Trends in Computational and Applied Mathematics, **23**, N. 3 (2022), 471-498
Sociedade Brasileira de Matemática Aplicada e Computacional
Online version ISSN 2676-0029
www.scielo.br/tcam
doi: 10.5540/tcam.2022.023.03.00471

Novas Versões para a Inversa Aproximada em Blocos: Uma Comparação Numérica

J. S. CRUZ[1*], M. C. ALMEIDA[2], L. M. CARVALHO[3] e M. SOUZA[4]

Recebido em 8 de dezembro de 2020 / Aceito em 15 de fevereiro de 2022

RESUMO. Propomos duas variações do precondicionador de aproximação da inversa em blocos (BAINV), originalmente desenvolvido por Benzi, Kouhia e Tůma em 2001. A primeira variação, a aproximação da inversa em blocos estabilizada para matrizes não simétricas (SBAINV-NS), é válida para matrizes não simétricas e não singulares. A segunda variação, a aproximação da inversa em blocos estabilizada combinada (SBAINV-VAR), é baseada nas relações dos fatores da inversa aproximada em blocos com a fatoração *LDU* em blocos de A e na relação de aproximação da inversa de Neumann. Demonstramos a consistência matemática dessas novas versões e apresentamos os algoritmos referentes a cada uma delas, além de exibir experimentos numéricos onde comparamos a densidade dos precondicionadores e o número de iterações quando aplicados ao método estabilizado de gradientes bi-conjugados (Bi-CGSTAB). Os principais resultados numéricos obtidos indicam que o uso da estrutura de blocos pode aumentar o desempenho do método iterativo de Krylov em comparação com a versão escalar. Além disso, nos experimentos apresentados, o SBAINV-VAR produz, em geral, precondicionadores que realizam menos iterações do Bi-CGSTAB e são menos densos do que o SBAINV-NS.

Palavras-chave: inversa aproximada, matrizes em bloco, precondicionadores, métodos de Krylov.

1 INTRODUÇÃO

Sistemas lineares da forma $Ax = b$, em que $A \in \mathbb{R}^{n\times n}$ é não singular e esparsa, $b \in \mathbb{R}^n$ é o vetor dos termos independentes, e $x \in \mathbb{R}^n$ é o vetor solução são de grande relevância em problemas da ciência e indústria. Quando n é muito grande ($n \geq 10^9$) e a matriz é densa, o uso de métodos

*Autor correspondente: Julia Sekiguchi da Cruz – E-mail: julia-seki@hotmail.com
[1]Universidade do Estado do Rio de Janeiro, FEN, Programa de Pós Graduação em Engenharia Mecânica - PPG-EM/UERJ, Rua Fonseca Teles, 121, 20940-903, Rio de Janeiro, RJ, Brasil – E-mail: julia-seki@hotmail.com https://orcid.org/0000-0002-7296-3928
[2]Instituto Federal do Rio de Janeiro, Departamento de Matemática, R. Sebastião Lacerda, S/N, 26600-000, Paracambi, RJ, Brasil – E-mail: moisesceni@gmail.com https://orcid.org/0000-0002-6170-2036
[3]Universidade do Estado do Rio de Janeiro, IME, Departamento de Matemática Aplicada, R. São Francisco Xavier, 524, 20550-000, Rio de Janeiro, RJ, Brasil – E-mail: luizmc@ime.uerj.br https://orcid.org/0000-0002-4829-2917
[4]Universidade Federal do Ceará, Departamento de Estatística e Matemática Aplicada, Campus do Pici, Bloco 910, 60440-900, Fortaleza, CE, Brasil – E-mail: souza.michael@gmail.com https://orcid.org/0000-0001-6751-2877

diretos para a sua resolução, como a eliminação gaussiana, é impraticável, pois a complexidade é $\mathcal{O}(n^3)$, fazendo-se necessário o uso de métodos iterativos, como, por exemplo, os métodos de Krylov [17]. Mesmo matrizes esparsas podem sofrer um grande preenchimento quando da utilização de métodos de fatoração completa, também implicando no uso de métodos iterativos [17]. Os métodos de Krylov convergem teoricamente em um número finito de passos, mas o mau condicionamento ou a distribuição dos autovalores da matriz pode dificultar a convergência destes métodos. Os precondicionadores são utilizados para melhorar o número de condicionamento ou a distribuição de autovalores [31]. Em geral, a ideia é construir um operador M tal que $MAx = Mb$ ou $AMy = b$ $(x = My)$. Geralmente, essas alterações no problema original fazem com que o sistema resultante seja solucionado em um menor número de iterações.

Há inúmeros precondicionadores e um exemplo bem conhecido é o precondicionador ILU [25] que é uma aproximação de A baseada na fatoração LU incompleta de A. Neste caso, quando o método de Krylov é aplicado ao sistema precondicionado, são resolvidos dois sistemas triangulares. Ele possui dezenas de variações [31] e, por ser uma aproximação de A, é chamado de precondicionador explícito. Outro exemplo é o precondicionador que aproxima uma fatoração da inversa de A, onde temos o AINV [7], FSAI [24,36], ISM-based [9], SPAI [18] e BIF [10,11] como alguns exemplos encontrados na literatura. Eles são chamados de precondicionadores implícitos e a vantagem no seu uso é que a aplicação do precondicionador da inversa é feita através da multiplicação matriz-vetor, uma operação paralelizável, sendo, assim, adequado para a utilização em computadores paralelos híbridos atuais, que contam com CPUs e GPUs [6, 7, 13, 18, 24, 36]. O objeto do nosso estudo é o precondicionador AINV (”Approximate Inverse”) que foi originalmente proposto por Benzi, Meyer e Tůma [6], em 1996, e encontra a fatoração da inversa aproximada de matrizes simétricas e positivas definidas. Em 1998, uma versão generalizada do AINV foi proposta [7], onde A pode ser qualquer matriz esparsa não singular. Como a fatoração produzida no AINV é densa e possui custo computacional elevado, comparável ao custo da eliminação gaussiana, algumas entradas dos fatores devem ser descartadas durante sua execução, daí a inversa ser aproximada.

Diversas variações do AINV foram desenvolvidas, como em [8, 28, 30, 32], porém apenas três versões foram apresentadas para matrizes em blocos [5, 8, 29]. Essas matrizes são muito comuns em problemas oriundos da discretização de equações diferenciais parciais (EDP's). Algumas das vantagens dos algoritmos em bloco são a sua maior estabilidade e o seu melhor desempenho computacional, especialmente com matrizes esparsas, onde o endereçamento indireto de esquemas de armazenamento esparsos reduz a localização durante o produto de matriz esparsa por vetor denso [1, 2, 20]. Além disso, também há a preservação de possíveis aspectos físicos associados aos blocos da matriz. Existem, na literatura, alternativas em blocos para alguns dos métodos de inversa aproximada. Entre eles podemos encontrar o BFSAI [14, 21, 22, 23], BSPAI [4, 33], Block ISM-based [12] e BAINV [5].

Diante das poucas contribuições referentes ao AINV em blocos encontradas na literatura e dos benefícios do uso do precondicionamento em blocos, consideramos importante estudar este tipo de precondicionador. Sendo assim, este trabalho tem o objetivo de analisar os principais aspectos

do AINV em blocos a fim de desenvolver novas versões e, por fim, testá-las em experimentos numéricos. Neste artigo propomos duas novas versões. A primeira é uma generalização, para matrizes não simétricas, do BAINV ("Block Approximate Inverse") proposto em [5] para matrizes simétricas. A segunda, tem como base o trabalho de Rafiei e Toutoniam [30] que apresentaram uma variação do AINV escalar, relacionando seus elementos com a fatoração *LDU* de A.

O restante deste trabalho está organizado da seguinte forma. Na Seção 2, explicamos as notações utilizadas. Na Seção 3, propomos o SBAINV-NS, na 4 demostramos alguns resultados relacionados aos fatores da inversa em blocos de A e dos fatores LDU em blocos de A e propomos o Algoritmo SBAINV-VAR. Na 5, descrevemos uma série de experimentos numéricos e realizamos a análise dos resultados obtidos e, finalmente, na 6, apresentamos as considerações finais e apontamentos para trabalhos futuros.

2 NOTAÇÕES

Seja A uma matriz em $\mathbb{R}^{n\times n}$, com n divisível por s. Nós consideraremos a estrutura em blocos:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1N} \\ A_{21} & A_{22} & \cdots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1} & A_{N2} & \cdots & A_{NN} \end{bmatrix},$$

onde $N = n/s$, para $1 \leq I,J \leq N$ e o bloco A_{IJ} é uma submatriz $s \times s$. Neste caso, temos que A é uma matriz-bloco ou matriz em blocos $N \times N$ com blocos de ordem s. Neste trabalho, a ordem do bloco é fixada como s e, assim, dizemos que A é formada por blocos *homogêneos*.

Definimos um vetor-bloco de tamanho N como uma matriz em blocos $N \times 1$ (i.e., uma matriz $Ns \times s$) e uma linha-bloco de tamanho N como uma matriz em blocos $1 \times N$ (i.e., uma matriz $s \times Ns$). Denotamos por A_J o vetor-bloco correspondente à J-ésima coluna-bloco de A e por A_{J*} a linha-bloco correspondente à J-ésima linha-bloco de A, ou seja,

$$A_J = \begin{bmatrix} A_{1J} \\ A_{2J} \\ \vdots \\ A_{NJ} \end{bmatrix}, \qquad A_{J*} = \begin{bmatrix} A_{J1} & A_{J2} & \cdots & A_{JN} \end{bmatrix}.$$

Nós também definimos o vetor-bloco canônico E_I de tamanho N, com $I = 1,2,\ldots,N$, como o produto de Kronecker do vetor canônico $e_I \in \mathbb{R}^N$ e a matriz identidade de ordem $s \times s$.

Definição 2.1. *Uma matriz quadrada A é uma Z-matriz se $a_{ij} \leq 0,\ \forall\, i \neq j$.*

Definição 2.2. *Uma Z-matriz A é uma M-matriz se $A = sI - B$ para alguma matriz $B \geq 0$ e algum escalar $s \geqslant \rho(B)$, onde $\rho(B)$ é o raio espectral de B.*

Definição 2.3. *Dada a matriz A, nós definimos as entradas c_{ij} da matriz comparação de A [27], designada por $C(A)$, como:*

$$c_{ij} = \begin{cases} -|a_{ij}|, & \text{if } i \neq j \\ |a_{ii}|, & \text{if } i = j \end{cases} .$$

Definição 2.4. *Uma matriz é uma H-matriz se a sua matriz comparação é uma M-matriz.*

3 SBAINV-NS

Em [5], foi proposto o BAINV que calcula a inversa aproximada de A como $A^{-1} \approx ZD^{-1}Z^T$, onde A é esparsa inversível e simétrica, Z é triangular superior em blocos, de ordem N, cujos blocos diagonais são formados pela identidade de ordem s, e D é uma matriz em blocos de ordem N, diagonal por blocos. Tal algoritmo é baseado no processo de A-ortogonalização [15] em blocos e sua consistência foi demonstrada em [3].

