leftarrow a_{ik} \leftarrow a_{ik}/a_{kk}$
    para $j = k+1, \ldots, n$ **faça**
      $a_{ij} \leftarrow a_{ij} - a_{ik}a_{kj}$
    fim
  fim
fim
$U \leftarrow A$

As matrizes L e U são, potencialmente, muito menos esparsas que A. Este preenchimento de posições originalmente nulas é chamado de *fill-in*, e consome memória

e tempo de processamento, inviabilizando o uso da fatoração LU completa como precondicionador, especialmente na solução de sistemas lineares de grande porte.

A fim de reduzir o *fill-in*, a fatoração ILU (0) emprega uma aproximação das matrizes L e U onde as posições originalmente nulas são descartadas, e assim são geradas as matrizes $\widetilde{L}$ e $\widetilde{U}$ sem qualquer *fill-in*. Este procedimento é exibido no Algoritmo 9, onde $nnz(A)$ é o conjunto de coeficientes não nulos em A.

Algoritmo 9: Fatoração ILUo(A)

Dados: Matriz A
Resultado: Matriz L, matriz U

```
para i = 2,...,n faça
    para k = 1,...,i−1 faça
        se (i,k) ∈ nnz(A) então
            l_ik ← a_ik ← a_ik / a_kk
            para j = k+1,...,n faça
                se (i,j) ∈ nnz(A) então
                    a_ij ← a_ij − a_ik a_kj
                fim
            fim
        fim
    fim
fim
U ← A
```

A acurácia do ILU (0) pode ser insuficiente para obter uma taxa de convergência adequada em algumas situações. Uma fatoração mais eficiente e mais confiável poderia permitir alguns *fill-ins*. Para tal, uma generalização do método ILU é o método ILU (p) exibido no Algoritmo 10, onde o descarte de posições originalmente não nulas só ocorre a partir da $(p+1)$-ésima etapa da fatoração. Assim, quanto maior o parâmetro p, mais robusto o método, porém, maior o custo computacional. Quando utilizado $p = 0$, nenhum *fill-in* é produzido, e quando $p = n - 1$, temos a fatoração LU completa.

Infelizmente, a quantidade de *fill-in* não é predizível quando $p > 0$, e existe o risco de algum coeficiente grande ser descartado, levando a uma aproximação demasiado imprecisa.

Quando o método ILU é empregado como precondicionador, é adotado $M = \widetilde{L}\widetilde{U}$. (SAAD, 2003)

Algoritmo 10: Fatoração ILU (A, p)

Dados: Matriz A; parâmetro p
Resultado: Matriz L, matriz U

para $i = 1, \ldots, n$ **faça**
  para $j = 1, \ldots, n$ **faça**
    se $a_{ij} > 0 \vee i = j$ **então**
      $lev_{ij} = 0$
    senão
      $lev_{ij} = \infty$
    fim
  fim
fim
para $i = 2, \ldots, n$ **faça**
  para $k = 1, \ldots, i-1$ **faça**
    se $lev_{ik} \leq p$ **então**
      $l_{ik} \leftarrow a_{ik} \leftarrow a_{ik}/a_{kk}$
      para $j = k+1, \ldots, n$ **faça**
        $a_{ij} \leftarrow a_{ij} - a_{ik}a_{kj}$
        se $a_{ij} = 0$ **então**
          $lev_{ij} = \min lev_{ij}, lev_{ik} + lev_{kj} + 1$
        fim
      fim
    fim
  fim
fim
$U \leftarrow A$

4.4 Precondicionador *Multigrid* Algébrico

Ao utilizar o AMG como precondicionador, em cada produto matriz-vetor do tipo da expressão (4.4) é executado um único ciclo-V do *multigrid* no sistema

$$Av = w, \tag{4.5}$$

seguindo as ideias descritas em (MCADAMS; SIFAKIS; TERAN, 2010).

Caso o AMG fosse executado até que se atingisse a tolerância ε do resíduo, tal como na implementação clássica do *multigrid*, teríamos que o uso do precondicionador AMG seria equivalente a utilizar o AMG como *solver*, pois a matriz de precondicionamento seria idêntica à matriz original do sistema A, e como o AMG iteraria até atingir a convergência, o sistema seria inteiramente solucionado pelo *multigrid* na

geração do vetor de termos independentes precondicionado expresso na Eq. (4.3)

$$\begin{aligned} \bar{f} &\leftarrow \overbrace{M^{-1}}^{M=A} b \\ &\leftarrow A^{-1}b. \end{aligned}$$

Porém, o uso do *multigrid* como *solver* é menos eficiente que a utilização do GMRES como *solver* aliado a um bom precondicionamento.

Isso também implicaria em

$$\begin{aligned} v &= A^{-1}Az, \\ &= z. \end{aligned}$$

Deste modo, em vez desta implementação clássica do AMG, é executado um único ciclo-V, o que não reduz o erro até a tolerância ε especificada, de modo que $v \neq z$, e consequentemente, M^{-1} não seja igual a A, mas sim igual a uma matriz completamente implícita tal que a solução do produto matriz vetor (4.4) resulte no mesmo que a aplicação de um único ciclo-V do AMG no sistema (4.5). Isto é:

$$Ciclo\_V(1, nr, \omega, A, z) \quad \longleftrightarrow \quad M^{-1}z$$

onde M é desconhecida.

Portanto segue-se que, como é realizado um único ciclo-V, a ordem de complexidade de cada aplicação do precondicionamento se torna linear em função do número de coeficientes não-nulos do sistema original, o que atrai atenção para o uso do AMG como precondicionador.

5 Resultados Experimentais

Nessa seção são apresentados os resultados dos experimento numéricos realizados com o AMG, executado tanto como *solver* quanto como precondicionador, na solução de sistemas lineares de grande porte. Os experimentos foram organizados em três conjuntos distintos. O primeiro composto por matrizes gerais oriundas de aplicações diversas. O segundo composto por matrizes estêncil, resultantes da discretização por diferenças finitas de problemas tridimensionais. O terceiro composto por matrizes oriundas da discretização por elementos finitos de problemas regidos pelas equações de Euler. Neste último caso o precondicionador AMG foi inserido no código FEMCodes[1] em desenvolvimento no Laboratório de Otimização e Modelagem Computacional da UFES - LabOtim.

Nos testes, foram utilizados como *solver* o método AMG, o GMRES não precondicionado e também o GMRES com os seguintes precondicionadores:

- **GS**: Fatoração Gauss-Seidel;
- **ILU**: Fatoração LU Incompleta;
- **Stb**: AMG com esquemas de engrossamento clássicos de Stüben;
- **DPA-S**: AMG com *Double Pairwise Aggregation* e SOR como relaxador;
- **DPA-G**: AMG com *Double Pairwise Aggregation* e GMRES como relaxador.

Em todos os testes, os parâmetros utilizados com valores fixos são apresentados na Tab. 2. Nenhum truncamento de interpolação foi utilizado e nos testes com esquemas clássicos de engrossamento optou-se pelo engrossamento agressivo, conforme descritos no Cap. 3.

A estratégia DPA-G tem por objetivo reduzir o problema a um sistema trivial. Uma vez que a cada engrossamento bem sucedido a ordem do sistema é reduzida a um quarto (dado que são aplicados dois *matchings* onde cada um reduz a ordem, idealmente, pela metade), é utilizado um número de níveis $NCL = \log_4 n - 2$. Desta forma, a estratégia realiza dois níveis de engrossamento a menos que o número estimado para reduzir o sistema à ordem 1. A redução do número de níveis de engrossamento em dois nesta estratégia se deve ao fato de termos observado nos testes realizados que o ganho de tempo proporcionado pela melhora da taxa de convergência nos domínios inferiores não superou o custo de produzir os mesmos. Em nossos testes preliminares,

[1] <https://github.com/mod-comp-ufes/FEM_CODES>

Parâmetro	**Descrição**	**Valor**
ϵ	Tolerância	10^{-8}
k	Tamanho da base de Krylov do GMRES	40
lmax	Número máximo de iterações. Acima deste valor, toda matriz foi considerada como não-convergente	1.000
nr	Número de relaxações em cada nível do ciclo V nos métodos Stb e DPA-S	2
ω	Parâmetro do SOR	1.5
NCL	Número de níveis do ciclo V do Stb e DPA-S	2
DPA-G-k	Tamanho da base de Krylov do GMRES utilizado como relaxação no DPA-G	10
DPA-G-lmax	Número máximo de iterações do GMRES utilizado como relaxação no DPA-G, exceto no último nível	1
DPA-G-ϵ	Tolerância do GMRES utilizado como relaxação no DPA-G	10^{-8}
x_0	Solução inicial	**0**

Tabela 2 – Parâmetros utilizados com valores fixos

a mesma estratégia não se mostrou frutífera com o engrossamento Stb. Os parâmetros utilizados no GMRES como precondicionador podem ser visualizados na Tab. 2.

Testes para comparação entre AMG como solver e como precondicionador foram realizados em uma máquina do Laboratório de Otimização e Modelagem Computacional (LabOtim) da UFES que conta com um Intel® Xeon® E5410 2.33GHz × 4 64 bits, 32 GB de memória RAM e Ubuntu 16.04.10.

