دسترسی در سایتِ http://jnrm.srbiau.ac.ir
سال پنجم، شماره هجدهم، خرداد و تیر ۱۳۹۸
شماره شاپا: ۲۵۸۸-۵۸۸X

JNRM
پژوهش‌های نوین در ریاضی

دانشگاه آزاد اسلامی، واحد علوم و تحقیقات

محاسبه شکل بلوکی پیش شرط ساز LU ناقص به عنوان محصول فرعی نسخه پیمایش چپ الگوریتم A–دو مزدوج سازی بلوکی

امین رفیعی[۱*]، بهناز طلوع حقیقی[۲]

(۱) گروه ریاضی کاربردی، دانشگاه حکیم سبزواری، سبزوار، ایران
(۲) گروه ریاضی محض، دانشگاه حکیم سبزواری، سبزوار، ایران

تاریخ ارسال مقاله: ۹۶/۰۴/۱۳ تاریخ پذیرش مقاله: ۹۷/۱۰/۰۱

چکیده

در این مقاله، شکل بلوکی پیش شرط ساز[۱] LU ناقصی را ارائه می‌کنیم که به عنوان محصول فرعی الگوریتم بلوکی A–دو مزدوج سازی[۲] محاسبه می‌شود. عناصر لولای این پیش شرط ساز، بلوک‌های یک در یک یا دو در دو می‌باشند. ماتریس‌های L و U این پیش شرط ساز به صورت مستقل از یکدیگر محاسبه خواهند شد. محک انتخاب عناصر لولا در این پیش شرط‌ساز بلوکی، همان محک به کار رفته در یکی از نسخه‌های بلوکی فرایند حذفی گاوس[۳] است. مبنای ارائه این پیش شرط‌ساز بلوکی، ارتباط میان فرایند حذفی گاوس و الگوریتم A–دو مزدوج سازی است. در بخش مثال‌های عددی این مقاله، ابتدا دستگاه‌های مصنوعی تولید کرده و سپس برای این دستگاه‌ها هر دو شکل ساده و بلوکی این پیش شرط ساز را محاسبه کرده‌ایم. پس از آن دستگاه‌های پیش شرط شده راست را تشکیل داده و از روش GMRES(۵۰)[۴] برای حل این دستگاه‌های پیش شرط شده استفاده گردیده است. نتایج عددی نشان می‌دهد که شکل بلوکی این پیش شرط‌ساز LU ناقص، روش GMRES(۵۰) را در تعداد تکرارهای کمتری نسبت به شکل ساده این پیش شرط‌ساز همگرا می‌نماید و بنابراین دارای کیفیت بهتری است.

واژه‌های کلیدی: پیش شرط ساز LU ناقص، نسخه پیمایش چپ[۵] پیش شرط ساز فاکتورسازی ناکامل قوی[۶]، نسخه بلوکی فرایند حذفی گاوس.

* عهده‌دار مکاتبات: Email: rafiei.am@gmail.com, a.rafiei@hsu.ac.ir

1. Preconditioner
2. A-biconjugation
3. Gaussian Elimination process
4. Generalized Minimum Residual
5. Left-looking version
6. Robust Incomplete Factorization

۱- مقدمه

فرض کنید $A \in \mathbb{R}^{n\times n}$ و $x, b \in \mathbb{R}^{n\times 1}$ باشد. برای حل دستگاه معادلات خطی وارون پذیر نامتقارن

$$Ax = b \qquad (۱)$$

می‌توان از روشهای زیر فضای کریلف[1] نظیر روش GMRES(m) استفاده کرد [۱]. برای دستگاه معادلات (۱) می‌توان از فرایند حذفی گاوس کمک گرفت و پیش شرط ساز LU ناقص M به شکل

$$A \approx M = LDU \qquad (۲)$$

را محاسبه نمود [۱]. در رابطه (۲)، ماتریس‌های L و U^T پایین مثلثی واحد[2] و D یک ماتریس قطری می‌باشد. در فرایند حذفی گاوس با ماتریس‌هایی به نام مکمل شور[3] سر و کار خواهیم داشت و محاسبه ماتریس-های L و U به یکدیگر مرتبط خواهند بود. اگر پیش شرط ساز M تقریب مناسبی از A باشد، آنگاه برای بدست آوردن نتایج همگرایی مناسب تر باید به جای حل دستگاه (۱) دستگاه پیش شرط شده راست

$$A \underbrace{M^{-1}u}_{x} = b$$

را تشکیل داد و سپس این دستگاه جدید را با روش GMRES(m) حل کرد.

هر گاه از عمل حذف کردن[4] در فرایند A - دومزدوج سازی استفاده کنیم، آنگاه پیش شرط ساز وارون تقریبی $\bar{M}$ که به صورت

$$A^{-1} \approx \bar{M} = ZD^{-1}W^T \qquad (۳)$$

می‌باشد محاسبه خواهد شد [۲]. ماتریس‌های Z و W موجود در این پیش شرط ساز بالا مثلثی واحد[5] و D یک ماتریس قطری می‌باشد. دو نسخه پیمایش چپ و راست[6] برای فرایند A- دومزدوج سازی وجود دارد و بنابراین پیش شرط ساز وارون تقریبی نیز دارای نسخه‌های پیمایش چپ و راست خواهد بود.

بنزی و توما در سال ۲۰۰۳ توانستند که یک پیش شرط ساز LU ناقص نظیر رابطه (۲) را به عنوان محصول فرعی فرایند A- دومزدوج سازی محاسبه نمایند. آنها نام این پیش شرط ساز را پیش شرط ساز فاکتورسازی ناکامل قوی در نظر گرفتند [۳]. ماتریس‌های L و U این پیش شرط ساز به صورت مستقل از یکدیگر محاسبه می‌شوند. پیش شرط ساز فاکتورسازی ناکامل قوی نیز دارای نسخه‌های پیمایش چپ و راست است [۴،۵].

در [۶]، کروشل یک نسخه بلوکی از فرایند حذفی گاوس ارائه کرده است که در آن عناصر لولا بلوک‌های یک در یک یا دو در دو می‌باشند. در این نسخه بلوکی، به طور همزمان امکان دسترسی به دو سطر و دو ستون پیشرو ماتریس مکمل شور فراهم است. این نویسنده همچنین پیش شرط ساز LU ناقصی بر اساس این نسخه بلوکی محاسبه کرده است.

در نسخه پیمایش چپ فرایند A - دومزدوج سازی می‌توان با استفاده از ستون‌های ماتریس‌های Z و W، دو ستون و دو سطر پیشرو یا ابتدایی ماتریسهای مکمل شور موجود در فرایند حذفی گاوس را به صورت ضمنی تولید نمود. بر مبنای همین مطلب و در مرجع [۷]، نویسندگان یک شکل بلوکی برای نسخه پیمایش چپ فرایند A - دومزدوج سازی مطرح نموده‌اند.

در این مقاله، یک شکل بلوکی از پیش شرط ساز فاکتورسازی ناکامل قوی را معرفی خواهیم کرد که به عنوان محصول فرعی شکل بلوکی نسخه‌ی پیمایش چپ فرایند A - دومزدوج سازی محاسبه می‌شود.

در بخش دوم این مقاله، نسخه بلوکی فرایند حذفی گاوس را یادآوری خواهیم کرد. در بخش سوم، نسخه پیمایش چپ پیش شرط ساز ناکامل قوی و شکل بلوکی آن بررسی می‌گردد. در بخش پایانی مقاله، مثال‌های عددی و مقایسه‌ها ارائه خواهند شد.

1. Krylov subspace methods
2. Unit lower triangular
3. Schur Complement
4. Dropping
5. Unit upper triangular
6. Right-looking version

فرض کنیم X یک ماتریس دلخواه است. در این مقاله از نمادهای $X_{i,:}$ و $X_{:,i}$ به ترتیب به عنوان سطر iام و ستون i ام ماتریس X استفاده کرده‌ایم. همچنین نماد $X_{i:j,k:l}$ نشان دهنده زیر ماتریسی از X است که اندیس سطری درایه‌های آن بین i تا j و اندیس ستونی درایه‌های آن بین k تا l است.

