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

Propriedades de Métodos de Biconjugação e Precondicionadores com Inversa Aproximada

Julia S. Cruz[1]
PPGM/FEN/UERJ, Rio de Janeiro, RJ
Moisés C. Almeida[2]
IFRJ, Rio de Janeiro, RJ
Luiz M. Carvalho[3]
DMA/IME/UERJ, Rio de Janeiro, RJ

Resumo. Este trabalho discute propriedades relativas ao método AINV ("Approximate Inverse"), que fornece a fatoração da inversa aproximada de matrizes grandes e esparsas, servindo como precondicionador em sistemas lineares de grande porte. Analisamos algumas características do método de biconjugação, base do AINV, como a quebra do algoritmo e as relações do método com a fatoração LDU da matriz A. O estudo de tais características ajuda a entender as principais versões do AINV existentes na literatura, assim como auxilia o processo de desenvolvimento de novas variações.

Palavras-chave. Precondicionador, AINV, Biconjugação, Sistemas Lineares

1 Introdução

Em diversas áreas da ciência e da indústria, é necessária a resolução de sistemas lineares grandes e esparsos da forma $Ax = b$, onde $A \in \mathbb{R}^{n\times n}$ é não singular, $b \in \mathbb{R}^n$ é o lado direito conhecido, e $x \in \mathbb{R}^n$ é o vetor solução. Sistemas lineares, onde n é muito grande, só podem ser resolvidos por métodos iterativos, já que os métodos diretos são impraticáveis nesses casos. Entretanto, métodos iterativos, em particular os métodos de projeção de subespaços de Krylov, possuem convergência lenta, tornando imprescindível o uso de precondicionadores. Precondicionadores são utilizados para transformar um sistema em outro com a mesma solução, porém computacionalmente mais fácil de se solucionar. Em geral, a ideia é construir um operador M tal que $MAx = Mb$ ou $AMy = b$ ($x = My$) sejam sistemas com convergências melhores e mais rápidas que o do sistema original. Uma alternativa é o uso de precondicionadores que aproximem a inversa de A (pois calcular a inversa exata de A é computacionalmente inviável), já que sua aplicação 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.

Em 1996, foi proposto o algoritmo "Approximate Inverse Preconditioner (AINV)", que calcula a inversa aproximada de matrizes simétricas positivas definidas (SPD), [1]. Pouco tempo depois, em 1998, uma versão generalizada do AINV foi proposta [2], onde A pode ser qualquer matriz esparsa não singular. A principal ideia do método é utilizar uma A-biortogonalização, que será referida aqui como biconjugação, que é um processo onde se obtém as aproximações das matrizes

[1]julia-seki@hotmail.com.
[2]moisesceni@gmail.com.
[3]luizmc@gmail.com.

inversíveis Z, W e D, tais que $A^{-1} = ZD^{-1}W^{\mathsf{T}}$, onde Z e W são triangulares superiores unitárias e D é diagonal. Calcular tais matrizes de forma exata é de ordem $\mathcal{O}(n^3)$, sendo computacionalmente inviável para problemas de grande magnitude. Portanto, alguns valores são descartados durante o processo, obtendo aproximações. Posteriormente, diversas variações do AINV foram publicadas, modificando-se alguns aspectos do algoritmo, como, por exemplo, as estratégias de descarte ou procedimentos que evitassem sua quebra [3, 6, 10]. Algumas variações fizeram uso do AINV como método livre de quebra para se computar a fatoração LDU de A a partir de suas relações com os fatores Z, W e D [4, 8, 9].

O objetivo deste trabalho é descrever e analisar algumas características do método de biconjugação, que é a base do AINV, como sua quebra e relações com a fatoração LDU de A. O estudo dessas características serve de fomento para as variações do AINV apresentadas em diversos trabalhos da literatura. O artigo é organizado da seguinte forma: Seção 2 que descreve o algoritmo de biconjugação, Seção 3 que analisa as principais características do método em aritmética exata e Seção 4 com as conclusões.

