Proceedings of the
47th Annual Iranian Mathematics Conference,
Kharazmi University, Karaj, Iran, 28-31 August 2016

مجموعه مقالات چهل و هفتمین کنفرانس ریاضی ایران
دانشگاه خوارزمی، کرج، ۱۰-۷ شهریور ۱۳۹۵

A block factorization format for the inverse of a nonsymmetric matrix

A. Rafiei*, M. Bollhöfer and T. Huckle
Hakim Sabzevari University, a.rafiei@hsu.ac.ir, rafiei.am@gmail.com
Technische Universität Braunschweig, m.bollhoefer@tu-bs.de
Technische Universität München, huckle@in.tum.de

Abstract

In this paper, we present an algorithm to compute the block factorization format of the inverse of a nonsymmetric matrix. This algorithm is based on the left-looking version of A-biconjugation process. In this algorithm, the pivot elements will be one by one or two by two blocks.

Keywords: left-looking A-biconjugation.

Mathematics Subject Classification (2010): 65F10, 65F50, 65F08.

1 Introduction

Consider a nonsymmetric matrix A which is also invertible. In 1998, Benzi and Tuma presented the left-looking A-biconjugation algorithm to compute the inverse factorization of A in the form of

$$A^{-1} = ZD^{-1}W, \tag{1.1}$$

where Z and W are unit upper triangular matrices and D is a diagonal matrix [1].

Suppose that A has the factorization $A = LDU$ in which L and U^T are unit lower triangular and D is a diagonal matrix. We can compute this factorization by using different versions of Gaussian Elimination process. In [2], Bollhöfer and Kruschel have presented a block format of the Gaussian Elimination process to obtain $A = LDU$. In this block format, L and U^T are the same as before while D is a block diagonal matrix with the pivot blocks of order one or two. They use a criterion to dynamically select one by one or two by two blocks. This criterion considers the rate of diagonal dominance or the rate of block diagonal dominance of the Schur-Complement matrices.

In this paper, we extend the method of Bollhöfer and Kruschel to the left-looking A-biconjugation algorithm.

2 Main Results

Algorithm 1, is a block format of left-looking A-biconjugation algorithm. At the end of this algorithm, the factorization (1.1) is obtained. In this factorization, D will be a block diagonal matrix with the blocks of order one or two.

*Corresponding author and speaker

Algorithm 1 (**A block format of left-looking A-biconjugation process**)

Input: Let $A \in \mathbb{R}^{n\times n}$ is a nonsymmetric and invertible matrix
Output: $A^{-1} = ZD^{-1}W^T$ where Z and W are unit upper triangular and D is block diagonal with blocks of order 1×1 or 2×2.