۲- نسخه بلوکی فرایند حذفی گاوس

الگوریتم ۱، نسخه بلوکی فرایند حذفی گاوس است که در [۶] ارائه شده است. این الگوریتم تجزیه ناقص موجود در رابطه (۲) را محاسبه می‌کند که در آن L و U^T ماتریس‌های پایین مثلثی واحد و D یک ماتریس بلوکی قطری است. عناصر قطری D بلوک‌های یک در یک یا دو در دو هستند. در اینجا گام iام این الگوریتم را به صورت مختصر شرح می‌دهیم.

همانگونه که در شکل ۱ نشان داده شده است در ابتدای گام iام الگوریتم ۱، تمام ماتریس مکمل شور $(S_{lk}^{(i-1)})_{l,k\geq i}$ در دسترس است. درون حلقه داخلی نظیر به $while$ و در خط ۴ الگوریتم، متغیر v_i به شکل

$$v_i = \max\left\{\frac{1}{|S_{ii}^{(i-1)}|}\sum_{j=i+1}^{n}|S_{ij}^{(i-1)}|, \frac{1}{|S_{ii}^{(i-1)}|}\sum_{j=i+1}^{n}|S_{ji}^{(i-1)}|\right\} \quad (۴)$$

معرفی می‌شود. در خطوط ۵ تا ۷ الگوریتم، دو متغیر جدید w_i^1 و w_i^2 به صورت

$$w_i^1 = \sum_{j=i+2}^{n}\left\|\begin{bmatrix} S_{ii}^{(i-1)} & S_{i,i+1}^{(i-1)} \\ S_{i+1,i}^{(i-1)} & S_{i+1,i+1}^{(i-1)} \end{bmatrix}^{-1}\begin{pmatrix} S_{ij}^{(i-1)} \\ S_{i+1,j}^{(i-1)} \end{pmatrix}\right\|_\infty$$

$$w_i^2 = \sum_{j=i+2}^{n}\left\|\begin{pmatrix} S_{ji}^{(i-1)} & S_{j,i+1}^{(i-1)} \end{pmatrix}\begin{bmatrix} S_{ii}^{(i-1)} & S_{i,i+1}^{(i-1)} \\ S_{i+1,i}^{(i-1)} & S_{i+1,i+1}^{(i-1)} \end{bmatrix}^{-1}\right\|_\infty \quad (۵)$$

محاسبه شده و سپس w_i برابر بیشینه این دو مقدار تعریف می‌شود. در خط ۸ الگوریتم بررسی می‌کنیم که آیا شرط $v_i < w_i$ برقرار است. اگر چنین باشد، آنگاه خطوط ۹ تا ۱۵ الگوریتم اجرا خواهند شد. در این خطوط ابتدا عنصر لولای یک در یک به عنوان درایه‌ی (۱و۱) ماتریس مکمل شور $(S_{lk}^{(i-1)})_{l,k\geq i}$ انتخاب می‌شود. سپس ستون i ام ماتریس L و سطر i ام ماتریس U محاسبه شده و برخی از درایه‌های آنها حذف خواهند شد. پس از آن، ستون و سطر محاسبه شده از ماتریس‌های L و U، بخش $S_{i+1:n,i+1:n}^{(i-1)}$ از ماتریس مکمل شور $(S_{lk}^{(i-1)})_{l,k\geq i}$ را به هنگام خواهند کرد. در پایان، به متغیر i یک واحد اضافه خواهد شد.

الگوریتم ۱: نسخه بلوکی از فرایند حذفی گاوس

Input: $A \in \mathbb{R}^{n\times n}$ is a nonsymmetric matrix and $\tau_L, \tau_U \in (0,1)$ are the drop tolerance parameters for matrices L and U

Output: $A \approx LDU$ where L^T and U are unit upper triangular matrices and D is a block diagonal matrix with diagonal blocks of order 1×1 or 2×2

1. $L = U = I_{n\times n}$, $S^{(0)} = A$, $D = 0 \in \mathbb{R}^{n\times n}$

2. $i = 1$

3. while $i < n$ **do**

4. $v_i = \max\left\{\frac{1}{|S_{ii}^{(i-1)}|}\sum_{j=i+1}^{n}|S_{ij}^{(i-1)}|, \frac{1}{|S_{ii}^{(i-1)}|}\sum_{j=i+1}^{n}|S_{ji}^{(i-1)}|\right\}$

5. $w_i^1 = \sum_{j=i+2}^{n}\left\|\begin{bmatrix} S_{ii}^{(i-1)} & S_{i,i+1}^{(i-1)} \\ S_{i+1,i}^{(i-1)} & S_{i+1,i+1}^{(i-1)} \end{bmatrix}^{-1}\begin{pmatrix} S_{ij}^{(i-1)} \\ S_{i+1,j}^{(i-1)} \end{pmatrix}\right\|_\infty$

6. $w_i^{\,2} = \sum_{j=i+2}^{n} \left\| \begin{pmatrix} S_{ji}^{(i-1)} & S_{j,i+1}^{(i-1)} \end{pmatrix} \begin{bmatrix} S_{ii}^{(i-1)} & S_{i,i+1}^{(i-1)} \\ S_{i+1,i}^{(i-1)} & S_{i+1,i+1}^{(i-1)} \end{bmatrix}^{-1} \right\|_{\infty}$

7. $w_i = \max\left\{w_i^{\,1}, w_i^{\,2}\right\}$

8. if $v_i < w_i$ **then**

9. $D_{ii} = S_{ii}^{(i-1)}$

10. $L_{i+1:n,i} = S_{i+1:n,i}^{(i-1)} D_{ii}^{-1}$

11. Suppose that $L_{i+1:n,i} = (\mathrm{L}_{ki})$**. For** $k \geq i+1$**, if** $|L_{ki}| < \tau_L$**, then set** $L_{ki} = 0$**.**

12. $U_{i,i+1:n} = D_{ii}^{-1} S_{i,i+1:n}^{(i-1)}$

13. Suppose that $U_{i,i+1:n} = (U_{ik})$**. For** $k \geq i+1$**, if** $|U_{ik}| < \tau_U$**, then set** $U_{ik} = 0$**.**

14. $S_{i+1:n,i+1:n}^{(i)} = S_{i+1:n,i+1:n}^{(i-1)} - L_{i+1:n,i} D_{ii} U_{i,i+1:n}$

15. $i = i+1$

16. else

17. $D_{i:i+1,i:i+1} = \begin{bmatrix} S_{ii}^{(i-1)} & S_{i,i+1}^{(i-1)} \\ S_{i+1,i}^{(i-1)} & S_{i+1,i+1}^{(i-1)} \end{bmatrix}$

18. $L_{i+2:n,i:i+1} = S_{i+2:n,i:i+1}^{(i-1)} D_{i:i+1,i:i+1}^{-1}$

19. Suppose that $L_{i+2:n,i:i+1} = (\mathrm{L}_{kj})$**. For** k **and** j**, if** $|L_{kj}| < \tau_L$**, then set** $L_{kj} = 0$**.**

20. $U_{i:i+1,i+2:n} = D_{i:i+1,i:i+1}^{-1} S_{i:i+1,i+2:n}^{(i-1)}$

21. Suppose that $U_{i:i+1,i+2:n} = (U_{kj})$**. For** k **and** j**, if** $|U_{kj}| < \tau_U$**, then set** $U_{kj} = 0$**.**

