電 301 数値解析
第 5 回
連立一次方程式の解法 (2) 反復解法

はじめに (1)

- 直接法は有限回の演算で連立一次方程式の解を求める手法.
- 有理数演算をサポートしている処理系では, 直接法は有理数に限定すれば, 数値計算の誤差なく連立一次方程式を解くことができる.

はじめに (2)

- 直接法は優れた方法ではあるが, 大規模疎行列を取り扱う際には他の手法を使った方がよいこともある.

- 要素の大部分が零の行列を疎行列という. 応用であらわれる大規模行列は多くの場合疎行列である (先週の復習).

はじめに (3)

- 反復法は繰り返しによって解を連立一次方程式の真の解に漸近させる手法であり, 係数行列のうち零でない要素だけを記憶しておけばよいという利点を持つ.

- 大規模疎行列には反復法が適している (と, もの本には書いてあるが …).

はじめに (4)

- 「大規模」「疎行列」といった概念は, 実用的な概念であって, 数学的な概念ではない.

- 大規模疎行列が応用上重要なのは, たとえば偏微分方程式を差分近似する際に疎行列があらわれるから.

はじめに (4)

- よって, 物理現象のシミュレ– ションをするときには, 大規模疎行列の取り扱いが必要になることがある.

- 偏微分方程式の数値解法は第 13 回, 第 14 回のテーマなので, 今回は深入りしない.

はじめに (5)

- 直接法は一意解を持つ連立一次方程式をすべて解けるが …
- 反復法は特殊な条件を満たす連立一次方程式にしか適用できない.

はじめに (6)

- 反復法の基本は Jacobi 法と Gauss-Seidel 法であるが (後述), これら以外にも, 偏微分方程式への応用をベースにした種々の解法がある.

- いくつか名前を挙げると, SOR 法, Chebyshev 加速法, ADI 法, マルチグリッド法など.

Jacobi 法 (1)

- $\boldsymbol{A}$ を n 行 n 列の行列, $\boldsymbol{x}$ および $\boldsymbol{b}$ を n 次のベクトルとする.

- $\boldsymbol{Ax} = \boldsymbol{b}$ を解きたい. ただし, 行列 $\boldsymbol{A}$ の対角要素はすべて零でないと仮定する (この条件が満たされない場合には Jacobi 法は適用できない).

Jacobi 法 (2)

- $\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots \\ a_{21} & a_{22} & a_{23} & \cdots \\ a_{31} & a_{32} & a_{33} & \cdots \\ \vdots & & & \end{pmatrix}$ としたとき …

Jacobi 法 (3)

- $\boldsymbol{A}$ の対角要素のみを抜き出した行列を $\boldsymbol{D}$ とすると,

$$\boldsymbol{D} = \begin{pmatrix} a_{11} & 0 & \cdots & \\ 0 & a_{22} & 0 & \\ \vdots & 0 & a_{33} & \ddots \\ \vdots & & \ddots & \ddots \end{pmatrix}$$

Jacobi 法 (4)

- $\boldsymbol{A} - \boldsymbol{D}$ 左下の部分を $\boldsymbol{E}$ とすると,

$$\boldsymbol{E} = \begin{pmatrix} 0 & \cdots & & \\ a_{21} & 0 & & \\ a_{31} & a_{32} & 0 & \\ \vdots & & \ddots & \ddots \end{pmatrix}$$

Jacobi 法 (4)

- $\boldsymbol{A} - \boldsymbol{D}$ 右上の部分を $\boldsymbol{F}$ とすると,

$$\boldsymbol{F} = \begin{pmatrix} 0 & a_{12} & a_{13} & \cdots \\ \vdots & 0 & a_{23} & \\ & & 0 & \ddots \\ & & \cdots & \ddots \end{pmatrix}$$

Jacobi 法 (5)

- $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ は, $(\boldsymbol{D} + \boldsymbol{E} + \boldsymbol{F})\,\boldsymbol{x} = \boldsymbol{b}$ と書ける.
- 上記の左辺第 2・3 項を右辺に移項し, 両辺に $\boldsymbol{D}^{-1}$ を掛けると,
  $\boldsymbol{x} = -\boldsymbol{D}^{-1}(\boldsymbol{E} + \boldsymbol{F})\,\boldsymbol{x} + \boldsymbol{D}^{-1}\boldsymbol{b}.$
- これに基づき, 次の漸化式を考える.

$$\boldsymbol{x}(k+1) = -\boldsymbol{D}^{-1}(\boldsymbol{E} + \boldsymbol{F})\,\boldsymbol{x}(k) + \boldsymbol{D}^{-1}\boldsymbol{b}$$

Jacobi 法 (6)

- 漸化式 $\boldsymbol{x}(k+1) = -\boldsymbol{D}^{-1}(\boldsymbol{E}+\boldsymbol{F})\boldsymbol{x}(k) + \boldsymbol{D}^{-1}\boldsymbol{b}$ の解が一定値 $\bar{\boldsymbol{x}}$ に収束するならば …
- この漸化式の両辺で $k \to \infty$ とすると …
- $\bar{\boldsymbol{x}} = -\boldsymbol{D}^{-1}(\boldsymbol{E}+\boldsymbol{F})\bar{\boldsymbol{x}} + \boldsymbol{D}^{-1}\boldsymbol{b}$ なので, $\bar{\boldsymbol{x}}$ は $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ の解である.

Jacobi 法 (7)

- 初期値 $\boldsymbol{x}(0)$ を定め (何でもよい), 漸化式 $\boldsymbol{x}(k+1) = -\boldsymbol{D}^{-1}(\boldsymbol{E}+\boldsymbol{F})\boldsymbol{x}(k)+\boldsymbol{D}^{-1}\boldsymbol{b}$ の解を $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ の近似解とする方法が Jacobi 法.

- Jacobi 法は行列 $\boldsymbol{A}$ の対角要素がすべて非零であれば動かせるが, 列 $(\boldsymbol{x}(k))$ が発散することもあり得る.