No presente trabalho, sugerimos uma generalização desse algoritmo para A não simétrica. Neste caso, devemos encontrar as matrizes em blocos Z, W e D, $N \times N$, tais que $A^{-1} \approx ZD^{-1}W$, baseando-se no processo de A-biortogonalização em blocos. O Algoritmo 1 apresenta o cálculo destes fatores.

Algorithm 1 SBAINV-NS

Dados: matriz A em blocos $N \times N$ nao singular.

Resultado: D, Z e W não singulares com $A^{-1} \approx ZD^{-1}W$.

$Z_I^{(0)} \leftarrow E_I, \qquad I \in \{1,...,N\} \quad W_{I*}^{(0)} \leftarrow E_I^T, \qquad I \in \{1,...,N\}$

para $I \leftarrow 1$ **até** N **faça**

  $Z_I \leftarrow Z_I^{(I-1)}$; $W_{I*} \leftarrow W_{I*}^{(I-1)}$

  $D_{II} \leftarrow A_{I*}Z_I$ ou $W_{I*}A_I \quad D_{II} \leftarrow (Z_I)^T AZ_I$ ou $W_{I*}A(W_{I*})^T$ se A for positiva definida

  para $J \leftarrow I+1$ **até** N **faça**

    $M_J^{(I-1)} \leftarrow A_{I*}Z_J^{(I-1)}$

    $Q_J^{(I-1)} \leftarrow W_{J*}^{(I-1)}A_I$

    $Z_J^{(I)} \leftarrow \text{descarte}_I(Z_J^{(I-1)} - Z_I D_{II}^{-1} M_J^{(I-1)})$

    $W_{J*}^{(I)} \leftarrow \text{descarte}_I(W_{J*}^{(I-1)} - Q_J^{(I-1)} D_{II}^{-1} W_{I*})$

  fim

fim

$D \leftarrow \text{Diag}(D_{11},\ldots,D_{NN})$, $Z \leftarrow \begin{bmatrix} Z_1 & \cdots & Z_N \end{bmatrix}$ e $W \leftarrow \begin{bmatrix} W_{1*} \\ \vdots \\ W_{N*} \end{bmatrix}$

Para garantir a esparsidade do precondicionador, tornando viável a sua utilização em sistemas com matrizes de grande porte, com ordem maior ou igual a 10^9, é utilizado o descarte$_I$, definido a seguir.

Definição 3.5. *(Baseada em [3]) Seja um vetor-bloco V com N blocos, a notação* descarte$_I$ *indica o vetor-bloco de tamanho N resultante do procedimento de descartar determinadas entradas de V, exceto aquelas que pertencem ao I-ésimo bloco, i.e.,*

$$(\text{descarte}_I(V))_{ij} = \begin{cases} v_{ij}, & se \;\; (I-1)b < i \leq Ib, \quad 1 \leqslant j \leqslant b, \\ v_{ij} \;\; ou \;\; 0, & para \; as \; demais \; entradas. \end{cases}$$

De forma análoga, seja uma linha-bloco V∗ com N blocos, a notação descarte$_I$ *indica a linha-bloco de tamanho N resultante do procedimento de descartar determinadas entradas de V∗, exceto aquelas que pertencem ao I-ésimo bloco, i.e.,*

$$(\text{descarte}_I(V*))_{ij} = \begin{cases} v*_{ij}, & se \;\; (I-1)b < j \leq Ib, \quad 1 \leqslant i \leqslant b, \\ v*_{ij} \;\; ou \;\; 0, & para \; as \; demais \; entradas. \end{cases}$$

O objetivo é produzir um vetor-bloco (ou linha-bloco) que seja esparso, porém, satisfatoriamente próximo do original. Uma opção natural seria descartar entradas de baixa magnitude. Também existem outras estratégias de descarte, como padrão de zeros pré definido, tolerância absoluta ou relativa, descarte de blocos ou elementos, etc.

Durante a execução do Algoritmo 1, é possível que D_{II} seja singular para algum $I \in 1,...,N$. Se isso ocorrer, dizemos que houve uma quebra no algoritmo. Caso não ocorra quebra, Z será bloco triangular superior não singular, W será bloco triangular inferior não singular, ambos com identidades nos blocos diagonais e D será bloco diagonal não singular. Isso pode ser demonstrado de forma análoga ao caso simétrico feito em [3]. Caso ocorra quebra, o processo de construção do precondicionador é parado e, assim, não o obtemos. Se A for positiva definida[1], uma alternativa livre de quebra seria utilizar as expressões $(Z_I)^T A Z_I$ ou $W_{I*}A(W_{I*})^T$ para o cálculo de D_{II} (linha 4 do Algoritmo 1). Isso é possível pois se A for positiva definida, então $(Z_I)^T A Z_I$ e $W_{I*}A(W_{I*})^T$ também serão positivas definidas e, portanto, não singulares.

A aplicação deste precondicionador se dá por meio da multiplicação das matrizes W, D^{-1} e Z pelo resíduo em cada iteração do Bi-CGSTAB.

Em [3] foi demonstrado que o BAINV (caso simétrico) não quebra, independente do descarte utilizado, se A for do tipo M-matriz. Também foi demonstrado que ele não quebra se A for do tipo H-matriz, com entradas da sua diagonal principal positivas e se sua matriz comparação $C(A)$ for uma M-matriz não singular. Resultados similares para o caso escalar já haviam sido demonstrados em [6, 7]. Pretendemos demonstrar resultados análogos para o BAINV-NS, com A não simétrica, em trabalhos futuros.

[1] Alguns autores restringem o uso do termo "positiva definida" ou "positiva" apenas para matrizes simétricas, aqui consideramos seu uso para matrizes quadradas quaisquer desde que $x^T Ax > 0, \forall x \neq 0$.

4 SBAINV-VAR

Nesta seção, propomos um algoritmo em blocos para a variação do AINV proposta por Rafiei e Toutounian em [30].

Consideremos novamente a matriz $A \in \mathbb{R}^{n\times n}$ e uma decomposição em matrizes blocos $N \times N$. Consideremos ainda a fatoração de A

$$A = W^{-1}DZ^{-1}, \tag{4.1}$$

onde Z é bloco triangular superior e W é bloco triangular inferior, ambas não singulares com identidades nos blocos diagonais e D é bloco diagonal não singular. Então (4.1) pode ser considerada a decomposição LDU em blocos de A, ou seja $A = LDU$, em que L e U são bloco triangulares inferior e superior, respectivamente, não singulares com identidades nas diagonais e D é bloco diagonal não singular. Como essa decomposição é única, para cada tamanho de bloco, temos que $W = L^{-1}$, $Z = U^{-1}$ e D é a mesma matriz.

Proposição 4.1. *Seja A matriz em blocos $N \times N$ esparsa e não singular. Considerando o Algoritmo 1 aplicado a A, sem descartes, temos*

$$M_J^{(K-1)} = W_{K*}A_J, \tag{4.2}$$

$$Q_J^{(K-1)} = A_{J*}Z_K. \tag{4.3}$$

Proof. Pelas linhas 1, 1, 8 e 9 do Algoritmo 1, temos que

$$Z_J = E_J - \sum_{I=1}^{J-1} Z_I D_{II}^{-1} M_J^{(I-1)} \text{ e } W_{J*} = E_J^T - \sum_{I=1}^{J-1} Q_J^{(I-1)} D_{II}^{-1} W_{I*}.$$

Seja $2 \le J \le N$ fixo. Como $W_{K*}AZ_J = 0$ para todo $K \neq J$ (pois $D = WAZ$), então, para $K = 1,...,J-1$, temos

$$\begin{array}{rcl} 0 = W_{K*}AZ_J & \Rightarrow & 0 = W_{K*}A(E_J - \sum_{I=1}^{J-1} Z_I D_{II}^{-1} M_J^{(I-1)}) \\ & \Rightarrow & 0 = W_{K*}A_J - \sum_{I=1}^{J-1} W_{K*}AZ_I D_{II}^{-1} M_J^{(I-1)} \\ & \Rightarrow & 0 = W_{K*}A_J - W_{K*}AZ_K D_{KK}^{-1} M_J^{(K-1)} \\ & \Rightarrow & 0 = W_{K*}A_J - M_J^{(K-1)} \\ & \Rightarrow & M_J^{(K-1)} = W_{K*}A_J \end{array},$$

obtendo (4.2). A equação (4.3) é obtida de forma análoga. □

Proposição 4.2. *Seja A uma matriz em blocos $N \times N$ esparsa e não singular. Seja a fatoração LDU de A em blocos, onde L e U são bloco triangulares inferior e superior, respectivamente, não singulares com identidades nas diagonais e D é bloco diagonal não singular. Então, o Algoritmo 1 quando aplicado a A, sem os descartes, produz*

$$U_{IJ} = D_{II}^{-1} M_J^{(I-1)}, \tag{4.4}$$

$$L_{JI} = Q_J^{(I-1)} D_{II}^{-1}, \tag{4.5}$$

onde $0 \leq I < J \leq N$.

Proof. De acordo com (4.1), para $0 \leq I < J \leq N$,

$$U = Z^{-1} = D^{-1}WA \Rightarrow U_{IJ} = D_{II}^{-1} W_{I*} A_J,$$

e

$$W^{-1} = L = AZD^{-1} \Rightarrow L_{JI} = A_{J*} Z_I D_{II}^{-1}.$$

Considerando (4.2) e (4.3) , temos que $U_{IJ} = D_{II}^{-1} M_J^{(I-1)}$ e $L_{JI} = Q_J^{(I-1)} D_{II}^{-1}$. □

Proposição 4.3. *Seja A uma matriz em blocos $N \times N$ esparsa e não singular. O Algoritmo 1 quando aplicado a A, sem os descartes, atende à seguinte propriedade*

$$Q_J^{(I-1)} = A_{JI} - \sum_{K=1}^{I-1} L_{JK} M_I^{(K-1)},$$

com $0 \leq I \leq J \leq N$.

Proof. Usando as mesmas hipóteses da Proposição 4.2, a partir da fatoração *LDU* em blocos de A, temos, para $0 \leq I \leq J \leq N$,

$$A_{JI} = \sum_{K=1}^{I} L_{JK} D_{KK} U_{KI} = L_{JI} D_{II} U_{II} + \sum_{K=1}^{I-1} L_{JK} D_{KK} U_{KI}.$$

A partir de (4.4), (4.5) e de que $U_{II} = I$,

$$\begin{array}{rcl} A_{JI} & = & Q_J^{(I-1)} D_{II}^{-1} D_{II} + \sum_{K=1}^{I-1} Q_J^{(K-1)} D_{KK}^{-1} D_{KK} D_{KK}^{-1} M_I^{(K-1)} \\ & = & Q_J^{(I-1)} + \sum_{K=1}^{I-1} Q_J^{(K-1)} D_{KK}^{-1} M_I^{(K-1)} \\ \Rightarrow & & Q_J^{(I-1)} = A_{JI} - \sum_{K=1}^{I-1} L_{JK} M_I^{(K-1)} \end{array}.$$

□

Com esses resultados, propomos o SBAINV-VAR descrito no Algoritmo 2. Baseado no SBAINV-NS, são produzidas aproximações dos fatores Z, L e D de A, sendo possível chegar na aproximação de L por meio da equação (4.5). Por sua vez, para se calcular $Q_J^{(I-1)}$ é utilizada a Proposição 4.3 e a equação (4.5), ou seja, seu cálculo é executado sem a utilização do fator W. Para o cálculo de D_{II} é dada a opção de se utilizar a expressão $Z_I^T A Z_I$ que evita a quebra em matrizes positivas definidas.