Os experimentos com matrizes estêncil e matrizes gerais foram executados no supercomputador Lobo Carneiro (LoboC) da UFRJ. O LoboC conta com 504 CPUs Intel® Xeon® E5-2670v3, totalizando 6048 núcleos de 2.3GHz, com 24 núcleos reais por nó de processamento e 48 com *Hyper-Threading* (HT), totalizando 252 nós de processamento. Cada nó possui 64 GB, totalizando 16 TB de memória RAM distribuída. O sistema operacional instalado no LoboC é o Suse Linux Enterprise.[2]

Os experimentos numéricos das aplicações oriundas da solução das equações de Euler resolvidas pelo método dos elementos finitos foram realizados em uma máquina do Laboratório de Computação de Alto Desempenho (LCAD) da UFES, que possui um Intel® Xeon® Processor E7-4850 v4, que conta com 16 núcleos com frequência base de 2.10 GHz e 40 MB de cache. A máquina possui 251 GB de memória RAM e sistema operacional Ubuntu 16.04.5 LTS.[3]

Comparações entre resultados serão realizadas apenas entre matrizes pertencentes a um mesmo grupo, com execuções na mesma máquina.

Uma vez que tanto o LoboC quanto a máquina disponibilizada pelo LCAD

[2] <http://www.nacad.ufrj.br/pt/recursos/sgiicex>
[3] <http://www.lcad.inf.ufes.br/>

são compartilhados, a carga varia ao longo da realizações dos testes, tornando os resultados instáveis. Devido a este fator, todas as execuções foram realizadas 5 vezes, eliminando-se os menores e o maiores valores encontrados e extraindo a média aritmética dos restantes em cada caso.

Todos os códigos implementados neste trabalho foram escritos na linguagem de programação C, compilados usando o compilador *gcc* versão 5.4.0 com nível de otimização -O3 e estão disponíveis em <https://github.com/mod-comp-ufes/AMG>.

Nas seções a seguir serão exibidos os resultados e observações extraídas em cada grupo de testes: matrizes gerais, matrizes estêncil e matrizes oriundas de problemas regidos pela equações de Euler discretizada pelo método dos elementos finitos.

5.1 Grupo 1 - Testes com matrizes gerais

Neste conjunto de testes foram utilizadas 17 matrizes obtidas na coleção de matrizes esparsas da Universidade da Flórida[4]. Estas matrizes são oriundas de aplicações diversas tais como problemas termodinâmicos, acústicos, estruturais ou de dinâmica dos fluidos em domínios bi e tridimensionais, discretizados através de diferentes métodos numéricos.

Tais matrizes são amplamente utilizadas como *benchmarks*, i.e., avaliações comparativas e estimativas de performances de métodos numéricos, pois, uma vez que não são geradas artificialmente, mas ao contrário, advém de aplicações reais, permitem uma avaliação mais robusta das estratégias. Dentre as vantagens na utilização dessas matrizes como *benchmarks*, pode-se destacar: (1) facilitar a repetição de testes, pois são disponibilizadas na internet e (2) possibilitar seu uso em métodos que utilizam diferentes formatos de armazenamento, uma vez que elas são disponibilizadas em três opções: MATLAB, Rutherford Boeing ou o formato próprio do repositório *Matrix Market*.

A Tab. 3 apresenta as principais características das matrizes escolhidas para os experimentos. As primeiras colunas indicam, da esquerda para a direita, o índice, nome, ordem, número de elementos não nulos, simetria e porcentagem de esparsidade. Como cada matriz possui origem diferente, não há motivo para se utilizar os mesmos parâmetros em todos os testes. Assim, foram testados valores entre 0 e 0.5 (inclusive) com passo de 0.05 para os parâmetros β e ε dos métodos Stb e DPA-S, respectivamente, e níveis de *fill-in* p para o precondicionador ILU entre 0 e 4. As três últimas colunas mostram as escolhas para os parâmetros β, ε e p baseadas nos melhores valores encontrados com relação ao tempo total de processamento.

[4] <https://sparse.tamu.edu/>

#	Nome	n	nnz	simétrica	esparsidade	ε	β	p
1	rail_5177	5.177	35.185	sim	99,8687%	0,50	0,05	4
2	afto1	8.205	125.567	sim	99,8135%	0,50	0,15	4
3	FEM_3D_thermal1	17.880	43.074	não	99,9865%	0,25	0,10	0
4	Dubcova2	65.025	1.030.225	sim	99,9756%	0,00	0,20	1
5	H2O	67.024	2.216.736	sim	99,9507%	0,20	0,00	0
6	FEM_3D_thermal2	147.900	3.489.300	não	99,9840%	0,50	0,05	0
7	parabolic_fem	525.825	3.674.625	sim	99,9987%	0,00	0,50	4
8	atmosmodj*	1.270.432	8.814.880	não	99,9995%	0,05	0,01	1
9	atmosmodd*	1.270.432	8.814.880	não	99,9995%	0,00	0,00	1
10	Serena	1.391.349	64.131.971	sim	99,9967%	0,50	0,10	3
11	Geo_1438	1.437.960	60.236.322	sim	99,9971%	0,00	0,10	0
12	atmosmodl	1.489.752	10.319.760	não	99,9995%	0,35	0,00	1
13	af_shell10	1.508.065	52.259.885	sim	99,9977%	0,00	0,00	1
14	G3_circuit	1.585.478	7.660.826	sim	99,9997%	0,40	0,20	4
15	Transport	1.602.111	23.487.281	não	99,9991%	0,40	0,15	4
16	Cube_Coup_dt6	2.164.760	124.406.070	sim	99,9973%	†	0,15	†
17	Bump_2911	2.911.419	127.729.899	sim	99,9985%	0,45	0,15	†

Tabela 3 – Características das Matrizes Gerais e parâmetros adotados.

As matrizes com asterisco são diagonal-dominantes. Os casos de não-convergência são exibidos como †. Informações referentes ao método DPA-G não aparecem na tabela pois todos os testes realizados com este método indicaram que o valor nulo para β é o mais adequado.

	Solver		precond.	
#	iter	t(s)	iter	t(s)
1	†	†	**174**	**0,32**
2	**7**	**0,02**	753	3,13
3	15	**0,19**	**12**	0,23
4	**15**	44,40	48	**4,51**
5	†	†	**26**	**6,01**
6	20	2,88	**11**	**2,39**
7	†	†	**585**	**264,69**
8	290	191,21	**44**	**43,81**
9	445	292,63	**43**	**44,73**
10	†	†	**139**	**612,89**
11	†	†	**197**	**838,55**
12	92	66,22	**23**	**26,04**
13	†	†	**7**	**16,47**
14	†	†	**326**	**284,71**
15	†	†	**561**	**1147,00**
16	–	–	–	–
17	–	–	–	–

Tabela 4 – Resultados do AMG como *solver* comparado aos resultados do AMG como precondicionador do GMRES com matrizes gerais. Execução realizada na máquina do LabOtim.

A Tab. 4 exibe os resultados em termos de número de iterações e tempo de processamento do AMG-S (método *Multigrid* algébrico, utilizando o método SOR como relaxamento) como *solver* e como precondicionador do GMRES. Por motivos de não disponibilidade de máquina, os casos 16 e 17 não puderam ser executados. O AMG-S como *solver* convergiu apenas em 7 das 15 matrizes, obtendo menor tempo em apenas 2 casos, enquanto o AMG-S como precondicionador convergiu em todos os testes realizados. Os resultados obtidos justificam nossa escolha de investigar o *Multigrid* como precondicionador e não como *solver*.

A seguir são apresentados os resultados considerando o GMRES sem precondicionamento, o GMRES precondicionado pela fatoração Gauss-Seidel, ILU, Stb, DPA-S e DPA-G, respectivamente, nas Tabs. 5 a 10. Para cada uma das matrizes, os melhores resultados em termos de número de iterações e tempo computacional estão em negrito. Entretanto é possível que em uma tabela, o número de iterações esteja em negrito - indicando que este resultado apresenta o menor número de iterações entre todos os métodos para essa matriz - mas o tempo computacional não está em negrito, pois um outro método foi mais rápido, mesmo que o seu número de iterações tenha sido maior. Em todas as tabelas as colunas indicam:

- **iter:** número de iterações;

- **coarse:** tempo de engrossamento;
- **galerk:** tempo da operação de Galerkin;
- **setup:** tempo de engrossamento + Galerkin;
- **relax:** tempo de relaxação;
- **transf:** tempo das transferências intergrid (restrições e interpolações);
- **ciclo-V:** tempo de relaxação e transferências;
- **solver:** tempo das operações do GMRES;
- **exec:** tempo total da fase de execução, compreendendo ciclo-V e solver;
- **total:** tempo de *setup* e execução.

#	iter	total
1	†	†
2	†	†
3	269	0,997
4	†	†
5	†	†
6	759	23,292
7	†	†
8	†	†
9	†	†
10	†	†
11	†	†
12	445	72,340
13	109	28,450
14	†	†
15	†	†
16	†	†
17	†	†

Tabela 5 – Resultados do GMRES sem precondicionamento em matrizes gerais. Execução realizada no LoboC.

