Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 8, n. 1, 2021.

Trabalho apresentado no XL CNMAC, Evento Virtual - Co-organizado pela Universidade do Mato Grosso do Sul (UFMS).

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Esquema de engrossamento com agregação dupla de pares para o precondicionador *Multigrid* Algébrico

Henrique Gomes de Jesus[1]
Maria Claudia Silva Boeres[2]
Lucia Catabriga[3]
Universidade Federal do Espírito Santo

Resumo. Este trabalho se destina à análise empírica do método *multigrid* algébrico (AMG) como precondicionador do método do Resíduo Mínimo Generalizado (GMRES) utilizando uma estratégia de engrossamento conhecida como agregação dupla de pares (DPA, do inglês - *Double Pairwise Aggregation*). Esta estratégia consiste em aplicar um algoritmo de *matching* em grafos duas vezes em cada nível da hierarquia a fim de produzir os operadores de restrição e interpolação. O esquema de engrossamento DPA é testado em um conjunto de matrizes do repositório de matrizes esparsas *SuiteSparse Matrix Collection*, relacionadas a variadas aplicações. O AMG como precondicionador, empregando tanto a técnica de engrossamento DPA quanto as clássicas, é comparado ao precondicionador ILU, ressaltando vantagens e desvantagens de cada escolha.

Palavras-chave. Métodos *multigrid*. *Multigrid* algébrico. *Double Pairwise Aggregation*. Métodos iterativos. Precondicionadores.

1 Introdução

Neste trabalho, utilizamos o Método do Resíduo Mínimo Generalizado (GMRES, do inglês): um método de solução de sistemas lineares assimétricos que aproxima a solução através do vetor de resíduo mínimo em um subespaço de Krylov [14]. Embora os métodos não estacionários sejam mais robustos que os estacionários, por convergirem com menos iterações, cada iteração requer mais operações de ponto flutuante, e portanto, sua robustez pode ser ampliada ainda mais por meio do uso de técnicas de precondicionamento. Tais técnicas consistem em modificar o sistema original $Au = f$, transformando-o num sistema equivalente, ou seja, que possui a mesma solução, mas que é melhor condicionado, ampliando a taxa de convergência e diminuindo o número de iterações. No precondicionamento à esquerda, o sistema precondicionado pode ser representado por $M^{-1}Au = M^{-1}f$, onde a matriz de precondicionamento M deve ser próxima a A, mas suficientemente simples de se construir.

A cada iteração i, o método GMRES executa i produtos escalares, i combinações lineares e 1 produto matriz-vetor. Deste modo, a ação do precondicionador no método iterativo ocorre no produto matriz vetor $v = M^{-1}Az$, sendo calculado em dois passos: No primeiro o produto matriz-vetor original é calculado ($w = Az$) e no segundo o sistema linear trivial $Mv = w$ é resolvido por algum método de baixa complexidade.

Já os métodos iterativos estacionários [14], ou *relaxações*, realizam uma sequência de iterações com um número bem menor de operações de ponto flutuante, entretanto podem convergir muito

[1]henrique.g.jesus@aluno.ufes.br
[2]boeres@inf.ufes.br
[3]luciac@inf.ufes.br

 © 2021 SBMAC

lentamente. Os métodos de relaxação podem ser melhorados através do emprego de uma estratégia conhecida como *Multigrid* [3], que transforma o problema em outro com menos variáveis, onde a execução do método de solução é mais barata. Chamamos esta nova versão do problema de "discretização grosseira", em oposição à original, dita "refinada". A solução encontrada no domínio grosseiro pode, por fim, ser transferida de volta ao domínio refinado original.

Os métodos *Multigrid* podem ser divididos em duas categorias: *Multigrid* Geométrico (GMG) e *Multigrid* Algébrico (AMG) [1]. Enquanto cada método GMG é construído a partir de um conjunto de discretizações de um domínio específico, 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*.

