元の行列の非零要素に基づく ib-IC(0,tol) 分解つき CG 法の収束性

塩出 亮 (九州大学大学院)

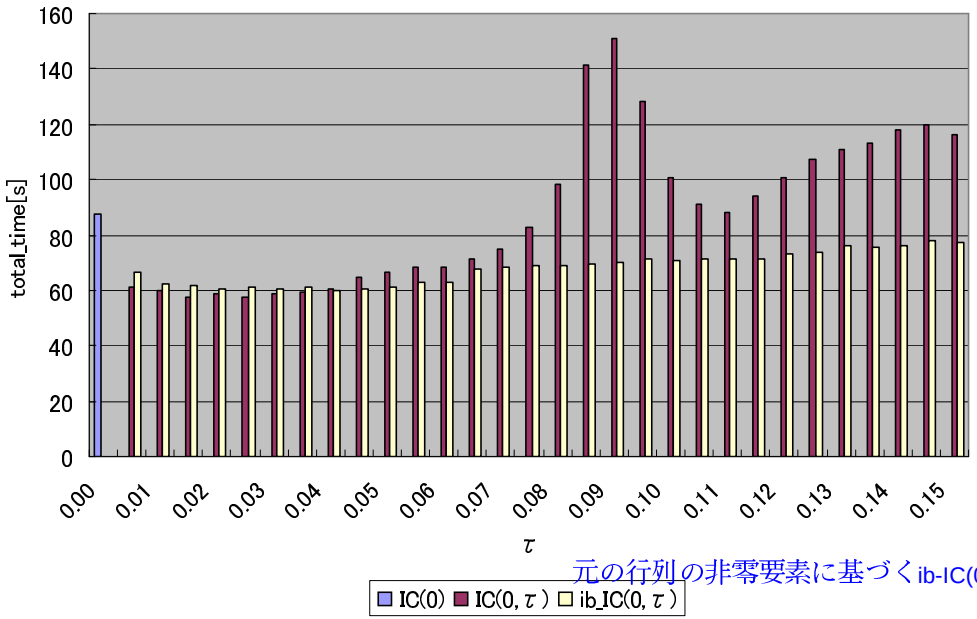
藤野清次 (九州大学情報基盤センター)

発表の概要

1. 研究の背景と目的
2. 通常の IC 分解で生じる誤差
3. Inverse-based IC 分解: ib-IC(0) 分解
4. 元の行列の非零要素に基づく ib-IC(0,tol) 分解
5. 数値実験
6. まとめ

研究背景

- 連立一次方程式 $A\boldsymbol{x} = \boldsymbol{b}$ を解くことを考える．ここで A は対称かつ正則な行列 $A = (a_{i,j}) \in R^{n \times n}$，解ベクトル $\boldsymbol{x} \in R^n$，右辺ベクトル $\boldsymbol{b} \in R^n$ とする．
- IC 分解では，前処理行列 $M = U^T U (\approx A)$ を生成し，その逆行列 M^{-1} を係数行列に作用させ反復法を適用する．
- 閾値つき IC(tol) 分解では，分解過程で生じたフィルインのうち或る閾値より小さいものは棄却．閾値の大きさで，反復法の収束性に大きなばらつきが生じる．



研究の目的

1. 非対称行列用前処理として Crout 版 ILU 分解 (ILUC 分解と略す) が提案され，逆行列 U^{-1} の誤差行列 X のノルムの大きさに基づくドロッピングが可能になった
2. Inverse-based ドロッピング手法を対称行列に拡張した
3. さらに，ib-IC 分解の変形版として、元の係数行列の非零要素に基づく ib-IC(0,tol) 分解を提案する。
4. ib-IC(0,tol) 分解が、必要なメモリ量が少なくかつ収束性も安定であることを検証する。

IC分解で生じる誤差-1

1. 前処理つきCG法では，解くべき方程式 $A\boldsymbol{x} = \boldsymbol{b}$ を

(1) $$(U^{-T}AU^{-1})(U\boldsymbol{x}) = U^{-T}\boldsymbol{b}$$

と変換して解く．変換後の係数行列は，$U^{-T}AU^{-1}$.