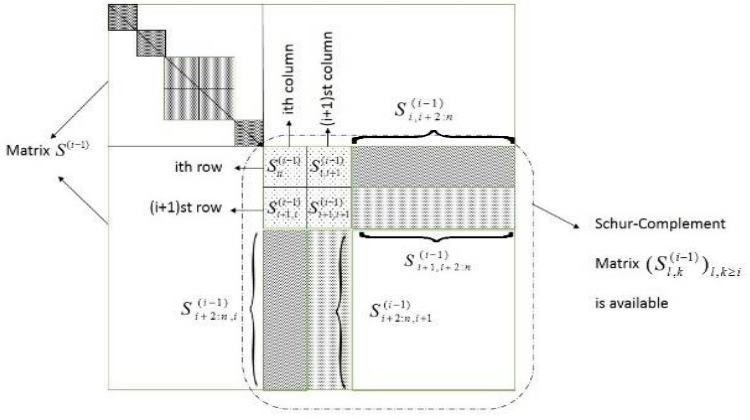
22. $S_{i+2:n,i+2:n}^{(i+1)} = S_{i+2:n,i+2:n}^{(i-1)} - L_{i+2:n,i:i+1} D_{i:i+1,i:i+1} U_{i:i+1,i+2:n}$

23. $i = i+2$

24. end if

25. end while

26. Return $L = (\mathrm{L}_{ij})_{1 \leq i,j \leq n}$, $U = (U_{ij})_{1 \leq i,j \leq n}$ **and** D



شکل ۱. گام i **ام نسخه بلوکی فرایند حذفی گاوس**

اگر در خط ۸ الگوریتم ۱ شرط $v_i \geq w_i$ برقرار باشد، آنگاه خطوط ۱۷ تا ۲۳ این الگوریتم اجرا خواهد شد. در این خطوط ابتدا زیر ماتریس اصلی دو در دو پیشرو از $(S_{lk}^{(i-1)})_{l,k\geq i}$ به عنوان بلوک لولا انتخاب می‌شود. سپس، در خطوط ۱۸ و ۱۹ ستون‌های i و $i+1$ از ماتریس L به طور همزمان محاسبه شده و برخی از درایه‌های این دو ستون حذف خواهند شد. پس از آن، در خطوط ۲۰ و ۲۱ الگوریتم، سطرهای i و $i+1$ از ماتریس U به طور همزمان معرفی خواهند شد. در خط ۲۲ الگوریتم، زیر ماتریس $S_{i+2:n,i+2:n}^{(i-1)}$ از ماتریس مکمل شور $(S_{lk}^{(i-1)})_{l,k\geq i}$ به هنگام گردیده و در پایان به متغیر i دو واحد اضافه خواهد شد.

۲- نسخه بلوکی پیش شرط ساز فاکتورسازی ناکامل قوی

الگوریتم ۲، نسخه پیمایش چپ پیش شرط ساز فاکتورسازی ناکامل قوی را به عنوان محصول فرعی فرایند A–دومزدوج سازی محاسبه می‌کند. در شکل ۲، چگونگی محاسبه ماتریس‌های L ، U و D این پیش شرط ساز ترسیم شده است. همانگونه که از این شکل پیداست در گام i ام این الگوریتم به ترتیب سطر i ام و ستون i ام ماتریس‌های L و U و درایه‌ی D_{ii} از ماتریس D محاسبه می‌شوند. در این الگوریتم ماتریس مکمل شوری وجود ندارد و ماتریس‌های L و U به ترتیب از طریق ماتریس‌های W و Z بدست می‌آیند. چون محاسبه ماتریس‌های Z و W به یکدیگر وابسته نیستند بنابراین ماتریس‌های L و U پیش شرط ساز فاکتورسازی ناکامل قوی نیز به صورت مستقل از یکدیگر تولید می‌شوند.

الگوریتم ۲: نسخه پیمایش چپ پیش شرط ساز فاکتورسازی ناکامل قوی

Input: $A \in \mathbb{R}^{n\times n}$ is a nonsymmetric matrix and $\tau_z, \tau_w, \tau_L, \tau_U \in (0,1)$ are the drop tolerance parameters for matrices Z, W, L and U

Output: $A \approx LDU$

1. $Z = [z_1^{(0)}, z_2^{(0)}, \ldots, z_n^{(0)}] = \mathrm{I}_{n\times n}$, $W = [w_1^{(0)}, w_2^{(0)}, \ldots, w_n^{(0)}] = \mathrm{I}_{n\times n}$, $L = U = I_{n\times n}$, $D = 0 \in \mathbb{R}^{n\times n}$

2. for $i = 1$ **to** n **do**

3. for $j = 1$ **to** $i-1$ **do**

4. $w_i^{(j)} = w_i^{(j-1)} - (\dfrac{(\mathrm{w}_i^{(j-1)})^T A e_j}{D_{jj}})\, \mathrm{w}_j^{(j-1)}$

5. $z_i^{(j)} = z_i^{(j-1)} - (\dfrac{\mathrm{e}_j^T \mathrm{A}\, z_i^{(j-1)}}{D_{jj}}) z_j^{(j-1)}$

6. For all $l \leq j$, **if** $\left|z_{li}^{(j)}\right| < \tau_Z$ **, then set** $z_{li}^{(j)} = 0$ **. Also, if** $\left|w_{li}^{(j)}\right| < \tau_W$ **, then set** $w_{li}^{(j)} = 0$

7. $U_{ji} = \dfrac{\mathrm{e}_j^T \mathrm{A}\, z_i^{(j-1)}}{D_{jj}}$**. If** $\left|U_{ji}\right| < \tau_U$ **, then set** $U_{ji} = 0$

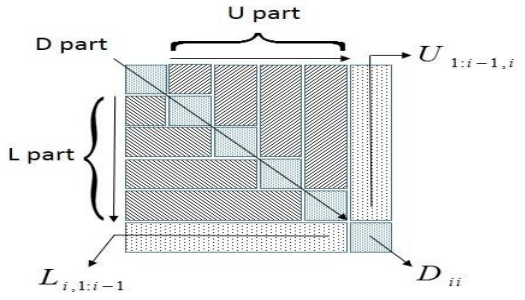
8. $L_{ij} = \dfrac{(\mathrm{w}_i^{(j-1)})^T A e_j}{D_{jj}}$ **. If** $\left|L_{ij}\right| < \tau_L$ **, then set** $L_{ij} = 0$

9. end for

10. $D_{ii} = e_i^T A z_i^{(i-1)}$

11. end for

12. Return $L = (\mathrm{L}_{ij})_{1\leq i,j\leq n}$, $U = (U_{ij})_{1\leq i,j\leq n}$ **and** $D = diag\,(D_{ii})_{1\leq i\leq n}$



شکل ۲. ساختار ماتریس‌ها در نسخه پیمایش چپ پیش شرط ساز فاکتورسازی ناکامل قوی

در گام i ام الگوریتم ۲ و با استفاده از ستون‌های $z_i^{(i-1)}$ و $w_i^{(i-1)}$ می‌توان سطر و ستون اول ماتریس مکمل شور $(S_{lk}^{(i-1)})_{l,k\geq i}$ موجود در الگوریتم ۱ را با استفاده از روابط

$$S_{ji}^{(i-1)} \approx e_j^T A z_i^{(i-1)},$$
$$S_{ij}^{(i-1)} \approx (w_i^{(i-1)})^T A e_j, \quad j \geq i. \qquad (۶)$$

تقریب زد [۸]. اگر در گام i ام الگوریتم ۲ ستون‌های $z_{i+1}^{(i-1)}$ و $w_{i+1}^{(i-1)}$ ماتریس‌های Z و W را نیز محاسبه کنیم در این صورت تقریبی از سطر و ستون دوم ماتریس مکمل شور مطابق روابط

$$S_{j,i+1}^{(i-1)} \approx e_j^T A z_{i+1}^{(i-1)},$$
$$S_{i+1,j}^{(i-1)} \approx (w_{i+1}^{(i-1)})^T A e_j, \quad j \geq i+1, \qquad (۷)$$

بدست خواهد آمد. بر مبنای روابط (۶) و (۷) می‌توان نسخه‌ای بلوکی برای پیش شرط ساز فاکتورسازی ناکامل قوی ارائه نمود. این نسخه بلوکی با استفاده از الگوریتم ۳ محاسبه می‌شود. در ابتدای الگوریتم ۳ و در خط ۱، دو پارامتر منطقی $\log ic\_z$ و $\log ic\_w$ معرفی گردیده و هر دو برابر $true$ در نظر گرفته شده‌اند. سپس در خط ۲، آرایه‌ای صحیح به نام $status$ مقدار دهی اولیه شده است. در خط ۳، ماتریس‌های Z، W، L و U برابر همانی و ماتریس D برابر صفر تعریف شده‌اند. اکنون، گام i ام این الگوریتم را به صورت خلاصه توضیح می‌دهیم.