Jacobi 法 (8)

- 列 $(\boldsymbol{x}(k))_{k\in\mathbb{N}}$ が $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$ の解に収束するための必要十分条件は, 差分方程式

$$\boldsymbol{x}(k+1) = -\boldsymbol{D}^{-1}\left(\boldsymbol{E}+\boldsymbol{F}\right)\boldsymbol{x}(k) + \boldsymbol{D}^{-1}\boldsymbol{b}$$

  が漸近安定, すなわち行列 $\boldsymbol{D}^{-1}\boldsymbol{E}$ のすべての固有値の絶対値が 1 未満となること.

Jacobi 法 (9)

- 以上の説明では便宜上 $\boldsymbol{D}$ の逆行列を明示的に書き表したが, 実際には $\boldsymbol{D}$ の逆行列を使うわけではない.
- $\boldsymbol{D}^{-1} = \begin{pmatrix} \frac{1}{a_{11}} & 0 & \cdots \\ 0 & \frac{1}{a_{12}} & \\ \vdots & & \ddots \end{pmatrix}$ だから …

Jacobi 法 (10)

- $\boldsymbol{D}^{-1}(\boldsymbol{E}+\boldsymbol{F})$ は, 行列 $\boldsymbol{E}+\boldsymbol{F}$ の第 1 行から第 n 行までにそれぞれ $1/a_{11}, \ldots, 1/a_{nn}$ を掛けたもの. なお零要素については計算不要.

- $\boldsymbol{D}^{-1}\boldsymbol{b}$ は, ベクトル $\boldsymbol{b}$ の第 1 行成分から第 n 成分までにそれぞれ $1/a_{11}, \ldots, 1/a_{nn}$ を掛けたもの. なお零要素については計算不要.

Gauss-Seidel 法 (1)

- Gauss-Seidel 法は $(\boldsymbol{D}+\boldsymbol{E}+\boldsymbol{F})\,\boldsymbol{x}=\boldsymbol{b}$ を Jacobi 法とは違った形で整理する．すなわち，初期値 $\boldsymbol{x}(0)$ を定め (何でもよい)，

$$(\boldsymbol{D}+\boldsymbol{E})\,\boldsymbol{x}(k+1)=\boldsymbol{b}-\boldsymbol{F}\boldsymbol{x}(k)$$

という漸化式を解く．

Gauss-Seidel 法 (2)

Gauss-Seidel 法を成分ごとに書くと

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(- \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} + b_i \right)$$

となる. ただし $x_i^{(k)}$ は第 k 回目の繰り返しにおけるベクトル $\boldsymbol{x}$ の第 i 成分. 成分ごとの差分方程式を使った方がメモリ消費を減らせるが, 行列を使った場合と比べてどちらが速いかは処理系によって変わる.

Gauss-Seidel 法 (3)

- Gauss-Seidel 法によって得られる列 $(\boldsymbol{x}(k))_{k\in\mathbb{N}}$ が $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ の解に収束するための必要十分条件は, $(\boldsymbol{D} + \boldsymbol{E})^{-1}\boldsymbol{F}$ のすべての固有値の絶対値が 1 未満となること.

SOR 法 (1)

- SOR 法とは, Successive Over-Relaxation 法の略であり, 逐次過大緩和法と訳される.
- SOR 法は設計パラメータ w を含む (ただし $0 < w < 2$).
- SOR 法とは, 初期値 $\boldsymbol{x}(0)$ を定め (何でもよい), 次ページで与える漸化式を解く方法.

SOR 法 (2)

次の漸化式を解く.

$$\frac{1}{w}\left(\boldsymbol{D}+w\boldsymbol{E}\right)\boldsymbol{x}(k+1)=\frac{1}{w}\left((1-w)\boldsymbol{D}-w\boldsymbol{F}\right)\boldsymbol{x}(k)+\boldsymbol{b}$$

成分ごとに書くと次のようになる.

$$y_i^{(k+1)}=\frac{1}{a_{ii}}\left(-\sum_{j<i}a_{ij}x_j^{(k+1)}-\sum_{j>i}a_{ij}x_j^{(k)}+b_i\right)$$

$$x_i^{(k+1)}=x_i^{(k)}+w\left(y_i^{(k+1)}-x_i^{(k)}\right)$$

SOR 法 (3)

- SOR 法で得られた列 $(\boldsymbol{x}(k))_{k\in\mathbb{N}}$ が $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$ の解に収束するための必要十分条件は, 行列

$$(\boldsymbol{D}+w\boldsymbol{E})^{-1}\left((1-w)\boldsymbol{D}-w\boldsymbol{F}\right)$$

のすべての固有値の絶対値が 1 未満となること.

SOR 法 (3)

- SOR 法の収束性はパラメータ w に依存する.
- パラメータ w の値によって収束の速さが変わるが, 大きい方がよいとも小さい方がよいともいえない.
- 実用上は, w を試行錯誤によって定めるが, w を解析的に求められる問題もある.

数値例 (1)

- $\boldsymbol{A}$ を, 次のような形の正方行列とする:

$$\boldsymbol{A} = \begin{pmatrix} 100 & 1 & & & \\ 1 & 100 & 1 & & \\ & 1 & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots \end{pmatrix}$$

(空白の部分の要素はすべて零).