2. 行列 A の完全分解の上三角行列を $\bar{U}$ とする

(2) $$U^T = \bar{U}^T + \tilde{X}^T, \quad U = \bar{U} + \tilde{X}$$

(3) $$U^{-T} = \bar{U}^{-T} + X^T, \quad U^{-1} = \bar{U}^{-1} + X.$$

3. 変換後の係数行列 $U^{-T}AU^{-1}$ は

(4) $$U^{-T}AU^{-1} = I + \bar{U}X + X^T\bar{U}^T + X^TAX.$$

IC分解で生じる誤差-2

1. ここで，式 (4) 中に式 (2) で定義した誤差行列 $\tilde{X}$ が現れていないことに注意を要する．
2. したがって，一般に行列 U の誤差行列 $\tilde{X}$ が小さい場合に逆行列 U^{-1} の誤差行列 X も小さいという保証はない
3. 式 (4) から，もし逆行列 U^{-1} の誤差行列 X が大きい場合には反復法の収束性に影響を及ぼす可能性がある．

IC分解で生じる誤差-3

1. 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になる行列の違いを表に示す

分解	U の誤差行列 $\tilde{X}$ と U^{-1} の誤差行列 X	ドロッピング対象行列
IC	$U \equiv \bar{U} + \tilde{X}$, $U^{-1} = \bar{U}^{-1} + X$	$U = \bar{U} + \tilde{X}$
ib_IC	同上	$U^{-1} = \bar{U}^{-1} + X$

2. 行列 $U^T = (u_{j,i})$ を下三角行列 $L = (l_{i,j})$ を用いて表現.

Inverse-based IC 分解

1. 行列 L_k を，1 から k 列目まで前処理行列 L の 1 から k 列目までと同値，$k+1$ から n 列目まで要素 $\overline{a_{i,i}}$ $(1 \leq i \leq n)$ を対角要素に持つ対角行列と同値な下三角行列．
2. 要素 $\overline{a_{i,i}}$ は行列 A の第 i 行の対角要素に対応し，分解過程において次々と更新される．

$$L_k = \begin{pmatrix} l_{1,1} & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ l_{2,1} & \ddots & \ddots & & & & \vdots \\ \vdots & \vdots & l_{k,k} & \ddots & & & \vdots \\ \vdots & \vdots & \vdots & \overline{a_{k+1,k+1}} & \ddots & & \vdots \\ \vdots & \vdots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ l_{n,1} & \cdots & l_{n,k} & 0 & \cdots & 0 & \overline{a_{n,n}} \end{pmatrix}. \tag{5}$$

逆行列 L_k^{-1} のノルム評価-1

1. 行列 L の k 列目を生成するとき，フィルイン $l_{j,k}$ $(j > k)$ が棄却されたと仮定する．
2. 行列 L_k は以下のように表現できる．ただし，単位ベクトル $\boldsymbol{e}_i$ $(1 \leq i \leq n)$ は i 行目が 1 で，i 以外の行はすべて 0 の列ベクトルとする．

(6) $$L_k = \bar{L}_k - l_{j,k}\boldsymbol{e}_j\boldsymbol{e}_k^T.$$

3. $j > k$ より，$\bar{L}_k\boldsymbol{e}_j = \overline{a_{j,j}}\boldsymbol{e}_j$ であることに注意すると，

(7) $$L_k = \bar{L}_k - l_{j,k}\boldsymbol{e}_j\boldsymbol{e}_k^T = \bar{L}_k(I - l_{j,k}\boldsymbol{e}_j\boldsymbol{e}_k^T/\overline{a_{j,j}})$$

逆行列 L_k^{-1} のノルム評価-2

1. 式 (7) より，行列 L_k^{-1} を次のように表現できる．

$$
\begin{aligned}
L_k^{-1} &= (I - l_{j,k}\boldsymbol{e}_j\boldsymbol{e}_k^T/\overline{a_{j,j}})^{-1}\bar{L}_k^{-1} \\
&\approx \bar{L}_k^{-1} + l_{j,k}\boldsymbol{e}_j\boldsymbol{e}_k^T\bar{L}_k^{-1}/\overline{a_{j,j}}. \qquad (8)
\end{aligned}
$$

