第8章『大規模行列の扱い方』

数値計算ツールで手軽に大規模線形問題を解く

荻田 武史 (JST/早大・理工), 大石 進一 (早大・理工)

第3回 講習会
神奈川大学 横浜キャンパス
2006年9月1日

はじめに

工学における問題を数値的に解く場合，連立一次方程式

$$Ax = b$$

や固有値問題

$$Ax = \lambda x$$

に帰着して，これを数値解法によって解くことが多い．

$\Longrightarrow$ 係数行列 A は大規模スパース行列 (sparse matrix)

$\Longrightarrow$ 大規模スパース行列の扱い方が重要

数値計算ツール

手軽に数値計算を実行できるインタプリタ型ツールは色々ある.

- Matlab (MathWorks, 商用)
- Octave (GNU, フリー)
- Scilab (INRIA, フリー)

今回は，スパース行列の扱いにも優れている Matlab を用いる.

(Matlab demo)

直接解法と反復解法

連立一次方程式の解法

- CG法，Bi-CG法などの反復解法 (計算量・メモリ量の節約)
- ガウス消去法などの直接解法 (ロバスト)

反復解法の特徴

一連の手続きを何度も繰り返し，解の精度を上げる方法

- 係数行列のスパース性を保持できる (メモリ量)
- 繰り返しの過程で解の精度をある程度推測しながら，適当なところで計算を打ち切ることができる (計算量)
- 丸め誤差によって，必ずしも真の解に収束するとは限らない

対称正定値行列などの一部の特殊な行列を除いては，決定的なアルゴリズムを作ることは困難

$\Longrightarrow$ 問題に応じてアルゴリズムを変える必要がある

直接解法の特徴

LU 分解 (ガウス消去法)，Cholesky 分解等

- 数値計算的に非常に安定 (ロバスト)
- 係数行列のスパース性を著しく失わせる (メモリ量)
- 常に一定の計算量が必要

直接解法は，本来，密行列向きの解法であり，スパース行列用には，それに特化したスパースダイレクトソルバ (sparse direct solver) を用いる．

Matlabにおける連立一次方程式の直接解法

直接解法は，必要とする計算量とメモリ量の点から，比較的小規模な問題(パソコンだと数千次元まで)に適用されることが多い.

大規模スパース系に適用する直接解法は，スパースダイレクトソルバ (sparse direct solver) と呼ばれ，構造解析などの数値解法でよく使われるスカイライン法もこれに属する.

Matlab におけるスパースダイレクトソルバ (UMFPACK) は，実は大変使いやすく設計されている．

密行列を係数行列とする連立一次方程式を解く場合と同様に

```
>> x = A \ b;
```

とするだけで，A がスパース行列であっても簡単に解を得ることができる．

(Matlab demo)

デフォルトでは，スパース行列のリオーダリング(再順序付け)には近似最小次数順序法 (approximate minimum degree ordering) を使っている．

リオーダリングについても，計算量・メモリ量節約のために，グラフ理論などを用いた様々な手法が提案されている．

これにより，計算量とメモリ量の両方が最適に近いくらいまで低減されているため，Matlabではほとんど何も考えなくても，比較的小規模なスパース系の連立一次方程式が直接解法によって解けてしまう．

また，これらのスパース行列手法のパラメタ設定については

```
>> help spparms
```

とすることによってその詳細を知ることができる．

現在，計算の高速化やメモリ量の低減などをテーマにしたスパースダイレクトソルバに関する研究は盛んに行われている．しかしながら，それでも問題がある程度以上の大きさになると直接解法で扱うのは困難になってくる (現在のパソコンでは，数十万次元程度が限度)．

Matlabにおける連立一次方程式の反復解法

- ガウス–ザイデル法 (Gauss-Seidel method) やSOR法 (逐次過剰緩和法; Successive Over-Relaxation method) を代表とする定常反復法 (stationary iterative method)

- CG法 (共役勾配法; Conjugate Gradient method) を代表とする非定常反復法 (non-stationary iterative method)

Matlabには，現時点 (version 2006a) では9つの反復解法が用意されているが，それらはすべて非定常反復法である．表 1 に列挙する．

Table 1: Matlabで使用できる反復解法

関数	反復解法の名前
bicg	Bi-CG法: 双共役勾配法
bicgstab	Bi-CGSTAB法: 安定化双共役勾配法
cgs	CGS法: 自乗共役勾配法
gmres	GMRES法: 一般化最小残差法
lsqr	LSQR法: 共役勾配法型の最小自乗法の実現
minres	MINRES法: 最小残差法
pcg	PCG法: 前処理付き共役勾配法
qmr	QMR法: 準最小残差法
symmlq	SYMMLQ法: 対称LQ法

Table 2: 各反復解法の適用可能な行列の範囲

関数	非対称 (nonsymmetric)	不定値 (indefinite)
bicg	○	○
bicgstab	○	○
cgs	○	○
gmres	○	○
lsqr	○	○
minres	×	○
pcg	×	×
qmr	○	○
symmlq	×	○

ここで，$A = (a_{i,j})$ を正方行列とすると，非対称 (nonsymmetric) とは，$a_{i,j} \neq a_{j,i}$ である (i, j) が存在することである．また，不定値 (indefinite) とは，正定値 (positive definite) でも負定値 (negative definite) でもないことで，A が実対称行列の場合は A が異なる符号の固有値を持つことと同値である．これらの反復解法は，A が複素行列の場合でも適用可能である．また，関数lsqrだけは，A が正方行列でなくてもよく，その場合はノルム $\|b - Ax\|_2$ を最小にするような最小自乗解 x が得られる．

以後，これら9つの関数の中でも実際によく使われていてポピュラなpcg，bicgstab，gmresの3つについて，Matlabでの使用法と数値実験結果を示す．

PCG法(前処理つき共役勾配法)

PCG法は係数行列 A が対称正定値の場合に最も効果的な解法である.