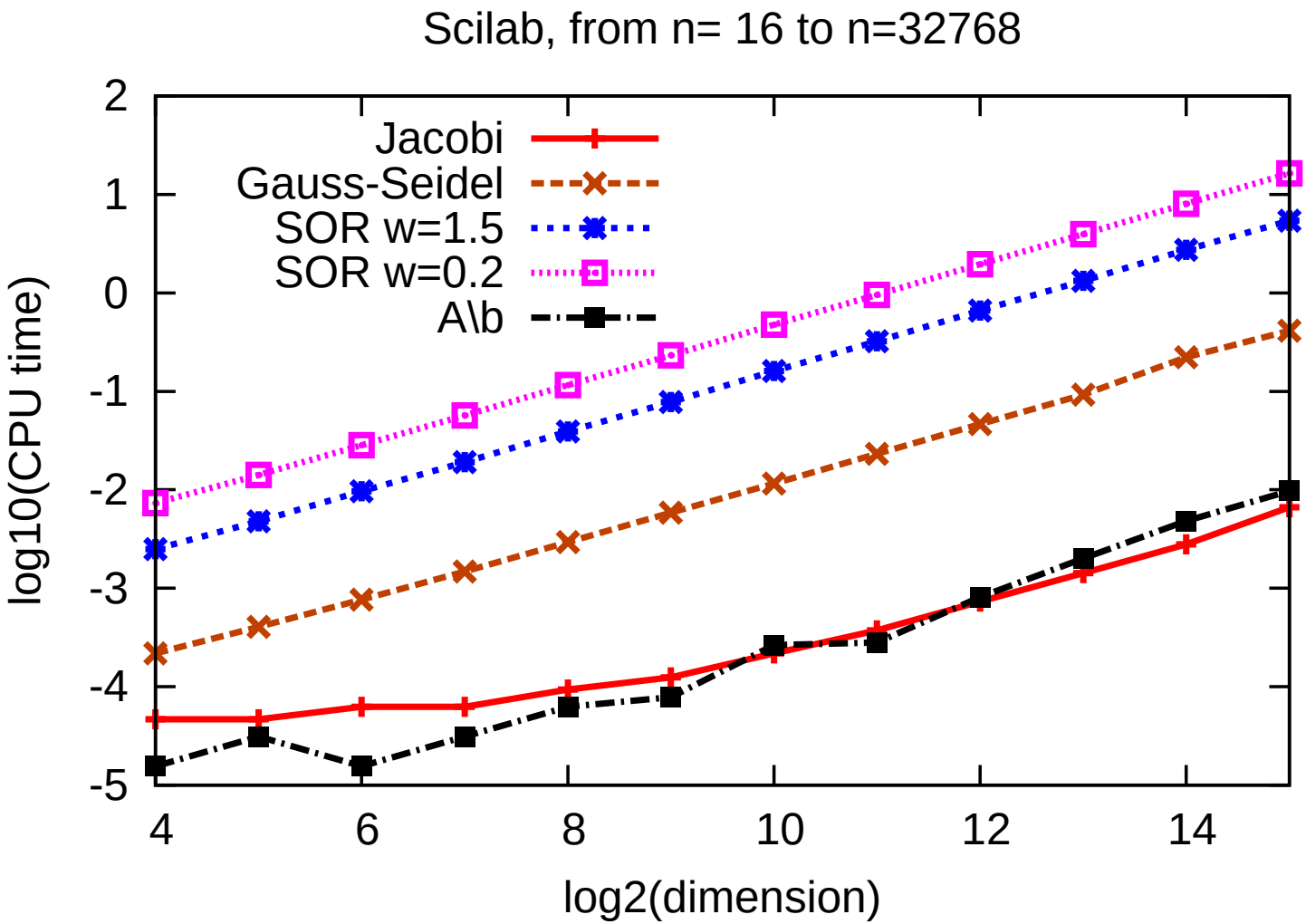
- このような行列を三重対角行列という.

数値例 (2)

- $\boldsymbol{A}$ の次元を n とする.
- Scilab および MATLAB において $n = 2^i, \quad 4 \leq i \leq 15$, $\boldsymbol{b} = (1, \ldots, 1)^T$ とし, $\boldsymbol{Ax} = \boldsymbol{b}$ を各アルゴリズムで 1000 回解いて平均時間を測定した (n が大きい方から順に数値実験).
- 反復法では, $\|\boldsymbol{Ax} - \boldsymbol{b}\| < 10^{-8}$ となった時点で求解成功とした.

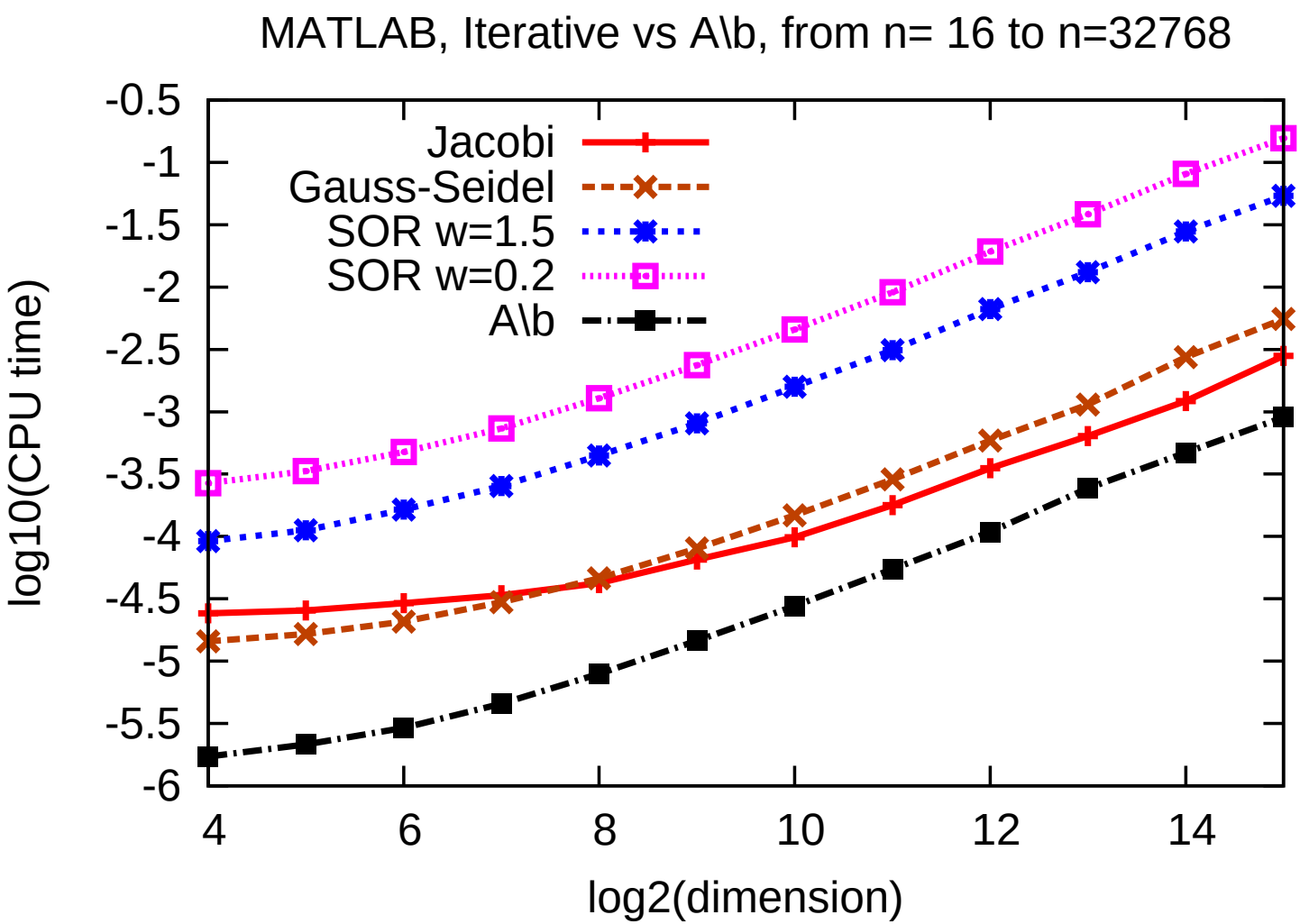
数値例 (3)