در خط ۵ الگوریتم ۳ وارد حلقه داخلی نظیر به $while$ خواهیم شد. در خط ۶ چنانچه مقدار $\log ic\_z$ برابر $true$ باشد، آنگاه خط ۷ اجرا می گردد که در آن الگوریتم ۴ صدا زده خواهد شد. این الگوریتم جدید، ستون iام ماتریس Z یعنی بردار $z_i^{(i-1)}$ را با استفاده از به هنگام سازی رتبه یک[1] یا به هنگام سازی رتبه دو[2] محاسبه خواهد کرد. در اینجا شرح مختصری از الگوریتم ۴ را بیان می‌کنیم. در خط ۱ این الگوریتم، متغیر j برابر ۱ قرار می‌گیرد. سپس حلقه داخلی نظیر به $while$ اجرا خواهد شد. در خط ۴ این الگوریتم، مقدار $status(j)$ ارزیابی می‌شود. چنانچه این مقدار برابر یک باشد، آنگاه خطوط ۵ تا ۸ الگوریتم اجرا خواهند شد. در این خطوط ابتدا مضربی از ستون $z_j^{(j-1)}$ از ستون $z_i^{(j-1)}$ کم شده و ستون $z_i^{(k)}$ ساخته خواهد شد. به این فرایند به هنگام سازی رتبه‌ی یک می‌گویند. سپس، درایه‌ی U_{ji} محاسبه شده و چنانچه قدر مطلق این درایه از τ_U کمتر باشد، آنگاه بر روی آن عملیات حذف صورت می‌گیرد یا به عبارت دیگر مقدار آن صفر خواهد شد. در نهایت و در خط ۸ الگوریتم ۴، یک واحد به متغیر j اضافه خواهد شد. اگر در خط ۹ الگوریتم ۴ مقدار $status(j)$ برابر دو باشد، آنگاه خطوط ۱۰ تا ۱۳ این الگوریتم اجرا خواهند شد. در این خطوط ابتدا مضربی از دو ستون $z_j^{(j-1)}$ و $z_{j+1}^{(j)}$ از ستون $z_i^{(j-1)}$ کم شده و بردار $z_i^{(k)}$ بدست خواهد آمد.

1. Rank-one update
2. Rank-two update

الگوریتم ۳: شکل بلوکی نسخه پیمایش چپ پیش شرط ساز فاکتورسازی ناکامل قوی

Input: $A \in \mathbb{R}^{n\times n}$ is a nonsymmetric matrix and $\tau_w, \tau_z, \tau_L, \tau_U \in (0,1)$ are the drop tolerance parameters
for matrices W, Z, L and U

Output: $A \approx LDU$ where L^T and U are unit upper triangular matrices and D is a block diagonal matrix with diagonal blocks of order 1×1 or 2×2

1. $\log ic\_z = \log ic\_w = true$

2. $status(i) = 0, \quad 1 \le i \le n$

3. $Z = [\mathrm{z}_1^{(0)}, \mathrm{z}_2^{(0)}, \ldots, z_n^{(0)}] = \mathrm{I}_{n\times n}$, $W = [w_1^{(0)}, w_2^{(0)}, \ldots, w_n^{(0)}] = \mathrm{I}_{n\times n}$, $L = U = I_{n\times n}$, $D = 0 \in \mathbb{R}^{n\times n}$

4. $i = 1$

5. while $i < n$ **do**

6. if $\log ic\_z$ **then**

7.call $Update\_factor(\mathrm{Z}, \tau_z, \mathrm{A}, \mathrm{D}, \mathrm{U}, \tau_U, \mathrm{i}, \mathrm{status})$

8.else

9. $\log ic\_z = true$

10. end if

11. call $Update\_factor(\mathrm{Z}, \tau_z, \mathrm{A}, \mathrm{D}, \mathrm{U}, \tau_U, \mathrm{i+1}, \mathrm{status})$

12. $S_{ii}^{(i-1)} = e_i^T A z_i^{(i-1)}$

13. for $j = i+1$ **to** n

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 $\log ic\_\mathrm{w}$ **then**

17. call $Update\_factor(W, \tau_W, \mathrm{A}^T, \mathrm{D}^T, \mathrm{L}^T, \tau_L, \mathrm{i}, \mathrm{status})$

18. else

19. $\log ic\_\mathrm{w} = true$

20. end if

21. call $Update\_factor(W, \tau_W, \mathrm{A}^T, \mathrm{D}^T, \mathrm{L}^T, \tau_L, \mathrm{i+1}, \mathrm{status})$

22. $S_{ij}^{(i-1)} = (\mathrm{w}_i^{(i-1)})^T A e_j, \quad j \ge i+1$

23. $S_{i+1,j}^{(i-1)} = (\mathrm{w}_{i+1}^{(i-1)})^T A e_j, \quad j \ge i+2$

24. $$v_i = \max\left\{\frac{1}{|S_{ii}^{(i-1)}|}\sum_{j=i+1}^{n}|\mathrm{S}_{ij}^{(i-1)}|, \frac{1}{|S_{ii}^{(i-1)}|}\sum_{j=i+1}^{n}|\mathrm{S}_{ji}^{(i-1)}|\right\}$$

25. $$w_i^{\,1} = \sum_{j=i+2}^{n}\left\|\begin{bmatrix} S_{ii}^{(i-1)} & S_{i,i+1}^{(i-1)} \\ S_{i+1,i}^{(i-1)} & S_{i+1,i+1}^{(i-1)}\end{bmatrix}^{-1}\begin{pmatrix} S_{ij}^{(i-1)} \\ S_{i+1,\mathrm{j}}^{(i-1)}\end{pmatrix}\right\|_\infty$$

26. $$w_i^{\,2} = \sum_{j=i+2}^{n}\left\|\begin{pmatrix} S_{ji}^{(i-1)} & S_{j,i+1}^{(i-1)}\end{pmatrix}\begin{bmatrix} S_{ii}^{(i-1)} & S_{i,i+1}^{(i-1)} \\ S_{i+1,i}^{(i-1)} & S_{i+1,i+1}^{(i-1)}\end{bmatrix}^{-1}\right\|_\infty$$

27. $w_i = \max\{w_i^{\,1}, w_i^{\,2}\}$

28. if $v_i < w_i$ **then**

29. $D_{ii} = S_{ii}^{(i-1)}$

30. $z_{i+1}^{(i)} = z_{i+1}^{(i-1)} - (\frac{e_i^T A z_{i+1}^{(i-1)}}{D_{ii}}) z_i^{(i-1)}$. **For all** $l \le i$, **if** $\left| z_{l,i+1}^{(i)} \right| < \tau_Z$, **then set** $z_{l,i+1}^{(i)} = 0$

31. $U_{i,i+1} = \frac{e_i^T A z_{i+1}^{(i-1)}}{D_{ii}}$. **If** $\left| U_{i,i+1} \right| < \tau_U$, **then set** $U_{i,i+1} = 0$

