1 はじめに

連立一次方程式 \( Ax = b \) を解くことを考える。ここで \( A \) は対称正定な行列 \( A = (a_{ij}) \in \mathbb{R}^{n \times n} \)，解ベクトル \( x \in \mathbb{R}^n \)，右辺ベクトル \( b \in \mathbb{R}^n \) とする。このように連立一次方程式を解く場合，前処理として IC(Incomplete Cholesky，以下 IC と略す) 方法が広く用いられている。IC 解法では，解ベクトル行列 \( x \) を \( U^{-1}U^T \) となる \( A \) に近似した前処行列を \( M = U^{-1}U^T(= A) \) を生成し，その解ベクトル行列 \( X \) に作用させて対称換行列を生成した後方行列に変換させた後方行列に対して共役勾配 (Conjugate Gradient，以下 CG と略す) 法などの反復法を適用する。さらに，ドロッピング処理 IC 解法では，元の解ベクトル行列 \( X \) を求めるのではなく，前処行列が解の反復過程で等でない値を持たず変数（以下，フィールドと呼ぶ）に対して，予め定めた閾値よりも小さい場合はそれを棄却するという不完全分解を行う。この際，解ベクトル行列の大きさによっては，反復法の収束性に大きな影響を及ぼすことがある。

近年，非対称行列列の前処理として Crout 版 ILU 解法 (以下，ICLUC 解法と略す) が提案され，さらに逆行列解（7）の誤差行列 \( X \) の具体的な評価方法について Inverse-based ドロッピング手法と呼ばれる方法が提案されている。Inverse-based ドロッピング手法の考え方は対称行列に適用すること，さらに原著では必ずしも十分明らかにされなかった Inverse-based ドロッピング手法が収束特性をより明らかにすることである。

2 IC 分解で生じる誤差

前処理用 CG 法では，解ベクトル行列 \( Ax = b \) を

\[
(U^TAU^{-1})(UX) = U^Tb
\]

と変換して解くのが一般的である。したがって，前処理後の係数行列は \( U^TAU^{-1} \) になる。このとき，前処行列 \( A \) を完全分解して得られた三角行列を \( U \) とするとき，従来の IC 解法により求められた解ベクトル行列 \( X \) は前処行列 \( U \) であると仮定して，次のように表せる。同様に，他の逆行列解行列 \( X \) を用いて，逆行列の逆行列 \( X \) は次のように表せる。

\[
U^T = U^T + X^T \quad U = U + X,
\]

（2）

\[
U^T = U^T + X^T \quad U = U + X.
\]

（3）

これらの式（2）、（3）を使うと，前処行列後の係数行列 \( U^TAU^{-1} \) は次のように表せる。

\[
(U^TAU^{-1}) = I + UX + X^TUX + XAX.
\]

（4）

ここで，式（4）中に式（2）、（3）で定義した誤差行列 \( X \) が含まれていないことに注意を要する。したがって，一般に解ベクトル行列 \( X \) の大小が小さい場合に逆行列解行列 \( X \) の誤差行列 \( X \) も小さいという保証は得られない。式（4）から，もし逆行列解行列 \( X \) の誤差行列 \( X \) が大きい場合には前処理を反復法の収束性に影響を及ぼす可能性があることが分かる。

表 1 に従来の IC 分解と ib.IC 分解における誤差行列およびドロッピングのとき対象となる行列の違いを示す。

<table>
<thead>
<tr>
<th>2</th>
<th>IC 分解で生じる誤差</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>前処理行列 CG 法</td>
</tr>
<tr>
<td>2.3</td>
<td>改</td>
</tr>
</tbody>
</table>

表 1: Error matrix and object matrix of dropping strategy in the IC and ib.IC decompositions.

<table>
<thead>
<tr>
<th>分解</th>
<th>( U ) の誤差行列 ( X ) と</th>
<th>ドロッピング</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC</td>
<td>( U^{-1} ) の誤差行列 ( X ) と</td>
<td>対象行列</td>
</tr>
<tr>
<td>IC</td>
<td>( U = U + X )</td>
<td>( U = U + X )</td>
</tr>
<tr>
<td>ib.IC</td>
<td>同上</td>
<td>( U^{-1} = U^{-1} + X )</td>
</tr>
</tbody>
</table>