No SBAINV-VAR, além da utilização do descarte$_I$ (definido na seção 3) também utilizamos o descarte para promover a esparsidade dos blocos L_{JI}, definido a seguir.

Definição 4.6. *Seja a matriz M de ordem s, a notação* descarte *indica a matriz resultante do procedimento de descartar determinadas entradas de M, i.e.,* $(\text{descarte}(M))_{ij} = m_{ij}$ *ou* $(\text{descarte}(M))_{ij} = 0$.

Algorithm 2 SBAINV-VAR

Dados: matriz A em blocos $N \times N$ não singular.

Resultado: matriz bloco diagonal D não singular e matrizes Z e L não singulares.

$Z_I^{(0)} \leftarrow E_I, \qquad I \in \{1,...,N\}$;
para $I \leftarrow 1$ **até** N **faça**
  $Z_I \leftarrow Z_I^{(I-1)}$ $D_{II} \leftarrow A_{I*}Z_I$ **ou** $Z_I^T A Z_I$ se A for positiva definida
  para $J \leftarrow I+1$ **até** N **faça**
    $M_J^{(I-1)} \leftarrow A_{I*}Z_J^{(I-1)}$
    $Q_J^{(I-1)} \leftarrow A_{JI} - \sum_{K=1}^{I-1} L_{JK}M_I^{(K-1)}$;
    $L_{JI} \leftarrow \text{descarte}(Q_J^{(I-1)}D_{II}^{-1})$;
    $Z_J^{(I)} \leftarrow \text{descarte}_I(Z_J^{(I-1)} - Z_I D_{II}^{-1} M_J^{(I-1)})$
  fim
fim
$D \leftarrow \text{Diag}(D_{11},\ldots,D_{NN})$, $Z \leftarrow \begin{bmatrix} Z_1 & \cdots & Z_N \end{bmatrix}$ e $L \leftarrow [L_{JI}]$

Para obter a aproximação de W, fazemos a aproximação da inversa de L ($W = L^{-1}$) por meio do trucamento da série de Neumann de grau l, (veja [26]):

$$W_l = I + F + F^2 + F^3 + ... + F^l, \tag{4.6}$$

onde $F = I - L$ e I é a identidade. Por meio de 4.6 e do método de Horner [19], obtemos a inversa aproximada $A^{-1} \approx ZD^{-1}W_l$.

Descreveremos brevemente o método de Horner [19]. Dado o polinômio

$$p(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + ... + a_n x^n,$$

em que $a_0 ... a_n$ são números reais (no nosso caso, consideramos matrizes com entradas reais). O objetivo é estimar o valor de $p(x_0)$ para algum valor específico x_0 (sendo nosso caso, alguma matriz específica). Para achar $p(x_0)$, o método de Horner se baseia escrevendo-se o polinômio da forma

$$p(x) = a_0 + x(a_1 + x(a_2 + ...x(a_{n-1} + a_n x)...).$$

Ou seja, para achar $p(x_0)$, faz-se o uso termos da sequência $(b_i)_{i=0}^n$, dada por

$$\begin{aligned} b_n &:= a_n \\ b_{n-1} &:= a_{n-1} + b_n x_0 \\ &\vdots \\ b_0 &:= a_0 + b_1 x_0, \end{aligned}$$

onde os valores $b_i's$ são obtidos recursivamente até b_0, que é o valor de $p(x_0)$.

No nosso estudo, os coeficientes do polinômio em 4.6 são a matriz identidade e x_0 é $I - L$. A ideia é usar a fórmula de Horner [19] para multiplicar W_l pelo resíduo do método iterativo, fazendo uso da multiplicação matriz-vetor. Ou seja, a aplicação do precondicionador gerado pelo SBAINV-VAR se dá por meio do método de Horner e da multiplicação das matrizes Z e D^{-1} pelo resíduo em cada iteração do Bi-CGSTAB. Este processo pode ser visto no Algoritmo 3.

Algorithm 3 Aplicação do precondicionador gerado pelo SBAINV-VAR

Dados: matrizes D, Z e L não singulares, vetor resíduo r e inteiro l (grau do polinômio Neumann).

Resultado: vetor v.

$r_0 \leftarrow r$
para $i \leftarrow 1$ **até** l **faça**
  $r_l \leftarrow r_{l-1} - Lr_{l-1}$
fim
$y \leftarrow \sum_{i=0}^{l} r_l$;
$v \leftarrow ZD^{-1}y$

5 EXPERIMENTOS NUMÉRICOS

Vamos analisar os comportamentos do SBAINV-NS, proposto na Seção 3, e do SBAINV-VAR, proposto na Seção 4. Os algoritmos foram implementados em Python 3.8, com o auxílio das bibliotecas NumPy e SciPy, e da biblioteca Matplotlib para criar os gráficos. Os experimentos foram simulados em um computador com 2 $\times$ CPU i5-7200U, de 2,5 GHz, e memória RAM de 8GB, com o sistema operacional Windows 64. As matrizes utilizadas pertencem aos repositórios Matrix Market[2] e Tim Davis' Collection[3]. Estas matrizes são provenientes de simulações de reservatórios de petróleo (SHERMAN1 e SHERMAN4), de um modelo proposto por H. Elman usando equações diferenciais parciais (PDE900 e PDE2961), de problemas de fluxo de rede (HOR131) e de problemas de eletroforese de DNA (CAGE8). Algumas destas matrizes possuem uma estrutura de blocos determinada pelo problema físico subjacente; no entanto, estabelecemos divisões arbitrárias para construir blocos homogêneos e avaliar o impacto da abordagem em blocos (b2 e b3) em relação à escalar (b1). A Tabela 1 mostra a ordem n de cada uma das matrizes, as respectivas quantidades de elementos não nulos nnz, os tamanhos dos blocos homogêneos utilizados nos experimentos com cada matriz e o número médio de elementos não nulos por linha (nnz/n) em cada uma delas.

Criamos um conjunto de dez vetores aleatórios b_i, $i = 1, ..., 10$, com entradas uniformemente distribuídas entre $[0, 1)$, para a solução do sistema $Ax = b_i$ por meio do método Bi-CGSTAB [34] (nativo da biblioteca SciPy) precondicionado por SBAINV-NS e SBAINV-VAR. A semente utilizada para gerar os vetores foi igual a zero. Com isso, simulamos os testes com o método iterativo dez vezes para cada matriz. Assim, os números de iterações indicados nas tabelas são as médias

[2]https://math.nist.gov/MatrixMarket.
[3]https://www.cise.ufl.edu/research/sparse/matrices.

Tabela 1: Ordens e número de elementos não nulos das matrizes e ordem dos blocos homogêneos (b1, b2 e b3).

Matriz	n	nnz	nnz$/n$	b1	b2	b3
SHERMAN1	1000	3750	3,75	1	2	8
SHERMAN4	1104	3786	3,43	1	2	6
PDE900	900	4380	4,87	1	2	6
PDE2961	2961	14585	4,93	1	3	7
HOR131	434	4710	10,85	1	2	7
CAGE8	1015	11003	10,84	1	5	7

das iterações obtidas nos dez testes realizados. O critério de parada foi 10^{-6} como cota superior para a norma 2 do resíduo relativo ao lado direito, sendo a aproximação inicial para o método iterativo Bi-CGStab o vetor com entradas iguais a zero e o número máximo de iterações igual a 1000. Nos experimentos do SBAINV-VAR, foram usados os primeiros quatro termos da série de Neumann para aproximar a inversa de W.

A norma de Frobenius foi utilizada para decidir quais blocos seriam descartados. Caso a norma de Frobenius de um bloco fosse menor que determinado valor, ele era descartado. Nos testes, estes valores foram variaram entre 0,1, 0,2, 0,3, 0,4 e 0,5. No SBAINV-NS, um bloco de $Z_J^{(I)}$ ou de $W_{J*}^{(I)}$ (ver Algoritmo 1) seria descartado caso a sua norma tivesse sido menor que um desses valores. O mesmo valor de descarte foi usado em $Z_J^{(I)}$ e $W_{J*}^{(I)}$. No SBAINV-VAR, um bloco de $Z_J^{(I)}$ seria descartado, caso sua norma fosse menor que a tolerância escolhida. Com em relação à matriz L, o bloco L_{JI} seria descartado caso sua norma fosse menor que determinada tolerância. Também utilizamos a mesma tolerância para os descartes em $Z_J^{(I)}$ e em L_{JI}.

As densidades dos precondicionadores SBAINV-NS e SBAINV-VAR foram medidas, respectivamente, pelas equações (5.1) e (5.2):

$$\delta_{\mathrm{NS}} = \frac{\mathrm{nnz}(Z) + \mathrm{nnz}(W) + \mathrm{nnz}(D)}{\mathrm{nnz}(A)} \tag{5.1}$$

e

$$\delta_{\mathrm{VAR}} = \frac{\mathrm{nnz}(Z) + \mathrm{nnz}(L) + \mathrm{nnz}(D)}{\mathrm{nnz}(A)}, \tag{5.2}$$

onde nnz(X) é o número de elementos não nulos da matriz X. Incluímos o número de não zeros da matriz D, pois, no caso bloco, quanto maior for o tamanho do bloco, maior será o preenchimento de D, uma vez que não há descarte nestes blocos. Vejamos que, em cada teste, o precondicionador é gerado uma vez para ser aplicado nas dez execuções do Bi-CGSTAB, relativas aos dez vetores aleatórios do lado direito. Com isso, enquanto os resultados apresentados sobre as iterações se referem às médias dessas dez execuções, as densidades são absolutas.

O número médio, mediana e desvio padrão (δ^2) dos números de iterações obtidos pelo Bi-CGSTAB sem precondicionamento com os dez lados direitos aleatórios associados a cada matriz

Tabela 2: Média, mediana e desvio padrão (δ^2) dos números de iterações e tempo médio do Bi-CGSTAB sem precondicionamento, em segundos.

Matriz	Iterações			Tempo
	Média	Mediana	δ^2	
SHERMAN1	360,6	358,5	16,44	7,64e-1
SHERMAN4	80,1	79	4,57	1,09e-2
PDE900	70,4	70,5	1,42	1,22e-1
PDE2961	132,6	132,5	1,96	2,34e-2
HOR131	†	†	†	†
CAGE8	11	11	0	1,56e-3

são exibidos na Tabela 2. O símbolo † indica que o sistema não reduziu o resíduo, na tolerância requerida, dentro do número máximo de iterações. Diante da proximidade entre os valores da média e mediana para cada matriz e do baixo desvio padrão em relação a esses valores, nós utilizaremos as médias apresentadas como base de comparação com os outros testes. Nesta tabela também apresentamos os tempos médios, em segundos, obtidos pelo Bi-CGSTAB sem precondicionamento para cada matriz.