2 Biconjugação Completa

Em 1998, foi proposta a generalização do precondicionador AINV em [1]. Seja $A \in \mathbb{R}^{n\times n}$ matriz esparsa e não singular. O AINV busca encontrar aproximações de três matrizes inversíveis W, Z e D tais que $W^T AZ = D$, onde D é diagonal, Z é triangular superior unitária e W^T é triangular inferior unitária. O algoritmo é baseado no método de biconjugação [5]. Este método fornece dois conjuntos de vetores, $\{z_i\}_{i=1}^n$ e $\{w_i\}_{i=1}^n$ biconjugados em relação à A, isto é, $w_i^T A z_j = 0$ para $i \neq j$. Esses vetores são as colunas das matrizes Z e W:

$$Z = [z_1, z_2, ..., z_n], W = [w_1, w_2, ..., w_n], \text{ tais que } W^T AZ = D = \begin{bmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_n \end{bmatrix}.$$

Os $p_i's$, $1 \leq i \leq$, são chamados pivôs. O processo de biconjugação, apresentado no Algoritmo 1, é iterativo e se inicia escolhendo Z_0 e W_0 como matrizes identidade $I_{n\times n}$. No final do método, são fornecidas W, D e Z tais que $A^{-1} = ZD^{-1}W^T$ ou $W^T AZ = D$. As matrizes W e Z geralmente são densas, mesmo A sendo esparsa. Portanto, o AINV descarta algumas entradas de Z e W durante as iterações. No Algoritmo 1, a_i^T e c_i^T são consideradas a i-ésima linha de A e A^T, respectivamente.

Existem duas formas de fazer o processo de biconjugação, conhecidas como right-looking, apresentado no Algoritmo 1, e left-looking. A principal diferença entre eles é a ordem de produção dos elementos, mas produzem os mesmos valores finais e intermediários. Neste trabalho, falaremos apenas da versão right-looking (todos os resultados são obtidos de forma análoga para a versão left-looking).

DOI: 10.5540/03.2021.008.01.0466

Algorithm 1: Biconjugação right-looking

Data: matriz A não singular e esparsa
Result: matriz D diagonal não singular e matrizes Z and W não singulares tais que $A^{-1} = ZD^{-1}W^{\mathsf{T}}$

$z_i^{(0)}, w_i^{(0)} \leftarrow e_i$, $i = 1, \ldots, n$;
for $i \leftarrow 1$ **to** n **do**
  $z_i \leftarrow z_i^{(i-1)}$;
  $w_i \leftarrow w_i^{(i-1)}$;
  $p_i \leftarrow a_i^{\mathsf{T}} z_i$;
  $q_i \leftarrow c_i^{\mathsf{T}} w_i$;
  for $j \leftarrow i+1$ **to** n **do**
    $r_j^{(i-1)} \leftarrow a_i^{\mathsf{T}} z_j^{(i-1)}$;
    $s_j^{(i-1)} \leftarrow c_i^{\mathsf{T}} w_j^{(i-1)}$;
    $z_j^{(i)} \leftarrow z_j^{(i-1)} - z_i \dfrac{r_j^{(i-1)}}{p_i}$;
    $w_j^{(i)} \leftarrow w_j^{(i-1)} - w_i \dfrac{s_j^{(i-1)}}{q_i}$;
$D \leftarrow Diag(p_1, p_2, ..., p_n)$, $Z \leftarrow (z_1, z_2, ..., z_n)$, e $W \leftarrow (w_1, w_2, ..., w_n)$

3 Propriedades do algoritmo de biconjugação

Aqui abordamos algumas características e relações do método de biconjugação.

3.1 Quebra no algoritmo

As principais dificuldades advindas desse procedimento estão relacionadas aos pivôs: é possível que p_i seja zero, ou muito pequeno, o que pararia o processo de execução do algoritmo. Quando tais situações acontecem, dizemos que o algoritmo quebra.