O AMG foi introduzido nos anos 80 [1], quando o princípio de Galerkin e as transferências entre discretizações baseadas em produtos matriz-vetor foram implementados no GMG com o objetivo de aumentar sua robustez [21]. Na metade dos anos 90 o interesse pelo AMG se intensificou tanto pelo aumento da complexidade geométrica das aplicações, quanto pela necessidade de ser disponibilizado um sistema *Multigrid* comercial [16]. Nesta época foram publicadas referências importantes para o estudo e compreensão do AMG, das quais destacamos [3, 9, 17, 19].

A fim de produzir as matrizes que existirão na versão reduzida do problema é utilizado um esquema de engrossamento, que decide quais variáveis existirão na fase grosseira do problema. Diversas estratégias já foram desenvolvidas, como as estratégias clássicas propostas em [17], o *Hybrid Modified Independent Set* (HMIS), o *Parallel Maximal Independent Set* (PMIS) [15], as agregações suavizadas [20], a *Compatible Relaxation* [2] e a Agregação Dupla por Pares (DPA) [11], apenas para citar algumas.

Neste trabalho, pretendemos apresentar um estudo do método AMG, seu esquema de engrossamento DPA e seu uso como precondicionador, bem como resultados experimentais comparativos na solução de matrizes gerais. Assim, as seções são organizadas como se segue: na Seção 2 é introduzido o método AMG como precondicionador do GMRES empregando o esquema de engrossamento DPA, na Seção 3 são apresentados os resultados numéricos obtidos utilizando um conjunto de matrizes do repositório de matrizes esparsas *SuiteSparse Matrix Collection* [4] e, na Seção 4, é feita uma análise dos resultados, destacando vantagens e desvantagens em cada método testado.

2 Precondicionador *Multigrid* Algébrico com DPA

O método *Multigrid* Algébrico acelera a convergência de métodos iterativos transformando o problema em outro com menos variáveis. Quando isso ocorre, dizemos que o problema foi discretizado em um domínio mais grosseiro. Este procedimento pode ser aplicado recursivamente como ilustrado na Fig. 1, onde o nível 1 representa a discretização original, mais refinada, e os níveis 2 e 3 representam discretizações sucessivamente mais grosseiras. Deste modo, o problema pode ser solucionado no domínio mais grosseiro, onde a execução das iterações do método de solução é naturalmente mais barata, e a solução pode ser levada de volta ao domínio mais refinado [3]. Tais estratégias se tornaram populares por, em tese, possuírem complexidade próxima da linear [12].

A transferência de um vetor do domínio refinado para o grosseiro é chamada de restrição e a transferência de um vetor do domínio grosseiro para o refinado é chamada de interpolação ou prolongação. Uma iteração do método *Multigrid* é caracterizada por sucessivas restrições e interpolações, que podem ser executadas em variados esquemas. Neste trabalho adotamos o esquema denominado Ciclo V. Nele, a solução é transferida até o domínio mais grosseiro e, então, de volta até o mais refinado, aplicando um número parametrizado de relaxações em cada nível. Ao utilizar o AMG como precondicionador, em cada produto matriz-vetor é executado um único Ciclo V do *Multigrid* no sistema $Mv = w$ seguindo as ideias descritas em [21] e [8].

O processo de restrições e interpolações define uma hierarquia de discretizações dos dados de

 © 2021 SBMAC

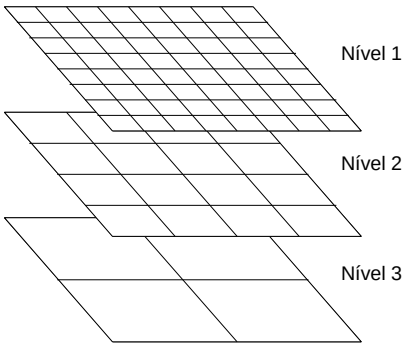

Figura 1: Hierarquia de discretizações.

um sistema. Considere o sistema original dado por $A^h u^h = f^h$ e a a versão grosseira da matriz A^h obtida através de uma operação de Galerkin [7] dada por:

$$A^{2h} = R^h A^h P^h \tag{1}$$

onde o índice sobrescrito h de cada variável indica o nível (ou domínio) ao qual pertence na hierarquia de discretizações. R^h e P^h são os operadores de interpolação e restrição, com $R^h = (P^h)^T$.