5.1 Variando blocos e descartes

Analisaremos os algoritmos de acordo com o número médio de iterações e densidades dos precondicionadores, ao variarmos os parâmetros. A Tabela 3 mostra o impacto da variação dos descartes e do tamanho dos blocos no número médio de iterações do Bi-CGSTAB, para ambos os precondicionadores, em relação à matriz SHERMAN1. Na última linha desta tabela, temos o quociente dos valores do número médio de iterações do SBAINV-NS e do SBAINV-VAR para o parâmetro de descarte de $0,5$ e $0,1$ ($0,5$ no numerador e $0,1$ no denominador), para cada tamanho de bloco. E na última coluna temos o quociente dos valores do número médio de iterações para os blocos de tamanho $b3$ e $b1$. A Tabela 4 apresenta as densidades dos precondicionadores gerados variando os descartes e tamanhos de bloco, em relação à mesma matriz. Na última linha desta tabela, temos o quociente dos valores das densidades dos precondicionadores obtidos pelo SBAINV-NS e SBAINV-VAR para o parâmetro de descarte de $0,5$ e $0,1$ ($0,5$ no numerador e $0,1$ no denominador), para cada tamanho de bloco. E na última coluna temos o quociente dos valores das densidades para os blocos de tamanho $b3$ e $b1$.

A Tabela 5 apresenta os tempos de construção do precondicionador (tprec) e os tempos médios de execução do Bi-CGSTAB (tsol), em segundos, quando se varia os descartes e os tamanhos de bloco para os dois precondicionadores, em relação à matriz SHERMAN1. Na última linha desta tabela, temos o quociente dos tempos obtidos pelo SBAINV-NS e SBAINV-VAR para o parâmetro de descarte de $0,5$ e $0,1$ ($0,5$ no numerador e $0,1$ no denominador), para cada tamanho de bloco. E na coluna b3/b1, temos o quociente dos tempos para os blocos de tamanho $b3$ e $b1$, referentes a tprec e tsol. Neste trabalho, apesar de analisarmos os resultados referentes

Tabela 3: Número médio de iterações obtidos na aplicação do Bi-CGSTAB com os precondicionadores SBAINV-NS e SBAINV-VAR na matriz SHERMAN1, com variação dos descartes.

Tolerância	Precondicionador	b1	b2	b3	b3/b1
0,1	SBAINV-NS	30,3	19,2	13,4	0,44
	SBAINV-VAR	25,2	15,2	11,5	0,46
0,2	SBAINV-NS	34,9	28,3	19,5	0,56
	SBAINV-VAR	29,9	22,8	16	0,54
0,3	SBAINV-NS	37,8	30,8	25	0,66
	SBAINV-VAR	30,7	24,3	19,4	0,63
0,4	SBAINV-NS	47,4	43,9	27,9	0,59
	SBAINV-VAR	34,6	28,5	21,4	0,62
0,5	SBAINV-NS	63	47,5	40,2	0,64
	SBAINV-VAR	42,6	29,9	28,1	0,66
0,5/0,1	SBAINV-NS	2,08	2,47	3	
	SBAINV-VAR	1,69	1,97	2,44	

aos tempos consumidos, não consideraremos o tempo como algo determinante pois geralmente nas aplicações, como, por exemplo em problemas de geomecânica [16], o precondicionador é construído uma vez para ser aplicado várias vezes na resolução do sistema linear. Em todo o trabalho, os tempos são representados utilizando notação científica.

A Tabela 6 apresenta as razões das médias das iterações e das densidades entre os precondicionadores (SBAINV-NS no numerador e SBAINV-VAR no denominador) na aplicação do Bi-CGSTAB precondicionado, para diversos tamanhos de bloco, para várias tolerâncias de descarte, com a matriz SHERMAN1. Acima de 1 o Bi-CGSTAB precondicionado pelo SBAINV-VAR faz menos iterações ou possui densidade menor. Ou seja, nesta tabela temos a razão entre as linhas da Tabela 3, na parte referente às iterações e a razão entre as linhas da Tabela 4, na parte referente às densidades, para cada tolerância de descarte.

A partir da Tabela 6, temos que o SBAINV-NS realiza mais iterações que o SBAINV-VAR, com uma média de aproximadamente 32% mais iterações que o SBAINV-VAR, considerando os três tamanhos de bloco. Este número foi obtido fazendo a média de todos os valores da Tabela 6 da parte de iterações, exceto a linha Média (ou fazendo a média dos valores da linha Média na parte de iterações) e diminuindo 1. Também vemos na linha Média desta tabela que o SBAINV-NS tende a ficar mais denso do que o SBAINV-VAR com o aumento do tamanho dos blocos. Ou seja, o SBAINV-VAR é melhor nos dois parâmetros estabelecidos para a análise, já que fornece um menor número médio de iterações e menor densidade de precondicionador, em relação ao SBAINV-NS. O mesmo comportamento foi observado para todas as demais matrizes testadas, como se confirmará na Seção 5.2.

Uma segunda observação em relação ao SBAINV-VAR, a partir da coluna b3/b1 da Tabela 3, é que o aumento do tamanho do bloco diminui o número médio de iterações entre 34% e 56%.

Tabela 4: Densidade dos precondicionadores (ver fórmulas (5.1) e (5.2)) SBAINV-NS e SBAINV-VAR para a matriz SHERMAN1, com variação dos descartes.

Tolerância	Precondicionador	b1	b2	b3	b3/b1
0,1	SBAINV-NS	2,05	3,70	21,75	10,61
	SBAINV-VAR	1,82	2,97	14,52	7.98
0,2	SBAINV-NS	1,50	2,12	11,17	7,45
	SBAINV-VAR	1,52	2,05	8,62	5,67
0,3	SBAINV-NS	1,33	1,69	8,10	6,09
	SBAINV-VAR	1,43	1,84	6,98	4,88
0,4	SBAINV-NS	1,18	1,48	6,50	5,51
	SBAINV-VAR	1,35	1,73	6,09	4.51
0,5	SBAINV-NS	1,04	1,28	5,00	4.81
	SBAINV-VAR	1,28	1,63	5,02	3.92
0,5/0,1	SBAINV-NS	0,51	0,35	0,23	
	SBAINV-VAR	0,70	0,55	0,35	

Além disso, a coluna b3/b1 da Tabela 4, mostra um aumento na densidade de quase 8 vezes para o descarte 0,1 e de quase 4 vezes para o descarte 0,5, quando aumentamos a ordem do bloco homogêneo. Para matrizes esparsas, o aumento de densidade para o descarte 0,1 em relação ao 0,5, significando maior utilização de memória, pode ser compensado pela diminuição em média de 51% do número de iterações do método de Krylov. Encontramos este número calculando o valor da média dos números da linha 0,5/0,1 da Tabela 3 e subtraindo de 1 a inversa deste valor. Isso pode indicar maior vantagem ao se utilizar o valor de $0,1$ como tolerância de descarte [6].

Uma outra indicação para o uso do descarte 0,1, aparece na análise das linhas 0,5/0,1 das Tabelas 3 e 4. Nelas, vemos que há um aumento na razão entre o número de iterações do SBAINV-NS e do SBAINV-VAR com o aumento da ordem do bloco, assim como uma diminuição na razão entre as densidades respectivamente. Porém, blocos densos em matrizes esparsas tendem a induzir uma melhor utilização das memórias rápidas nas arquiteturas usuais de CPU's e GPU's, como pode ser visto em [2].

Analisemos os tempos de execução dos testes na Tabela 5. Primeiramente, comparemos o SBAINV-NS com o SBAINV-VAR. Podemos ver que em todos os testes, o SBAINV-NS precisou de mais tempo para construir o precondicionador (tprec), utilizando, em média, aproximadamente 18% mais tempo que o SBAINV-VAR, para todos os tamanhos de bloco e todas as tolerâncias de descarte. Esse valor foi obtido dividindo o tempo consumido pelo SBAINV-NS pelo tempo consumido pelo SBAINV-VAR em todos os casos, fazendo a média desse valores e subtraindo 1. Em relação ao tempo médio utilizado pelo método iterativo, temos que o SBAINV-NS também precisou de mais tempo na maioria dos casos, exceto em três testes. Porém, as diferenças foram pequenas. Portanto, vejamos que em relação ao tempo, o SBAINV-VAR também apresentou maior vantagem que o SBAINV-NS.

Tabela 5: Tempo de construção do precondicionador (tprec) e tempo médio do Bi-CGSTAB (tsol), em segundos, ao se utilizar os precondicionadores SBAINV-NS (NS) e SBAINV-VAR (VAR) na matriz SHERMAN1, com variação dos descartes.

Tol	Prec	b1		b2		b3		b3/b1	
		tprec	tsol	tprec	tsol	tprec	tsol	tprec	tsol
0,1	NS	1,74e1	8,44e-2	1,71e1	4,53e-2	1,27e1	3,59e-2	7,3e-1	4,25e-1
	VAR	1,44e1	6,87e-2	1,34e1	4,37e-2	9	4,06e-2	6,25e-1	5,91e-1
0,2	NS	1,38e1	8,28e-2	1,12e1	6,4e-2	7,84	5,29e-2	5,68e-1	6,39e-1
	VAR	1,25e1	7,19e-2	9,53	5,78e-2	5,95	5,48e-2	4,76e-1	7,62e-1
0,3	NS	1,25e1	9,37e-2	9,22	7,19e-2	6,17	7,47e-2	4,94e-1	7,97e-1
	VAR	1,15e1	1,17e-1	8,09	6,4e-2	4,95	5,94e-2	4,3e-1	5,08e-1
0,4	NS	1,2e1	1,08e-1	8,44	1e-1	5,2	6,7e-2	4,33e-1	6,2e-1
	VAR	1,09e1	8,59e-2	7,61	7,64e-2	4,3	6,39e-2	3,94e-1	7,44e-1
0,5	NS	1,1e1	1,48e-1	7,95	1,06e-1	4,55	9,46e-2	4,14e-1	6,39e-1
	VAR	1,03e1	1e-1	7,01	7,79e-2	3,81	8,23e-2	3,7e-1	8,23e-1
0,5/0,1	NS	6,32e-1	1,75	4,65e-1	2,34	3,58e-1	2,64		
	VAR	7,1e-1	1,46	5,23e-1	1,78	4,23e-1	2,03		

Vejamos a influência da tolerância de descarte e do tamanho dos blocos nos tempos de execução dos testes. Podemos observar na última linha da Tabela 5, que para o SBAINV-NS e SBAINV-VAR, ao se utilizar $0,5$ como tolerância, o tempo de construção do precondicionador foi menor quando comparado ao valor de tolerância de $0,1$. Vemos que as razões obtidas diminuem com o aumento do bloco. Em relação ao tempo consumido pelo método iterativo, temos que, tanto para o SBAINV-NS quanto para o SBAINV-VAR, ao se utilizar $0,5$ como tolerância, o tempo foi maior do que em relação à tolerância de $0,1$. Também verificamos que as razões aumentaram com o aumento do bloco. Vemos, então, uma vantagem no uso do descarte de $0,5$ no tempo de construção do precondicionador. Observemos que, ao se aumentar o tamanho dos blocos, ocorre a diminuição tanto no tempo de construção do precondicionador, quanto no tempo utilizado pelo método iterativo. De acordo com a coluna b3/b1, ao utilizarmos tamanho de bloco b3, o tempo de construção do precondicionador diminuiu entre 27% e 63% e o tempo de execução do método iterativo diminuiu entre $17,5\%$ e $57,5\%$ que em relação ao tamanho de bloco b1 (caso escalar). Ou seja, quanto maior o tamanho dos blocos, melhores foram os resultados obtidos em relação ao tempo, mesmo tendo produzido matrizes mais densas.