- 実行環境は以下の通り: ソフトのバージョン: Scilab 5.5.2 (64bit), MATLAB R2015b, OS: Windows7 Professional Service Pack 1 (64bit), CPU: Intel Core i5-4690 3.5GHz, メモリ: 32GB
- 以下にグラフを示す. 横軸は $\log_2$(問題の次元), 縦軸は ($\log_{10}$(計算時間)).
- まず Scilab の結果を見る.


Scilab, from n= 16 to n=32768
Jacobi
Gauss-Seidel
SOR w=1.5
SOR w=0.2
A\b
log10(CPU time)
log2(dimension)
2
1
0
-1
-2
-3
-4
-5
4
6
8
10
12
14

数値例 (5)

- $n = 2^{11}$ のあたりで Jacobi 方は `A\b` より速くなる.
- この例では Gauss-Seidel 法と SOR 法には良いところがない. 対数軸になおさずに, $n = 2^{15}$ において Jacobi 法の計算時間を 1 に正規化して比較すると Gauss-Seidel 法:62.2, SOR 法 ($w = 1.5$):815.0, SOR 法 ($w = 0.2$):2456.9, `A\B`:1.5 となる.
- MATLAB は …


MATLAB, Iterative vs A\b, from n= 16 to n=32768
log10(CPU time)
log2(dimension)
Jacobi
Gauss-Seidel
SOR w=1.5
SOR w=0.2
A\b
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
-4.5
-5
-5.5
-6
4
6
8
10
12
14

数値例 (7)

- MATLAB ではこの例では一貫して `A\b` が速い. 反復法には優位性なし.
- 対数軸になおさずに, $n = 2^{15}$ において Jacobi 法の計算時間を 1 に正規化して比較すると Gauss-Seidel 法:2.08, SOR 法 ($w = 1.5$):19.2, SOR 法 ($w = 0.2$):55.7, `A\B`:0.3 となる.
- 次に, MATLAB と Scilab を比較してみる.

数値例 (8)

- $n = 2^4 (= 16)$ および $n = 2^{15} (= 32768)$ において, MATLAB の A\b を 1 に規格化して計算時間を比較.
- $n = 16$:

	J	GS	SOR1.5	SOR0.2	A\b
Scilab	27.19	126.90	1450.27	4260.18	9.06
MATLAB	14.06	8.35	53.17	155.26	1.00

- $n = 32768$:

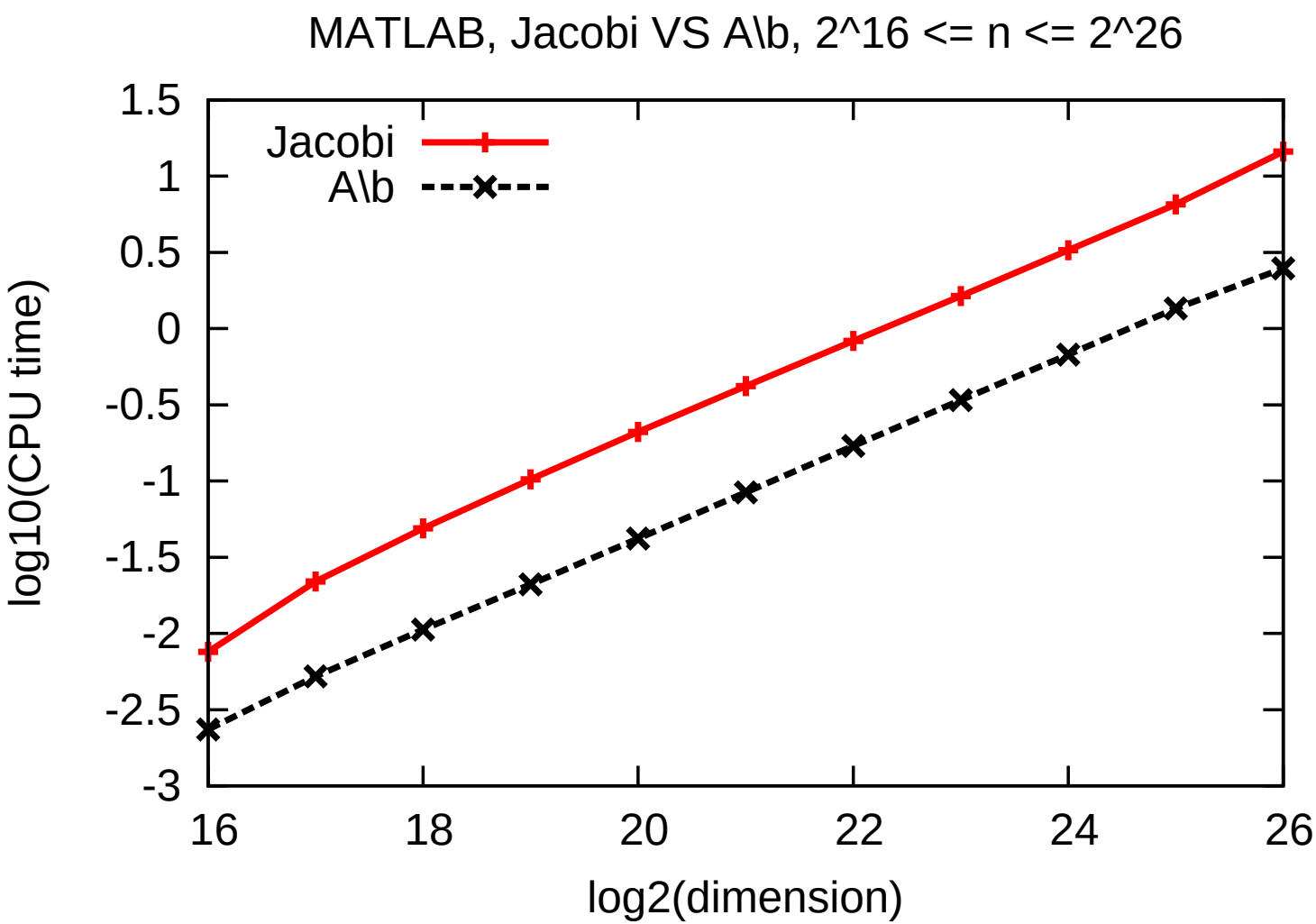
	J	GS	SOR1.5	SOR0.2	A\b
Scilab	7.35	457.15	5987.36	18048.62	10.76
MATLAB	3.10	6.14	59.46	172.71	1.00