A utilização do método *Multigrid* em sistemas lineares definidos a partir de matrizes gerais – advindas de aplicações diversas – demanda uma fase de pré-processamento chamada de *setup*, onde os operadores R^h e P^h são construídos. Nessa fase, é empregado um esquema de engrossamento entre níveis, que visa decidir quais variáveis serão criadas nos domínios mais grosseiros e suas relações de dependência com as variáveis dos domínios mais refinados [18].

As estratégias clássicas de engrossamento propostas por Stüben [17] se baseiam em aplicar heurísticas de agregação de vértices em grafos que representam a matriz de coeficientes do sistema linear. Quando se utiliza o esquema DPA [11], este processo é realizado através de um algoritmo de *matching* [6]. A Fig. 2 ilustra um processo de *matching* e o engrossamento a ele correspondente.

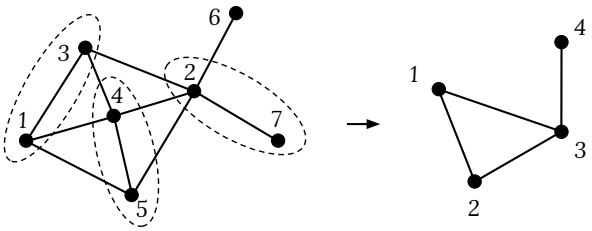

Figura 2: *Matching* sobre grafo à esquerda. As arestas selecionadas estão circuladas. Na matriz grosseira, cada aresta terá se tornado uma única variável, como representado à direita.

Em esquemas de engrossamento por agregação, as variáveis de um domínio mais refinado podem ser separadas em conjuntos disjuntos (agregados), associados a variáveis do domínio grosseiro. A fim de ampliar a redução da ordem do problema, o processo de *matching* é realizado duas vezes em cada nível, de forma que os agregados sejam formados por pares de pares de variáveis (vértices no grafo correspondente), elevando o potencial de redução da ordem do sistema linear em até quatro vezes [11]. Deste modo, o número máximo de agregados por variáveis do sistema grosseiro

 © 2021 SBMAC

é constante e igual a quatro, enquanto que nos esquemas clássicos de engrossamento, cada variável do sistema grosseiro está associada a todos os vizinhos – cujo número não se conhece *a priori* – de uma variável diretamente correspondente no sistema refinado.

3 Resultados Experimentais

Os experimentos numéricos foram realizados com 17 matrizes do repositório de matrizes esparsas *SuiteSparse Matrix Collection*. Os resultados obtidos são analisados com respeito ao número de iterações e tempo computacional. Para cada matriz é definido um sistema linear trivial, cujo vetor dos termos independentes é calculado considerando que a solução exata é o vetor $(1, 1, \ldots, 1)^t$ de ordem n. As principais características dessas matrizes estão descritas na Tabela 1. As colunas indicam, da esquerda para a direita, o índice de identificação da matriz (#), nome (Nome), ordem (n), número de elementos não nulos (nnz), simetria (simétrica) e porcentagem de esparsidade (esparsidade (%)). O asterisco (*) indica que a matriz é diagonal dominante. Para cada matriz, foram testados os seguintes precondicionadores: ILU - Fatoração LU Incompleta [14]; Stb - AMG com esquemas de engrossamento clássicos de Stüben [17]; DPA-S - AMG com DPA e SOR (Método sobre relaxação sucessiva [14]) como relaxador; e DPA-G - AMG com DPA e GMRES como relaxador.

Tabela 1: Principais características das matrizes.