但し，与えられた係数行列が正定値かどうかを判定するのは困難であるため，あらかじめ係数行列が正定値になることが分かっている問題に有効である.（最近は，不定値でも使えるという報告もある）

A を $n \times n$ 対称正定値行列，b を n 次ベクトルとすると，関数 pcgの最も簡単な使い方は，

```
>> x = pcg(A,b);
```

である．このとき，反復停止条件はデフォルトの

$$\frac{\|b - Ax\|_2}{\|b\|_2} < 10^{-6}$$

が仮定され，最大反復回数はデフォルトの $\min\{n, 20\}$ となる．反復停止条件は，

```
>> x = pcg(A,b,tol);
```

とすることによって

$$\frac{\|b - Ax\|_2}{\|b\|_2} < \mathrm{tol}$$

が仮定される．また，

```
>> x = pcg(A,b,tol,maxit);
```

とすることによって，最大反復回数は maxit となる．これらは，前処理 (precondition) を使用していないので，CG法と同値である．

(Matlab demo)

前処理を使用する場合は,

```
>> x = pcg(A,b,tol,maxit,M);
```

あるいは

```
>> x = pcg(A,b,tol,maxit,M1,M2);
```

のようにする．関数pcgでは，前処理行列 (preconditioner) M あるいは $M = M_1 M_2$ のような下三角行列 M_1 と上三角行列 M_2 を用いて

$$Ax = b$$

の代わりに

$$M^{-1}Ax = M^{-1}b$$

を解く．前処理は，係数行列 A の条件を良くして問題を解きやすくするための技法である．前処理の技法はこれ以外にも色々存在するが，それらを実現するためには，自分でMatlabファイルを書き換えなければならない．逆に言うと，Matlabの反復解法の関数は組み込み関数と違ってある程度編集可能であるため，自分の使いやすい様に変形することができる．

前処理行列には，IC分解 (不完全Cholesky分解) によって得られる三角行列が有効である．IC分解を行う関数cholincは，

```
>> R = cholinc(A,droptol);
```

のように使う．出力引数 R は $A \approx R^H R$ であることを理想とする上三角行列である．droptolは，IC分解過程におけるfill-inの基準となるしきい値であり，例えば，$\mathrm{droptol} = 10^{-3}$ にしたいときは，

```
>> droptol = 1e-3;
```

のようにする．この値を小さくすればするほど，完全なCholesky分解に近づくが，fill-inがその分増加して，必要なメモリ量が莫大になる．また，

```
>> droptol = '0';
```

のときは，fill-inは行わず，したがって，このとき R は A の上三角部分と同じスパースパターンを持つ．また，2つめの入力引数を構造体にして呼び出すこともできる．以下のような最大3つのフィールドを持つ構造体optsに対して

opts.droptol: IC分解過程におけるfill-inの基準となるしきい値

opts.michol: $\mathrm{michol} = 0$: IC分解 (デフォルト)，$\mathrm{michol} = 1$: MIC分解 (修正不完全Cholesky分解; Modified Incomplete Cholesky factorization)

opts.rdiag: 特異な要素を避けるため，R の対角上のゼロを置

き換える．rdiag $= 0$: 置き換えない (デフォルト)，michol $=$ 1: 置き換える

のうち，興味のあるフィールドを選んで設定することができる．例えば，以下のように使う．

```
>> opts = struct('droptol',1e-3,'michol',1);
>> R = cholinc(A,opts);
```

このとき，droptol $= 10^{-3}$ が仮定され，michol $= 1$ により MIC 分解が実行される．また，

```
>> x = pcg(A,b,tol,maxit,M1,M2,x0);
```

とすることによって，初期推定値 x_0 を設定することもできる．逆に，x_0 を省略した場合は，すべてがゼロ要素のベクトルが初期推定値となる．

より詳細な結果が欲しいときは，出力引数を増やすことが可能で，最も出力引数が多い場合では

```
>> [x,flag,relres,iter,resvec] = pcg(A,b,...);
```

と呼び出すことができる．それぞれ，x には数値解，flag には反復停止時の収束に関する情報 (flag が 0 なら収束した，1 なら収

束しなかった，など)，relres には相対残差

$$\text{relres} = \frac{\|b - Ax\|_2}{\|b\|_2}$$

が出力される．iter には収束までの反復回数が出力される (解が収束しなかった場合は，iter = maxit)．最後の resvec には，残差ノルム $\|b - Ax\|_2$ の履歴がベクトル形式で出力される．

対称正定値行列に対するPCG法の適用例

対称正定値行列の例として，テスト行列にはpoisson (Poisson方程式を差分法で離散化して得られるようなブロック三重対角行列) を用いることにする．例えば,

```
>> m = 3; A = gallery('poisson',m)
```