2. 行列 L_k の逆行列 L_k^{-1} は $l_{j,k}/\overline{a_{j,j}}$ と $\bar{L}_k^{-1}$ の k 行目の積の値の大きさだけ影響を受ける
3. $||l_{j,k}\boldsymbol{e}_j\boldsymbol{e}_k^T\bar{L}_k^{-1}/\overline{a_{j,j}}||_\infty = |l_{j,k}/\overline{a_{j,j}}|\ ||\boldsymbol{e}_j\boldsymbol{e}_k^T\bar{L}_k^{-1}||_\infty$ でドロッピングを行うのが好ましい
4. 概算の値ではあるが，逆行列 L_k^{-1} の第 k 行のノルムの大きさを見積もれるようになった．

行列 A の最大ノルム

1. $||A||_\infty$ は行列 A の最大ノルムで，

$$||A||_\infty = \max_{\boldsymbol{x}\neq 0} \frac{||A\boldsymbol{x}||_\infty}{||\boldsymbol{x}||_\infty} = \max_i \sum_{j=1}^{n} |a_{i,j}| \tag{9}$$

と定義される．

2. $||\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}_k^{-1}||_\infty$ はすべての要素が 0 でないあるベクトル $\boldsymbol{b}$ を用いて，次のように近似できる．

$$||\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}_k^{-1}||_\infty \quad = \quad \max_{\boldsymbol{b}\neq 0} \frac{||\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}^{-1}\boldsymbol{b}||_\infty}{||\boldsymbol{b}||_\infty}. \tag{10}$$

最大値を与えるベクトルbの決定-1

1. したがって，$||\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}^{-1}||_\infty$ を推定するためには，式 (10) を満たすようなベクトル $\boldsymbol{b}(\neq \boldsymbol{0})$ を決定すればよい．
2. 最大ノルムの定義式 (9) において，$i = m$ のとき，

$$\sum_{j=1}^{n} |a_{i,j}| \tag{11}$$

が最大のとき，最大値を与えるベクトル $\boldsymbol{x}$ の第 j 成分は，

$$x_j = \mathrm{sign}(a_{m,j}), \quad (j = 1, 2, \ldots, n) \tag{12}$$

となることが知られている．
3. ただし，sign は符号関数

$$\mathrm{sign}(t) \;=\; +1 \; (t \geq 0),\; \mathrm{sign}(t) = -1 \; (t < 0) \tag{13}$$

ベクトルbの決定-2

1. 式 (10) では，行列 $\bar{L}^{-1}$ をそのまま計算するのは手間がかかる．ベクトル $\boldsymbol{b}$ の要素の符号の決定は困難． そこで，

$$\begin{aligned} \boldsymbol{b} &= (-b_1, -b_2, \ldots, -b_n)^T, \\ & \quad (b_1, b_2, \ldots, b_n = \pm 1) \end{aligned} \tag{14}$$

の中から，$||\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}^{-1} \boldsymbol{b}||_\infty$ をできるだけ大きくするベクトル $\boldsymbol{b}$ を探すことを考える．

2. すなわち，解の成分を１つずつ求めていく過程で，その成分の絶対値がなるべく大きくなるようにベクトル $\boldsymbol{b}$ の対応する成分を+1 か-1 に定める，という便法を採用する

3. 上の考え方は条件数を推定するとき使用される．
(参考)： 森正武著，数値計算プログラミング，岩波書店，2002.

ベクトルbの決定-3

1. ベクトル $\boldsymbol{b}$ が見つかれば，次の近似式が成り立つ．

$$||\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}^{-1}||_\infty \approx \frac{||\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}^{-1} \boldsymbol{b}||_\infty}{||\boldsymbol{b}||_\infty}. \tag{15}$$

2. 式 (14) の右辺項は，$\bar{L}\boldsymbol{\xi} = \boldsymbol{b}$ の解 $\boldsymbol{\xi}$ の第 k 成分で評価でき，ベクトル $\boldsymbol{\xi}$ の第 k 成分 ξ_k は，右辺の分母のベクトル $\boldsymbol{b}$ の第 k 成分を用いて次の式で求められる．$\boldsymbol{\xi}_k = (\xi_1, \xi_2, \ldots, \xi_k, 0, \ldots, 0)^T$, $\boldsymbol{b}_k = (b_1, b_2, \ldots, b_k, 0, \ldots, 0)^T$

$$\xi_k = (b_k - \boldsymbol{e}_k^T \bar{L}_{k-1} \boldsymbol{\xi}_{k-1}) / l_{k,k}. \tag{16}$$