32. $w_{i+1}^{(i)} = w_{i+1}^{(i-1)} - (\frac{(w_{i+1}^{(i-1)})^T A e_i}{D_{ii}}) w_i^{(i-1)}$. **For all** $l \le i$, **if** $\left| w_{l,i+1}^{(i)} \right| < \tau_W$, **then set** $w_{l,i+1}^{(i)} = 0$

33. $L_{i+1,i} = \frac{(w_{i+1}^{(i-1)})^T A e_i}{D_{ii}}$. **If** $\left| L_{i+1,i} \right| < \tau_L$, **then set** $L_{i+1,i} = 0$

34. $\log ic\_z = false, \quad \log ic\_w = false$

35. $status(i) = 1$

36. $i = i + 1$

37. else

38. $D_{i:i+1,i:i+1} = \begin{bmatrix} S_{ii}^{(i-1)} & S_{i,i+1}^{(i-1)} \\ S_{i+1,i}^{(i-1)} & S_{i+1,i+1}^{(i-1)} \end{bmatrix}$

39. $z_{i+1}^{(i)} = z_{i+1}^{(i-1)}, \quad w_{i+1}^{(i)} = w_{i+1}^{(i-1)}$

40. $status(i) = 2$

41. $i = i + 2$

42. end if

43. end while

44. if $status(n-1) = 0$ **then**

45. call $Update\_factor(\mathrm{Z}, \tau_z, \mathrm{A}, \mathrm{D}, \mathrm{U}, \tau_U, \mathrm{n}, \mathrm{status})$

46. call $Update\_factor(W, \tau_W, \mathrm{A}^T, \mathrm{D}^T, \mathrm{L}^T, \tau_L, \mathrm{n}, \mathrm{status})$

47. $D_{nn} = e_n^T A z_n^{(n-1)}$

48. end if

49. if $status(n-1) = 1$ **then**

50. $D_{nn} = e_n^T A z_n^{(n-1)}$

51. end if

الگوریتم ۴: به هنگام کردن ماتریس‌های Z و U

$Update\_factor(\mathrm{Z}, \tau_z, \mathrm{A}, \mathrm{D}, \mathrm{U}, \tau_U, i, \mathrm{status})$

Input: $Z = \left[z_1^{(0)}, z_2^{(1)}, \ldots, z_{i-1}^{(i-2)}, z_i^{(0)}, \ldots z_n^{(0)} \right] \in \mathbb{R}^{n \times n}$, $\tau_z \in (0,1)$ is the drop tolerance parameter for matrix Z, $A \in \mathbb{R}^{n \times n}$ is a nonsymmetric matrix, $D \in \mathbb{R}^{n \times n}$ is a block diagonal matrix, U is an unit upper triangular matrix, $\tau_U \in (0,1)$ is the drop tolerance parameter for matrix U, i is an integer value and *status* is an integer array

Output: updated version of matrices Z and U

1. $j = 1$

2. while $j \le i - 1$ **do**

3. $k = j + status(\mathrm{j}) - 1$

4. if $status(j) = 1$ **then**

5. $z_i^{(k)} = z_i^{(j-1)} - z_j^{(\mathrm{j}-1)} \times \frac{1}{D_{jj}} \times A_{j,:} \times z_i^{(j-1)}$

6. $U_{ji} = \frac{1}{D_{jj}} \times A_{j,:} \times z_i^{(j-1)}$

7. If $|U_{ji}| < \tau_U$ **, then set** $U_{ji} = 0$

8. $j = j + 1$

9. else if $status(j) = 2$ **then**

10. $z_i^{(k)} = z_i^{(j-1)} - \left[z_j^{(j-1)} \quad z_{j+1}^{(j)} \right] \times D^{-1}_{j:j+1,j:j+1} \times A_{j:j+1,:} \times z_i^{(j-1)}$

11. $U_{j:j+1,\mathrm{i}} = D^{-1}_{j:j+1,j:j+1} \times A_{j:j+1,:} \times z_i^{(j-1)}$

12. If $|U_{ji}| < \tau_U$ **and** $|U_{j+1,i}| < \tau_U$ **, then set** $U_{ji} = 0$ **and** $U_{j+1,i} = 0$

13. $j = j + 2$

14. end if

15. Suppose that $z_i^{(k)} = (\mathrm{z}_{li}^{(k)})$ **. For all** l **, if** $|\mathrm{z}_{li}^{(k)}| < \tau_Z$ **, then set** $\mathrm{z}_{li}^{(k)} = 0$

16. end while

17. Return Z **and** U

به این فرایند به هنگام سازی رتبه‌ی دو گفته می‌شود. سپس، دو درایه‌ی $U_{j:j+1,i}$ محاسبه گردیده و چنانچه قدر مطلق هر یک از این دو درایه از پارامتر τ_U کوچکتر باشد، آنگاه آنها را برابر صفر قرار می‌دهیم. در نهایت و در خط ۱۳ الگوریتم، به متغیر j دو واحد افزوده خواهد شد. در خط ۱۵ الگوریتم ۴، عملیات حذف کردن بر روی درایه‌های بردار $z_i^{(k)}$ اجرا می‌گردد. پس از این خط، الگوریتم به ابتدای حلقه $while$ برگشته و تا زمانی که شرط $j \le i - 1$ برقرار باشد این حلقه تکرار خواهد شد. در این قسمت به ادامه توضیح الگوریتم ۳ باز می‌گردیم. چنانچه در خط ۶ این الگوریتم مقدار $\log ic\_z$ برابر $true$ نباشد، آنگاه در خط ۹ الگوریتم مقدار این متغیر را برابر $true$ قرار می‌دهیم. در خط ۱۱ و با ورودی‌های متفاوت، بار دیگر الگوریتم ۴ صدا زده خواهد شد. این بار این زیر برنامه ستون $z_{i+1}^{(i-1)}$ را محاسبه خواهد کرد. در خطوط ۱۲ تا ۱۵ الگوریتم ۳، از بخش اول روابط (۶) و (۷) استفاده کرده و دو ستون اول ماتریس مکمل شور $(S_{lk}^{(i-1)})_{l,k \ge i}$ را تقریب خواهیم زد. اگر مقدار $\log ic\_\mathrm{w}$ در خط ۱۶ الگوریتم ۳ برابر $true$ باشد، آنگاه خط ۱۷ اجرا می‌گردد که در آن بار دیگر الگوریتم ۴ و با پارامترهای ورودی متفاوت نسبت به دفعات قبل صدا زده خواهد شد. این زیر برنامه این بار ستون i ام ماتریس W یا به عبارت دیگر بردار $w_i^{(i-1)}$ را محاسبه می‌کند.

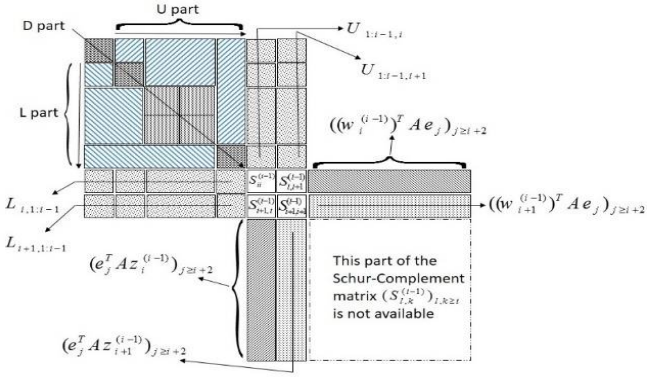
چنانچه مقدار $\log ic\_\mathrm{w}$ در خط ۱۶ الگوریتم ۳ برابر $false$ باشد، آنگاه در خط ۱۹ مقدار این متغیر را برابر $true$ قرار می‌دهیم. در خط ۲۱ الگوریتم ۳، الگوریتم ۴ را برای چهارمین بار فراخوانی می‌کنیم. نتیجه این امر محاسبه بردار $w_{i+1}^{(i-1)}$ است. در خطوط ۲۲ و ۲۳ الگوریتم ۳ و با استفاده از بخش دوم هر یک از روابط (۶)