とすると,

$$A = \left[\begin{array}{ccc|ccc|ccc} 4 & -1 & 0 & -1 & 0 & 0 & & & \\ -1 & 4 & -1 & 0 & -1 & 0 & & O & \\ 0 & -1 & 4 & 0 & 0 & -1 & & & \\ \hline -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ \hline & & & -1 & 0 & 0 & 4 & -1 & 0 \\ & O & & 0 & -1 & 0 & -1 & 4 & -1 \\ & & & 0 & 0 & -1 & 0 & -1 & 4 \end{array}\right]$$

が得られる．行列 A のサイズは，$m^2 \times m^2$ である．このとき，A はsparse型であり，実際には非ゼロ要素だけがメモリに格納されて

いる．右辺ベクトルは，ここでは

```
>> b = ones(m^2,1)
```

として，要素がすべて 1 の列ベクトル，すなわち $b = (1, 1, \ldots, 1)^T$ を人工的に生成する．

今回は $m = 100$ として，$Ax = b$ をPCG法で解く．よって，行列 A のサイズは 10000×10000 であり，非ゼロ要素数は49,600個である．反復停止条件は，$\mathrm{tol} = 10^{-9}$，$\mathrm{maxit} = 200$ とする．以下の3つのタイプで実行する．

タイプ	備考
no precondition	前処理なし (CG法) `pcg(A,b,tol,maxit);`

`droptol='0'`	前処理付き:
	`K1 = cholinc(A,'0');`
	`pcg(A,b,tol,maxit,K1',K1);`

`droptol=1e-3`	前処理付き:
	`K2 = cholinc(A,1e-3);`
	`pcg(A,b,tol,maxit,K2',K2);`

結果を図 1 に示す．これは，図のX軸がPCG法の反復回数で，Y座標に反復毎の残差ノルム $\|b - Ax\|_2$ をプロットしている．

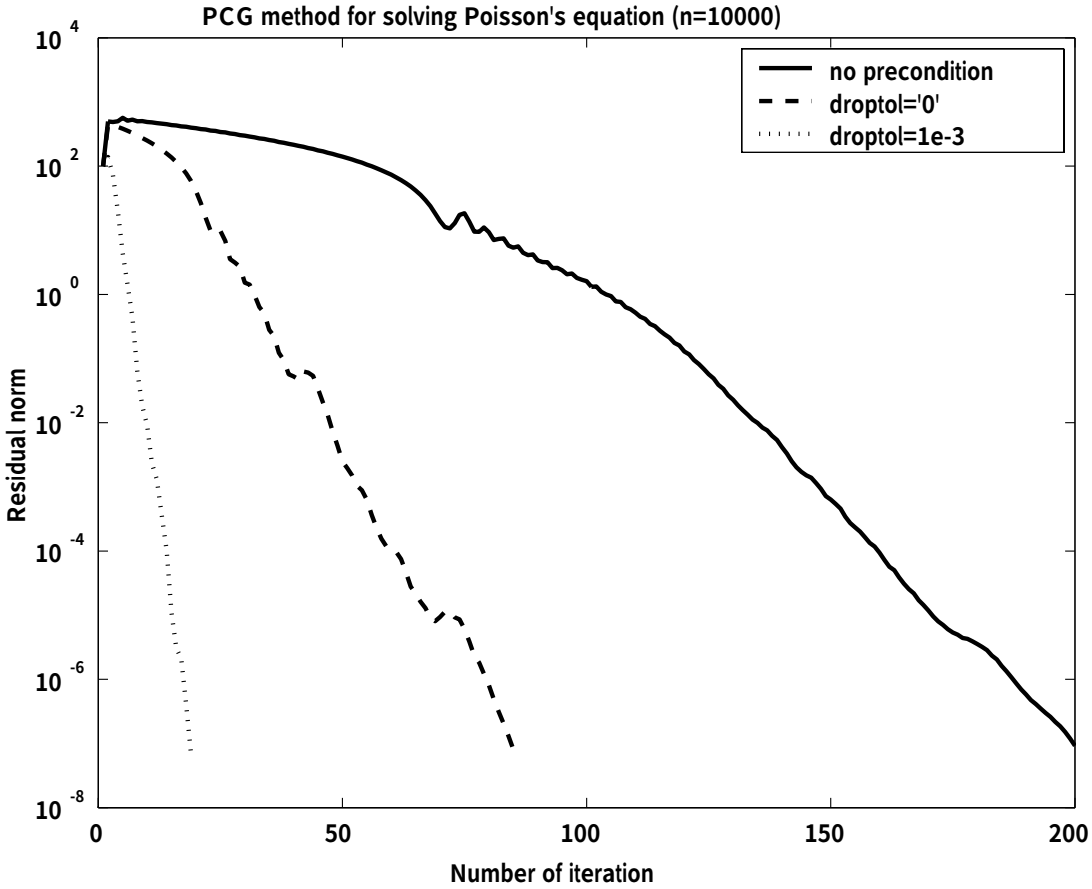

Figure 1: PCG法による残差ノルム $\|b - Ax\|_2$ の履歴

PCG 法によって問題がうまく解けていることがわかる．前処理によって劇的に収束が加速されている．但し，前処理行列の非ゼロ要素数を調べてみると (コマンドは nnz)，K_1 の非ゼロ要素数は 29,800 個で，K_2 の非ゼロ要素数は 149,672 個であった．droptol をさらに小さく設定すると，さらに収束が加速されるが，IC 分解過程での fill-in が多くなり，必要なメモリ量がさらに増加する．係数行列 A の非ゼロ要素数が 49,600 個であるから，droptol の値には注意が必要である．