3. 以上から，$||\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}^{-1}||_\infty$ を見積もる以下のアルゴリズムが得られる．ξ_k と ν_k は，$||\boldsymbol{e}_j \boldsymbol{e}_k^T \bar{L}^{-1}||_\infty$ と $\boldsymbol{e}_k^T \bar{L}_{k-1} \boldsymbol{\xi}_{k-1}$ に各々対応．

$||e_j e_k^T \bar{L}^{-1}||_\infty$ を見積る算法

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

for $k = 2, n$

$temp_+ = 1 - \nu_k$

$temp_- = -1 - \nu_k$

if $|temp_+| > |temp_-|$ then

$\xi_k = temp_+ / l_{k,k}$

else

$\xi_k = temp_- / l_{k,k}$

end if

for $j = k + 1, n$ $(l_{j,k} \neq 0)$

$\nu_j = \nu_j + \xi_k l_{j,k}$

end for

end for.

Inverse-based ドロッピング手法

1. Inverse-based ドロッピング手法では，フィルイン $l_{j,k}$ に対して，

(17) $$|l_{j,k}/\overline{a_{j,j}}|||\boldsymbol{e}_j\boldsymbol{e}_k^T\bar{L}^{-1}||_\infty \quad = \quad |l_{j,k}||\xi_k|/|\overline{a_{j,j}}| \leq \tau$$

のとき棄却する．ただし，パラメータ τ は閾値とする．

2. 棄却された行列 U の要素の集合 P は次のようになる．

(18) $$P = \{(i,j) \mid (|a^*_{i,j}||\xi_k|)/(|\overline{a_{j,j}}|\sqrt{a_{i,i}}) \leq \tau\}.$$

ib-IC(0,tol)分解：集合の定義

1. 係数行列 A の零要素のパターンの集合:
   $\phi \equiv \{(j,k) \mid a_{j,k} = 0\}$.

2. L の従来のドロッピングで棄却された要素の集合:
   $\psi \equiv \{(j,k) \mid |a^*_{j,k}|/\sqrt{(\overline{a_{j,j}})(\overline{a_{k,k}})} \leq \tau\}$

3. L の Inverse-based ドロッピングで棄却された要素の集合:
   $\psi_{\rm ib} \equiv \{(j,k) \mid |a^*_{j,k}||\xi_k|/(|\overline{a_{j,j}}|\sqrt{a_{k,k}}) \leq \tau\}$.

3種類のIC(0)型分解

分解法	ドロッピング対象行列	棄却要素
IC(0)	-	$u_{j,k} \in \phi$
IC$(0,\tau)$	$U = \bar{U} + \tilde{X}.$	$u_{j,k} \in (\phi \cup \psi)$
ib_IC $(0,\tau)$	$U^{-1} = \bar{U}^{-1} + X.$	$u_{j,k} \in (\phi \cup \psi_{\mathrm{ib}})$