و (۷) سطرهای اول و دوم ماتریس مکمل شور $(S_{lk}^{(i-1)})_{l,k\geq i}$ تقریب زده خواهد شد. حال که تقریبی از ستون‌های اول و دوم و سطرهای اول و دوم ماتریس مکمل شور را در اختیار داریم می‌توانیم در خطوط ۲۴ تا ۲۷ از روابط (۴) و (۵) استفاده کرده و مقدار متغیرهای v_i و w_i را محاسبه کنیم. از این مرحله به بعد باید محک موجود در الگوریتم ۱ را به کار گرفت و با استفاده از آن عنصر لولای گام i ام الگوریتم ۳ را مشخص نمود. برای این منظور چنانچه در خط ۲۸ الگوریتم ۳ شرط $v_i < w_i$ برقرار باشد، آنگاه خطوط ۲۹ تا ۳۶ اجرا می‌گردد. در این خطوط ابتدا عنصر لولای یک در یک به عنوان درایه‌ی (۱و۱) ماتریس مکمل شور معرفی می‌گردد. سپس، با استفاده از ستون $z_i^{(i-1)}$ بردار $z_{i+1}^{(i)}$ که همان ستون $(\mathrm{i}+1)$ام ماتریس Z است، محاسبه می‌شود. اگر قدر مطلق درایه‌ای از این ستون از τ_Z کوچکتر باشد، آنگاه این درایه برابر صفر قرار خواهد گرفت. پس از آن، درایه‌ی $U_{i,i+1}$ معرفی گردیده و بر روی آن فرایند حذف کردن اجرا خواهد شد. سپس، مضربی از ستون $w_i^{(i-1)}$ از بردار $w_{i+1}^{(i-1)}$ کم شده و بردار $w_{i+1}^{(i)}$ ساخته خواهد شد. این بردار همان ستون $(\mathrm{i}+1)$ام ماتریس W است. اگر قدر مطلق درایه ای از ستون $w_{i+1}^{(i)}$ از τ_W کوچکتر باشد، آنگاه این درایه برابر صفر قرار خواهد گرفت. درایه‌ی $L_{i+1,i}$ در خط ۳۳ الگوریتم معرفی گردیده و بر روی آن فرایند حذف کردن اجرا خواهد شد. در پایان، متغیرهای منطقی $\log ic\_z$ و $\log ic\_w$ را برابر $false$ قرار داده، مقدار $status(i)$ را یک در نظر گرفته و به متغیر i یک واحد اضافه می‌کنیم.

اگر در خط ۲۸ الگوریتم ۳ شرط $v_i \geq w_i$ برقرار باشد، آنگاه خطوط ۳۸ تا ۴۱ این الگوریتم اجرا می‌گردد. در این خطوط ابتدا بلوک لولای دو در دو را برابر زیر ماتریس اصلی دو در دو پیشرو از ماتریس مکمل شور $(S_{lk}^{(i-1)})_{l,k\geq i}$ تعریف می‌کنیم. سپس، ستونهای $(\mathrm{i}+1)$ ام ماتریس‌های Z و W را به ترتیب برابر بردارهای $z_{i+1}^{(i-1)}$ و $w_{i+1}^{(i-1)}$ در نظر می‌گیریم. در پایان، $status(i)$ را برابر ۲ قرار داده و به متغیر i دو واحد اضافه می‌کنیم.

در خط ۴۳ الگوریتم ۳ چنانچه شرط $i < n$ همچنان برقرار باشد، آنگاه به ابتدای حلقه $while$ باز می‌گردیم و خطوط داخل این حلقه دوباره اجرا خواهند شد. فرض کنید که $i \geq n$ است و از این حلقه داخلی خارج شده‌ایم. در خط ۴۴ الگوریتم اگر $status(\mathrm{n}-1)$ برابر صفر باشد، آنگاه در خطوط ۴۵ و ۴۶ دو بار الگوریتم ۴ صدا زده خواهد شد. حاصل این امر محاسبه ستون n ام ماتریس‌های Z و W یا بردارهای $z_n^{(n-1)}$ و $w_n^{(n-1)}$ می‌باشد. در خط ۴۷ الگوریتم و با استفاده از ستون $z_n^{(n-1)}$، مقدار D_{nn} محاسبه می‌شود.

در پایان الگوریتم ۳ و در خط ۴۹ بررسی می‌کنیم که آیا $status(\mathrm{n}-1)$ برابر یک است. اگر چنین باشد، تنها کافیست که در خط ۵۰ الگوریتم مقدار D_{nn} محاسبه شود. شکل ۳، شرح مختصری از گام i ام الگوریتم ۳ را بیان می‌کند.



شکل ۳. ساختار ماتریس‌ها در گام i ام شکل بلوکی نسخه پیمایش چپ پیش شرط ساز فاکتورسازی ناکامل قوی

۴- مثال‌های عددی

در این بخش، کیفیت شکل بلوکی نسخه پیمایش چپ پیش شرط ساز فاکتورسازی ناکامل قوی با کیفیت شکل ساده این پیش شرط ساز مقایسه شده است. برای این منظور تعداد ۴۸ ماتریس نامتقارن از مرجع [۹] را در نظر گرفته‌ایم. نام این ماتریس‌ها در جدول شماره ۱ درج شده است. فرض کنید یکی از این ماتریس‌ها برابر A باشد. اگر $x=[1,\ldots,1]^T$ و $b=A[1,\ldots,1]^T$ باشد، آنگاه دستگاه مصنوعی $Ax=b$ را ساخته‌ایم. به عنوان پیش پردازش از روش مرتب سازی تو در توی چندگامی[1] و روش جورسازی با بیشترین وزن[2] [۱۰،۱۱] برای این دستگاه استفاده کرده‌ایم. روش مرتب سازی تو در توی چندگامی با نام بسته نرم افزاری[3] METIS نیز شناخته می‌شود و در مرجع [۱۲] به زبان برنامه نویسی سی[4] پیاده‌سازی شده است. ما نسخه دیگری از این روش مرتب سازی را استفاده کرده‌ایم که در مرجع [۱۳] موجود است. روش جورسازی با بیشترین وزن با نام بسته نرم افزاری MC64 عنوان می‌شود و در مرجع [۱۴] به زبان برنامه نویسی فرترن[5] پیاده سازی گردیده است. با استفاده از این دو پیش پردازنده، دستگاه مصنوعی $Ax=b$ به دستگاه جدیدی نظیر $\tilde{A}x=\tilde{b}$ تبدیل می‌شود. جواب هر دو دستگاه یکسان است. سپس، از روش GMRES(۵۰) [۱] برای حل این دستگاه جدید استفاده می‌کنیم. برنامه فرترن روش GMRES(۵۰) در مرجع [۱۵] موجود است. محک توقف این روش به صورت

$$\frac{\left\|\tilde{b}-\tilde{A}x_k\right\|_2}{\left\|\tilde{b}\right\|_2}\le 10^{-8} \qquad (۸)$$

در نظر گرفته شده است که x_k، k امین تکرار می‌باشد. به دلیل زیاد بودن اعداد و ارقام، تنها ۱۶ مورد از ماتریس‌های جدول ۱ را انتخاب کرده و جزییات مربوط به تمامی آزمایش‌های انجام گرفته بر روی این ماتریس‌ها را در جدول‌های ۲ و ۳ ارائه می‌نماییم. این در حالی است که نظیر این آزمایش‌ها بر روی بقیه ماتریس‌ها نیز انجام گرفته و نتایج مشابهی بدست آمده است.