#	iter	precond	solver	exec	total
1	†	†	†	†	†
2	†	†	†	†	†
3	249	1,246	0,964	2,210	2,210
4	817	11,926	10,259	22,180	22,180
5	427	12,350	9,238	21,588	21,588
6	392	17,523	13,811	31,334	31,334
7	†	†	†	†	†
8	340	40,806	51,772	92,578	92,578
9	222	27,446	35,411	62,857	62,857
10	3	3,682	1,683	5,366	5,366
11	†	†	†	†	†
12	111	16,559	20,702	37,260	37,260
13	49	16,157	13,060	29,216	29,216
14	†	†	†	†	†
15	†	†	†	†	†
16	†	†	†	†	†
17	†	†	†	†	†

Tabela 6 – Resultados do GMRES precondicionado pela fatoração Gauss-Seidel em matrizes gerais. Execução realizada no LoboC.

#	iter	setup	precond	solver	exec	total
1	149	0,123	0,288	0,149	0,437	0,561
2	**37**	0,213	0,131	0,079	0,211	**0,424**
3	9	0,119	0,037	0,055	0,092	**0,211**
4	86	2,316	2,711	1,392	4,103	6,496
5	299	0,676	5,605	8,329	13,933	14,609
6	8	0,810	0,254	0,347	0,602	**1,413**
7	707	4,283	76,884	42,103	118,987	123,284
8	147	2,823	22,404	31,285	53,734	56,597
9	118	3,244	17,705	24,959	42,664	45,907
10	77	2211,188	287,672	38,674	326,385	2537,467
11	491	25,456	270,929	292,647	534,067	622,124
12	61	3,517	11,074	16,188	27,322	30,867
13	**1**	2,315	0,404	0,556	0,960	**3,299**
14	195	6,491	56,226	43,380	99,823	106,532
15	688	277,254	1487,493	268,893	1756,385	2033,698
16	†	†	†	†	†	†
17	†	†	†	†	†	†

Tabela 7 – Resultados do GMRES precondicionado pelo ILU com níveis de *fill-in* definidos na Tab. 3. Execução realizada no LoboC.

#	iter	coarse	galerk	setup	relax	transf	ciclov	solver	exec	total
1	187	0,017	0,011	0,028	0,731	0,082	0,186	0,224	1,157	1,185
2	479	0,053	0,026	0,079	4,405	0,299	1,153	1,299	6,890	6,968
3	12	0,083	0,068	0,150	0,062	0,005	0,013	0,009	0,084	0,235
4	77	1,606	0,312	1,918	5,082	0,670	1,841	1,707	8,658	10,576
5	**26**	0,934	1,982	3,051	10,606	0,148	0,820	0,777	12,202	15,253
6	11	0,561	0,595	1,155	0,603	0,033	0,102	0,079	0,785	1,939
7	**406**	3,598	2,128	5,728	235,570	26,797	49,513	45,360	330,423	336,151
8	**44**	2,628	2,310	4,918	9,750	0,747	1,602	2,173	13,567	18,461
9	**43**	2,612	2,363	4,975	9,533	0,730	1,544	2,221	13,298	18,411
10	93	19,347	21,084	40,435	408,561	10,698	57,390	54,098	520,049	560,202
11	268	104,137	18,123	127,143	587,483	63,614	184,133	140,319	911,935	1043,599
12	**23**	3,114	2,796	5,896	6,115	0,461	0,968	1,218	8,301	14,185
13	9	7,033	1,704	8,909	2,563	0,241	0,709	0,614	3,881	12,789
14	353	3,404	1,644	5,064	185,198	18,919	46,055	55,573	286,844	291,817
15	**524**	4,116	5,168	9,289	232,099	8,608	30,440	46,585	308,975	318,245
16	†	†	†	†	†	†	†	†	†	†
17	662	32,080	31,864	63,915	4175,454	98,056	652,477	658,623	5486,554	**5550,469**

Tabela 8 – Resultados do Stb em matrizes gerais. Execução realizada no LoboC.

#	iter	coarse	galerk	setup	relax	transf	ciclov	solver	exec	total
1	230	0,059	0,006	0,065	0,853	0,069	0,264	0,365	1,508	1,574
2	247	0,109	0,014	0,133	2,670	0,086	0,613	0,833	4,163	4,308
3	12	0,124	0,030	0,154	0,070	0,005	0,015	0,012	0,097	0,251
4	42	1,340	0,224	1,565	5,318	0,134	1,002	1,131	7,473	9,038
5	40	2,552	0,574	3,151	8,786	0,154	1,783	2,043	12,617	15,768
6	11	1,109	0,206	1,315	0,473	0,034	0,109	0,088	0,670	1,985
7	863	2,343	0,559	2,902	195,286	7,433	35,674	49,416	280,376	283,275
8	53	4,247	0,631	4,878	8,367	0,401	1,511	2,733	12,576	17,454
9	53	3,213	0,549	3,814	8,734	0,450	1,692	3,135	13,564	17,407
10	105	26,075	6,388	32,463	232,617	5,412	50,382	51,765	334,764	**367,184**
11	**168**	27,008	6,144	33,526	347,171	8,596	76,243	79,444	502,857	**536,808**
12	27	3,325	0,611	3,974	5,226	0,255	0,970	1,588	7,784	11,761
13	7	13,497	2,547	16,089	3,028	0,174	0,598	0,507	4,103	20,191
14	436	13,258	2,725	15,984	683,608	40,724	150,604	239,047	1073,258	1088,610
15	901	8,454	1,412	9,867	241,511	17,643	54,750	84,625	387,098	396,869
16	**298**	41,701	11,426	52,947	1933,590	26,177	372,048	382,424	2686,326	**2739,273**
17	**488**	125,788	26,903	152,821	6010,666	100,867	1276,312	1400,675	8687,652	8840,473

Tabela 9 – Resultados do DPA-S em matrizes gerais. Execução realizada no LoboC.

#	iter	coarse	galerk	setup	relax	transf	ciclov	solver	exec	total
1	**1**	0,022	0,006	0,028	0,003	0,001	0,003	0,001	0,006	**0,034**
2	†	†	†	†	†	†	†	†	†	†
3	**1**	0,229	0,066	0,305	0,019	0,003	0,020	0,009	0,048	0,353
4	**1**	0,458	0,100	0,555	0,031	0,007	0,033	0,015	0,079	**0,634**
5	**1**	1,670	0,523	2,193	0,107	0,010	0,109	0,047	0,262	**2,464**
6	**1**	1,711	0,460	2,177	0,145	0,021	0,153	0,075	0,373	2,550
7	**1**	3,337	0,758	4,107	0,341	0,058	0,232	0,104	0,678	**4,826**
8	**1**	6,079	1,463	7,590	0,528	0,104	0,502	0,245	1,275	**9,041**
9	**1**	5,704	1,323	7,054	0,586	0,115	0,578	0,268	1,432	**8,746**
10	†	†	†	†	†	†	†	†	†	†
11	†	†	†	†	†	†	†	†	†	†
12	**1**	7,393	1,850	9,287	0,611	0,120	0,590	0,286	1,486	**10,893**
13	**1**	26,927	6,202	33,291	1,336	0,213	1,438	0,600	3,373	37,039
14	**1**	7,717	1,962	9,783	0,678	0,202	0,689	0,298	1,664	**11,619**
15	**1**	17,070	4,380	21,451	1,135	0,216	1,246	0,548	2,928	**24,617**
16	†	†	†	†	†	†	†	†	†	†
17	†	†	†	†	†	†	†	†	†	†

Tabela 10 – Resultados do DPA-G em matrizes gerais. Execução realizada no LoboC.

#	**DPA-G**	**DPA-S**	**Stüben**	**ILU**
1	0,022	1,000	0,753	0,356
2	†	0,618	1,000	0,061
3	1,000	0,711	0,665	0,598
4	0,060	0,855	1,000	0,614
5	0,156	1,000	0,967	0,927
6	1,000	0,778	0,761	0,554
7	0,014	0,843	1,000	0,367
8	0,160	0,308	0,326	1,000
9	0,191	0,379	0,401	1,000
10	†	0,145	0,221	1,000
11	†	0,514	1,000	0,596
12	0,353	0,381	0,460	1,000
13	1,000	0,545	0,345	0,089
14	0,011	1,000	0,268	0,098
15	0,012	0,195	0,156	1,000
16	†	1,000	†	†
17	†	1,000	0,628	†
Média	0,332	0,663	0,622	0,617

Tabela 11 – Normalização dos tempos de execução com matrizes gerais. Execução realizada no LoboC.

Para cada matriz, os resultados foram comparados entre os precondicionadores (não entre matrizes). Devido às distintas origens e perfis de cada matriz, não há um padrão claro nos resultados, mas é possível constatar que em 15 das 17 matrizes os precondicionadores AMG (Tabs. 8, 9 e 10) obtiveram menor número de iterações (em uma delas empatado com o ILU).