数値例 (9)

- MATLAB は一貫して Scilab より速い.
- 経験上, Scilab では, プログラム中に多数の for 文等の繰り返し文が含まれる場合, 実行が顕著に遅くなる. Scilab において Gauss-Seidel 法と SOR 法の実行が遅いのは, これが原因であると推測される.

数値例 (10)

- MATLAB で, $n = 2^i, \quad 16 \leq i \leq 26$ として, 同様の数値実験をおこなった結果を次ページに示す. この例では, 各次元でのサンプルは 1 個で, 平均を取っていない.
- この例題は反復解法向きであると思われるが, Jacobi 法の優位性は見られない.


MATLAB, Jacobi VS A\b, 2^16 <= n <= 2^26
Jacobi
A\b
log10(CPU time)
log2(dimension)
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
16
18
20
22
24
26

共役勾配法とは (1)

- 連立一次方程式の代表的な解法は大別すると直接法と反復法であるが …
- 直接法と反復法を組み合わせた解法があり，共役勾配法と呼ばれる．

共役勾配法とは (2)

- 共役勾配法は非線形最小化問題に適用される最急降下法という手法から派生した手法.
- この手法は 1952 年に提案されたが, 数値計算の誤差に弱いため不遇の時代が続いた. しかし, 前処理によって特性が改善されることが判明し, 見直されている.

共役勾配法とは (3)

- 共役勾配法はいまだん研究が続いている方法.
- 非線形最適化問題の解法としての拡張が可能.
- 線形計算の研究を志すのであれば共役勾配法は必須であるが, 一般的な工学系の選択科目としては専門的すぎる内容と思われるので, この講義では概要のみ紹介する.

最急降下法 (1)

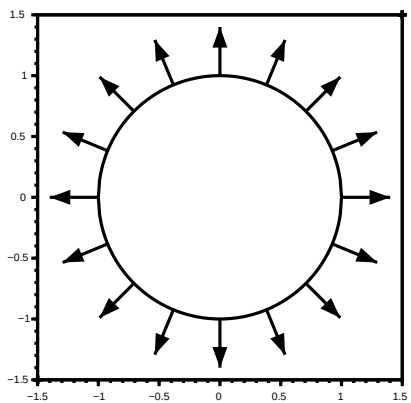
- 最急降下法は, 変数ベクトル $\boldsymbol{x}$ に関する実数値関数 $f(\boldsymbol{x})$ を最小化 (あるいは最大化) する手法のひとつ.
- 関数 f の勾配ベクトルを $\nabla f = (\frac{\partial f}{\partial \boldsymbol{x}})^T$ とする.
- ∇f は関数 f の等高線の外向き法線ベクトルを与える.

最急降下法 (2)

- ∇f は関数 f の等高線の外向き法線ベクトルだから, 関数 f が一定の条件を満たすときには, 解を $-\nabla f$ の方向に少しずつ動かせば, 解は f の最小値を与える点 $\boldsymbol{x}_*$ に収束する. この方法を最急降下法という.

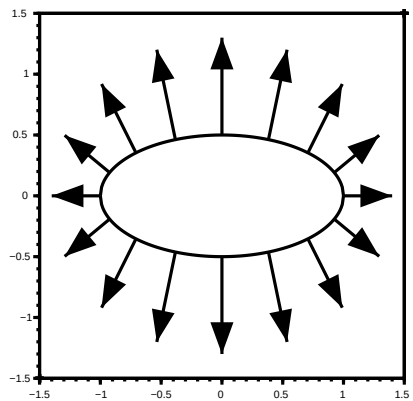
最急降下法 (3)

- $f(x, y) = x^2 + y^2$ のように, 内向き法線ベクトルと関数が最小となる点の方向が近い場合には最急降下法はそれなりに高効率だが …



最急降下法 (3)

- 関数の等高線が細長い楕円になっているような場合には効率が悪い.



共役方向法と共役勾配法 (1)

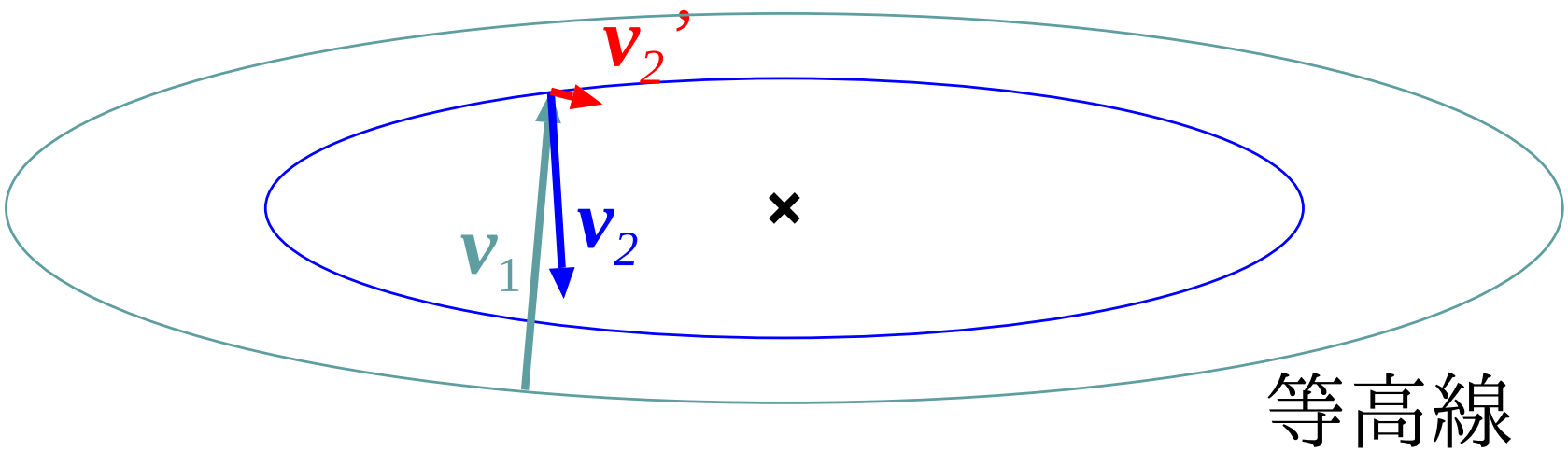
- 共役勾配法は, 共役方向法の一種.
- 以下, 解を動かす方向を探索ベクトルと呼ぶ.
- 共役方向法は最急降下法の改良版. 過去の勾配ベクトルの系列を直交化して探索ベクトルを作ることが特徴.