جدول ۱: ماتریس‌های مورد استفاده در آزمایش‌های عددی

نام ماتریس		
airfoil_2d	ASIC_100ks	cavity11
coupled	ex36	hcircuit
mult_dcop_02	poisson3Da	poisson3Db
raefsky2	sherman3	thermal
venkat25	venkat50	wang4
fs_541_2	fs_541_3	fs_541_4
gemat11	lnsp_511	mahindas
orsreg_1	pores_2	pores_3
Chebyshev3	chipcool0	chipcool1
Kaufhold	lung2	Memplus
rajat27	rajat28	raefsky1
TSOPF_RS_b39_c7	TSOPF_RS_b162_c1	TSOPF_RS_b162_c4
bp_200	bp_400	bp_600
fs_680_1	fs_680_2	fs_680_3
orani678	orsirr_1	orsirr_2
sherman2	sherman5	steam2

1. Multilevel nested dissection
2. Maximum weighted matching
3. Software package
4. C programming language
5. Fortran programming language

برای هر یک از ماتریس‌های جدول ۲ دستگاه‌های مصنوعی تولید کرده‌ایم. سپس، دستگاه‌های مصنوعی را با روش GMRES(۵۰) حل کرده‌ایم. در این جدول، بعد و تعداد درایه‌های غیر صفر هر ماتریس گزارش شده است. همچنین، تعداد تکرارها و زمان لازم برای روش GMRES(۵۰) نیز قابل مشاهده است. در این جدول، زمان بر حسب ثانیه می‌باشد و نماد + نشان می‌دهد که محک همگرایی (۸) در ۲۰۰۰ تکرار برقرار نشده است.

در الگوریتم‌های ۲ و ۳ پارامترهای حذف τ_w، τ_z، τ_L و τ_U را برابر ۰/۱ در نظر گرفته و شکل ساده و بلوکی نسخه پیمایش چپ پیش شرط ساز فاکتورسازی ناکامل قوی را برای ۱۶ ماتریس موجود در جدول ۲ محاسبه کرده‌ایم. فرض کنید هر یک از این دو پیش شرط ساز برابر M باشد. با استفاده از M می‌توان دستگاه پیش شرط شده راست

$$\tilde{A}\underbrace{M^{-1}u}_{x}=\tilde{b}$$

را تشکیل داد و این دستگاه را با روش GMRES(۵۰) حل کرد. پیش شرط سازی که برای برقراری محک (۸) به تعداد تکرارهای کمتری از روش GMRES(۵۰) نیاز داشته باشد دارای کیفیت بهتری خواهد بود. در اینجا لازم به توضیح است که برنامه کامپیوتری هر یک از الگوریتم‌های ۲ و ۳ به زبان برنامه نویسی سی نوشته شده است. در این پیاده‌سازی از روشهای برنامه نویسی موجود در بسته نرم افزاری ITSOL [۱۶] استفاده شایانی شده است. نتایج این آزمایش‌های عددی در جدول شماره ۳ ثبت شده است.

جدول ۲: ویژگی ماتریس‌ها و نتایج روش GMRES(۵۰)

نام ماتریس	ویژگی ماتریس		GMRES(۵۰)	
	بعد ماتریس	تعداد درایه‌های غیر صفر	تعداد تکرارها	زمان
ASIC_100ks	۹۹۱۹۰	۵۷۸۸۹۰	۹۸۷	۷/۰۴۹
coupled	۱۱۳۴۱	۹۷۱۹۳	+	+
hcircuit	۱۰۵۶۷۶	۵۱۳۰۷۲	+	+
lung2	۱۰۹۴۶۰	۴۹۲۵۶۴	+	+
memplus	۱۷۷۵۸	۹۹۱۴۷	+	+
poisson3Db	۸۵۶۲۳	۲۳۷۴۹۴۹	۴۶۹	۶/۱۵۰
rajat27	۲۰۶۴۰	۹۷۳۵۳	+	+
TSOPF_RS_b39_c7	۱۴۰۹۸	۲۵۲۴۴۶	+	+
TSOPF_RS_b162_c4	۲۰۳۷۴	۸۱۲۷۴۹	+	+
venkat25	۶۲۴۲۴	۱۷۱۷۷۹۲	+	+
venkat50	۶۲۴۲۴	۱۷۱۷۷۹۲	+	+
bp_600	۸۲۲	۴۱۷۲	+	+
fs_541_4	۵۴۱	۴۲۷۳	+	+
fs_680_3	۶۸۰	۲۶۴۶	+	+
gemat11	۴۹۲۹	۳۳۱۸۵	+	+
sherman5	۳۳۱۲	۲۰۷۹۳	+	+

جدول ۳: مشخصات پیش شرط‌سازها و نتایج همگرایی روش GMRES(۵۰) برای حل دستگاه‌های پیش شرط شده راست

نام ماتریس	GMRES(۵۰) + (نسخه پیمایش چپ پیش شرط ساز فاکتورسازی ناکامل قوی)				GMRES(۵۰) + (شکل بلوکی نسخه پیمایش چپ پیش شرط ساز فاکتورسازی ناکامل قوی)					
	زمان کل	تعداد تکرارها	چگالی	زمان پیش شرط سازی	زمان کل	تعداد تکرارها	تعداد لولاهای یک در یک	تعداد لولاهای دو در دو	چگالی	زمان پیش شرط سازی
ASIC_100ks	۳/۴۳۴	۵۹	۰/۸۳۶	۰/۳۷۴	۱۴/۴۳۲	۲۱	۸۹۳۸۲	۴۹۰۴	۰/۸۲۵	۱۳/۴۸۷
coupled	+	+	۰/۵۹۲	۰/۱۵۲	۰/۳۹۶	۲۷	۱۰۶۴۱	۳۵۰	۰/۵۸۸	۰/۳۱۷
hcircuit	+	+	۰/۷۷۳	۰/۲۷۶	۱۸/۵۳۳	۶۷	۵۳۹۰۶	۲۵۸۸۵	۰/۷۲۶	۱۴/۹۸
lung2	۴/۹۷۹	۸۴	۱/۱۱	۰/۲۵۵	۱۸/۳۵۸	۴۷	۱۲۰۴	۵۴۱۲۸	۱/۰۱۴	۱۵/۷۴۵
memplus	+	+	۰/۴۳۶	۰/۰۴۸	۱/۰۵۲	۷۶	۱۷۱۶۲	۲۹۸	۰/۴۳۵	۰/۴۷۸
poisson3Db	۲۰/۸۱۹	۲۸۲	۰/۲۹۳	۱/۳۲۶	۲۶/۱۵۹	۱۹۴	۸۲۹۰۹	۱۳۵۷	۰/۲۹۵	۱۱/۸۷۲
rajat27	+	+	۰/۷۳۷	۰/۰۶۷	۱/۴۰۵	۸۷	۱۵۱۲۶	۲۷۵۷	۰/۷۳۱	۰/۶۴۸
TSOPF_RS_b39_c7	۰/۶۷۶	۴۵	۰/۳۱۳	۰/۳۷۴	۱/۰۲۹	۳۷	۱۱۶۴۶	۱۲۲۶	۰/۳۰۸	۰/۷۸۱
TSOPF_RS_b162_c4	۳/۱۱۵	۱۴۹	۰/۱۴۱	۰/۹۹۵	۳/۵۹۷	۹۶	۲۰۲۳۶	۶۹	۰/۱۴۱	۲/۰۰۳
venkat25	۵۳/۸۳۱	۷۹۶	۰/۷۹۵	۱/۶۳۲	۵۱/۱۹۶	۶۳۸	۴۹۴۸۶	۶۴۶۹	۰/۷۸۶	۷/۳۵۴
venkat50	۸۶/۶۶	۱۳۱۳	۰/۷۹۹	۱/۶۲۷	۷۸/۵۳۴	۱۰۲۱	۴۹۵۷۸	۶۴۲۳	۰/۷۹۰	۷/۵۹۲
bp_600	۰/۰۱۲	۷۴	۱/۰۹۲	۰/۰۰۳	۰/۰۱۰	۳۰	۵۸۸	۱۱۷	۱/۱۳۱	۰/۰۰۶
fs_541_4	۰/۰۰۴	۳۵	۰/۵۲۷	۰/۰۰۱	۰/۰۰۴	۲۶	۴۴۵	۴۸	۰/۵۱۸	۰/۰۰۲
fs_680_3	۰/۰۰۲	۱۷	۰/۷۷۲	۰/۰۰۱	۰/۰۰۳	۱۰	۴۸۴	۹۸	۰/۷۶۸	۰/۰۰۲
gemat11	+	+	۰/۹۰۵	۰/۰۱۹	۰/۴۱۸	۲۹۹	۲۳۵۱	۱۲۸۹	۰/۸۰۵	۰/۰۵۵
sherman5	۰/۰۵۳	۸۸	۰/۷۰۳	۰/۰۱	۰/۰۵۸	۴۲	۱۱۵۲	۱۰۸۰	۰/۶۹۷	۰/۰۲۹