A Figura 1 mostra a relação entre densidade e número médio de iterações quando aumentamos a tolerância nos diversos tamanhos de bloco. Dentro das caixas de texto há a indicação da tolerância de descarte utilizada.O número de iterações reduz com o aumento da densidade. Este, por sua vez, proporcionado pela redução da tolerância de descarte. Caso o gráfico seja observado tendo como variável o fator de tolerância (dentro das caixas), então o aumento do fator de tolerância reduz a densidade e ocasiona o aumento do número de iterações. Esta constatação nos indica que

Tabela 6: Razões entre as iterações e densidades dos precondicionadores (SBAINV-NS/SBAINV-VAR) na aplicação do Bi-CGSTAB precondicionado, com a matriz SHERMAN1, com variação de descartes.

Tolerância	iterações			densidade		
	b1	b2	b3	b1	b2	b3
0,1	1,2	1,26	1,17	1,13	1,25	1,50
0,2	1,17	1,24	1,22	0,99	1,03	1,30
0,3	1,23	1,27	1,29	0,93	0,92	1,16
0,4	1,37	1,54	1,3	0,87	0,86	1,07
0,5	1,48	1,59	1,43	0,81	0,79	1,00
Média	1,29	1,38	1,28	0,95	0,97	1,20

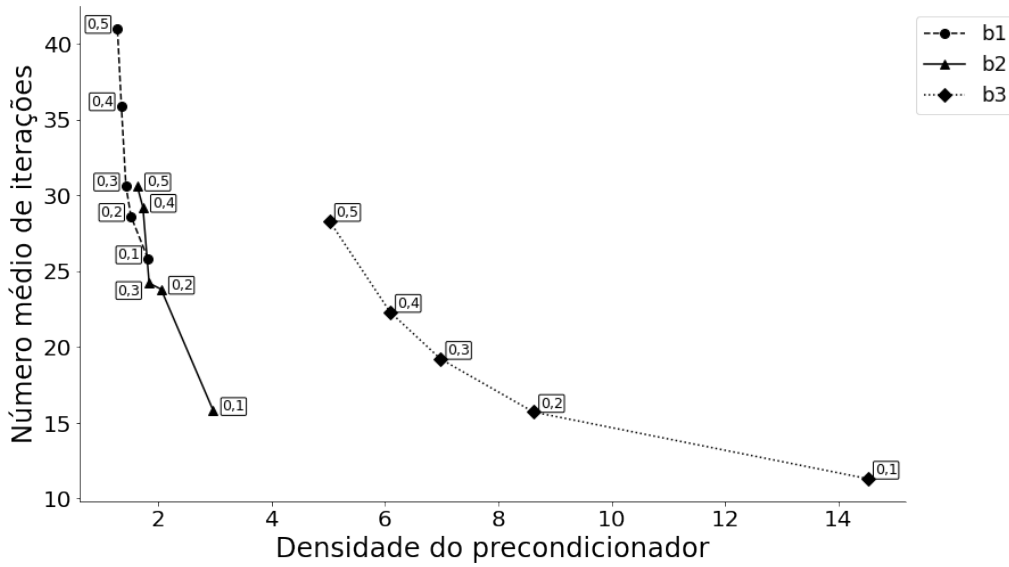

Figura 1: Relação entre a densidade do SBAINV-VAR e o número médio de iterações do Bi-CGSTAB precondicionado, com variação dos descartes, para a matriz SHERMAN1. Os números dentro das caixas de texto indicam a tolerância de descarte utilizada.

outros tipos de descartes devem ser testados para se tentar controlar melhor este crescimento, sendo esta uma indicação para trabalhos futuros.

5.2 Variando blocos e mantendo descarte constante

Utilizando as matrizes descritas na Tabela 1, a Tabela 7 mostra o número médio de iterações do Bi-CGSTAB para os dois precondicionadores quando há variação de tamanho dos blocos homogêneos. Na última coluna desta tabela temos o quociente dos valores do número médio

de iterações para os blocos de tamanho $b3$ e $b1$. Na Tabela 8, apresentamos as densidades dos precondicionadores em relação aos tamanhos dos blocos. Na última coluna desta tabela temos o quociente dos valores das densidades para os blocos de tamanho $b3$ e $b1$. Nestes testes, a tolerância para o descarte foi fixada em 0,1, seguindo as observações realizadas na Seção 5.1.

Tabela 7: Número médio de iterações na aplicação do Bi-CGSTAB precondicionado por SBAINV-NS e SBAINV-VAR, para diversos tamanhos de bloco, com tolerância de descarte de 0,1.

Matriz	Precondicionador	b1	b2	b3	b3/b1
SHERMAN1	SBAINV-NS	30,3	19,2	13,1	0,44
	SBAINV-VAR	25,2	15,2	11,5	0,46
SHERMAN4	SBAINV-NS	30,4	24,4	18,3	0,6
	SBAINV-VAR	25,4	20,4	14,8	0,58
PDE900	SBAINV-NS	19,8	12,4	7,1	0,36
	SBAINV-VAR	15,8	9,1	5	0,32
PDE2961	SBAINV-NS	40,3	18,1	12	0,3
	SBAINV-VAR	31,1	15,8	11,9	0,38
HOR131	SBAINV-NS	28,4	31,1	9,8	0,35
	SBAINV-VAR	25,1	20,1	9,7	0,39
CAGE8	SBAINV-NS	6,1	4,1	4	0,66
	SBAINV-VAR	5,4	4,1	4	0,74

Na Tabela 9, utilizando os dados das Tabelas 7 e 8, apresentamos a razão entre os precondicionadores (SBAINV-NS no numerador e SBAINV-VAR no denominador) em relação aos números médios de iterações e densidade, respectivamente, para diversos tamanhos de bloco, com tolerância para o descarte de 0,1. Acima de 1 o Bi-CGSTAB precondicionado pelo SBAINV-VAR faz menos iterações (na parte de iterações) e é menos denso (na parte de densidade).

Na Tabela 10 temos os tempos, em segundos, obtido pelo SBAINV-NS e SBAINV-VAR na construção do precondicionador (tprec) e os tempos médios obtidos na execução do Bi-CGSTAB (tsol), para todas as matrizes, com o valor de descarte fixado em 0,1.

De acordo com a Tabela 7, observamos que para o SBAINV-VAR, em média, os blocos de tamanho b2 precisaram de aproximadamente 67% (com desvio padrão de 13%) do número médio de iterações necessárias para as versões escalares (b1) enquanto que os blocos b3 precisaram de aproximadamente 48% (com desvio padrão de 16%) em comparação às versões escalares. Estes números foram obtidos fazendo a média e desvio padrão dos quocientes entre o número médio de iterações relativo ao bloco de tamanho b2 e b1 de todas as matrizes. O mesmo raciocínio foi utilizado para os blocos de tamanho b3.

De acordo com a Tabela 9, podemos concluir que o SBAINV-VAR teve um desempenho melhor ou igual no número médio de iterações, que em relação ao SBAINV-NS. Esse fato ocorre em todas as matrizes testadas e em quaisquer tamanhos de blocos utilizados. Observando a linha

Tabela 8: Densidade dos precondicionadores para diversos tamanhos de bloco, com tolerância de descarte de 0,1.

Matriz	Precondicionador	b1	b2	b3	b3/b1
SHERMAN1	SBAINV-NS	2,05	3,70	21,75	10,61
	SBAINV-VAR	1,82	2,97	14,52	7,98
SHERMAN4	SBAINV-NS	1,62	2,58	8,93	5,51
	SBAINV-VAR	1,60	2,46	7,33	4,58
PDE900	SBAINV-NS	2,57	5,42	21,77	8,47
	SBAINV-VAR	2,41	5,02	16,96	7,04
PDE2961	SBAINV-NS	2,54	11,41	37,45	14,74
	SBAINV-VAR	2,28	7,78	22,78	9,99
HOR131	SBAINV-NS	1,81	4,61	13,90	7,68
	SBAINV-VAR	1,71	4,02	11,82	6,91
CAGE8	SBAINV-NS	0,34	2,14	3,50	10,29
	SBAINV-VAR	0,58	2,46	3,94	6,79

Média desta tabela, o SBAINV-NS obteve, em média, aproximadamente 20% mais iterações do que o SBAINV-VAR na versão escalar, 25% mais iterações nos blocos de tamanho b2 e 14% mais iterações nos blocos de tamanho b3. Vemos que o SBAINV-VAR é melhor em 16 testes e empata em dois para o SBAINV-NS.

A Tabela 8 nos permite observar que o SBAINV-VAR produz um precondicionador menos denso do que o SBAINV-NS para cinco matrizes e mais denso para a matriz CAGE8. Como comentado anteriormente, uma densidade maior não é necessariamente uma limitação relevante, já que a localidade dos dados armazenados pelos blocos pode ter um impacto positivo no desempenho das operações de básicas de álgebra linear (uma outra fonte para este fato é [35]). Os precondicionadores nos blocos de ordem b2 foram, em média, 2,53 vezes mais densos que os precondi-

Tabela 9: Razões entre as iterações e densidades dos precondicionadores (SBAINV-NS/SBAINV-VAR), com tolerância de descarte de 0,1.

Matriz	iterações			densidade		
	b1	b2	b3	b1	b2	b3
SHERMAN1	1,2	1,26	1,17	1,13	1,25	1,50
SHERMAN4	1,2	1,2	1,24	1,01	1,05	1,22
PDE900	1,25	1,36	1,42	1,07	1,08	1,28
PDE2961	1,3	1,15	1,01	1,11	1,47	1,64
HOR131	1,13	1,55	1,01	1,06	1,15	1,18
CAGE8	1,13	1	1	0,59	0,87	0,89
Média	1,20	1,25	1,14	0,99	1,14	1,28

Tabela 10: Tempo de construção do precondicionador (tprec) e tempo médio do Bi-CGSTAB (tsol), em segundos, ao se utilizar os precondicionadores SBAINV-NS e SBAINV-VAR, para diversos tamanhos de bloco, com tolerância de descarte de 0,1.