共役方向法と共役勾配法 (2)

- 探索ベクトルを作るには, 勾配ベクトルから過去の探索ベクトルと線形独立な成分を抽出する (射影を用いる).

- これがなぜ効率的かは, 関数 $f(\boldsymbol{x})$ の等高線が楕円の場合を考えればわかる (次ページ).

共役方向法と共役勾配法 (3)



等高線が楕円の場合の内向き法線ベクトル
($(-1)\times$ 勾配ベクトル) とその直交化

共役方向法と共役勾配法 (4)

- 共役方向法は「探索ベクトルの直交化」ということしか主張していない.
- 共役勾配法は, 共役方向法の枠内で, より具体的に探索ベクトルの構成法を与える.
- 以下では $(\boldsymbol{x}, \boldsymbol{y})$ により $\boldsymbol{x}$ と $\boldsymbol{y}$ の内積を表す.

共役方向法と共役勾配法 (5)

- もっとも単純な共役勾配法は, 行列 $\boldsymbol{A}$ が正定対称行列である場合を対象とする.
- 解くべき問題は, $\boldsymbol{Ax} = \boldsymbol{b}$ の解 $\boldsymbol{x}$ を求めることである.
- この場合の共役勾配法のアルゴリズムは次ページに示す通り (典拠は杉原・室田, p. 150).

共役勾配法

(初期化) $k = 0$ とし, 初期値 $\boldsymbol{x}_0$ を定め, $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$, $\boldsymbol{p}_0 = \boldsymbol{r}_0$ とする. 終了条件に相当するパラメータ $\varepsilon > 0$ を定める.

(ループ) $\|\boldsymbol{r}_k\| < \varepsilon\|\boldsymbol{b}\|$ であれば終了. そうでなければ, $\alpha_k = (\boldsymbol{r}_k, \boldsymbol{p}_k)/(\boldsymbol{p}_k, \boldsymbol{A}\boldsymbol{p}_k)$, $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$, $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \boldsymbol{A}\boldsymbol{p}_k$, $\beta_k = -(\boldsymbol{r}_{k+1}, \boldsymbol{A}\boldsymbol{p}_k)/(\boldsymbol{p}_k, \boldsymbol{A}\boldsymbol{p}_k)$, $\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{p}_k$ とし, $k = k + 1$ としてループ冒頭に戻る.

共役方向法と共役勾配法 (6)

- 共役勾配法は数値計算の誤差の影響を受けやすいので, 実用上は, $\boldsymbol{C}$ をある正則行列とし, 連立一次方程式 $\boldsymbol{Ax} = \boldsymbol{b}$ を, $(\boldsymbol{C}^{-1}\boldsymbol{AC}^{-T})(\boldsymbol{C}^{T}\boldsymbol{x}) = \boldsymbol{C}^{-1}\boldsymbol{b}$ というふうに変形してから共役勾配法を適用する. 行列 $\boldsymbol{C}$ を使って問題を変形する操作を前処理という.

- 前処理のしかたは色々あるが, 決定版と言うべき方法はない.

共役方向法と共役勾配法 (7)

- 行列 $\boldsymbol{A}$ が正定対称行列でない場合の共役勾配法は, たとえば目的関数 $(\boldsymbol{Ax}-\boldsymbol{b}, \boldsymbol{Ax}-\boldsymbol{b})$ に関する最小化問題を解く, といったような形で定式化される.

- 上記の方法を一般化共役残差法 (Generalized Conjugate Residual 法; GCR 法) とよぶ.

共役方向法と共役勾配法 (8)