Remark 3.1. *Se o Algoritmo 1 não quebrar, temos $a_j^{\mathsf{T}} z_i = c_j^{\mathsf{T}} w_i = 0$, para $1 \leqslant j < i \leqslant n$, e $z_{ii} = w_{ii} = 1$, para $1 \leqslant i \leqslant n$, então*

$$p_i = a_i^{\mathsf{T}} z_i = w_i^{\mathsf{T}} A z_i = c_i^{\mathsf{T}} w_i = q_i. \tag{1}$$

Assim, o produto escalar $q_i = c_i^{\mathsf{T}} w_i$ pode ser substituído por p_i, ou vice versa.

Algumas condições podem evitar a quebra do algoritmo, como veremos a seguir.

Proposition 3.1. *Seja A uma matriz inversível de ordem n e p_i's ($1 \leq i \leq n$) os pivôs gerados aplicando-se o Algoritmo 1 em A, sem fazer descartes e tal que ele não quebre ($p_i \neq 0, \forall i$). Se denotarmos por Δ_i o i-ésimo menor principal líder de A, com $\Delta_0 = 1$, então*

$$p_i = \frac{\Delta_i}{\Delta_{i-1}}, (i = 1, \ldots, n). \tag{2}$$

Demonstração. Se o Algoritmo 1 não quebrar, são geradas W, Z e D, tais que

$$A = W^{-T} D Z^{-1} \tag{3}$$

onde W^{-T} é triangular inferior unitária, Z^{-1} é triangular superior unitária e D é diagonal cujas entradas são formadas pelos pivôs p_i. Portanto, considerando (3), temos que

$$\Delta_i = p_1 \cdot ... \cdot p_i, (i = 1, \ldots, n). \tag{4}$$

Façamos indução em i. Para $i = 1$, por (4), temos que $p_1 = \Delta_1 = \dfrac{\Delta_1}{\Delta_0}$. Consideremos, então, que (2) vale para os pivôs $p_0, ..., p_{i-1}$ e provemos que vale para p_i. Por (4) temos

$$\Delta_i = p_1 \cdot ... \cdot p_{i-1} \cdot p_i \Rightarrow p_i = \frac{\Delta_i}{p_1 \cdot ... \cdot p_{i-1}} = \frac{\Delta_i}{\Delta_1 \cdot \dfrac{\Delta_2}{\Delta_1}\dfrac{\Delta_3}{\Delta_2} \cdot ... \cdot \dfrac{\Delta_{i-1}}{\Delta_{i-1}}} = \frac{\Delta_i}{\Delta_{i-1}}.$$

□

Portanto, de acordo com a Proposição 3.1, o algoritmo completo não quebra se e somente se todos os menores principais líderes de A são não nulos. Como consequência disso, o algoritmo não quebra para matrizes SPD. Também temos que para qualquer matriz não-singular A existe uma matriz permutação P (ou Q), tal que os menores principais líderes de PA (AQ) são diferentes de zero [7]. Desta forma, multiplicar A por tais matrizes antes de aplicar o algoritmo pode servir de ferramenta para evitar a quebra.

3.2 Relação com os fatores L e U

De acordo com o Algoritmo 1, W e Z são triangulares superior unitárias e

$$A = W^{-\mathsf{T}} D Z^{-1}. \tag{5}$$

Se a matriz diagonal D for não singular (ou seja, quando o Algoritmo 1 não quebra) então a fatoração LDU de A é única e (5) corresponde a essa fatoração. Ou seja,

$$W = L^{-\mathsf{T}}, \quad Z = U^{-1} \tag{6}$$

e D é a mesma matriz. Com isso, temos os seguintes resultados:

Lemma 3.1. *Considerando o Algoritmo 1, para qualquer j fixo, com $1 \leqslant j \leqslant n$,*

$$z_j = z_j^{(i)} - \sum_{k=i+1}^{j-1} z_k \frac{r_j^{(k-1)}}{p_k} \text{ e } w_j = w_j^{(i)} - \sum_{k=i+1}^{j-1} w_k \frac{s_j^{(k-1)}}{q_k} \tag{7}$$

para $0 \leqslant i < j$.