Matriz	Prec	b1		b2		b3	
		tprec	tsol	tprec	tsol	tprec	tsol
SHERMAN1	NS	1,74e1	8,44e-2	1,71e1	4,53e-2	1,27e1	3,59e-2
	VAR	1,44e1	6,87e-2	1,34e1	4,37e-2	9	4,06e-2
SHERMAN4	NS	1,91e1	6,25e-3	1,3e1	4,68e-3	1,02e1	4,69e-3
	VAR	1,67e1	7,81e-3	1,18e1	6,25e-3	7,98	7,81e-3
PDE900	NS	2,24e1	4,45e-2	1,4e1	2,34e-2	6,9	1,87e-2
	VAR	1,82e1	3,7e-2	1,09e1	1,87e-2	4,81	1,72e-2
PDE2961	NS	1,24e2	1,09e-2	1,09e2	1,3e-2	5,37e1	2e-2
	VAR	1,06e2	1,56e-2	6,7e1	1,39e-2	4,17e1	2,66e-2
HOR131	NS	5,04e1	2,66e-2	3,79e1	3,51e-2	9,86	1,09e-2
	VAR	4,85e1	2,81e-2	3,71e1	2,77e-2	9,15	1,41e-2
CAGE8	NS	2,66e1	1,4e-3	2,33e1	1,56e-3	2,03e1	1,56e-3
	VAR	3,13e1	2,19e-3	2,52e1	3,12e-3	2,22e1	4,69e-3

cionadores das versões escalares (note que pela Tabela 1, os valores de b2 são, em média 2,66 vezes maiores que b1) enquanto que os precondicionadores produzidos pelo SBAINV-VAR com os tamanhos de bloco b3 são, em média, 7,19 vezes mais densos que as versões escalares (note que pela Tabela 1, os valores de b3 são, em média 6,83 vezes maiores que b1). Assim, os testes indicaram que a densidade foi aproximadamente proporcional ao aumento na ordem dos blocos. Isto também pode ser visto no gráfico da Figura 2. Nele mostramos a relação entre a densidade dos precondicionadores do SBAINV-VAR e o tamanho de bloco utilizado, com tolerância de descarte de $0,1$ em todas as matrizes. A linha tracejada relaciona as médias das iterações de todas as matrizes com as médias dos tamanhos de bloco de todas as matrizes referentes aos tamanhos b1, b2 e b3.

De acordo com a Tabela 9, observando a linha Média desta tabela, o SBAINV-NS obteve, em média, aproximadamente 99% da densidade do SBAINV-VAR na versão escalar, foi 14% mais denso nos blocos de tamanho b2 e 28% mais denso nos blocos de tamanho b3 que o SBAINV-VAR. Vemos, novamente, que apenas na matriz CAGE8 o SBAINV-NS produziu precondicionadores menos densos que os do SBAINV-VAR. Considerando todos os testes, o SBAINV-VAR produziu, em média, precondiconadores menos densos.

O SBAINV-VAR novamente foi superior nos dois quesitos aqui avaliados. Adicionalmente, observamos na Tabela 9 que para três casos as razões entre as iterações se mantêm estáveis e que para outros três existe uma redução desta razão com o aumento da ordem dos blocos. Também vemos que a densidade do SBAINV-NS tende a crescer mais rapidamente do que densidade do SBAINV-VAR.

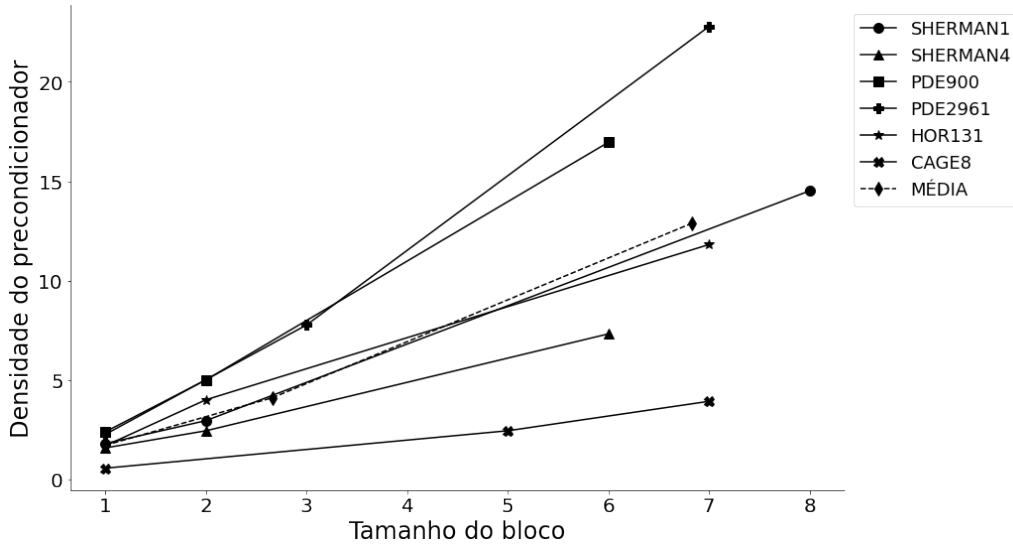

Figura 2: Relação entre a densidade dos precondicionadores do SBAINV-VAR e o tamanho de bloco utilizado, com tolerância de descarte de $0,1$ em todas as matrizes.

Pela Tabela 10 podemos comparar os tempos executados pelos dois precondicionadores. Vejamos que o SBAINV-VAR consome menos tempo para a construção do precondicionador em todos os testes, exceto na matriz CAGE8, onde ele consome mais tempo que o SBAINV-NS. Já, no tempo de solução do método iterativo, as diferenças entre os tempos foram pequenas, tendo casos em que o SBAINV-NS foi melhor e em outros ocorreu o contrário. Então, podemos dizer que, na maior parte dos casos, o SBAINV-VAR foi mais eficiente no uso do tempo.

Em relação aos tamanhos de bloco, observemos que o aumento da ordem dos blocos causou diminuição do tempo consumido para a construção do precondicionador. Em média, os blocos de tamanho b2 precisaram de aproximadamente 77% (com desvio padrão de 13%) do tempo (tprec) necessário para a para as versões escalares (b1) enquanto que os blocos b3 precisaram de aproximadamente 47% (com desvio padrão de 21%) em comparação às versões escalares. Estes números foram obtidos fazendo a média e desvio padrão dos quocientes entre o tempo de construção do precondicionador relativo ao bloco de tamanho b2 e b1 de todas as matrizes. O mesmo raciocínio foi utilizado para os blocos de tamanho b3. Em relação aos tempos executados pelo método iterativo, as diferenças também foram pequenas, tendo tanto casos em que o b3 foi mais rápido que a versão escalar quanto casos em ocorreu o contrário. Observamos que, de uma forma geral, aumentar a ordem dos blocos diminuiu significativamente o tempo consumido nos testes, em todas as matrizes, mesmo tendo produzido precondicionadores mais densos.

A Tabela 11 mostra a razão entre as o número médio de iterações do Bi-CGSTAB sem precondicionamento, apresentadas na Tabela 2, e os números médios das iterações com o SBAINV-VAR, para os três tamanhos de bloco b1, b2 e b3, conferir Tabela 7. Acima de 1 o Bi-CGSTAB precon-

Tabela 11: Razões entre as iterações médias do Bi-CGSTAB sem precondicionamento (no numerador) e precondicionado pelo SBAINV-VAR (no denominador), para os três tamanhos de bloco e com tolerância de descarte de 0,1.

Matriz	b1	b2	b3
SHERMAN1	14,31	23,72	31,36
SHERMAN4	3,15	3,93	5,41
PDE900	4,45	7,74	14,08
PDE2961	4,26	8,39	11,14
CAGE8	2,04	2,68	2,75

dicionado pelo SBAINV-VAR faz menos iterações. De acordo com ela, o método precondicionado se comportou melhor para as cinco matrizes, fazendo em média 82% menos iterações para blocos de tamanho b1, 89% menos iterações para blocos de tamanho b2 e 92% menos iterações para blocos de tamanho b3. Lembrando que para a matriz HOR131, sem precondicionamento, o método não convergiu dentro do número máximo de iterações. Vejamos que a redução percentual do número médio de iterações ao se utilizar precondicionador foi maior para a matriz SHERMAN1, que reduziu, em média 96%, considerando os três tamanhos de bloco. Já a que teve a menor redução foi a matriz CAGE8, com uma redução média de 60%, considerando os três tamanhos de bloco. Esta diferença pode ser vista no gráfico da Figura 3, onde exibimos em gráfico de linhas os valores da Tabela 11. Ou seja, ele exibe as relações entre as razões mencionadas neste parágrafo e cada tamanho de bloco, para todas as matrizes.

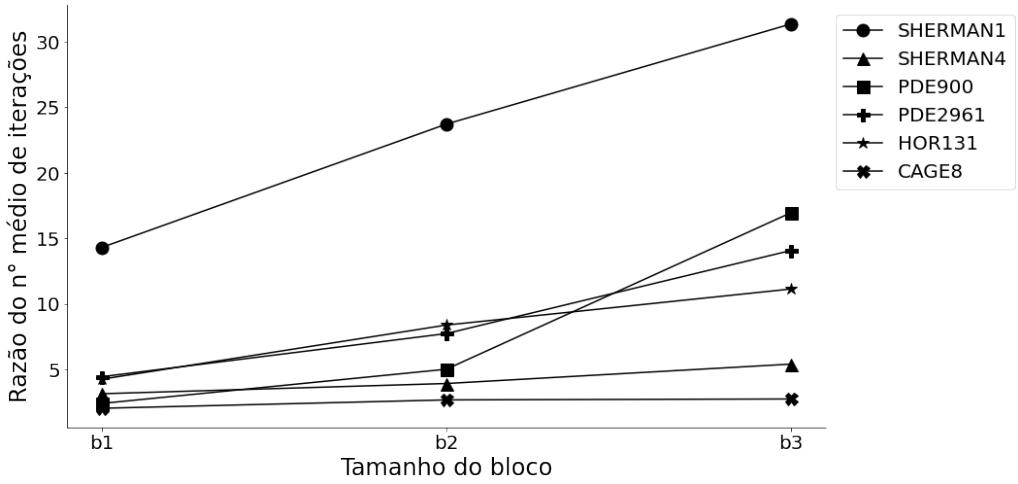

Figura 3: Relação entre as razões do número médio de iterações do sistema precondicionado pelo SBAINV-VAR e não precondicionado e os tamanhos de bloco utilizados para gerar estes precondicionadores.

6 CONCLUSÕES

Propusemos duas variações do BAINV [5] denominadas SBAINV-NS e SBAINV-VAR para matrizes não simétricas, esparsas e não singulares, tendo como base as versões escalares encontradas em [7, 30]. Para provar suas consistências, utilizamos a fundamentação teórica proposta em [3] e as demonstrações dos resultados da Seção 4. Nessas versões também sugerimos o uso do passo de estabilidade para o cálculo de D_{II}, caso A seja positiva definida.