Bi-CGSTAB法(安定化双共役勾配法)

Bi-CGSTAB法はランチョス原理に基づく反復解法で，係数行列 A が非対称行列の場合によく使われる解法である．Bi-CG法系統の解法は，後述のGMRES法系統の解法と違い，リスタートなどのパラメータ設定が基本的に不要であるため使い勝手が良く，ユーザに好まれる傾向がある．Matlabでは，関数bicgstabが用意されており，基本的な使い方は関数pcgと同じである．但し，係数行列が非対称行列であることから前処理の方法が若干異なるため，それを中心に説明する．ここでは，Bi-CGSTAB法の前処理にILU分解 (不完全LU分解; Incomplete LU factorization) を用いることにする．

まず，関数bicgstabの最も簡単な使い方は，A を $n \times n$ 非対称行列，b を n 次ベクトルとすると

```
>> x = bicgstab(A,b);
```

である．ILU分解を行う関数luincは，関数cholincと同様に

```
>> [L,U,P] = luinc(A,droptol);
```

のように使う．出力引数 L, U, P はそれぞれ，下三角行列，上三角行列，置換行列であり，$PA \approx LU$ であることを理想とする．droptolは，ILU分解過程におけるfill-inの基準となるしきい値であり，例えば，$\mathrm{droptol} = 10^{-3}$ にしたいときは，

```
>> droptol = 1e-3;
```

のようにする．この値を小さくすればするほど，完全なLU分解に近づくが，fill-inがその分増加して，必要なメモリが莫大になる．また，

```
>> droptol = '0';
```

のときは，fill-inは行わず，したがって，このとき L, U はそれぞれ PA の下三角部分あるいは上三角部分と同じスパースパターンを持つ．また，2つめの入力引数には構造体を使って

```
>> [L,U,P] = luinc(A,opts);
```

のように呼び出すことができる．２つめの入力引数optsは最大4つのフィールドを持つ構造体で，

opts.droptol: ILU分解過程におけるfill-inの基準となるしきい値

opts.milu: milu = 0: ILU分解 (デフォルト), milu = 1: MILU分解 (修正不完全LU分解; Modified Incomplete LU factorization)

opts.udiag: 特異な要素を避けるため, U の対角上のゼロを置き換える. rdiag = 0: 置き換えない (デフォルト), michol = 1: 置き換える

opts.thresh: ILU分解過程における軸交換の基準となる値

のうち, 興味のあるフィールドを選んで設定することができる. 具体的な使い方は, 前述の関数cholincの説明を参照されたい.

前処理を使用する場合は，関数pcgと同様に

```
>> x = bicgstab(A,b,tol,maxit,M);
```

あるいは

```
>> x = bicgstab(A,b,tol,maxit,M1,M2);
```

のようにする．但し，関数luincを前処理に使う場合は,

```
>> x = bicgstab(P*A,P*b,tol,maxit,L,U);
```

のように，あらかじめ置換行列 P を $Ax = b$ の両辺に掛けて

$$PAx = Pb$$

について解く形になる.

解の初期推定値の設定方法や，詳細な結果を出力する方法については，前述の関数pcgとまったく同様である.

非対称行列に対するBi-CGSTAB法の適用例

次に，非対称行列の例として，テスト行列にはランダム行列 (但し，スパースパターンはpoissonと同じ) を用いることにする．例えば，

```
>> m = 3; A = gallery('poisson',m);
>> randn('state',0); A = sprandn(A);
```

とする．関数randnは，ここでは乱数発生器の初期化に使っている．関数sprandnは，要素が正規分布乱数であるような行列を出力する．右辺ベクトルは，ここでは

```
>> b = ones(m^2,1)
```

として，要素がすべて 1 の列ベクトル，すなわち $b = (1, 1, \ldots, 1)^T$ を人工的に生成する．

今回は $m = 100$ として，$Ax = b$ をBi-CGSTAB法で解く．よって，行列 A のサイズは 10000×10000 であり，非ゼロ要素数は49,600個である．反復停止条件は，$\mathrm{tol} = 10^{-9}$，$\mathrm{maxit} = 200$ とする．以下の3つのタイプで実行する．

タイプ	備考
no prec.	前処理なし `bicgstab(A,b,tol,maxit)`

opts1	前処理付き:
	`opts1 = struct('droptol','0','udiag',1);`
	`[L1,U1,P1] = luinc(A,opts1);`
	`bicgstab(P1*A,P1*b,tol,maxit,L1,U1);`

opts2	前処理付き:
	`opts2 = struct('droptol',1e-3,'udiag',1);`
	`[L2,U2,P2] = luinc(A,opts2);`
	`bicgstab(P2*A,P2*b,tol,maxit,L2,U2);`