O DPA-S foi o único método que convergiu em todos os casos. O Stb não obteve a convergência em apenas um caso, o ILU não obteve convergência em dois casos e o DPA-G não obteve a convergência em cinco. Assim, observa-se que a maior robustez foi apresentada pelo DPA-S. Tanto o DPA-S quanto o Stb alcançaram melhor robustez que o ILU, enquanto que o DPA-G apresentou robustez inferior.

A Tab. 11 trás um resumo dos tempos computacionais dos métodos Stb, DPA-S, DPA-G e ILU normalizados pelo maior tempo obtido em cada matriz (linha), para fins de comparação. O maior valor em cada linha foi igualado a $1,000$ e os demais foram proporcionalmente ajustados. Os casos de não-convergência são exibidos como †.

Pela normalização observa-se que em média os menores tempos foram obtidos pelo DPA-G, com 46.19% de vantagem em relação ao ILU, que apresentou o segundo menor tempo. Os maiores tempos em média foram obtidos pelo DPA-S.

Em 13 das 17 matrizes os precondicionadores AMG obtiveram os menores tempos, sendo que 9 foram obtidos pelo DPA-G. Todas as matrizes que convergiram com o DPA-G o fizeram em uma única iteração.

Em resumo, para este grupo de testes, observa-se:

- Superioridade do DPA-S em robustez e inferioridade em tempo em relação a todos os demais métodos,
- Superioridade do DPA-G em tempo e inferioridade em robustez em relação a todos os demais métodos.

5.2 Grupo 2 - Testes com matrizes estêncil

Em aproximações oriundas do método das diferenças finitas, as matrizes geradas possuem estruturas conhecidas de dependência entre os pontos (Estêncil). O padrão Estêncil é regularmente usado para representar a dependência entre pontos em uma aproximação de derivadas por diferenças finitas (SAAD, 2003) e podem ser expressas através de $2d+1$ pontos para problemas de primeira e segunda ordem de d dimensões, onde para cada linha i da matriz, teremos coeficientes em i, $i+1$, $i-1$, $i+n$, $i-n$, $i+mn$, $i-mn$, $\ldots$; conforme ilustrado na Fig. 17 para problemas tridimensionais, onde n é o número de incógnitas na direção x, m é o número de incógnitas na direção y e p o número de incógnitas na direção z. O sistema linear resultante tem ordem nmp.

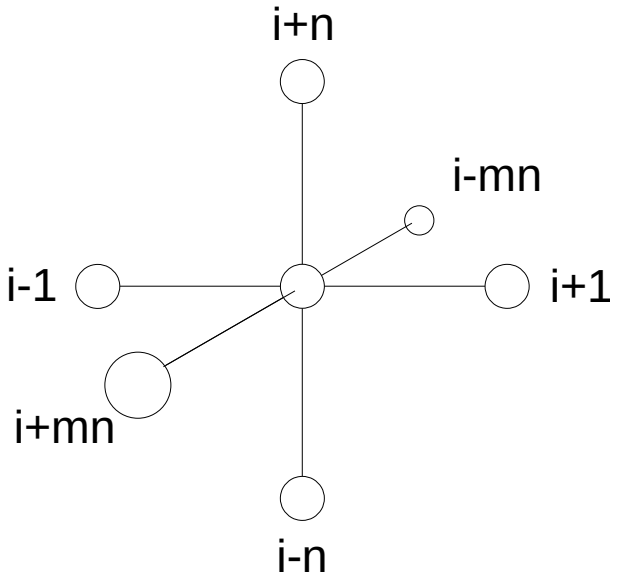


Figura 17 – Estêncil de sete pontos para alguma aproximação por diferenças finitas.

Neste conjunto de testes foram utilizadas matrizes hepta-diagonais que não apresentam a propriedade diagonal dominante, oriundas da discretização por diferen-

	m	DPA-G	Stb	DPA-S	ILU
Total	50	0,60	1,42	1,01	0,80
	100	5,12	11,97	8,31	7,13
	150	19,55	42,66	31,48	25,59
	200	45,51	102,42	69,15	58,44
	250	92,11	230,48	160,00	110,90
	300	170,27	356,67	251,54	199,64
	n	**DPA-G**	**Stb**	**DPA-S**	**ILU**
Iter	50	1	7	9	28
	100	1	7	9	30
	150	2	7	9	30
	200	1	7	9	30
	250	2	7	9	29
	300	2	7	9	29
	n	**DPA-G**	**Stb**	**DPA-S**	**ILU**
Setup	50	0,50	0,64	0,39	0,12
	100	4,20	5,64	3,29	1,02
	150	15,30	20,78	14,02	3,75
	200	39,06	50,87	28,79	8,23
	250	75,19	112,84	67,66	16,54
	300	140,64	180,67	113,93	29,83
	n	**DPA-G**	**Stb**	**DPA-S**	**ILU**
Exec	50	0,10	0,79	0,63	0,68
	100	0,93	6,33	5,02	6,11
	150	4,24	21,88	17,46	21,87
	200	6,31	51,42	40,36	50,22
	250	16,93	117,64	92,34	94,37
	300	29,63	174,37	137,61	169,81

Tabela 12 – Resultados com matrizes estêncil. Execução realizada no LoboC.

ças finitas para problemas tridimensionais regidos pela equação de Poisson. Assim, uma malha com m incógnitas em cada direção resulta em um sistema linear com m^3 equações. Na matriz que representa a malha, cada elemento fora da diagonal possui valor -1 e a diagonal é toda preenchida com o valor fixo igual a 6.

A Tab. 12 apresenta os principais resultados dos testes realizados para matrizes de ordem $1,2 \times 10^5$ ($m = 50$) até ordem $2,7 \times 10^7$ ($m = 300$) incógnitas. Em todos os experimentos o precondicionador Stb utilizou $\varepsilon = 0.2$ e as duas versões do DPA, $\beta = 0.0$. Não foi considerado nenhum nível de *fill-in* no precondicionador ILU, ou seja, foi utilizado ILU(0). Esta escolha está baseada em experimentos preliminares que indicaram menor tempo de processamento para o precondicionador ILU(0) neste conjunto de testes. Na Tab. 12 encontramos o tempo total de execução (**Total**), o número de iterações (**Iter**), o tempo necessário para preparo dos precondicionadores (**Setup**) e o tempo de solução do sistema linear até atingir a tolerância fixada (**Exec**).

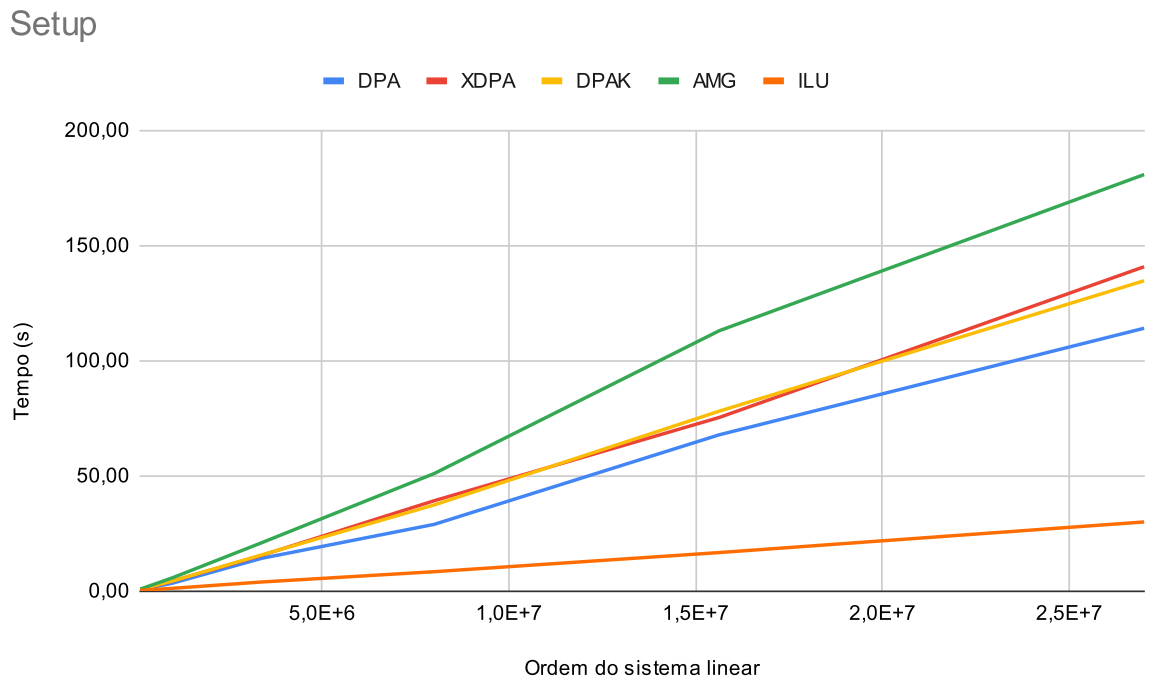


Figura 18 – Tempo de *setup* com matrizes estêncil.

Em todos os tamanhos de malha, os tempos totais (**Total**) foram obtidos, do menor para o maior, pelo DPA-G, ILU, DPA-S e Stb. Observe que, apesar do tempo de **Setup** do ILU ter sido menor, o tempo de execução (**Exec**) do DPA-G foi menor suficiente para compensar o maior tempo do **Setup**.