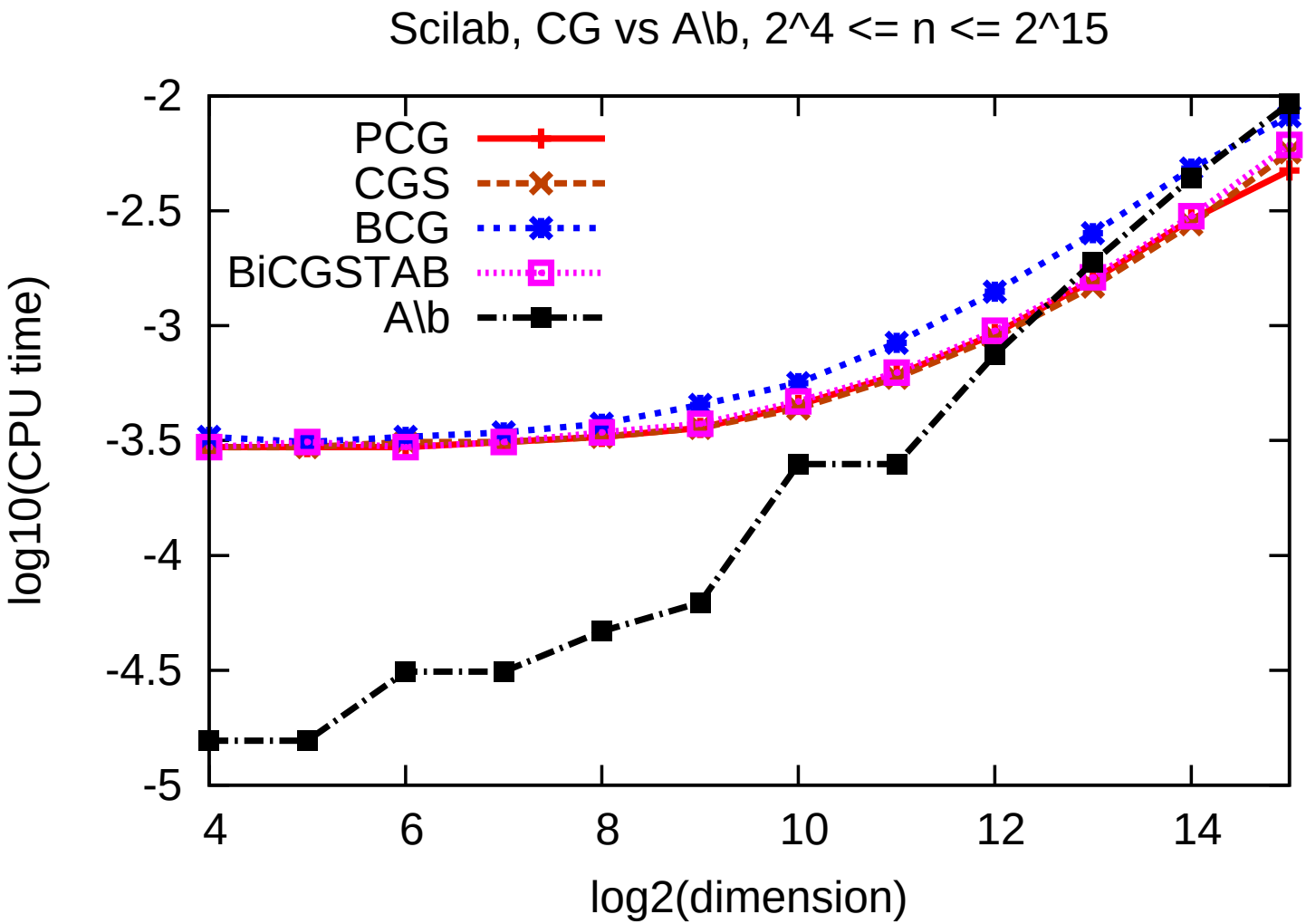
- これ以外に, GCR(m) 法, Orthomin(m) 法, 一般化最小残差法 (Generalized Minimal RESidual 法; GMRES 法), 双共役勾配法 (BiConjugate-Gradient 法; BCG 法), 擬似最小残差法 (Quasi-Minimal Residual 法; QMR 法), 安定化双共役勾配法 (BiConjugate Gradient STABilized 法; BiCGSTAB 法), Conjugate Gradient Squared 法 (CGS 法) など, 様々な方法がある.

Scilab・MATLABの共役勾配法 (1)

- 組み込み関数 conjgrad により共役勾配法が使える.
- オプション指定により前処理付き共役勾配法, 前処理付き 2 乗共役勾配法, 前処理付き BCG 法, 前処理付き BiCGSTAB 法を選択することができる.

Scilab・MATLABの共役勾配法 (2)

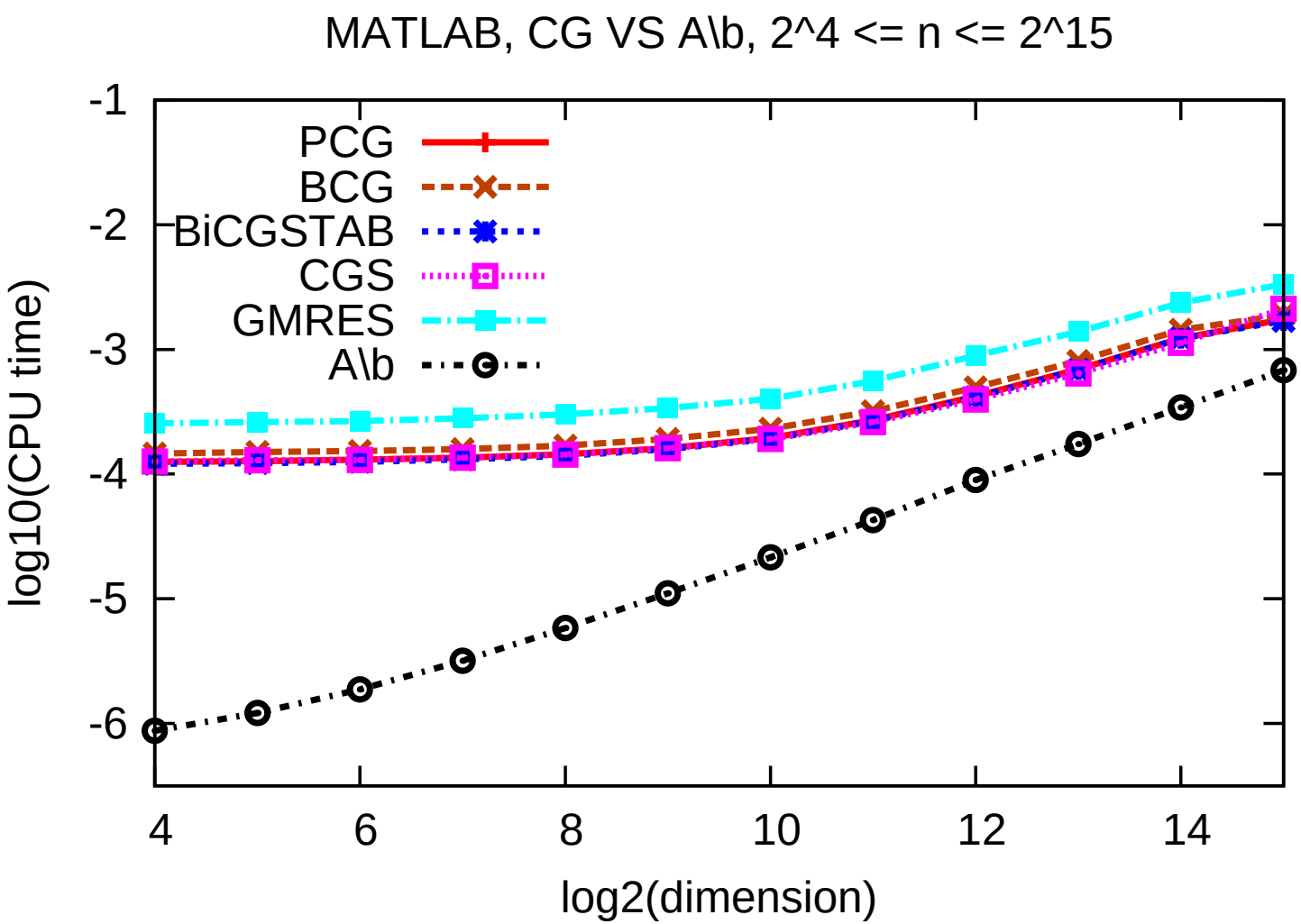
- 先に使った三重対角行列の問題を解いてみると $\cdots$ ($n = 2^i, \quad 4 \leq i \leq 15$, 1000 回解いた平均時間, 横軸, 縦軸とも対数).