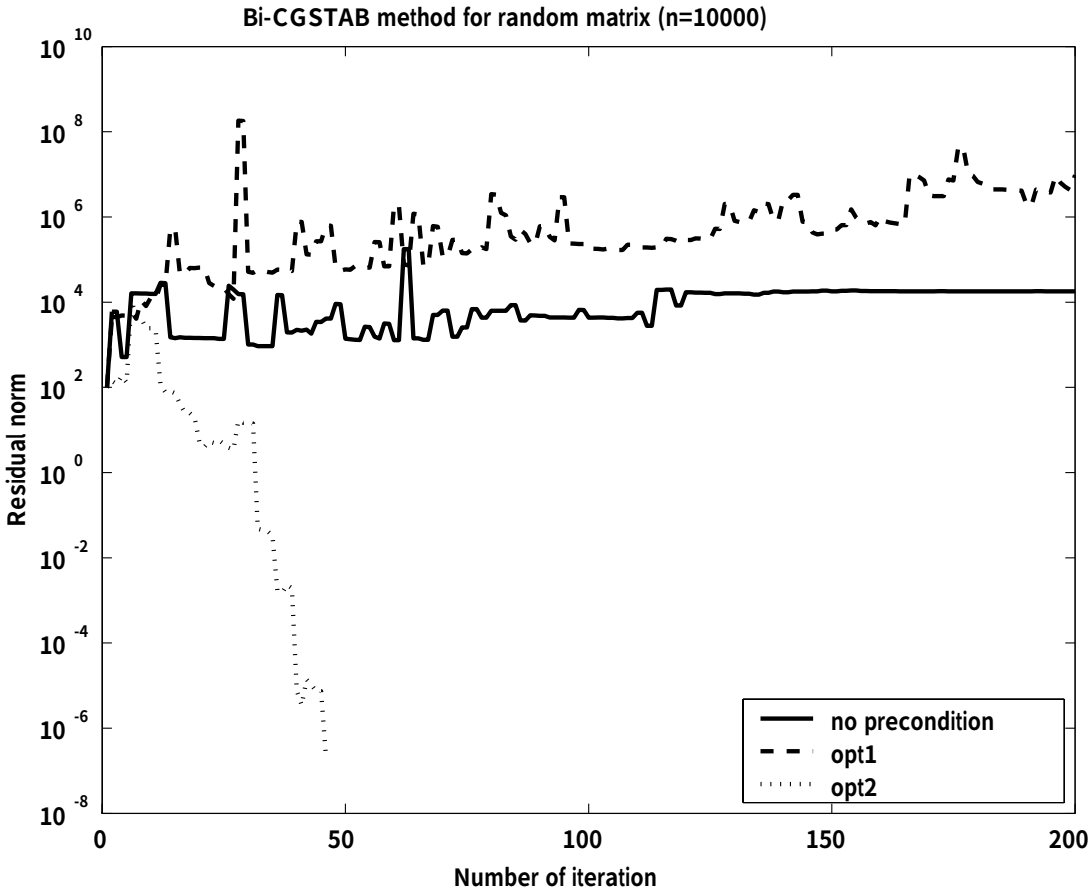

Figure 2: Bi-CGSTAB法による残差ノルム $\|b - Ax\|_2$ の履歴

この例では，前処理付きのタイプ opts2 でのみ収束している．このとき，L_2 の非ゼロ要素数は 384,762 個，U_2 の非ゼロ要素数は 350,540 個であり，係数行列 A の非ゼロ要素数と比べると，かなりのメモリ量が必要である．このように，反復解法では非常に解き辛い問題を簡単に作ることができてしまう．

GMRES法: 一般化最小残差法

GMRES法はアーノルディ原理に基づく反復解法で，Bi-CGSTAB法と同様に，係数行列 A が非対称行列の場合によく使われる解法である．オリジナルのGMRES法は計算量およびメモリ量の点で実用的でなく，適当な正整数 k に対して，k 回毎にリスタートすることによって必要な記憶容量を減らしたGMRES(k)法が実際には用いられる．Matlabでは，関数gmresが用意されており，基本的な使い方は関数bicgstabと同じである．但し，リスタートのためのパラメータ設定が必要である．

関数gmresの最も簡単な使い方は，A を $n \times n$ 非対称行列，b を

n 次ベクトルとすると

```
>> x = gmres(A,b);
```

である．これはリスタートを行わないGMRES法である．GMRES(k)法は，

```
>> x = gmres(A,b,k);
```

とすることによって実現され，反復 k 回毎にリスタートされる．この k の値は本来，大きいほどオリジナルのGMRESに近づき収束も速くなる傾向にあるが，その分，必要なメモリ量も増加する．これは，解きたい問題に応じて変える必要があるだろう．それ以外は，関数bicgstabと同様であり，例えば，ILU分解による前処理を

加えた場合は,

```
>> x = gmres(P*A,P*b,k,tol,maxit,L,U);
```

のように使う．但し，総反復回数は，(maxit$\times k$)回となる．

非対称行列に対するGMRES法の適用例

別の例として，五重対角 Toeplitz行列を用いる．例えば,

```
>> n = 7; gamma = 1.3;
>> A = gallery('toeppen',n,gamma,0,2,1,0)
```

$$A = \begin{bmatrix} 2 & 1 & 0 & & & & \\ 0 & 2 & 1 & 0 & & O & \\ \gamma & 0 & 2 & 1 & 0 & & \\ & \gamma & 0 & 2 & 1 & 0 & \\ & & \gamma & 0 & 2 & 1 & 0 \\ & O & & \gamma & 0 & 2 & 1 \\ & & & & \gamma & 0 & 2 \end{bmatrix}$$

が得られる (但し，この例では $\gamma = 1.3$)．この行列は，反復解法のテスト問題としてよく使われている．行列 A のサイズは，$n \times n$ である．このとき，A はsparse型であり，実際には非ゼロ要素だけがメモリに格納されている．右辺ベクトルは，ここでは

```
>> b = ones(n,1)
```