Os gráfico das Fig. 18 a Fig. 20 exibe, respectivamente, o tempos de **Setup**, **Exec** e **Total**. Nos gráficos, os tempos estão em função da ordem do sistema linear, que é igual a m^3 em uma malha de tamanho m em cada dimensão.

Observando-se os gráficos é possível notar um crescimento linear dos tempos em relação à ordem da matriz, o que indica boa escalabilidade dos métodos na extensão dos tamanhos testados. Também é possível observar que todos os métodos possuem uma constância no número de iterações em relação à ordem da matriz. O DPA-G exibiu os menores números de iterações, e o ILU, os maiores, conforme pode ser observado na Tab. 12.

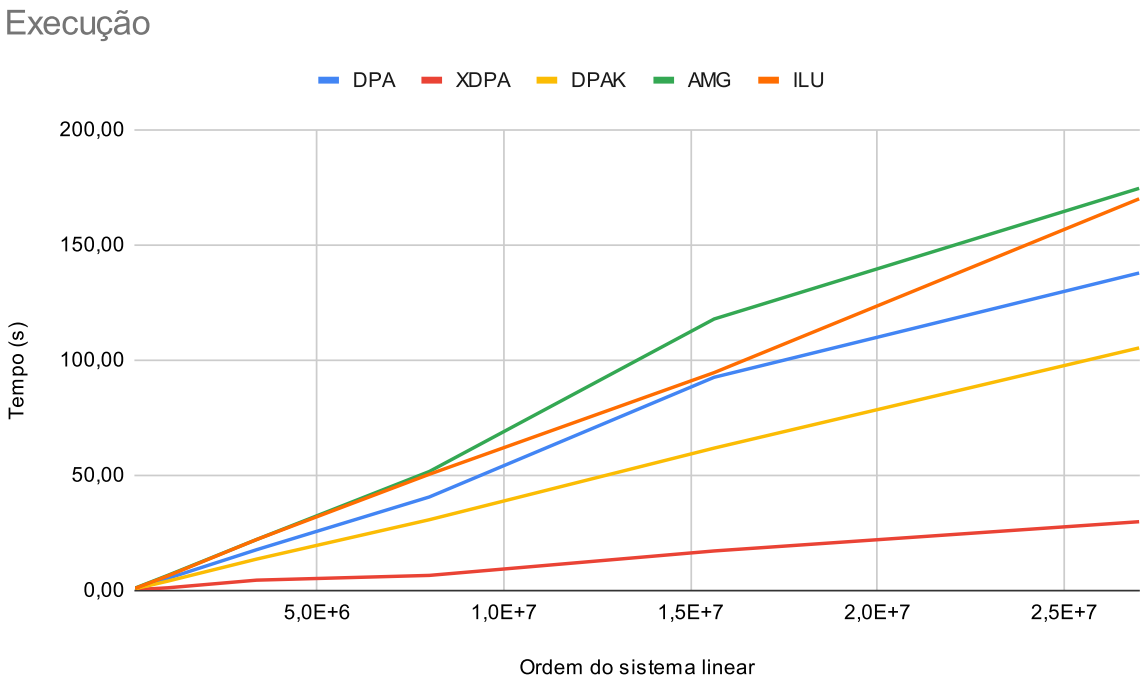


Figura 19 – Tempo de execução com matrizes estêncil.

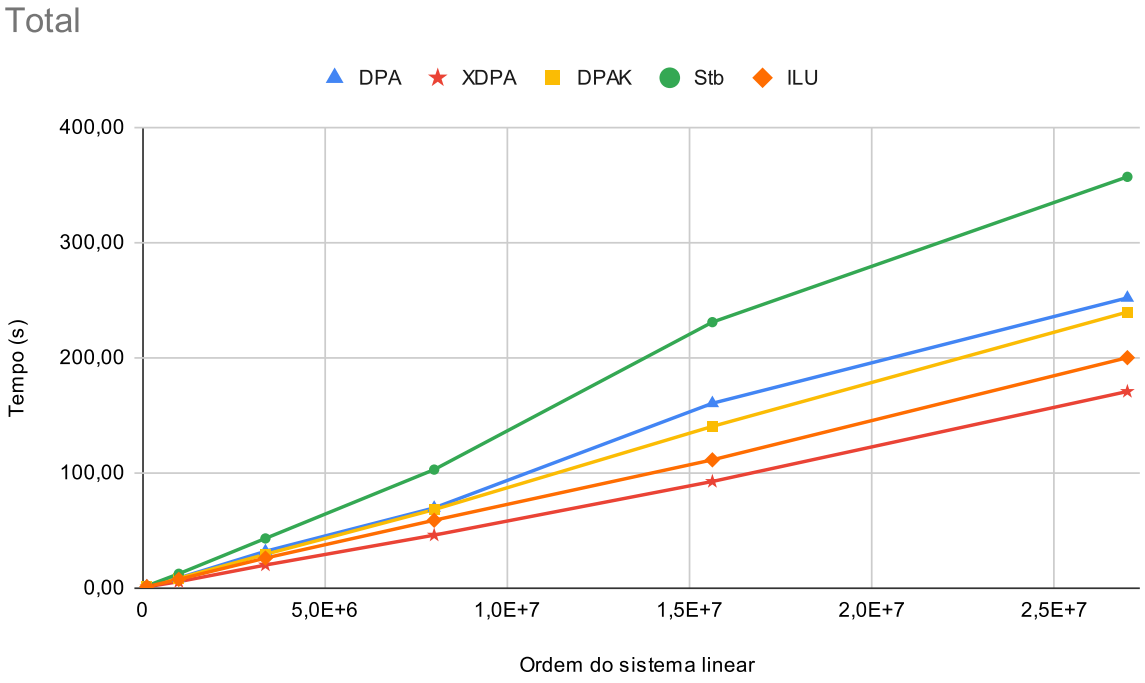


Figura 20 – Tempo total com matrizes estêncil.

5.3 Grupo 3 - Testes com matrizes oriundas de problemas regidos pelo sistemas de equações de Euler

Para a realização deste conjunto de testes, o precondicionador AMG foi incluído no código FEMCodes[5] em desenvolvimento no Laboratório de Otimização e Modelagem Computacional da UFES - LabOtim. Neste grupo de matrizes executamos experimentos similares aos realizados nos Grupos 1 e 2 com matrizes oriundas de um problema regido pela equação de Euler discretizada pelo método dos elementos finitos chamado Explosão (TORO, 2013), onde a solução exibe uma onda de choque circular viajando do centro para o contorno de um domínio quadrado, uma superfície de contato circular viajando na mesma direção e uma rarefação circular viajando para a origem. A Fig.21 apresenta o domínio do problema e a Fig. 22 a distribuição da densidade considerando o precondicionador DPA-S.

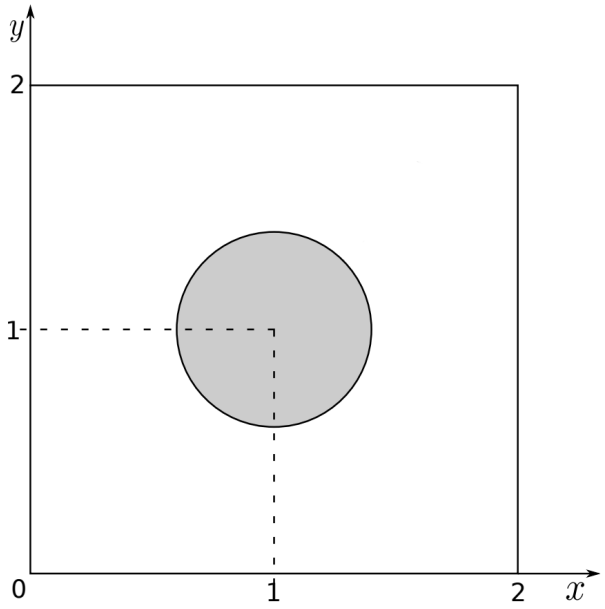


Figura 21 – Ilustração do problema Explosão. É indicada na área sombreada a expansão do choque circular sobre o domínio em um determinado momento. – Reproduzida de Bento (2018).

Foram comparados os precondicionadores ILU, AMG com esquema clássico de engrossamento (Stb) e AMG com *double pairwise aggregation* (DPA-S). As execuções foram feitas tanto considerando o reordenamento do sistema resultante pelo método de *Reverse Cuthill-McKee* (RCM) (GEORGE; LIU, 1981) quanto sem considerar.