#	**Nome**	**n**	**nnz**	**simétrica**	**esparsidade (%)**
1	rail_5177	5.177	35.185	sim	99,8687
2	aft01	8.205	125.567	sim	99,8135
3	FEM_3D_thermal1	17.880	430.740	não	99,865
4	Dubcova2	65.025	1.030.225	sim	99,9756
5	H2O	67.024	2.216.736	sim	99,9507
6	FEM_3D_thermal2	147.900	3.489.300	não	99,9840
7	parabolic_fem	525.825	3.674.625	sim	99,9987
8	atmosmodj*	1.270.432	8.814.880	não	99,9995
9	atmosmodd*	1.270.432	8.814.880	não	99,9995
10	Serena	1.391.349	64.131.971	sim	99,9967
11	Geo_1438	1.437.960	60.236.322	sim	99,9971
12	atmosmodl	1.489.752	10.319.760	não	99,9995
13	af_shell10	1.508.065	52.259.885	sim	99,9977
14	G3_circuit	1.585.478	7.660.826	sim	99,9997
15	Transport	1.602.111	23.487.281	não	99,9991
16	Cube_Coup_dt6	2.164.760	124.406.070	sim	99,9973
17	Bump_2911	2.911.419	127.729.899	sim	99,9985

Os experimentos foram executados no supercomputador Lobo Carneiro (LoboC) da UFRJ. O LoboC conta com 504 CPUs Intel® Xeon® E5-2670v3, totalizando 6048 núcleos de CPU, distribuídos em 252 nós computacionais com 64 GBytes de memória cada. O sistema operacional instalado no LoboC é o Suse Linux Enterprise [10]. Uma vez que o LoboC é uma máquina compartilhada, a carga varia ao longo da realizações dos testes, tornando os resultados instáveis. Devido a este fato, todas as execuções foram realizadas 5 vezes, eliminando os menores e os maiores valores encontrados e extraindo a média aritmética dos restantes em cada caso. Todos os

 © 2021 SBMAC

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`.

Em todos os testes adotamos a tolerância de $\varepsilon = 10^{-8}$ e o número máximo de iterações igual a 1.000 – qualquer matriz que dependa de mais de 1.000 iterações para convergir foi considerada não convergente. O fator de preenchimento p do precondicionador ILU foi escolhido dentre os valores $p = 0, 1, 2, 3, 4$. Os resultados apresentados para esse precondicionador variam de acordo com o valor de p com o qual obteve-se o menor tempo computacional, uma vez que as matrizes são oriundas de aplicações diversas.

Nos precondicionadores Stb e DPA-S são consideradas duas relaxações do SOR com parâmetro de relaxação igual a $1,5$ e apenas um nível de engrossamento no Ciclo V. No caso do precondicionador DPA-G, que utiliza o relaxador GMRES, foram considerados $\log_4 n - 2$ níveis de engrossamento no Ciclo V, sendo n a ordem do sistema, com o objetivo de reduzi-lo a uma ordem trivial no nível mais grosseiro [5]. Testes preliminares utilizando essa mesma estratégia com Stb e DPA-S não produziram bons resultados.

O GMRES empregado como relaxador no precondicionador DPA-G utiliza uma base de Krylov com 10 vetores e apenas uma iteração GMRES, exceto no último nível, onde são consideradas no máximo 1.000 iterações GMRES e tolerância de 10^{-8}. Assim, no nível mais grosseiro o método é iterado até atingir a tolerância (ou número máximo de 1.000 iterações), diferentemente do DPA-S, onde se executam apenas 2 iterações fixas nas etapas de restrição e interpolação.

A Tabela 2 apresenta para cada matriz e precondicionador Stb, DPA-S, DPA-G e ILU, o número de iterações do GMRES (**Iter**) e os tempos computacionais (**Tempo**), normalizados pelo maior tempo. Os testes que não convergiram são representados por †.

Tabela 2: Normalização dos tempos de execução em segundos e número de iterações. Os melhores resultados dos tempos computacionais estão destacados em negrito para cada matriz.