として，要素がすべて 1 の列ベクトル，すなわち $b = (1, 1, \ldots, 1)^T$ を人工的に生成する．

今回は $n = 10000, \gamma = 1.7$ として，$Ax = b$ をGMRES(k)法で解く．よって，行列 A のサイズは 10000×10000 であり，非ゼロ要素数は約30,000個である．反復停止条件は，$\mathrm{tol} = 10^{-9}$，$\mathrm{maxit} = 200$

とする．リスタートまでの反復回数 k による収束特性の違いを見るために，前処理は行わないことにする．

結果を図 3 に示す．図の見方は今までと同様である．

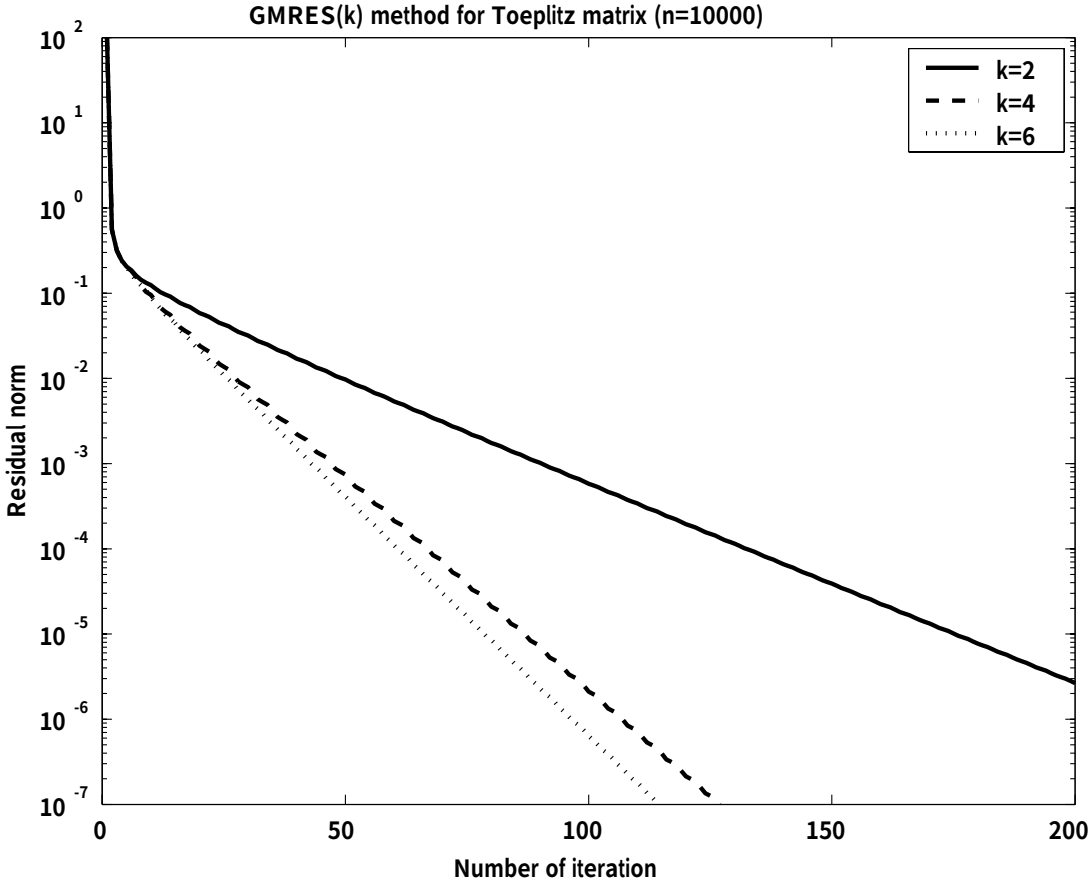

Figure 3: GMRES(k)法による残差ノルム $\|b - Ax\|_2$ の履歴

外部データをMatlabで使う方法

これまでの数値例では，Matlabであらかじめ用意されているテスト行列を使ったが，他のアプリケーションなどを使って作成した行列をファイルに保存しておけば，sparse型の行列としてMatlabに取り込むことができる．ファイルの取り込み方の例は，

```
>> load mymatrix.dat
```

で，このとき，「mymatrix.dat」が読み込まれる．このとき，ファイルのフォーマットは，以下のようにする．

「mymatrix.dat」の中身

```
% sample matrix data
% i, j, element
3  1   6.0
1  2  -2.0
4  3   0.0
2  4   7.0
```

```
>> A = spconvert(mymatrix);
```

とすることによって，sparse型の配列 A が作成される．試しに，ス

パース行列を密行列として出力する関数 full を使って配列 A の値を表示させてみると,

```
>> full(A)
ans =
     0    -2     0     0
     0     0     0     7
     6     0     0     0
     0     0     0     0
```

となり，ファイルからデータを正しく読み込めていることがわかる．また，複素行列を扱いたい場合は,

複素行列の場合の形式

```
% sample matrix data (complex case)
% i, j, real, img
3  1  6.0  1.0
1  2 -2.0  2.0
4  3  0.0  3.0
2  4  7.0  4.0
```

のようにする．ファイル内のデータが4列の場合は，自動的に3列目と4列目がそれぞれ複素数の実部と虚部として読み込まれる．

(Matlab demo)