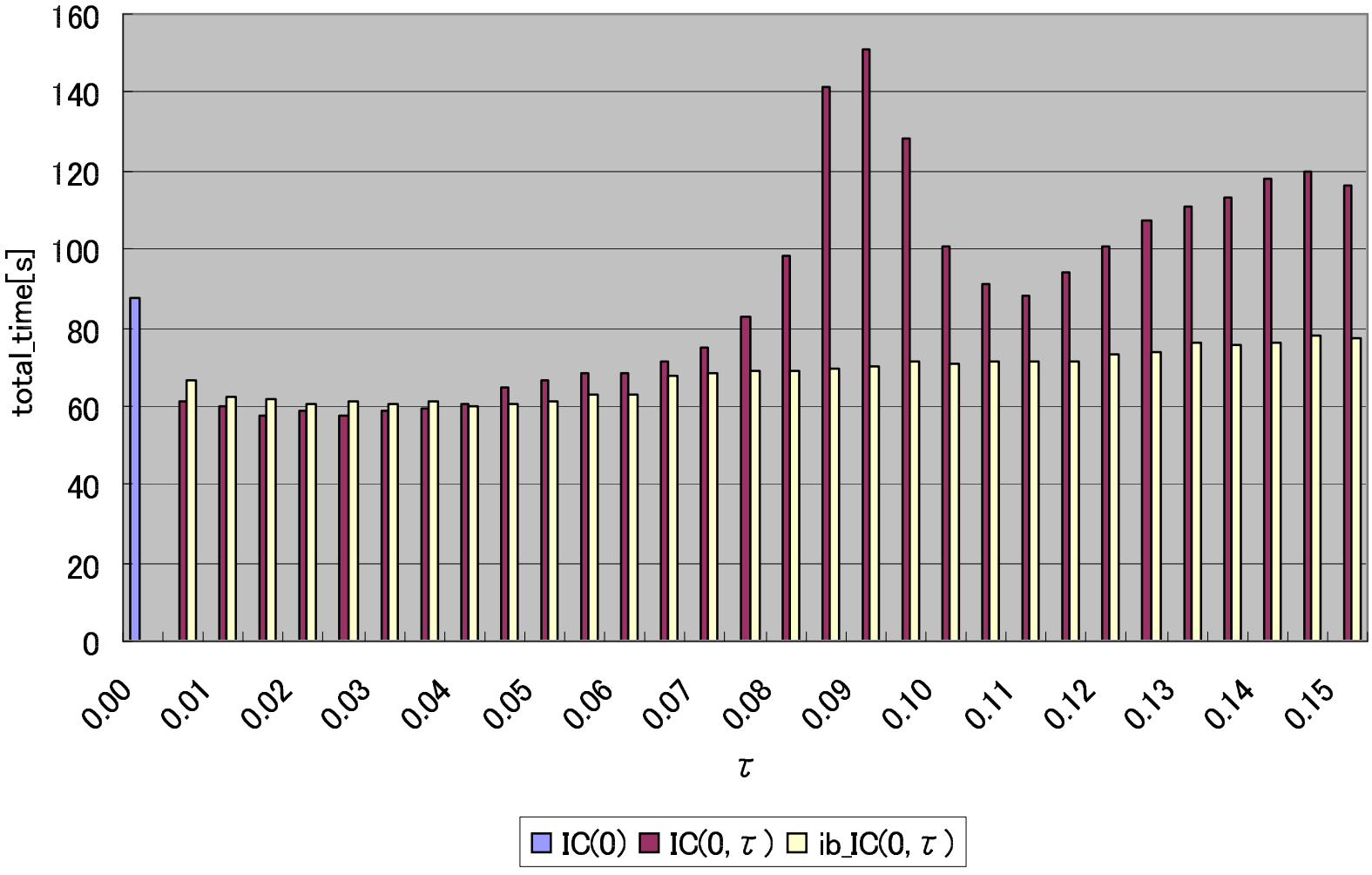
数値実験

1. Pentium4 (クロック周波数 3.8GHz) を搭載した PC
2. プログラムは Fortran90 で実装，計算は倍精度浮動小数点演算．最適化オプションは-O3
3. 反復法には CG 法を使用
4. 前処理は IC(0)_s, WIC(0)_s, IC(0), IC(0,τ), ib_IC(0,τ) 分解の 5 種類
5. IC(0)_s は対角要素のみ求める前処理，WIC(0)_s はさらに対角要素に 1 より大きな加速係数を掛ける前処理．
6. 係数行列 A は予め対角項を 1 に揃える正規化処理
7. CG 法の初期近似解 $\boldsymbol{x}_0$ はすべて $\boldsymbol{0}$
8. 収束判定は残差ベクトル l_2 ノルム比:$||\boldsymbol{r}_k||_2/||\boldsymbol{r}_0||_2 \leq 10^{-8}$
9. 閾値 τ は，0.005 から 0.15 まで 0.005 刻みで合計 30 通り