Na execução dos testes com ILU foi utilizado $p = 0$, uma vez que este valor obteve os melhores resultados em Muller (2017) para problemas regidos pela equação de Euler. O AMG foi executado com $\varepsilon = 0.2$, restrição agressiva no caso do Stb e

[5] <https://github.com/mod-comp-ufes/FEM_CODES>

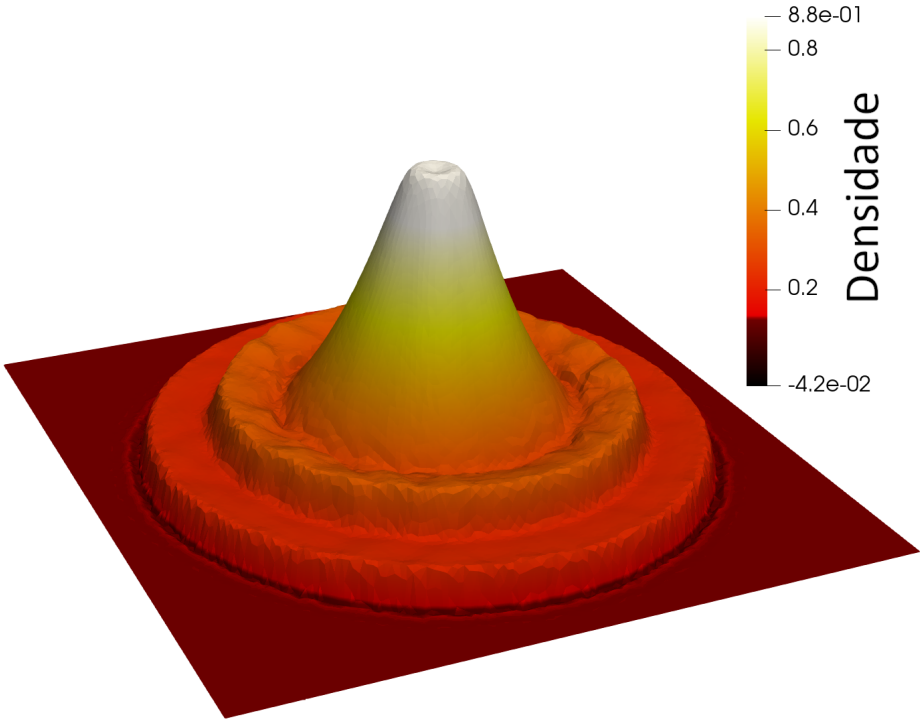


Figura 22 – Problema explosão solucionado pelo DPA-S.

$\beta = 0.0$ nos casos do DPA-S. Ambos com um nível de descida, parâmetro do SOR $\omega = 1.5$ e nenhum truncamento de interpolação.

O código implementado no FEMCodes para o problema Explosão simula um problema com um conjunto de passos de tempo fixos iguais a $\Delta t = 1 \times 10^{-3}$ até atingir o tempo final de $0,25$ segundos. Detalhes podem ser consultados em Bento (2018). A cada passo de tempo é considerado um conjunto de iterações não-lineares, resultando em 2.500 iterações não-lineares, e em cada uma delas é necessário resolver um sistema linear de equações. Nos experimentos consideramos 3 tamanhos de malhas: malha mais grossa, com 92.564 elementos e 46.683 nós (183.533 incógnitas); malha intermediária, com 144.708 elementos e 72.855 nós (287.420 incógnitas) e malha mais refinada 258.018 elementos e 129.678 nós (513.368 incógnitas).

Os resultados são exibidos na Tab. 13. As colunas indicam, da esquerda para a direita, a ordem do sistema (número de incógnitas), o tempo total de processamento, o número de iterações lineares, o número médio de iterações lineares a cada iteração não-linear, o tempo médio por iteração não-linear, e o tempo médio por iteração linear. A tabela apresenta os resultados considerando os casos com e sem reordenamento.

Em todos os casos o precondicionador ILU apresentou os menores tempos totais de execução comparados aos tempos obtidos pelos precondicionadores *Multigrid*, sendo os tempos do Stb menores que os tempos do DPA-S.

Tabela 13 – Resultados com equações de Euler. Execução realizada no LCAD.

Sem reordenamento

ordem	tempo	iter	iter/gmres	tempo/gmres	tempo/iteração
			ILU(0)		
183.532	2.174,75	9.995	4,00	0,87	0,22
287.420	3.551,23	10.018	4,01	1,42	0,35
513.368	6.613,89	9.953	3,98	2,64	0,66
			Stb		
183.532	3.568,45	2.500	1,00	1,43	1,43
287.420	5.907,53	2.500	1,00	2,36	2,36
513.368	11.012,40	2.500	1,00	4,40	4,40
			DPA-S		
183.532	12.889,43	46.879	18,75	5,15	0,27
287.420	19.350,10	35.122	14,05	7,74	0,55
513.368	38.890,03	43.725	17,49	15,55	0,89

Com reordenamento

ordem	tempo	iter	iter/gmres	tempo/gmres	tempo/iter
			ILU(0)		
183.532	1.225,13	13.309	5,32	0,49	0,09
287.420	1.947,53	12.668	5,07	0,77	0,15
513.368	3.489,13	11.300	4,52	1,38	0,31
			Stb		
183.532	3.557,60	2.500	1,00	1,42	1,42
287.420	5.854,89	2.500	1,00	2,34	2,34
513.368	10.879,27	2.500	1,00	4,35	4,35
			DPA-S		
183.532	9.676,39	47.482	18,99	3,87	0,20
287.420	13.985,66	34.519	13,81	5,59	0,40
513.368	27.341,64	42.769	17,11	10,93	0,64

Enquanto o ILU teve os tempos expressivamente reduzidos com o uso do reordenamento, os precondicionadores *Multigrid* com Stb e DPA-S foram pouco afetados.

O precondicionador Stb convergiu em apenas 1 iteração linear em cada iteração não linear (**iter/gmres**) em todos os casos testados, enquanto que o precondicionador DPA-S necessitou de 16,76 iterações em média para o caso sem reordenamento e 16,63 para o caso com reordenamento. Já para o precondicionador ILU o número médio de iterações para o caso sem reordenamento é de 3,99 e para o caso com reordenamento é de 4,97. Desta forma, observamos que o precondicionador Stb diminui o número de iterações quando comparado com o precondicionador ILU, entretanto o tempo computacional por iteração é maior como pode ser observado na coluna **tempo/iteração**. Este comportamento pode ser explicado uma vez que o **Setup** dos precondicionadores AMG são mais caros computacionalmente que o **Setup** do precondicionador ILU, conforme observado nos Grupos 1 e 2. O precondicionador DPA-S se destaca entretanto, na redução do tempo por iteração quando comparado com o precondicionador

Stb, já que o seu **Setup** é mais barato computacionalmente.

Os gráficos das Figs. 23 e 24 exibem o aumento gradativo do tempo computacional em função da ordem do sistema linear, respectivamente com e sem reordenamento. Em ambos os casos, a curva de crescimento do precondicionador DPA-S possui maior inclinação em relação aos demais, indicando menor escalabilidade.

Para este grupo de testes os precondicionadores *Multigrid* implementados não superaram o desempenho obtido pelo precondicionador ILU. No entanto, em termos de taxa de convergência, o precondicionador Stb se destaca por ter convergido com apenas uma iteração em todos os casos (com e sem ordenamento).