大規模固有値問題の数値解法

固有値問題に関して，Matlabにおける数値解法を示す．Matlabにはスパース行列の固有値と固有ベクトルを求める関数eigsが用意されている．eigsは，内部でARPACK (Arnoldi Package) [4] という大規模固有値問題のための数値計算ライブラリを呼び出している．eigsの最も簡単な呼び出し方は，$n \times n$ 正方行列 A に対して

```
>> d = eigs(A);
```

で，このとき A の固有値の中の大きいものから6個をベクトルと

して出力する．これは，標準固有値問題

$$Ax = \lambda x$$

を部分的に解いている．また，

```
>> [V,D] = eigs(A);
```

とすると，固有値の中の大きいものから6個を要素に持つ対角行列 D と，それに対応する固有ベクトルを並べた行列 V を出力する．

(Matlab demo)

次に，B を A と同じサイズで対称(またはエルミート)な正定値行列とすると，

```
>> eigs(A,B);
```

は，一般固有値問題

$$Ax = \lambda Bx$$

を解く．出力に関しては，標準固有値問題の場合と同様である．

```
>> eigs(A,k);
```

と

```
>> eigs(A,B,k);
```

は，固有値の中の大きいものから k 個を出力する．さらに，

```
>> eigs(A,k,sigma);
```

と

```
>> eigs(A,B,k,sigma);
```

は，以下のルールで定まる sigma を基準に k 個 $(k \leq n)$ の固有値を出力する:

'LM' または 'SM': 絶対値が最大または最小の固有値から

A が対称行列またはエルミート行列の場合は

'LA' または 'SA': 最大または最小の固有値から
'BE': 最大・最小の固有値の両端から $\frac{k}{2}$ 個ずつ，k が奇数の場合は，大きい方を1個多く

それ以外の場合は

'LR' または 'SR': 実数部が最大または最小の固有値から
'LI' または 'SI': 虚数部が最大または最小の固有値から.

そして，sigmaがスカラの場合は，sigmaに最も近い固有値を基準とする．少なくとも，B は対称(またはエルミート)な半正定値行列でなければならない．他に，構造体optsを利用して，

```
>> eigs(A,k,sigma,opts);
```

あるいは

```
>> eigs(A,B,k,sigma,opts);
```

のように呼び出すことができる．詳しい情報については

```
>> help eigs
```

とするか，あるいはMatlabヘルプを参照されたい．

固有値問題の数値例

スパース行列系の固有値問題をMatlabで解く例を示す．テスト行列にはMatrix Marketにある固有値問題用の行列を使う．

Matrix Marketの行列データファイルには，

- MatrixMarket format (.mtx)
- Harwell-Boeing format (.rua)
- それらをMatlab形式にしたもの (.mat)

など，いくつかのデータ形式の種類があるので，それぞれの計算環境に応じて選択できる．

標準固有値問題の数値例

テスト行列として，mhd4800a.mtx
MHD4800A: Alfven Spectra in Magnetohydrodynamics
を係数行列 A にする．このとき，A はサイズが 4800×4800 の非対称な実行列で非ゼロ要素は102,252個である．A の固有値の内，大きいほうから10個を計算する．手順は，

```
>> load mhd4800a.mtx
>> A = spconvert(mhd4800a);
```

固有値を求めるときは，関数eigsを使って以下のようにした．

```
>> k = 10; d = eigs(A,k)
```

```
d =
   -1.652891460154018e+001 +5.426775225142485e+002i
   -1.652891460154018e+001 -5.426775225142485e+002i
   -6.130614229281726e+000 +2.004991269111328e+002i
   -6.130614229281726e+000 -2.004991269111328e+002i
   -3.822225777264265e+000 +1.257332759988957e+002i
   -3.822225777264265e+000 -1.257332759988957e+002i
   -2.783787261314251e+000 +9.173882176752058e+001i
   -2.783787261314251e+000 -9.173882176752058e+001i
   -2.189861591085109e+000 +7.226211695559431e+001i
   -2.189861591085109e+000 -7.226211695559431e+001i
```