3 Inverse-based IC 分解

ここでは，行列 \( U^T = (u_{ij}) \) を下三角行列 \( L = (l_{ij}) \) を用いて表現する。同様に，行列 \( U^T = (u_{ij}) \) を下三角行列 \( L = (l_{ij}) \) を用いて表現する。また，行列 \( Lk \) と，1 から \( k \) 列目まで前処行列解行列 \( L \) と \( k \) 列目までの同値であり，\( k \) から \( n \) 列目まで要素 \( O_k(1 \leq i \leq n) \) は対角要素に求める対角行列と同値であるような下三角行列とする。ただし，要素 \( O_k \) は行列 \( L \) の第 \( k \) 行の対角要素に対応し，分解過程において新しく生成される。
の行のノルムを制限することができる。ただし、$||A||_{\infty}$ は行列 $A$ の最大ノルムで、

$$\begin{align*}
||A||_{\infty} &= \max_{\alpha \in \mathbb{R}} ||\alpha e_i||_{\infty} \\
&= \max_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{i,j}|
\end{align*}$$

(9)

と定義される。$||e_i e_j^T L^{-1}||_{\infty}$ はすべての要素が 0 でないあるベクトル $b$ を用いて、次のように近似できる。

$$\begin{align*}
||e_i e_j^T L^{-1}||_{\infty} &= \max_{b \in \mathbb{R}^n, \|b\|_{\infty}=1} \|e_i e_j^T L^{-1} b\|_{\infty} \\
&= \max_{j=1}^{n} |a_{i,j}|
\end{align*}$$

(10)

したがって、$||e_i e_j^T L^{-1}||_{\infty}$ を推定するためには、式 (10) を満たすようなベクトル $b$ を求める必要がある。最大ノルムの定義式 (9) において、$i = m$ のとき、

$$\begin{align*}
\sum_{j=1}^{n} |a_{i,j}|
\end{align*}$$

(11)

が最大になったとき、その最大値を与えるベクトル $x_i$ の第 $j$ 成分は、

$$x_i = \text{sign}(a_{i,j}), \quad (j = 1, 2, \ldots, n)$$

(12)

t となることが知られている。ただし、sign は符号関数で、$\text{sign}(t) = +1 (t \geq 0)$、$\text{sign}(t) = -1 (t < 0)$ で定義される。しかし、式 (10) では、行列 $L^{-1}$ の計算をせず、ベクトル $b$ の要素の符号を決定できない。そこで、

$$b = (-b_1, -b_2, \ldots, -b_n)^T, \quad (b_1, b_2, \ldots, b_n = \pm 1)$$

(13)

のうちから、$||e_i e_j^T L^{-1} b||_{\infty}$ をできるだけ大きくするベクトル $b$ を探すことができる。このようなベクトル $b$ を見つければ、次のような近似式が成り立つ。

$$||e_i e_j^T L^{-1}||_{\infty} \approx \frac{||e_i e_j^T L^{-1} b||_{\infty}}{||b||_{\infty}}$$

(14)

式 (14) の右边部、$L\xi = b$ の解 $\xi$ の第 $k$ 成分で評価でき、ベクトル $\xi$ の第 $k$ 成分 $\xi_k$ は、右边の分母のベクトル $b$ の第 $k$ 成分を用いて次のように求められる。ただし、$\xi_k = (\xi_1, \xi_2, \ldots, \xi_k, 0, \ldots, 0)^T, b_k = (b_1, b_2, \ldots, b_k, 0, \ldots, 0)^T$ とする。

$$\xi_k = (b_k - e_i^T L^{-1} \xi_{k-1})/l_{k,k}$$

(15)

以上から、$||e_i e_j^T L^{-1}||_{\infty}$ を見積もる以下のアルゴリズムが得られる。ただし、アルゴリズム中の $\xi_k$ と $\nu_k$ は、$||e_i e_j^T L^{-1}||_{\infty}$ と $e_i^T L^{-1} \xi_{k-1}$ にそれぞれ対応する。

アルゴリズム 1 $||e_i e_j^T L^{-1}||_{\infty}$ を見積もるアルゴリズム

set $\xi_1 = 1/l_{1,1}; \nu_0 = 0 (i = 1, \ldots, n)$

for $k = 2, n$

$\text{temp}_+ = 1 - \nu_k$

$\text{temp}_- = -1 - \nu_k$

if $|\text{temp}_+| > |\text{temp}_-|$ then

$\xi_k = \text{temp}_+/l_{k,k}$

else

$\xi_k = \text{temp}_- / l_{k,k}$

end if

end for

Inverse-based ドロッピング手法は、フィルイン $l_{k,k}$ に対し、

$$l_{k,k} ||e_i e_j^T L^{-1}||_{\infty} = ||l_{k,k}| \xi_k | ||e_i e_j^T L^{-1}||_{\infty} \leq \tau$$

(16)

のとき適用する。ただし、パラメータ $\tau$ は閾値とする。すなわち、収束された行列 $L^{-1}$ の要素の集合 $P$ は次のように表せる。

$$P = \{ (i, j) \mid |a_{i,j}|/|\xi_k| \sqrt{||\xi_k||_{\infty}} \leq \tau \}$$

(17)