Os principais resultados dos experimentos numéricos indicam que o SBAINV-VAR possui melhor desempenho em comparação ao SBAINV-NS, tanto em número médio de iterações do Bi-CGSTAB, quanto em densidade do precondicionador. Além disso, nestes experimentos, o valor da tolerância do descarte de 0,1 foi adequado e o uso de uma estrutura em blocos representou um ganho importante no número de iterações, ainda que tenha tornado os precondicionadores mais densos, com o aumento da ordem dos blocos.

Como trabalhos futuros, pretendemos utilizar outros critérios de descarte que não sejam puramente pela tolerância da magnitude dos blocos, além de realizar reordenamentos e escalamentos nas matrizes. Pretendemos ainda realizar uma comparação mais detalhada do tempo computacional de aplicação e construção dos precondicionadores. Por fim, realizaremos implementações em C++, tanto com MPI quanto com OpenMP, para medir o desempenho destas alternativas em computadores paralelos híbridos e compará-las a outros precondicionadores conhecidos como, por exemplo, as fatorações incompletas e o multigrid, para matrizes reais oriundas de problemas de simulação de reservatórios de petróleo e de geomecânica.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), processo número 8882.450906/2019-01.

ABSTRACT. We propose two variations of the block approximate inverse preconditioner (BAINV), presented by Benzi, Kouhia and Tůma in 2001. The first variation, the stabilized block approximate inverse for non-symmetric matrices (SBAINV-NS), is used for non-symmetric and non-singular matrices. The second variation, the combined stabilized block approximate inverse (SBAINV-VAR), is based on the relations between the block approximate inverse factors with the block *LDU* factors of *A*, as we will demonstrate, and on the relations between the approximate inverse and Neumann series. We prove the mathematical consistency of these new versions and present the algorithms for each one. We also present the numerical experiments, where we compare the density of the preconditioners and the number of iterations when applying the biconjugate gradient stabilized method (Bi-CGSTAB). The main numerical results indicate that the use of the block structure can increase the performance of the Krylov's iterative method compared to the scalar version. Furthermore, the experiments show that SBAINV-VAR preconditioners, in general, perform less iterations of Bi-CGSTAB and are less dense than SBAINV-NS preconditioners.

Keywords: approximate inverse, block matrices, preconditioners, krylov methods.

REFERÊNCIAS

[1] J.L. A. Buttari & J. Dongarra. A class of parallel tiled linear algebra algorithms for multicore architectures. *Parallel Computing*, **35**(1) (1998), 38–53.

[2] A. Abdelfattah, H. Ltaief, D. Keyes & J. Dongarra. Performance optimization of Sparse Matrix-Vector Multiplication for multi-component PDE-based applications using GPUs. *Concurrency and Computation: Practice and Experience*, **28**(12) (2016), 3447–3465. doi:https://doi.org/10.1002/cpe.3874. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3874`.

[3] M. Almeida, J. Cruz, P. Goldfeld, L.M. Carvalho & M. Souza. Supporting theory for a block approximate inverse preconditioner. *Linear Algebra and its Applications*, **614** (2020), 325–342. doi:10.1016/j.laa.2020.06.017.

[4] S.T. Barnardy & M.J. Grotez. A block version of the spai preconditioner. In "Proceedings of the 9th SIAM conference on Parallel Processing" (1999).

[5] M. Benzi, R. Kouhia & M. Tůma. Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics. *Computer Methods in Applied Mechanics and Engineering*, **190**(49-50) (2001), 6533–6554. doi:10.1016/S0045-7825(01)00235-3.

[6] M. Benzi, C.D. Meyer & M. Tůma. A Sparse Approximate Inverse Preconditioner for the Conjugate Gradient Method. *SIAM Journal on Scientific Computing*, **17**(5) (1996), 1135–1149. doi:10.1137/S1064827594271421.

[7] M. Benzi & M. Tůma. A Sparse Approximate Inverse Preconditioner for Nonsymmetric Linear Systems. *SIAM Journal on Scientific Computing*, **19**(3) (1998), 968–994. doi:10.1137/S1064827595294691.

[8] R. Bridson & W.P. Tang. Refining an approximate inverse. *Journal of Computational and Applied Mathematics*, **123**(1) (2000), 293–306. doi:10.1016/S0377-0427(00)00399-X. Numerical Analysis 2000. Vol. III: Linear Algebra.

[9] R. Bru, J. Cerdán, J. Marín & J. Mas. Preconditioning sparse nonsymmetric linear systems with the Sherman-Morrison formula. *SIAM Journal on Scientific Computing*, **25**(2) (2003), 701–715.

[10] R. Bru, J. Marín, J. Mas & M. Tůma. Balanced incomplete factorization. *SIAM Journal on Scientific Computing*, **30**(5) (2008), 2302–2318.

[11] R. Bru, J. Marín, J. Mas & M. Tůma. Improved balanced incomplete factorization. *SIAM Journal on Matrix Analysis and Applications*, **31**(5) (2010), 2431–2452.

[12] J. Cerdan, T. Faraj, N. Malla, J. Marín & J. Mas. Block approximate inverse preconditioners for sparse nonsymmetric linear systems. *Electronic Transactions on Numerical Analysis*, **37** (2010), 23–40.

[13] E. Chow & Y. Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM Journal on Scientific Computing*, **19**(3) (1998), 995–1023.

[14] M. Ferronato, C. Janna & G. Pini. A generalized block FSAI preconditioner for nonsymmetric linear systems. *Journal of Computational and Applied Mathematics*, **256**(0) (2014), 230–241.

[15] L. Fox. “An introduction to numerical linear algebra”. Mono. Num. Analys. Clarendon Press, Oxford (1964).

[16] L. Gasparine, J.R.P. Rodrigues, D.A. Augusto, L.M. Carvalho, C. Conopoima, P. Goldfeld, J. Panetta, J.P. Ramirez, M. Souza, M.O. Figueireido & V.M.D.M. Leite. Hybrid parallel iterative sparse linear solver framework for reservoir geomechanical and flow simulation. *Journal of Computational Science*, **51** (2021). doi:10.1016/j.jocs.2021.101330.

[17] G.H. Golub & C.F. van Loan. “Matrix Computations”. Johns Hopkins University Press, 4rd ed. (2013).

[18] M.J. Grote & T. Kuckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, **18**(3) (1997), 838–853.

[19] N.J. Higham. “Accuracy and stability of numerical algorithms”. SIAM, 2 ed. (2002).

[20] D.C.S. J. Dongarra, I. S. Duff & H. van der Vorst. “Numerical Linear Algebra for High-Performance Computers”. SIAM, Philadelfia (1998).

[21] C. Janna, M. Ferronato & G. Gambolati. A block FSAI-ILU parallel preconditioner for symmetric positive definite linear systems. *SIAM Journal on Scientific Computing*, **32**(5) (2010), 2468–2484.

[22] C. Janna, M. Ferronato & G. Gambolati. Enhanced block fsai preconditioning using domain decomposition techniques. *SIAM Journal on Scientific Computing*, **35**(5) (2013), S229–S249.

[23] C. Janna, M. Ferronato & G. Gambolati. The use of supernodes in factored sparse approximate inverse preconditioning. *SIAM Journal on Scientific Computing*, **37**(1) (2015), C72–C94.

[24] L.Y. Kolotilina & A.Y. Yeremin. Factorized sparse approximate inverse preconditionings I. theory. *SIAM Journal on Matrix Analysis and Applications*, **14**(1) (1993), 45–58. doi:10.1016/j.cam.2007.07.003.

[25] J.A. Meijerink & H.A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, **31** (1977), 148–162. doi:10.1090/S0025-5718-1977-0438681-4.

[26] G. Meurant. “Computer solution of large linear systems”. Elsevier (1999).

[27] R.J. Plemmons. M-Matrix Characterizations. I—Nonsingular M-Matrices. *Linear Algebra and Its Applications*, **18**(2) (1977), 175–188. doi:10.1016/0024-3795(77)90073-8.

[28] A. Rafiei. Left-looking version of AINV preconditioner with complete pivoting strategy. *Linear Algebra and its Applications*, **445** (2014), 103–126. doi:10.1016/j.laa.2013.11.046.

[29] A. Rafiei, M. Bollhöfer & F. Benkhaldoun. A block version of left-looking AINV preconditioner with one by one or two by two block pivots. *Applied Mathematics and Computation*, **350** (2019), 366–385. doi:10.1016/j.amc.2019.01.012.

[30] A. Rafiei & F. Toutounian. New breakdown-free variant of AINV method for nonsymmetric positive definite matrices. *Journal of Computational and Applied Mathematics*, **219**(1) (2008), 72–80. doi:10.1016/j.cam.2007.07.003.

[31] Y. Saad. “Iterative Methods for Sparse Linear Systems”. SIAM, 2nd ed. (2003).

[32] D.K. Salkuyeh. A sparse approximate inverse preconditioner for nonsymmetric positive definite matrices. *Journal of Applied Mathematics & Informatics*, **28**(5-6) (2010), 1131–1141.

[33] M. Sedlacek. “Sparse Approximate Inverses for Preconditioning, Smoothing, and Regularization”. Ph.D. thesis, Universität München (2012).

[34] H. van der Vorst. BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of non-symmetric linear systems. *SIAM Journal on Scien. and Stat. Computing*, **13**(2) (1992), 631–644.

[35] S. Williams, A. Waterman & D. Patterson. Roofline: An Insightful Visual Performance Model for Multicore Architectures. *Commun. ACM*, **52**(4) (2009), 65–76. doi:10.1145/1498765.1498785. URL `https://doi.org/10.1145/1498765.1498785`.

[36] A. Yeremin, , L. Kolotinila & A. Nikishin. Factorized sparse approximate inverse preconditionings. III. iterative construction of preconditioners. *Journal of Mathematical Sciences*, **101**(4) (2000), 3237–3254.

APÊNDICE A DEMONSTRAÇÃO DA CONSISTÊNCIA DO ALGORITMO 1

Aqui demonstraremos a consistência do Algoritmo 1, primeiramente, sem considerar os descartes. Ou seja, se A for uma matriz N × N em bloco não singular e caso o Algoritmo 1, sem os descartes, não quebre, então as matrizes D, Z e W são não singulares, com D bloco diagonal e tais que $A^{-1} = ZD^{-1}W$. Para as demonstrações, denotamos como $A_{I_0:I_f,J_0:J_f}$ a submatriz-bloco

$$A_{I_0:I_f,J_0:J_f} = \begin{bmatrix} A_{I_0,J_0} & A_{I_0,J_0+1} & \cdots & A_{I_0,J_f} \\ A_{I_0+1,J_0} & A_{I_0+1,J_0+1} & \cdots & A_{I_0+1,J_f} \\ \vdots & \vdots & \ddots & \vdots \\ A_{I_f,J_0} & A_{I_f,J_0+1} & \cdots & A_{I_f,J_f} \end{bmatrix}.$$

Por coveniência, denotamos "$1:N$" simplesmente por ":". Por exemplo, o vetor-bloco A_J e a linha-bloco A_{J*} também podem ser expressos como $A_{:,J:J}$ e $A_{J:J,:}$, respectivamente.