	DPA-G		**DPA-S**		**Stb**		**ILU**	
#	**Iter**	**Tempo**	**Iter**	**Tempo**	**Iter**	**Tempo**	**Iter**	**Tempo**
1	1	**0,022**	230	1,000	187	0,753	149	0,356
2	†	†	247	0,618	479	1,000	37	**0,061**
3	1	1,000	12	0,711	12	0,665	9	**0,598**
4	1	**0,060**	42	0,855	77	1,000	86	0,614
5	1	**0,156**	40	1,000	26	0,967	299	0,927
6	1	1,000	11	0,778	11	0,761	8	**0,554**
7	1	**0,014**	863	0,843	406	1,000	707	0,367
8	1	**0,160**	53	0,308	44	0,326	147	1,000
9	1	**0,191**	53	0,379	43	0,401	118	1,000
10	†	†	105	**0,145**	93	0,221	77	1,000
11	†	†	168	**0,514**	268	1,000	491	0,596
12	1	**0,353**	27	0,381	23	0,460	61	1,000
13	1	1,000	7	0,545	9	0,345	1	**0,089**
14	1	**0,011**	436	1,000	353	0,268	195	0,098
15	1	**0,012**	901	0,195	524	0,156	688	1,000
16	†	†	298	**1,000**	†	†	†	†
17	†	†	488	1,000	662	**0,628**	†	†
Média de Tempo		**0,332**		0,663		0,622		0,617
#convergências	12		17		16		15	

 © 2021 SBMAC

Para esse conjunto de matrizes o uso de precondicionamento é essencial, pois possibilitou o aumento da convergência de 23% para o GMRES sem precondicionamento – conforme relatado em [5] – para 100% das matrizes para o GMRES com precondicionamento: o DPA-S convergiu para todas; o Stb não obteve a convergência em apenas um caso; o ILU, em dois casos e o DPA-G não convergiu para cinco matrizes. Desta forma, para esse conjunto de matrizes, os precondicionadores avaliados podem ser classificados, do mais para o menos robusto em termos de convergência, na seguinte ordem: DPA-S > Stb > ILU > DPA-G.

O GMRES com o precondicionador DPA-G convergiu para 12 das 17 matrizes apenas com 1 iteração. Acreditamos que esse resultado se deve ao esquema de engrossamento adotado, já que neste caso, no nível mais grosseiro, o GMRES como relaxador calcula o erro até que a tolerância de 10^{-8} seja atingida. Na maioria dos casos onde o DPA-G não convergiu, o algoritmo de *matching* não obteve boa redução na ordem do problema após um determinado nível. Como consequência foi gerado um alto índice de variáveis não pareadas pelo *matching*, não reduzindo satisfatoriamente a ordem da matriz original nos níveis mais grosseiros e prejudicando a convergência do relaxador GMRES.

A expressiva redução em número de iterações do método de solução com o uso de precondicionadores AMG não está diretamente relacionada com a melhor eficiência quando comparado ao uso do precondicionador ILU, uma vez que a iteração do GMRES com o precondicionador ILU é, em geral, muito mais rápida (em tempo computacional) que a iteração com os precondicionadores AMG.

Os melhores tempos computacionais foram alcançados pelo precondicionador DPA-G, seguido pelo ILU, Stb e, por último, o DPA-S. Em média, o DPA-G reduziu o tempo de execução em 46.19% em relação ao ILU, 46.62% em relação ao Stb e 49.92% em relação ao DPA-S.