1. $logic\_\ z = logic\_\ w = true$
2. $status(i) = 0, \qquad 1 \le i \le n$
3. $Z = [z_1^{(0)}, z_2^{(0)}, \cdots, z_n^{(0)}] = I_{n\times n}$, $W = [w_1^{(0)}, w_2^{(0)}, \cdots, w_n^{(0)}] = I_{n\times n}$
4. $i = 1$
5. **while** $i < n$ **do**
6. **if** $logic\_\ z$ **then**
7. call $Column\_\ Const(Z, A, D, i, status)$
8. **else**
9. $logic\_\ z = true$
10. **end if**
11. call $Column\_\ Const(Z, A, D, i+1, status)$
12. $S_{ii}^{(i-1)} = e_i^T A z_i^{(i-1)}$
13. **for** $j = i+1$ to n **do**
14. $S_{ji}^{(i-1)} = e_j^T A z_i^{(i-1)}, \quad S_{j,i+1}^{(i-1)} = e_j^T A z_{i+1}^{(i-1)}$
15. **end for**
16. **if** $logic\_\ w$ **then**
17. call $Column\_\ Const(W, A^T, D^T, i, status)$
18. **else**
19. $logic\_\ w = true$
20. **end if**
21. call $Column\_\ Const(W, A^T, D^T, i+1, status)$
22. $S_{ij}^{(i-1)} = (w_i^{(i-1)})^T A e_j, \quad j \ge i+1$
23. $S_{i+1,j}^{(i-1)} = (w_{i+1}^{(i-1)})^T A e_j, \quad j \ge i+2$
24. $v_i = \max\{\frac{1}{|S_{ii}^{(i-1)}|}\sum_{j=i+1}^n |S_{ij}^{(i-1)}|, \frac{1}{|S_{ii}^{(i-1)}|}\sum_{j=i+1}^n |S_{ji}^{(i-1)}|\}$
25. $w_i^1 = \sum_{j=i+2}^n \| \begin{bmatrix} S_{ii}^{(i-1)} & S_{i,i+1}^{(i-1)} \\ S_{i+1,i}^{(i-1)} & S_{i+1,i+1}^{(i-1)} \end{bmatrix}^{-1} \begin{pmatrix} S_{ij}^{(i-1)} \\ S_{i+1,j}^{(i-1)} \end{pmatrix} \|_\infty$
26. $w_i^2 = \sum_{j=i+2}^n \| \begin{pmatrix} S_{ji}^{(i-1)} & S_{j,i+1}^{(i-1)} \end{pmatrix} \begin{bmatrix} S_{ii}^{(i-1)} & S_{i,i+1}^{(i-1)} \\ S_{i+1,i}^{(i-1)} & S_{i+1,i+1}^{(i-1)} \end{bmatrix}^{-1} \|_\infty$
27. $w_i = \max\{w_i^1, w_i^2\}$
28. **if** $v_i < w_i$ **or** $(|S_{i+1,i}^{(i-1)}| + |S_{i,i+1}^{(i-1)}|) \le 10^{-2}\min\{|S_{i,i}^{(i-1)}|, |S_{i+1,i+1}^{(i-1)}|\}$ **then**
29. $D_{ii} = S_{ii}^{(i-1)}$
30. $z_{i+1}^{(i)} = z_{i+1}^{(i-1)} - (\frac{e_i^T A z_{i+1}^{(i-1)}}{D_{ii}}) z_i^{(i-1)}, \qquad w_{i+1}^{(i)} = w_{i+1}^{(i-1)} - (\frac{(w_{i+1}^{(i-1)})^T A e_i}{D_{ii}}) w_i^{(i-1)}$
31. $logic\_\ z = false, \qquad logic\_\ w = false$
32. $status(i) = 1$
33. $i = i+1$
34. **else**
35. $D_{i:i+1,i:i+1} = \begin{bmatrix} S_{ii}^{(i-1)} & S_{i,i+1}^{(i-1)} \\ S_{i+1,i}^{(i-1)} & S_{i+1,i+1}^{(i-1)} \end{bmatrix}$
36. $z_{i+1}^{(i)} = z_{i+1}^{(i-1)}, \quad w_{i+1}^{(i)} = w_{i+1}^{(i-1)}$
37. $status(i) = 2$
38. $i = i+2$
39. **end if**
40. **end while**
41. **if** $status(n-1) = 1$ **or** $status(n-2) = 2$ **then**
42. call $Column\_\ Const(Z, A, D, n, status)$
43. call $Column\_\ Const(W, A^T, D^T, n, status)$
44. $D_{n,n} = e_n^T A z_n^{(n-1)}$
45. **end if**
46. Return $Z = [z_1^{(0)}, z_2^{(1)}, \cdots, z_n^{(n-1)}]$, $W = [w_1^{(0)}, w_2^{(1)}, \cdots, w_n^{(n-1)}]$ and D

Here we explain step i of this algorithm. In lines 6-11 of the algorithm, we first compute the columns $z_i^{(i-1)}$ and $z_{i+1}^{(i-1)}$ of matrix Z. This is done by calling the function $Column\_\ Const$ which you can find it in Algorithm 2. In lines 12-15 of the algorithm, we use these two columns to obtain the first and the second columns of the Schur-Complement matrix, implicitly. This is the Schur-Complement matrix of the block version of Gaussian Elimination process which is presented by Bollhöfer and Kruschel. In lines 16-21, we first generate the columns $w_i^{(i-1)}$ and $w_{i+1}^{(i-1)}$ of matrix W by calling the function $Column\_\ Const$ two more

times. Then, in lines 22 and 23, we implicitly get the first and the second rows of the Schur-Complement matrix. In lines 24-39 of the algorithm, we first check whether we should have a one by one or two by two diagonal (pivot) block of matrix D. The criterion for such a selection considers the rate of the diagonal dominance or the rate of the block diagonal dominance of the Schur-Complement matrix. If the one by one pivot entry is preferable, then i or the step number is incremented by one and we should also update the i-th column of matrices Z and W one more time. This is done in line 30. If we should switch to the two by two block pivot, then i will be incremented by two. This is written in line 38 of the algorithm.