Demonstração. As igualdades (7) seguem imediatamente das linhas 3, 5, 8 e 10 do Algoritmo 1, para a primeira igualdade, e das linhas 4, 6, 9 e 11 do Algoritmo 1 para a segunda igualdade. □

Proposition 3.2. *Seja $A \in \mathbb{R}^{n \times n}$ não singular com todos os seus menores principais diferentes de zero. Seja a fatoração $A = LDU$ única, onde L é triangular inferior unitária, U é triangular superior unitária e D é diagonal não singular. Com $1 \leqslant i \leqslant j \leqslant n$, assumimos que $L = [l_{ji}]$, $U = [u_{ij}]$ e $p_i = q_i$ são as entradas da diagonal D. Além disso, seja $r_j^{(i-1)}$ e $s_j^{(i-1)}$, onde $1 \leqslant i < j \leqslant n$, os escalares produzidos pelo Algoritmo 1. Então,*

$$u_{ij} = \frac{r_j^{(i-1)}}{p_i} \text{ e } l_{ji} = \frac{s_j^{(i-1)}}{q_i}. \tag{8}$$

Demonstração. Seja e_j o j-ésimo vetor canônico, pelo Lema 3.1, temos $z_j = e_j - \sum_{k=1}^{j-1} z_k \frac{r_j^{(k-1)}}{p_k}$

Fixando j, com $1 \leqslant i < j \leqslant n$, então,

$$w_i^{\mathsf{T}} A z_j = w_i^{\mathsf{T}} A e_j - \sum_{k=1}^{j-1} w_i^{\mathsf{T}} A z_k \frac{r_j^{(k-1)}}{p_k} \Rightarrow 0 = w_i^{\mathsf{T}} A e_j - \frac{r_j^{(i-1)}}{p_i} w_i^{\mathsf{T}} A z_i \Rightarrow r_j^{(i-1)} = w_i^{\mathsf{T}} A e_j, \tag{9}$$

com $1 \leqslant i < j \leqslant n$, pois $w_i A z_j = 0$, para $j \neq i$, e $w_i^{\mathsf{T}} A z_i = p_i$, de (1). De (5), (6) e (9)

$$Z^{-1} = U = D^{-1} W^{\mathsf{T}} A \Rightarrow u_{ij} = \frac{w_i^{\mathsf{T}} A e_j}{p_i} = \frac{r_j^{(i-1)}}{p_i}, \text{ com } 1 \leqslant i < j \leqslant n.$$

Analogamente, pelo Lema 3.1, $s_j^{(i-1)} = z_i^{\mathsf{T}} A^{\mathsf{T}} e_j$ e, portanto

$$W^{-\mathsf{T}} = L = AZD^{-1} \Rightarrow l_{ji} = \frac{z_i^{\mathsf{T}} A^{\mathsf{T}} e_j}{q_i} = \frac{s_j^{(i-1)}}{q_i}, \text{ com } 1 \leqslant i < j \leqslant n;$$

□

Além das entradas de L^{-1} and U^{-1}, o método de biconjugação também computa implicitamente as entradas dos fatores L e U de A. Esta computação pode ser feita adaptando-se o laço interno do Algoritmo 1, como é mostrado no Algoritmo 2.

Algorithm 2: Computando as entradas dos fatores L e U.

```
1 for j ← i + 1 to n do
2     r_j^(i-1) ← a_i^T z_j^(i-1);
3     s_j^(i-1) ← c_i^T w_j^(i-1);
4     l_ji = s_j^(i-1) / q_i;
5     u_ij = r_j^(i-1) / p_i;
6     z_j^(i) ← z_j^(i-1) − z_i u_ij;
7     w_j^(i) ← w_j^(i-1) − w_i l_ji;
```

Proposition 3.3. *Sejam p_i, q_i, $r_j^{(i-1)}$'s e $s_j^{(i-1)}$'s os escalares produzidos pelo Algoritmo 1; Portanto,*

$$p_i = a_{ii} - \sum_{k=1}^{i-1} \frac{r_i^{(k-1)} s_i^{(k-1)}}{p_k} = q_i, \text{ para } 1 \leqslant i \leqslant n. \tag{10}$$

$$r_j^{(i-1)} = a_{ij} - \sum_{k=1}^{i-1} \frac{r_j^{(k-1)} s_i^{(k-1)}}{p_k} \text{ e} \tag{11}$$

$$s_j^{(i-1)} = a_{ji} - \sum_{k=1}^{i-1} \frac{r_i^{(k-1)} s_j^{(k-1)}}{q_k}, \text{ para } 1 \leqslant i \leqslant j \leqslant n. \tag{12}$$