4 Conclusões

Neste trabalho introduzimos o uso do método *Multigrid* Algébrico como precondicionador do método GMRES. Especificamente, enfatizamos o uso da estratégia de engrossamento por agregação dupla de pares (DPA) [11], comparando-a às estratégias clássicas propostas em [13] e à fatoração LU incompleta [14]. Os resultados computacionais obtidos indicam a superioridade dos métodos *Multigrid* avaliados em relação ao ILU. Em termos de tempo computacional, o uso desses métodos melhorou os tempos obtidos pelo ILU em 13 das 17 matrizes. E em 12 dessas 13 matrizes, a melhora foi obtida pelo uso do *Multigrid* com o DPA. Em termos de número de iterações, o DPA-G convergiu com apenas 1 iteração para 12 matrizes, enquanto que, em sua maioria, os outros precondicionadores necessitaram de um número muito maior de iterações.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Além disso, agradecemos ao Núcleo Avançado de Computação de Alto Desempenho (NACAD) da COPPE/UFRJ pela disponibilização do supercomputador Lobo Carneiro (LoboC).

Referências

[1] A. Brandt, S. McCormick e J. Ruge, Algebraic multigrid (AMG) for automatic multigrid solutions with application to geodetic computations, Report, *Institute for Computational Studies*, CO, 1982.

 © 2021 SBMAC

[2] A. Brandt, General highly accurate algebraic coarsening schemes, em *Proceedings of the Ninth Copper Mountain Conference on Multigrid Methods, Copper Mountain*, Citeseer, 1999.

[3] W. L. Briggs, V. E. Henson e S. F. McCormick, A *multigrid tutorial*. Siam, 2000, vol. 72.

[4] T. A. Davis e Y. HU. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38:1-25, 2011.

[5] H. G. de Jesus, Precondicionador multigrid algébrico para métodos iterativos não estacionários na solução de sistemas lineares de grande porte, Mestrado em Informática, UFES, Vitória, 2021.

[6] R. Diestel, *Graph Theory*. Springer Berlin Heidelberg, 2017, vol. 173.

[7] P. W. Hemker, A note on defect correction processes with an approximate inverse of deficient rank, *Journal of Computational and Applied Mathematics*, 8:137–139, 1982.

[8] A. McAdams, E. Sifakis e J. Teran, A parallel multigrid Poisson solver for fluids simulation on large grids, em *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, 2010, pages 65–74.

[9] S. F. McCormick, *Multigrid methods*. SIAM, 1987.

[10] SGI ICE-X. NACAD, Rio de Janeiro, 2021. Disponível em: http://www.nacad.ufrj.br/pt/recursos/sgiicex. Acesso em: 14 de junho de 2021.

[11] Y. Notay, Aggregation-based algebraic multilevel preconditioning, *SIAM journal on matrix analysis and applications*, 27:998–1018, 2006.

[12] J. Park, M. Smelyanskiy, U. M. Yang, D. Mudigere e P. Dubey, High-performance algebraic multigrid solver optimized for multi-core based distributed parallel systems, em *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ACM, 2015.

[13] J. W. Ruge e K. Stüben, Algebraic multigrid, em *Multigrid methods*, SIAM, 1987, pages 73–130.

[14] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Minneapolis: Society for Industrial e Applied Mathematics, 2003.

[15] H. De Sterck, U. M. Yang e J. J. Heys, Reducing complexity in parallel algebraic multigrid preconditioners, *SIAM Journal on Matrix Analysis and Applications*, 27:1019– 1039, 2006.

[16] K. Stuben, A review of Algebraic Multigrid, *CMD Report*, 1999.

[17] Algebraic multigrid (AMG): an introduction with applications, *Multigrid*, 2000.

[18] K. Stüben, A review of algebraic multigrid, em *Numerical Analysis: Historical Developments in the 20th Century*, Elsevier, 2001, pages 331–359.

[19] U. Trottenberg, C. W. Oosterlee e A. Schuller, *Multigrid*. Elsevier, 2000.

[20] P. Vaněk, J. Mandel e M. Brezina, Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems, *Computing*, 56:179–196, 1996.

[21] U. M. Yang, Parallel algebraic multigrid methods—high performance preconditioners, em *Numerical solution of partial differential equations on parallel computers*, Springer, 2006, pages 209–236.

 © 2021 SBMAC