Algorithm 2 (**Column construction of a matrix**)

$Column\_Const(Z, A, D, i, status)$
1. $j = 1$
2. **while** $j < i - 1$ **do**
3. $\quad k = j + status(j) - 1$
4. $\quad$ **if** $status(j) = 1$ **then**
5. $\quad\quad z_i^{(k)} = z_i^{(j-1)} - z_j^{(j-1)} \times \frac{1}{D_{jj}} \times A_{j,:} \times z_i^{(j-1)}$
6. $\quad\quad j = j + 1$
7. $\quad$ **else if** $status(j) = 2$ **then**
8. $\quad\quad z_i^{(k)} = z_i^{(j-1)} - [z_j^{(j-1)} \quad z_{j+1}^{(j)}] \times [D(j : j+1, j : j+1)]^{-1} \times A_{j:j+1,:} \times z_i^{(j-1)}$
9. $\quad\quad j = j + 2$
10. $\quad$ **end if**
11. **end while**
12. Return Z

Inside Algorithm 1, we use the array *status*. We first initialize it as the zero array in line 2. Then, as it is stated in line 37 of the algorithm, we should set $status(i) = 2$ if at step i, the pivot is the two by two block. If in this step, we should use the one by one element as the pivot entry, then we should set $status(i) = 1$. This has been mentioned in line 32 of the algorithm. At the end of *while* loop, if $status(n-1) = 1$ or $status(n-2) = 2$, then the n-th column of matrices Z and W should be computed. We have considered this in lines 41-45.

Example 2.1. Here we give an example. Consider the nonsymmetric and invertible matrix A as:

$$A = \begin{bmatrix} 0.8147 & 0.5469 & 0.8003 & 0.0357 & 0.6555 & 0.8235 & 0.7655 \\ 0.9058 & 0.9575 & 0.1419 & 0.8491 & 0.1712 & 0.6948 & 0.7952 \\ 0.1270 & 0.9649 & 0.4218 & 0.9340 & 0.7060 & 0.3171 & 0.1869 \\ 0.9134 & 0.1576 & 0.9157 & 0.6787 & 0.0318 & 0.9502 & 0.4898 \\ 0.6324 & 0.9706 & 0.7922 & 0.7577 & 0.2769 & 0.0344 & 0.4456 \\ 0.0975 & 0.9572 & 0.9595 & 0.7431 & 0.0462 & 0.4387 & 0.6463 \\ 0.2785 & 0.4854 & 0.6557 & 0.3922 & 0.0971 & 0.3816 & 0.7094 \end{bmatrix}.$$

After running Algorithm 1, the computed Z and W are

$$Z = \begin{bmatrix} 1.0000 & -0.6713 & -0.9823 & 0.2802 & 0.4796 & -0.6240 & -0.2088 \\ 0 & 1.0000 & 0 & -1.2273 & -0.9337 & -0.0323 & -0.7834 \\ 0 & 0 & 1.0000 & 0.5088 & -0.6895 & -0.3693 & -0.0536 \\ 0 & 0 & 0 & 1.0000 & 0.4548 & -0.0545 & 0.5126 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 & 0.4281 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 & -0.5137 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 \end{bmatrix},$$

$$W = \begin{bmatrix} 1.0000 & -1.1118 & -0.1559 & -1.4108 & -0.4916 & -0.5458 & -0.1291 \\ 0 & 1.0000 & 0 & 0.1990 & -0.0397 & 0.6359 & 0.0341 \\ 0 & 0 & 1.0000 & 0.4388 & -0.6680 & -1.3024 & 0.0349 \\ 0 & 0 & 0 & 1.0000 & -0.1216 & -0.0695 & -0.1354 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 & -0.0715 \\ 0 & 0 & 0 & 0 & 0 & 1.0000 & -0.4078 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 \end{bmatrix},$$

and matrix D is

$$D = \begin{bmatrix} 0.8147 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.3494 & -0.7479 & 0 & 0 & 0 & 0 \\ 0 & 0.8796 & 0.2970 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.2072 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.5276 & -0.7254 & 0 \\ 0 & 0 & 0 & 0 & -1.1244 & -0.0479 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.2825 \end{bmatrix}.$$

References

[1] M. Benzi and M. Tuma, A sparse approximate inverse preconditioner for nonsymmetric linear systems, *SIAM J. Sci. Comput.* **19(3)**, (1998) 968-994.

[2] Ch. Kruschel, Lösen von positiv definiten, unsymmetrischen Matrizen mit Matching-Methoden am Beispiel von Konvektion-Diffusionsgleichungen, *Bachelor of Sciences thesis*, Technische Universität Braunschweig, 2009.