Demonstração. Utilizando as mesmas hipóteses da Proposição 3.2, seja $A = LDU$, $L = [l_{ji}]$, $U = [u_{ij}]$ e p_i as entradas diagonais de D, então as entradas da diagonal de A, a_{ii} $(1 \leqslant i \leqslant n)$, podem ser expressas como

$$a_{ii} = \sum_{k=1}^{i} l_{ik} p_k u_{ki} = l_{ii} p_i u_{ii} + \sum_{k=1}^{i-1} l_{ik} p_k u_{ki}, \text{ considerando (8)},$$

DOI: 10.5540/03.2021.008.01.0466

$$a_{ii} = p_i + \sum_{k=1}^{i-1} \frac{s_i^{(k-1)}}{q_k} p_k \frac{r_i^{(k-1)}}{p_k}, \text{ já que } p_k = q_k, \;\; p_i = a_{ii} - \sum_{k=1}^{i-1} \frac{r_i^{(k-1)} s_i^{(k-1)}}{p_k} = q_i,$$

provando (10). Para $1 \leqslant i \leqslant j \leqslant n$

$$a_{ij} = \sum_{k=1}^{i} l_{ik} p_k u_{kj} = l_{ii} p_i u_{ij} + \sum_{k=1}^{i-1} l_{ik} p_k u_{kj}, \text{ considerando (8)},$$

$$a_{ij} = p_i \frac{r_j^{(i-1)}}{p_i} + \sum_{k=1}^{i-1} \frac{s_i^{(k-1)}}{q_k} p_k \frac{r_j^{(k-1)}}{p_k}, \text{ já que } p_k = q_k, \;\; r_j^{(i-1)} = a_{ij} - \sum_{k=1}^{i-1} \frac{r_j^{(k-1)} s_i^{(k-1)}}{p_k},$$

provando (11). Para $1 \leqslant i \leqslant j \leqslant n$

$$a_{ji} = \sum_{k=1}^{j} l_{jk} p_k u_{ki} = l_{ji} p_i u_{ii} + \sum_{k=1}^{j-1} l_{jk} p_k u_{ki}, \text{ considerando (8)},$$

$$a_{ji} = p_i \frac{s_j^{(i-1)}}{q_i} + \sum_{k=1}^{j-1} \frac{s_j^{(k-1)}}{q_k} p_k \frac{r_i^{(k-1)}}{p_k}, \text{ já que } p_i = q_i, \;\; s_j^{(i-1)} = a_{ji} - \sum_{k=1}^{j-1} \frac{r_i^{(k-1)} s_j^{(k-1)}}{q_k},$$

provando (12). □

Utilizando a Proposição 3.3, podemos computar $r_j^{(i-1)}$'s recursivamente usando os $s_i^{(k-1)}$ previamente gerados. Tais escalares podem ser calculados independente de Z. Portanto, a computação das entradas de U também pode ser feita independente de Z. É possível, então, modificar o laço interno do Algoritmo 1 para esse propósito, como é mostrado no Algoritmo 3.

Algorithm 3: Computando as entradas dos fatores L e U sem computar Z.

for $j \leftarrow i+1$ **to** n **do**
- $s_j^{(i-1)} \leftarrow c_i^{\mathsf{T}} w_j^{(i-1)}$;
- $r_j^{(i-1)} \leftarrow a_{ij} - \sum_{k=1}^{i-1} \frac{r_j^{(k-1)} s_i^{(k-1)}}{p_k}$;
- $l_{ji} = \frac{s_j^{(i-1)}}{q_i}$;
- $u_{ij} = \frac{r_j^{(i-1)}}{p_i}$;
- $w_j^{(i)} \leftarrow w_j^{(i-1)} - w_i^{(i-1)} l_{ji}$;

O mesmo raciocínio é válido para $s_j^{(i-1)}$, W e L. Podemos modificar o laço interno do Algoritmo 1, resultando no Algoritmo 4. Percebemos que a matriz W não é computada de forma explícita.