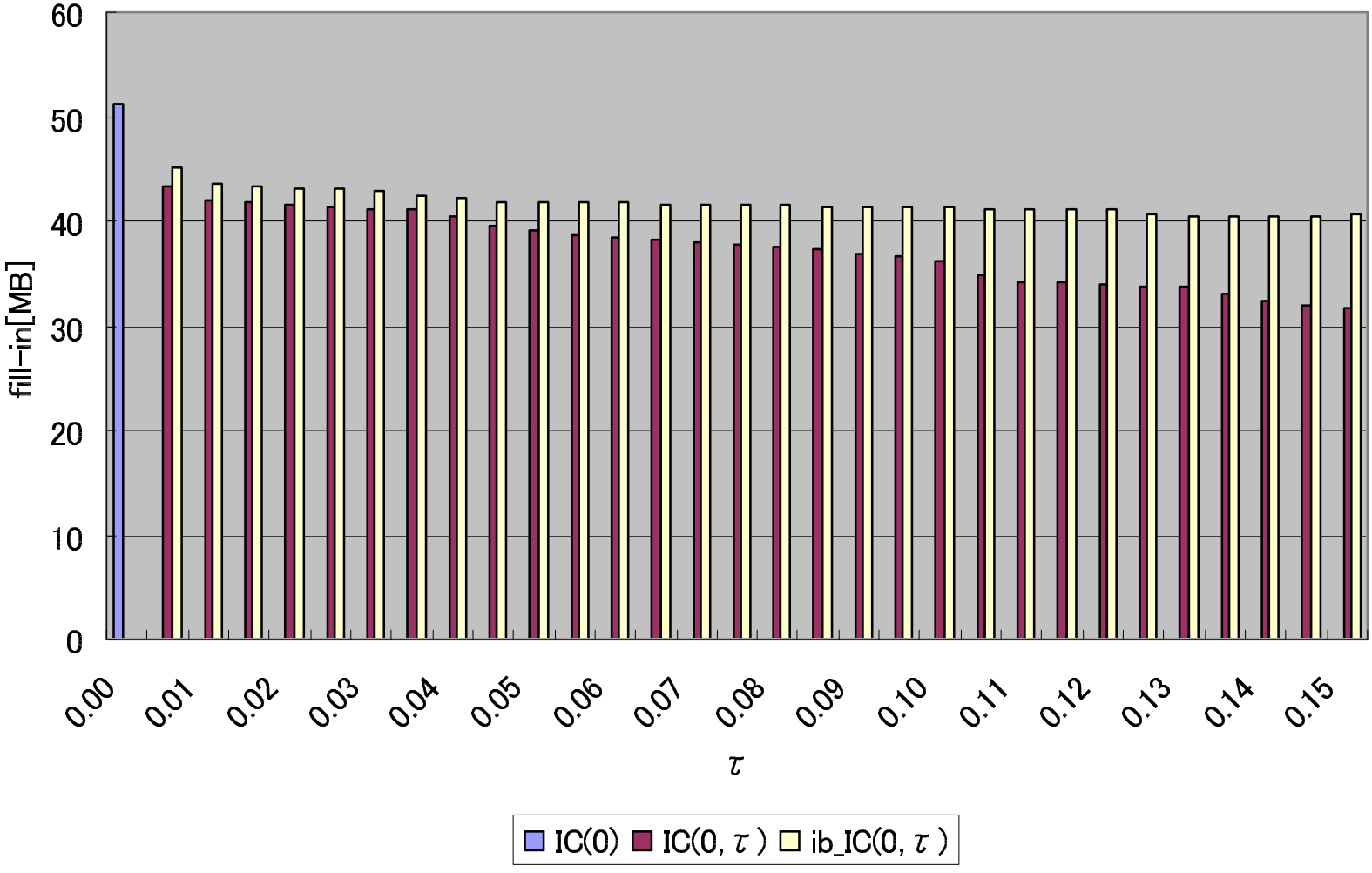
テスト行列

SuperP10	115,248	1,592,277	13.8	構造解析 (重合メッシュ法)
SuperP40	417,524	6,053,860	14.5	同上