در این جدول، زمان پیش شرط سازی بر حسب ثانیه می‌باشد. منظور از زمان کل، مجموع زمان پیش شرط سازی و زمان لازم برای اجرای روش GMRES(۵۰) است که بر حسب ثانیه محاسبه شده است. همچنین، چگالی هر دو پیش شرط ساز برابر کسری است که صورت آن مجموع تعداد درایه‌های غیر صفر ماتریس‌های L و U و مخرج آن تعداد درایه‌های غیر صفر ماتریس $\tilde{A}$ می‌باشد. در این جدول، تعداد لولاهای دو در دو و یک در یک نظیر به شکل بلوکی نسخه پیمایش چپ پیش شرط ساز فاکتورسازی ناکامل قوی ارائه گردیده است. برای برخی از ماتریس‌ها، در ستون مربوط به تعداد تکرارها و زمان کل از نماد + استفاده شده است. این نماد نشان می‌دهد که محک همگرایی روش GMRES(۵۰) برای حل دستگاه‌های پیش شرط شده راست در ۲۰۰۰ تکرار برقرار نیست. با توجه به اعداد گزارش شده در جدول ۳ می‌توان نتیجه گرفت که:

- برای تمامی ماتریس‌ها، زمان پیش شرط سازی نسخه بلوکی از زمان پیش شرط سازی نسخه ساده بیشتر است.
- برای تمامی ماتریس‌ها، چگالی هر دو پیش شرط ساز اعدادی نزدیک به یکدیگرند. بنابراین، مقایسه کیفیت آنها با یکدیگر مفهوم دارد.
- برای تمامی ماتریس‌ها، شکل بلوکی نسخه پیمایش چپ پیش شرط ساز فاکتورسازی ناکامل قوی روش GMRES(۵۰) را در تعداد تکرارهای کمتری نسبت به شکل ساده این پیش شرط ساز همگرا کرده است. لازم به توضیح است که برای ماتریس‌های coupled، hcircuit، memplus، rajat27 و gemat11، شکل بلوکی نسخه پیمایش چپ پیش شرط ساز فاکتورسازی ناکامل قوی باعث همگرایی روش GMRES(۵۰) شده

است در حالی که شکل ساده این پیش شرط ساز نتوانسته است که همگرایی روش GMRES(۵۰) را در ۲۰۰۰ تکرار تضمین نماید.

- برای ماتریس‌های ASIC_100ks، lung2، poisson3Db، TSOPF_RS_b39_c7 ، TSOPF_RS_b162_c4، fs_680_3، bp_600 و sherman5، زمان کل نظیر به شکل بلوکی نسخه پیمایش چپ پیش شرط ساز فاکتور سازی ناکامل قوی بیشتر از زمان کل نظیر به شکل ساده این پیش شرط‌ساز است. برای ماتریس‌های venkat25 و venkat50 عکس این مطلب برقرار است. همچنین، در مورد ماتریس fs_541_4 زمان کل نظیر به هر دو پیش شرط ساز یکسان می‌باشد.

از مقایسه اعداد موجود در جدول‌های ۲ و ۳ در می‌یابیم که اگر روش GMRES(۵۰) با نسخه پیمایش چپ پیش شرط ساز فاکتور سازی ناکامل قوی یا شکل بلوکی آن ترکیب شود، آنگاه تعداد تکرارهای بهتری برای همگرایی بدست خواهد آمد.

۵- نتیجه‌گیری

در این مقاله، شکل بلوکی نسخه پیمایش چپ پیش شرط ساز فاکتورسازی ناکامل قوی ارائه گردیده است. عناصر لولای پیش شرط ساز بلوکی معرفی شده، بلوک‌های یک در یک یا دو در دو می‌باشند. کیفیت شکل بلوکی نسخه پیمایش چپ پیش شرط ساز فاکتورسازی ناکامل قوی با کیفیت شکل ساده این پیش شرط ساز مقایسه شده است. نتایج عددی نشان می‌دهد که اگر شکل بلوکی این پیش شرط ساز با روش GMRES(۵۰) ترکیب گردد، آنگاه تعداد تکرارهای لازم برای همگرایی کمتر از زمانی است که ترکیبی از شکل ساده این پیش شرط ساز با روش GMRES(۵۰) استفاده می‌شود.

فهرست منابع

[1] Y. Saad, Iterative methods for sparse linear systems. SIAM Publications, Philadelphia, second edition (2003).

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

[3] M. Benzi, and M. Tuma, M. A Robust Incomplete Factorization Preconditioner for Positive Definite Matrices, Numer. Linear Alg. Appl., 10 (2003) 385-400.

[4] A. Rafiei, B. Tolue, and M. Bollhoefer, Complete pivoting strategy for the left-looking Robust Incomplete Factorization preconditioner, Comput. Math. Appl., 67 (2014) 2055-2070.

[5] A. Rafiei, A complete pivoting strategy for the right-looking Robust Incomplete Factorization preconditioner. Comput. Math. Appl., 64 (2012) 2682-2694.

[6] Ch. Kruschel, Losen von positiv definiten, unsymmetrischen Matrizen mit Matching-Methoden am Beispiel von Konvektion-Diffusionsgleichungen. Bachelor of Sciences thesis, Technische Universitat Braunschweig (2009).

[7] A. Rafiei, M. Bollhoefer and F. Benkhaldoun, A block version of left-looking AINV preconditioner with one by one or two by two block pivots, Revised for Appl. Math. Comput., (2018).

[8] A. Rafiei, Left-looking version of AINV preconditioner with complete pivoting strategy, Linear Alg. Appl., 445 (2014) 103-126.

[9] T. Davis, The SuiteSparse Matrix Collection, http://www.cise.ufl.edu/research/sparse/matrices/. Accessed 2017.

[10] I. S. Duff, and J. Koster, The design and use of algorithms for permuting large entries to the diagonal of sparse matrices. SIAM J. Matrix Anal. Appl., 20(4) (1999) 889-901.

[11] I. S. Duff, and S. Parlett Strategies for scaling and pivoting for sparse symmetric indefinite problems. SIAM J. Matrix Anal. Appl., 27(2) (2005) 313-340.

[12] G. Karypis, and V. Kumar, METIS a Software Package for partitioning Unstructured Graphs and Computing Fill-Reduced Orderings of Sparse Matrices, http://glaros.dtc.umn.edu/gkhome/metis/metis/download

[13] M. Bollhoefer, ILUPACK software package. http://www.icm.tu-bs.de/~bolle/ilupack.

[14] The HSL Mathematical Software Library, http://www.hsl.rl.ac.uk.

[15] Y. Saad, Sparskit and sparse examples, http://www-users.cs.umn.edu/~saad/software. Accessed 2017.

[16] Y. Saad, ITSOL software package. http://www-users.cs.umn.edu/~saad/software.