Scilab, CG vs A\b, 2^4 <= n <= 2^15
log10(CPU time)
log2(dimension)
PCG
CGS
BCG
BiCGSTAB
A\b
-2
-2.5
-3
-3.5
-4
-4.5
-5
4
6
8
10
12
14

Scilab・MATLABの共役勾配法 (3)

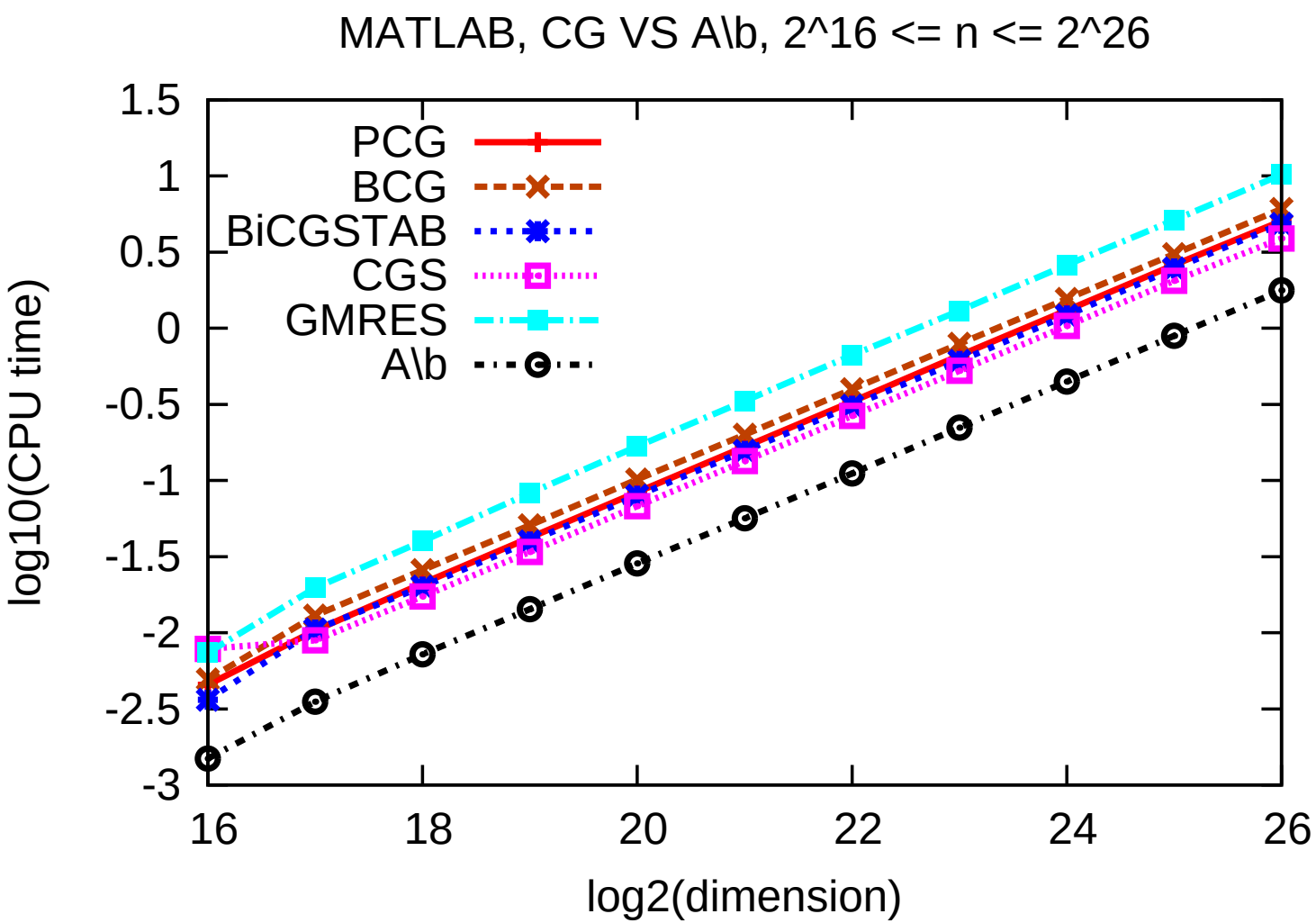
- $n = 12$ までは `A\b` がどの共役勾配法より速いが, $n = 13$ で BCG を除き逆転する. $n = 15$ で BCG も `A\b` より速くなる.

- MATLAB はどうかというと … (条件は先と同様).


MATLAB, CG VS A\b, 2^4 <= n <= 2^15
log10(CPU time)
log2(dimension)
PCG
BCG
BiCGSTAB
CGS
GMRES
A\b
-1
-2
-3
-4
-5
-6
4
6
8
10
12
14

Scilab・MATLABの共役勾配法 (5)

- 共役勾配法に `A\b` に対する優位性なし.
- $n = 2^i, \quad 16 \leq i \leq 26$ として, 同様の数値実験をおこなった結果 (サンプル 1 個, 平均なし) を次ページに示す.
- やはり共役勾配法には `A\b` に対する優位性なし.


MATLAB, CG VS A\b, 2^16 <= n <= 2^26
log10(CPU time)
log2(dimension)
PCG
BCG
BiCGSTAB
CGS
GMRES
A\b
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
16
18
20
22
24
26