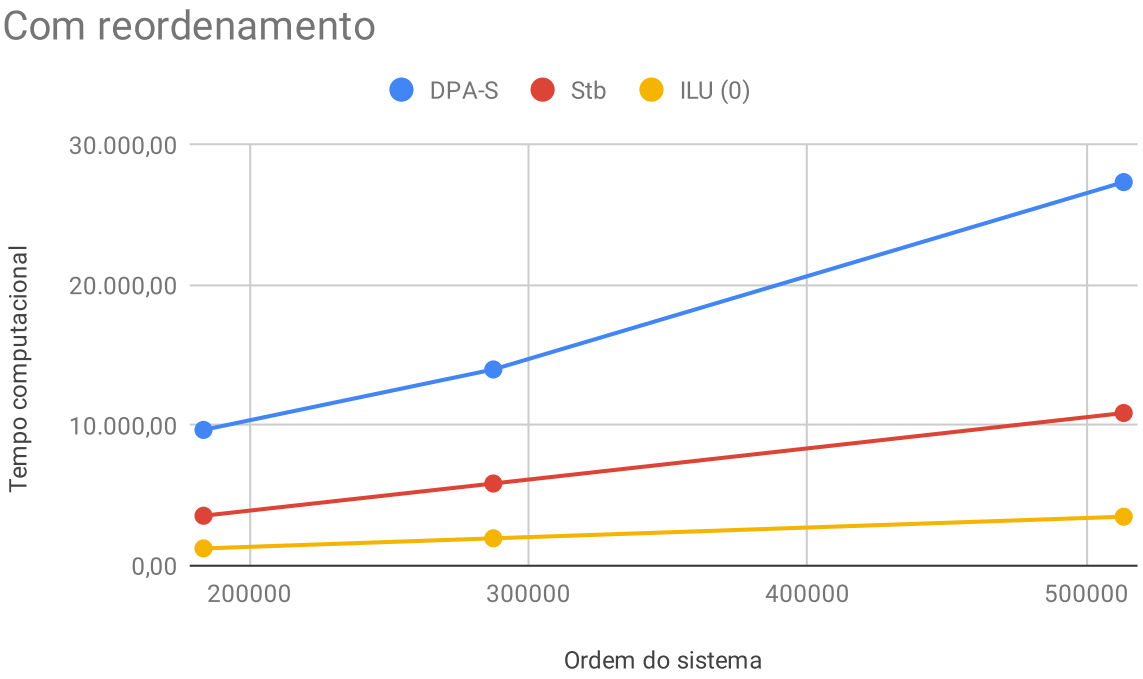


Figura 23 – Variação dos tempos nas equações de Euler com uso de reordenamento

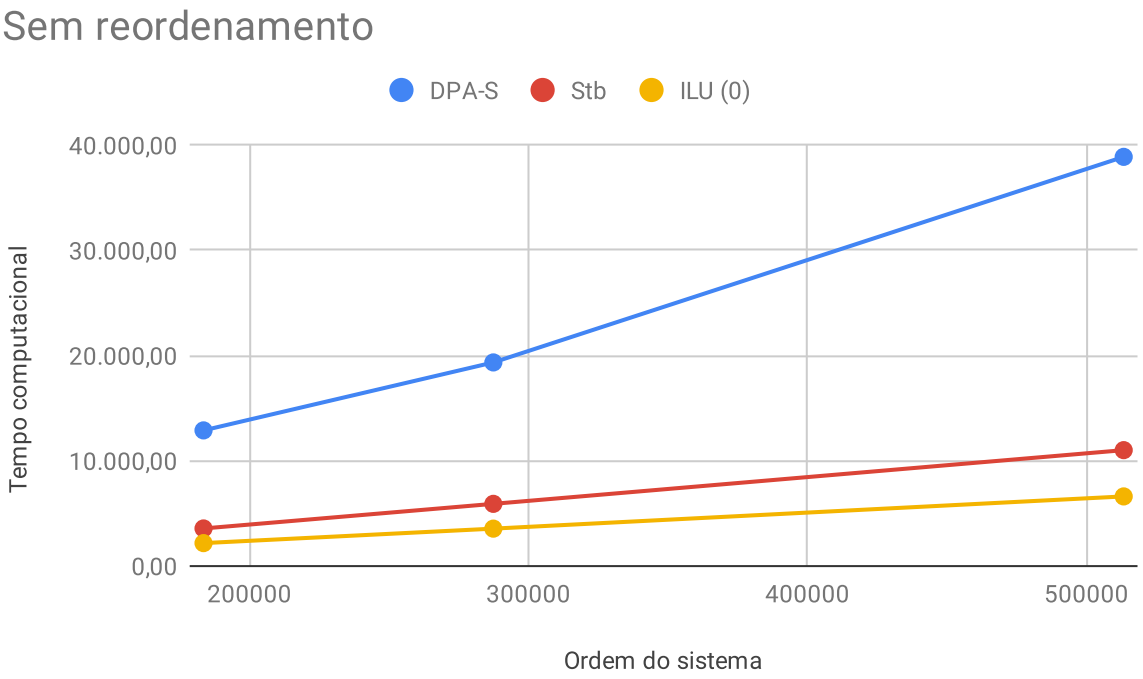


Figura 24 – Variação dos tempos nas equações de Euler sem uso de reordenamento

6 Conclusões

Neste trabalho investigamos diversas estratégias de solução de sistemas lineares no contexto dos métodos *Multigrid*. Especificamente, enfatizamos o uso de sua versão algébrica (AMG), que generaliza a estratégia *Multigrid* para problemas sobre os quais não temos informações a respeito do domínio. O AMG utiliza diferentes estratégias de engrossamento e interpolação, que são componentes fundamentais do *Multigrid* e afetam suas taxas de tempo e convergência, dentre as quais foram implementadas as estratégias clássicas propostas em Ruge e Stüben (1987) e o *Double Pairwise Aggregation* (NOTAY, 2006). Além da aplicação do *Multigrid* como *solver*, também investigamos sua utilização como precondicionador do método GMRES.

Também foram discutidas as ordens de complexidade dos métodos de engrossamento, e por fim, foram analisados os resultados experimentais comparativos do *Multigrid* Algébrico como precondicionador obtidos com matrizes gerais, matrizes estêncil e matrizes oriundas da solução de problemas regidos pelas equações de Euler discretizadas pelo método dos elementos finitos.

Com as implementações desenvolvidas neste trabalho aprimoramos tanto a robustez quanto os tempos de convergência do método GMRES quando executado com o AMG como precondicionador. Na análise dos resultados com matrizes estêncil o precondicionador *Multigrid* com uso do *Double Pairwise Aggregation* (DPA-S e DPA-G) obteve tempos consistentemente mais baixos que o *Multigrid* com esquemas clássicos de engrossamento (Stb), assim como com o precondicionador ILU. Dentre os resultados com matrizes gerais, das 17 matrizes, 13 obtiveram resultados melhores com os precondicionadores *Multigrid* do que com o precondicionador ILU, e destas, 12 através do *Double Pairwise Aggregation* (DPA-S e DPA-G).

Porém, nos experimentos com matrizes de Euler o AMG não se mostrou adequado, especialmente quando utilizando o DPA. Além de ter apresentado tempos piores em todos os casos, as curvas de aumento do tempo em relação ao tamanho do problema foram mais inclinadas, indicando menor escalabilidade. O AMG com esquemas clássicos de restrição apresentou maior robustez, pois reduziu para 1 o número de iterações, demonstrando uma taxa de convergência superior. Porém, esta redução também apresentou custo computacional consideravelmente superior, não compensado pelo ganho no número de iterações.

Assim, identifica-se a necessidade de investigar mais estratégias para adequar o *Multigrid* ao contexto de matrizes oriundas de discretização do método dos elementos finitos. Nosso estudo do método *Multigrid* ainda se encontra em sua fase inicial.

Ainda há muito a se explorar neste assunto. Para trabalhos futuros, são sugeridos:

- Implementar o Ciclo-K (YANG et al., 2011), que é o principal esquema utilizado em conjunto com o *Double Pairwise Aggregation*, onde são realizadas acelerações por subespaços de Krylov através de um método tal como o *Flexible Inner-Outer Preconditioned* GMRES (FGMRES) (SAAD, 1993) ou a variante recursiva do GMRES (GMRESR) (VORST; VUIK, 1994);
- Testar outros algoritmos de *matching* no *Double Pairwise Aggregation*, por exemplo, o algoritmo de Preis, disponível no BootCMatch (DAMBRA; FILIPPONE; VASSILEVSKI, 2018);
- Implementar e comparar novas estratégias de engrossamento, dentre as quais se destacam o *Compatible Relaxation* (DAMBRA; FILIPPONE; VASSILEVSKI, 2018), as agregações suavizadas (VANĚK; MANDEL; BREZINA, 1996), o *Hybrid Modified Independent Set* (HMIS) e o *Parallel Maximal Independent Set* (PMIS) (STERCK; YANG; HEYS, 2006);
- Testar de forma mais exaustiva os valores dos parâmetros dos métodos apresentados, variando-se a quantidade de níveis de descida do ciclo-V, considerando o uso de truncamento de interpolação e o reordenamento de matrizes em casos gerais;
- Considerar diferentes esquemas para a realização da multiplicação tripla de matrizes, como as sugeridas por (MCCOURT; SMITH; ZHANG, 2015);
- Investigar formas de paralelização para os métodos de engrossamento, interpolação, operação de Galerkin ou relaxação.

Referências

BARRETT, R.; BERRY, M.; CHAN, T. F.; DEMMEL, J.; DONATO, J.; DONGARRA, J.; EIJKHOUT, V.; POZO, R.; ROMINE, C.; VORST, H. Van der. *Templates for the solution of linear systems: building blocks for iterative methods*. [S.l.]: SIAM, 1994. Citado na página 48.

BENTO, S. S. *Nonlinear Multiscale Viscosity Methods and Time Integration Schemes for Solving Compressible Euler Equations*. 90 f. Tese (Doutorado em Ciência da Computação) — UFES, Vitória, 2018. Citado 2 vezes nas páginas 76 e 77.

BRANDT, A. Algebraic multigrid theory: The symmetric case. *Applied mathematics and computation*, Elsevier, v. 19, n. 1-4, p. 23–56, 1986. Citado na página 15.

BRANDT, A. General highly accurate algebraic coarsening schemes. In: CITESEER. *Proceedings of the Ninth Copper Mountain Conference on Multigrid Methods, Copper Mountain*. [S.l.], 1999. Citado na página 15.

BRANDT, A.; MCCORMICK, S.; RUGE, J. Algebraic multigrid (amg) for automatic multigrid solutions with application to geodetic computations, report. *Institute for Computational Studies, Fort Collins, CO*, 1982. Citado na página 15.

BRANDT, A.; MCCORUICK, S.; HUGE, J. Algebraic multigrid (amg) for sparse matrix equations. *Sparsity and its Applications*, v. 257, 1985. Citado 2 vezes nas páginas 15 e 34.

BREZINSKI, C.; WUYTACK, L. *Numerical analysis: Historical developments in the 20th century*. [S.l.]: Elsevier, 2012. Citado 2 vezes nas páginas 19 e 20.

BRIGGS, W. L.; HENSON, V. E.; MCCORMICK, S. F. *A multigrid tutorial*. [S.l.]: Siam, 2000. v. 72. Citado 7 vezes nas páginas 14, 15, 18, 23, 24, 28 e 30.