Verifiquemos, agora, alguns resultados.

Lema 1.1. *Caso o Algoritmo 1, sem os descartes, não quebre (i.e., D_{II}'s singulares não são gerados), ele fornece uma matriz Z bloco triangular superior e uma matriz W bloco triangular inferior. Além disso, os blocos diagonais de Z e W são compostos pela identidade.*

Proof. Façamos por indução em I para provar que, para $I = 0, 1, \ldots, N-1$,

$$E_L^T Z_J^{(I)} = W_{J*}^{(I)} E_L = \delta_{LJ} \quad \forall\, J, L \text{ tal que } I < J \leq L \leq N, \tag{A.1}$$

onde δ_{LJ} é dado como

$$\delta_{LJ} = \begin{cases} \text{bloco identidade}, & \text{se } L = J \\ 0, & \text{se } L \neq J \end{cases}.$$

A expressão (A.1) é notavelmente verdadeira para $I = 0$, já que $Z_J^{(0)} = E_J$ e $W_{J*}^{(0)} = E_J^T$. Agora, assumamos que (A.1) é verdadeira para $0 \leq I < N-1$. A partir disso e da linha 8 do Algoritmo 1,

$$E_L^T Z_J^{(I+1)} = E_L^T Z_J^{(I)} - E_L^T Z_{I+1}^{(I)} D_{(I+1)(I+1)}^{-1} M_J^{(I)} = E_L^T Z_J^{(I)} = \delta_{LJ},$$

e

$$W_{J*}^{(I+1)} E_L = W_{J*}^{(I)} E_L - Q_J^{(I)} D_{(I+1)(I+1)}^{-1} W_{I+1*}^{(I)} E_L = W_{J*}^{(I)} E_L = \delta_{LJ},$$

o que completa a indução. □

Lema 1.2. *Se o Algoritmo 1 sem descartes não quebrar, temos que, para qualquer $1 \leq J \leq N$ fixo,*

$$Z_J = Z_J^{(K)} - \sum_{I=K+1}^{J-1} Z_I D_{II}^{-1} M_J^{(I-1)}, \qquad W_{J*} = W_{J*}^{(K)} - \sum_{I=K+1}^{J-1} Q_J^{(I-1)} D_{II}^{-1} W_{I*} \tag{A.2}$$

para todo $0 \leq K \leq I-1$. **Proof.** Segue diretamente das linhas 3, 8 e 9 do Algoritmo 1. □

Theorem 1.1. *Seja A uma matriz $N \times N$ em bloco tal que $A_{1:J,1:J}$ é não singular para $J = 1, 2, \ldots, N$. Então, Algoritmo 1, sem os descartes, não quebra e retorna uma matriz bloco diagonal D não singular e matrizes em bloco Z e W não singulares tais que $A^{-1} = ZD^{-1}W$ (ou, de forma equivalente, $WAZ = D$).*

Proof. Vamos fazer por indução. É suficiente provar que as hipóteses em H_J são verdadeiras para todo $J \in \{1,\ldots,N\}$.

$$H_J : \begin{cases} A_{I*}Z_J = 0, & I \in \{1,2,\ldots,J-1\}; \quad \text{(A.3a)} \\ W_{J*}A_I = 0, & I \in \{1,2,\ldots,J-1\}; \quad \text{(A.3b)} \\ W_{I*}AZ_J = W_{J*}AZ_I = 0, & I \in \{1,2,\ldots,J-1\}; \quad \text{(A.3c)} \\ W_{J*}AZ_J = D_{JJ}; \quad \text{(A.3d)} \\ W_{J*}A_J = D_{JJ}; \quad \text{(A.3e)} \\ D_{JJ} \text{ é não singular}; \quad \text{(A.3f)} \end{cases}$$

As hipóteses (A.3c), (A.3d), e (A.3f) são suficientes para chegar no resultado desejado. Já as hipóteses (A.3a), (A.3b), e (A.3e) são auxiliares.

Para $J = 1$, as condições (A.3a), (A.3b) e (A.3c) são alcançadas por vacuidade e as condições (A.3d), (A.3e) e (A.3f) seguem pelos fatos de que $Z_1 = E_1$, $W_{1*} = E_1^T$ e $D_{11} = A_{11}$. Para o passo de indução, assumimos que $H_1, H_2, \ldots, H_{J-1}$ são verdadeiras e utilizadas para demonstrar H_J.

Para provar (A.3a), nós primeiramente multiplicamos a linha 8 do Algorimto 1 por A_{I*} para $I < J$, obtendo

$$A_{I*}Z_J^{(I)} = A_{I*}Z_J^{(I-1)} - A_{I*}Z_I D_{II}^{-1} M_J^{(I-1)}.$$

Temos que o primeiro termo do lado direito é $M_J^{(I-1)}$ (linha 6 do Algoritmo 1) e, sendo $A_{I*}Z_I = D_{II}$ (linha 4 do Algoritmo 1), nós concluímos que

$$A_{I*}Z_J^{(I)} = 0 \quad \forall\, I < J. \tag{A.4}$$

Agora, a partir do Lemma 1.2, para $1 \leqslant K < J$,

$$A_{K*}Z_J = A_{K*}Z_J^{(K)} - \sum_{I=K+1}^{J-1} A_{K*}Z_I D_{II}^{-1} M_J^{(I-1)}.$$

Foi provado em (A.4) que o primeiro termo do lado direito da expressão é zero. Como a hipótese de indução garante que $A_{K*}Z_I$ é zero para $1 \leqslant K < I < J$, o somatório do lado direito também é zero, provando (A.3a).

Analogamente, para provar (A.3b) primeiramente multiplicamos do lado direito a linha 9 do Algoritmo 1 por A_I para $I < J$, obtendo

$$W_{J*}^{(I)}A_I = W_{J*}^{(I-1)}A_I - Q_J^{(I-1} D_{II}^{-1} W_{I*}A_I.$$

Temos que o primeiro termo do lado direito é $Q_J^{(I-1)}$ (linha 7 do Algoritmo 1) e, sendo $W_{I*}A_I = D_{II}$ (pela hipóteses de indução (A.3e)), concluímos que

$$W_{J*}^{(I)}A_I = 0 \quad \forall\, I < J. \tag{A.5}$$

Agora, a partir do Lemma 1.2, para $1 \leqslant K < J$,

$$W_{J*}A_K = W_{J*}^{(K)}A_K - \sum_{I=K+1}^{J-1} Q_J^{(I-1)} D_{II}^{-1} W_{I*}A_K.$$

Foi provado em (A.5) que o primeiro termo do lado direito da expressão é zero. Como a hipótese de indução garante que $W_{I*}A_K$ é zero para $1 \leqslant K < I < J$, o somatório do lado direito também é zero, provando (A.3b).

A partir do que já foi provado, temos que a $N \times J$ bloco-matriz Z_J e $W_{1:J,:}A$ são bloco triangulares superiores e a $J \times N$ bloco matriz $W_{1:J,:}$ e AZ_J são bloco triangular inferiores. Notemos que $W_{1:J,:}AZ_J$ é o produto de duas matrizes bloco triangular inferiores, dado por $W_{1:J,:}(AZ_J)$, resultando em uma matriz bloco triangular inferior. Da mesma forma, $W_{1:J,:}AZ_J$ é o produto de duas matrizes bloco triangular superiores, dado por $(W_{1:J,:}A)Z_J$, resultando em uma matriz bloco triangular superior. Isto prova (A.3c).

Escrevendo a matriz identidade como $I = \sum_{K=1}^{N} E_K E_K^T$, temos que

$$W_{J*}AZ_J = \sum_{K=1}^{N} (W_{J*}E_K)(E_K^T A)Z_J = \\ = \sum_{K=1}^{J-1} (W_{J*}E_K)A_{K*}Z_J + (W_{J*}E_J)A_{J*}Z_J + \sum_{K=J+1}^{N} (W_{J*}E_K)A_{K*}Z_J.$$

Notemos que o primeiro somatório do último termo é igual a zero por conta da hipótese de indução (A.3a) e que o segundo somatório é zero de acordo com o Lema 1.1. Então chegamos ao (A.3d) pelo Lema 1.1 e linha 4 do Algoritmo 1.

A partir de (A.3d), Lema 1.1 e linha 1 do Algoritmo 1 nós temos

$$D_{JJ} = W_{J*}AZ_J = W_{J*}AE_J - \sum_{I=1}^{J-1} W_{J*}AZ_I D_{II}^{-1} M_J^{(I-1)}. \tag{A.6}$$

O somatório no último termo é zero por conta de (A.3c). Portanto, temos $D_{JJ} = W_{J*}A_J$, provando (A.3e).

A hipóteses (A.3c) e (A.3d) resultam que

$$W_{1:J,:}AZ_{:,1:J} = diag(D_{11}, \ldots, D_{JJ}),$$

uma matriz bloco diagonal com D_{II}, $1 \leqslant I \leqslant J$, na sua diagonal em blocos. Além disso, o Lema 1.1 garante que $Z_{J+1:N,1:J}$ e $W_{1:J,J+1:N}$ são zero. Desta forma

$$diag(D_{11}, \ldots, D_{JJ}) = W_{1:J,:}AZ_{:,1:J} = W_{1:J,1:J}A_{1:J:,1:J}Z_{1:J:,1:J}.$$

As matrizes $Z_{1:J,1:J}$ e $W_{1:J,1:J}$ são bloco triangulares e seus blocos diagonais são formados por identidades, portanto elas são não singulares. Como $A_{1:J:,1:J}$, por hipótese, é também não singular, chegamos a (A.3f).

□

Corolário 1.0.1. *Considerando os elementos do Algoritmo 1, sem descartes, temos que* $D_{II} = W_{I*}A_I$.

Com isso, demonstramos a consistência do algoritmo de biconjugação em blocos que é base do SBAINV-NS. No SBAINV-NS podem ser efetuados descartes em Z e W com o uso do $descarte_I$ nas linhas 8 e 9 e, portanto, obtemos uma aproximação $A^{-1} \approx ZD^{-1}W$. Vejamos que no caso do SBAINV-NS, mesmo se A for não singular, existe a possibilidade de quebra. A seguir conferimos mais um resultado SBAINV-NS.

Lema 1.3. *Caso o Algoritmo 1 não quebre, ele gera uma matriz bloco triangular inferior Z e uma matriz bloco triangular superior W. Além disso, os bloco diagonais de Z e W são formados pela identidade.*

Proof. A prova é análoga a do Lema 1.1. As únicas observações adicionais são de que $E_I^T descarte_I(V) = E_I^T V$ e que $E_L^T V = 0 \Rightarrow E_L^T descarte_I(V) = 0$ (assim como, $descarte_I(V)E_I = VE_I$ e que $VE_I = 0 \Rightarrow descarte_I(V)E_I = 0$). □

Desta forma, sendo A uma matriz em blocos não singular, se o Algoritmo 1 não quebrar então D é bloco diagonal não singular e Z e W são bloco triangulares inferior e superior, respectivamente, com blocos diagonais formados pela identidade e, portanto, também não singulares.