Algorithm 4: Computando as entradas dos fatores L e U sem computar W.

for $j \leftarrow i+1$ **to** n **do**
- $r_j^{(i-1)} \leftarrow a_i^{\mathsf{T}} z_j^{(i-1)}$;
- $s_j^{(i-1)} \leftarrow a_{ji} - \sum_{k=1}^{i-1} \frac{s_j^{(k-1)}}{q_k} r_i^{(k-1)}$;
- $l_{ji} = \frac{s_j^{(i-1)}}{q_i}$;
- $u_{ij} = \frac{r_j^{(i-1)}}{p_i}$;
- $z_j^{(i)} \leftarrow z_j^{(i-1)} - z_i u_{ij}$;

Diante dos resultados aqui demonstrados, podemos ver como é possível utilizar o AINV para a obtenção dos fatores LDU de A, sendo facultativo o cálculo de um dos fatores W e Z de forma explícita.

4 Conclusões

Neste trabalho, buscamos revisar as características mais relevantes do método de biconjugação, que é base do AINV, como os artifícios para se evitar a quebra e suas relações com a fatoração LDU de A. Vimos que é possível, além de Z e W, também computar as matrizes L e U utilizando a biconjugação. Devido a isso, alguns autores utilizam o AINV como método para encontrar a fatoração LDU de A. Porém, as matrizes L e U também podem ser densas e, portanto, algumas entradas de L e U devem ser descartadas segundo algum critério [4, 8, 9]. Por fim, acreditamos que o conhecimento dessas características pode vir a ajudar no desenvolvimento e aprimoramento de novas versões do AINV, bem como no estudo dos precondicionadores de um modo geral.

Referências

[1] Benzi, M., Meyer, C. D. and Tůma, M. A Sparse Approximate Inverse Preconditioner for the Conjugate Gradient Method, *SIAM Journal on Scientific Computing*, 17:1135–1149, 1996. DOI:10.1137/S1064827594271421.

[2] Benzi, M. and Tůma, M. A Sparse Approximate Inverse Preconditioner for Nonsymmetric Linear Systems, *SIAM Journal on Scientific Computing*, 19:968–994, 1998. DOI:10.1137/S1064827595294691.

[3] Benzi, M., Cullum, J. K. and Tůma, M. Robust Approximate Inverse Preconditioning for the Conjugate Gradient Method, *SIAM Journal on Scientific Computing*, 22:1318–1332, 2000. DOI:10.1137/S1064827599356900.

[4] Benzi, M., Kouhia, R. and Tůma, M. A Robust Preconditioner with Low Memory Requirements for Large Sparse Least Squares Problems, *SIAM Journal on Scientific Computing*, 25:499–512, 2003. DOI:10.1137/S106482750240649X.

[5] Fox, L. *An introduction to numerical linear algebra, 1a. edição*. Clarendon Press, Oxford, 1964.

[6] Gomes, L. T., De Barros, L. C. and Bede, B. Fujino, S. and Ikeda, An improvement of SAINV and RIF preconditionings of CG method by double dropping strategy. In *Proceedings. Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region.* IEEE, 2004. DOI: 10.1109/HPCASIA.2004.1324029.

[7] Horn, R. A. and Johnson, C. R. *Matrix Analysis, 2a. edição*. Cambridge University Press, New York, 2012.

[8] Rafiei, A. and Toutounian, F. New breakdown-free variant of AINV method for nonsymmetric positive definite matrices *Journal of Computational and Applied Mathematics*, 119:72–80, 2008. DOI:10.1016/j.cam.2007.07.003.

[9] Rafiei, A. and Bollhöfer, M. Robust incomplete factorization for nonsymmetric matrices *Numerische Mathematik*, 118:247–269, 2011. DOI:10.1007/s00211-010-0336-1.

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