行列 SuperP10:計算時間



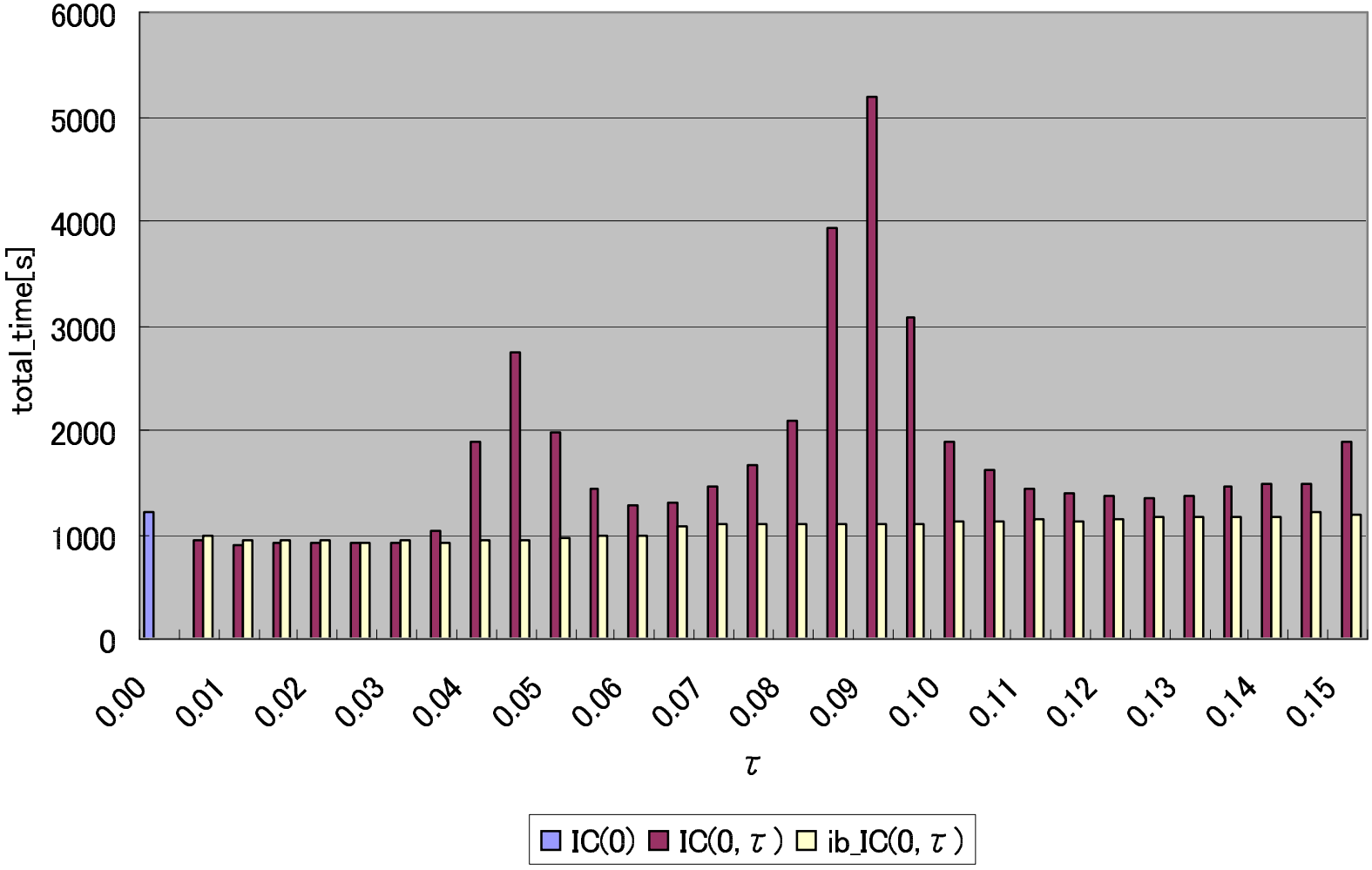
行列 SuperP10:メモリ量



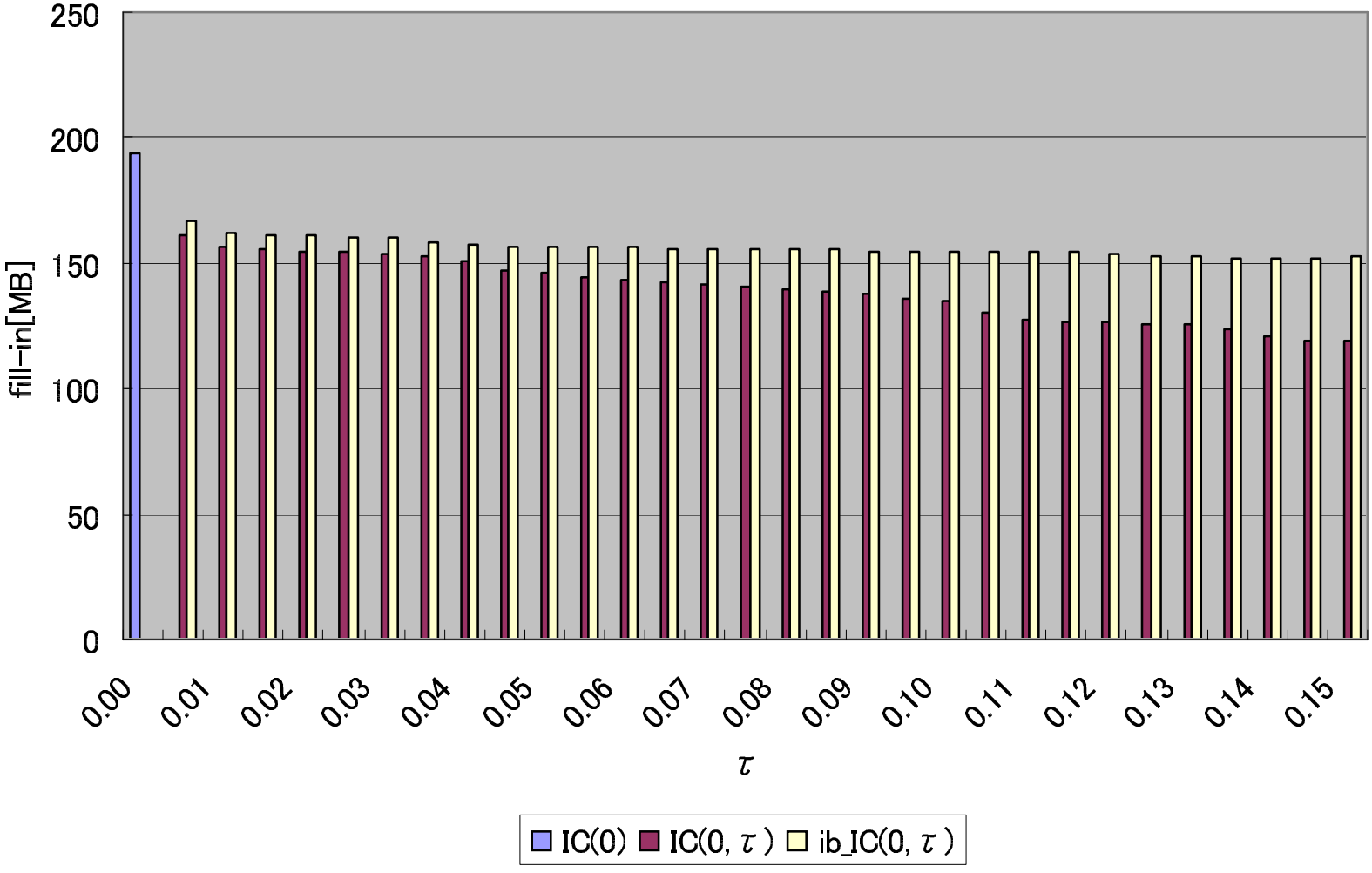
行列 SuperP10

前処理	τ	U の非零要素	反復回数	pre-時間	CG 時間	合計時間	メモリ量
IC(0)_s	破綻	-	-	-	-	-	-
WIC(0)_s	-	-	5185	0.02	132.9	132.9	24.8
IC(0)	-	1.59M	2912	14.7	72.7	87.5	51.3
IC(0,τ)	.025	.94M	2844	0.20	57.1	57.3	41.4
	.090	.64M	8476	0.13	150.8	150.9	36.9
ib_IC(0,τ)	.020	1.05M	2873	0.37	60.4	60.8	43.2
	.145	.87M	3925	0.18	77.6	77.8	40.4

行列 SuperP40:計算時間



行列 SuperP40:メモリ量



行列 SuperP40

前処理	τ	U の非零要素	反復回数	pre-時間	CG 時間	合計時間	メモリ量
IC(0)_s	破綻	-	-	-	-	-	-
WIC(0)_s	-	-	25885	0.07	2579.8	2579.9	93.1
IC(0)	-	6.05M	11629	80.7	1131.5	1212.3	193.5
IC(0,τ)	.010	3.64M	11429	0.93	896.9	897.9	156.7
	.090	2.37M	75927	0.49	5189.9	5190.4	137.4
ib_IC(0,τ)	.025	3.87M	11587	1.24	927.2	928.4	160.2
	.145	3.30M	16024	0.70	1211.2	1211.9	151.6

まとめ

1. 逆行列 U^{-T} の行のノルムの大きさに基づきドロッピングを行う対称行列用の ib_IC 分解を紹介した.
2. 元の行列の非零要素に基づく ib-IC(0,tol) 分解を考えた
3. ib_IC(0,τ) 分解 CG 法の収束は，従来のドロッピング手法に基づく IC(0) 分解や IC(0,τ) 分解 CG 法と比較して，

   (a) 収束が速く，
   (b) 使用メモリ量も少なく，
   (c) 収束が安全である

   ことが分かった.