BROUGHTON, S. A.; BRYAN, K. Discrete fourier analysis and wavelets. In: *Applications to signal and image processing*. [S.l.]: Wiley Online Library, 2009. Citado na página 23.

CARRION, M. T. P. *An Experimental Study of the Geometric and Algebraic Multigrid Strategies*. 63 f. Dissertação (Mestrado em Informática) — UFES, Vitória, 2016. Citado 10 vezes nas páginas 9, 23, 24, 25, 28, 30, 32, 33, 40 e 48.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. *Introduction to algorithms*. [S.l.]: MIT press, 2009. Citado na página 51.

DIESTEL, R. *Graph Theory*. [S.l.]: Springer Berlin Heidelberg, 2017. v. 173. Citado 3 vezes nas páginas 44, 45 e 46.

DAMBRA, P.; FILIPPONE, S.; VASSILEVSKI, P. S. Bootcmatch: a software package for bootstrap amg based on graph weighted matching. *ACM Transactions on Mathematical Software (TOMS)*, ACM New York, NY, USA, v. 44, n. 4, p. 1–25, 2018. Citado 3 vezes nas páginas 14, 16 e 82.

FALGOUT, R. D. An introduction to algebraic multigrid computing. *Computing in science & engineering*, IEEE Computer Society, v. 8, n. 6, p. 24, 2006. Citado na página 30.

FEDORENKO, R. P. A relaxation method for solving elliptic difference equations. *USSR Computational Mathematics and Mathematical Physics*, Elsevier, v. 1, n. 4, p. 1092–1096, 1962. Citado na página 30.

FILHO, F. F. C. *Algoritmos Numericos*. [S.l.]: LTC, 2007. v. 1. Citado 4 vezes nas páginas 18, 20, 21 e 22.

FRANCO, N. M. B. F. *Cálculo Numérico*. [S.l.]: Pearson Universidades, 2006. v. 1. Citado 4 vezes nas páginas 18, 20, 21 e 22.

GEORGE, A.; LIU, J. W. *Computer solution of large sparse positive definite*. [S.l.]: Prentice Hall Professional Technical Reference, 1981. Citado na página 76.

HEMKER, P. W. A note on defect correction processes with an approximate inverse of deficient rank. *Journal of Computational and Applied Mathematics*, Elsevier, v. 8, n. 2, p. 137–139, 1982. Citado na página 34.

KINCAID, D. R. Celebrating fifty years of david m. youngs successive overrelaxation method. In: *Numerical mathematics and advanced applications*. [S.l.]: Springer, 2004. p. 549–558. Citado na página 21.

LEE, C.-O.; CAI, X.-C.; KEYES, D. E.; KIM, H. H.; KLAWONN, A.; PARK, E.-J.; WIDLUND, O. B. *Domain Decomposition Methods in Science and Engineering XXIII*. [S.l.]: Springer, 2017. Citado na página 19.

MCADAMS, A.; SIFAKIS, E.; TERAN, J. A parallel multigrid poisson solver for fluids simulation on large grids. In: EUROGRAPHICS ASSOCIATION. *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. [S.l.], 2010. p. 65–74. Citado na página 59.

MCCORMICK, S. F. *Multigrid methods*. [S.l.]: SIAM, 1987. Citado na página 15.

MCCOURT, M.; SMITH, B.; ZHANG, H. Sparse matrix-matrix products executed through coloring. *SIAM Journal on Matrix Analysis and Applications*, SIAM, v. 36, n. 1, p. 90–109, 2015. Citado na página 82.

MOHAMED, K. S. *Neuromorphic Computing and Beyond: Parallel, Approximation, Near Memory, and Quantum*. [S.l.]: Springer Nature, 2020. Citado na página 20.

MULLER, L. K. *Estudo comparativo de precondicionadores locais e globais no contexto do método dos elementos finitos*. 85 f. Dissertação (Mestrado em Informática) — UFES, Vitória, 2017. Citado na página 76.

NOTAY, Y. Aggregation-based algebraic multilevel preconditioning. *SIAM journal on matrix analysis and applications*, SIAM, v. 27, n. 4, p. 998–1018, 2006. Citado 6 vezes nas páginas 15, 16, 44, 45, 47 e 81.

NOTAY, Y. An aggregation-based algebraic multigrid method. *Electronic transactions on numerical analysis*, v. 37, n. 6, p. 123–146, 2010. Citado na página 16.

PARK, J.; SMELYANSKIY, M.; YANG, U. M.; MUDIGERE, D.; DUBEY, P. High-performance algebraic multigrid solver optimized for multi-core based distributed parallel systems. In: ACM. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. [S.l.], 2015. p. 54. Citado na página 30.

RUGE, J. W.; STÜBEN, K. Algebraic multigrid. In: *Multigrid methods*. [S.l.]: SIAM, 1987. p. 73–130. Citado 3 vezes nas páginas 15, 37 e 81.

RUGGIERO, V. L. D. R. L. M. A. G. *Cálculo Numerico*. [S.l.]: Pearson Makron Books, 1996. v. 1. Citado 3 vezes nas páginas 20, 21 e 22.

SAAD, Y. A flexible inner-outer preconditioned gmres algorithm. *SIAM Journal on Scientific Computing*, SIAM, v. 14, n. 2, p. 461–469, 1993. Citado na página 82.

SAAD, Y. *Iterative Methods for Sparse Linear Systems*. Minneapolis: Society for Industrial and Applied Mathematics, 2003. Citado 11 vezes nas páginas 14, 16, 17, 18, 19, 20, 21, 30, 54, 58 e 72.

STEINBACH, O.; YANG, H. Comparison of algebraic multigrid methods for an adaptive space–time finite-element discretization of the heat equation in 3d and 4d. *Numerical Linear Algebra with Applications*, Wiley Online Library, v. 25, n. 3, p. e2143, 2018. Citado na página 15.

STERCK, H. D.; FALGOUT, R. D.; NOLTING, J. W.; YANG, U. M. Distance-two interpolation for parallel algebraic multigrid. *Numerical Linear Algebra with Applications*, Wiley Online Library, v. 15, n. 2-3, p. 115–139, 2008. Citado 4 vezes nas páginas 10, 40, 42 e 44.

STERCK, H. D.; MILLER, K.; SANDERS, G.; WINLAW, M. Recursively accelerated multilevel aggregation for markov chains. *SIAM Journal on Scientific Computing*, SIAM, v. 32, n. 3, p. 1652–1671, 2010. Citado 2 vezes nas páginas 16 e 48.

STERCK, H. D.; YANG, U. M.; HEYS, J. J. Reducing complexity in parallel algebraic multigrid preconditioners. *SIAM Journal on Matrix Analysis and Applications*, SIAM, v. 27, n. 4, p. 1019–1039, 2006. Citado na página 82.

STUBEN, K. A review of algebraic multigrid. *CMD Report*, 1999. Citado 2 vezes nas páginas 14 e 15.

STUBEN, K. Algebraic multigrid (amg): an introduction with applications. *Multigrid*, Academic Press, 2000. Citado 11 vezes nas páginas 9, 10, 15, 16, 30, 36, 38, 39, 41, 44 e 45.

STÜBEN, K. An introduction to algebraic multigrid. *Multigrid*, p. 413–532, 2001. Citado 2 vezes nas páginas 42 e 45.

STÜBEN, K. A review of algebraic multigrid. In: *Numerical Analysis: Historical Developments in the 20th Century*. [S.l.]: Elsevier, 2001. p. 331–359. Citado na página 34.

TORO, E. F. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. [S.l.]: Springer Science & Business Media, 2013. Citado na página 76.

TROTTENBERG, U.; OOSTERLEE, C. W.; SCHULLER, A. *Multigrid*. [S.l.]: Elsevier, 2000. Citado na página 15.

VAN, P.; BREZINA, M.; MANDEL, J. et al. Convergence of algebraic multigrid based on smoothed aggregation. *Numerische Mathematik*, Springer, v. 88, n. 3, p. 559–579, 2001. Citado na página 15.

VANĚK, P.; MANDEL, J.; BREZINA, M. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, Springer, v. 56, n. 3, p. 179–196, 1996. Citado 2 vezes nas páginas 15 e 82.

VORST, H. A. Van der; VUIK, C. Gmresr: a family of nested gmres methods. *Numerical linear algebra with applications*, Wiley Online Library, v. 1, n. 4, p. 369–386, 1994. Citado na página 82.

YANG, J.; LI, Z.; CAI, Y.; ZHOU, Q. Powerrush: A linear simulator for power grid. In: IEEE. *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. [S.l.], 2011. p. 482–487. Citado na página 82.

YANG, U. M. Parallel algebraic multigrid methodshigh performance preconditioners. In: *Numerical solution of partial differential equations on parallel computers*. [S.l.]: Springer, 2006. p. 209–236. Citado 5 vezes nas páginas 14, 15, 23, 30 e 54.

YANG, U. M. On long-range interpolation operators for aggressive coarsening. *Numerical Linear Algebra with Applications*, Wiley Online Library, v. 17, n. 2-3, p. 453–472, 2010. Citado na página 43.