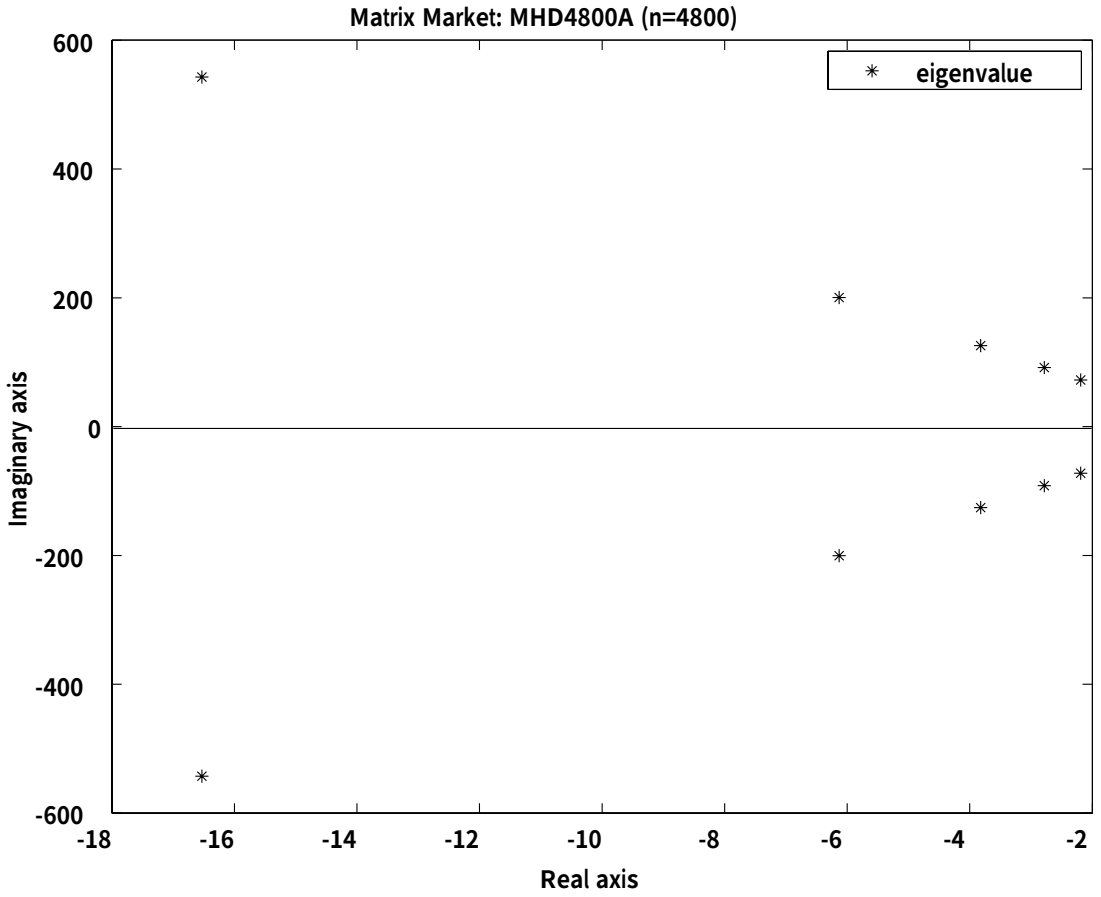

Figure 4: 大きいほうから10個の固有値

一般化固有値問題の数値例

テスト行列として，BCSSTRUC1: BCS Structural Engineering Matrices (eigenvalue matrices) セットの中のBCSSTK12 (ファイル名: bcsstk12.mat) を係数行列 A に，BCSSTM12 (ファイル名: bcsstm12.mat) を係数行列 B とする．このとき，A はサイズが 1473×1473 の実対称行列で非ゼロ要素は34,241個，B は A と同じサイズの実対称半正定値行列で非ゼロ要素は19,659個である．

$Ax = \lambda Bx$ の固有値の内，大きいほうと小さいほうの両方から4個ずつを計算する．

手順は,

```
>> load bcsstk12
>> A = Problem.A;
>> load bcsstm12
>> B = Problem.A;
```

固有値を求めるときは，関数eigsを使って

```
>> k = 4; sigma = 'LM'; dl = eigs(A,B,k,sigma)
>> k = 4; sigma = 'SM'; ds = eigs(A,B,k,sigma)
```

とした.

結果は以下のようになった.

```
dl =
    9.906714612101550e+008
    9.895278765762475e+008
    9.883785578422949e+008
    9.852508397202565e+008

ds =
    5.538521160256889e+003
    3.670949681098981e+003
    3.469352276996439e+003
    1.406547498093064e+003
```

References

[1] Barrett, R., et al.: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*, SIAM, Philadelphia, 1994.

[2] Freund, R. W., Nachtigal, N. M.: *QMR: A Quasi-Minimal Residual Method for Non-Hermitian Linear Systems*, Numer. Math. 60 (1991), 315–339.

[3] Higham, N. J.: *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.

[4] Lehoucq, R. B., Sorensen, D. C., Yang, C.: *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.
`http://www.caam.rice.edu/software/ARPACK/`

[5] Paige, C. C., Saunders, M. A.: *Solution of Sparse Indefinite Systems of Linear Equations*, SIAM J. Numer. Anal. 12 (1975), 617–629.

[6] ——: *LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares*, ACM TOMS 8:1 (1982), 43–71.

[7] —: *Algorithm 583: LSQR: Sparse Linear Equations and Least Squares Problems*, ACM TOMS 8:2 (1982), 195–209.

[8] Reichel, L., Trefethen, L. N.: *Eigenvalues and Pseudo-eigenvalues of Toeplitz Matrices*, Lin. Alg. Appl. 162-4 (1992), 153–185.

[9] Saad, Y.: *Iterative methods for sparse linear systems*, PWS Publishing Company, 1996.
`http://www-users.cs.umn.edu/~saad/books.html`

[10] Saad, Y., Schultz, M. H.: *GMRES: A Generalized Minimal*

Residual Algorithm for Solving Nonsymmetric Linear Systems, SIAM J. Sci. Stat. Comput. 7:3 (1986), 856–869.

[11] Sonneveld, P.: *CGS: A Fast Lanczos-type Solver for Nonsymmetric Linear Systems*, SIAM J. Sci. Stat. Comput. 10:1 (1989), 36–52.

[12] Van der Vorst, H. A.: *Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems*, SIAM J. Sci. Stat. Comput. 13:2 (1992), 631–644.

[13] Van Rienen, U.: *Numerical Methods in Computational*

Electrodynamics, Lecture Notes in Computational Science and Engineering; 12, Springer, 2001.

[14] 長谷川里美, 長谷川秀彦, 藤野清次 (訳): 反復法 Templates, 朝倉書店, 1996.

[15] 藤野清次, 張紹良: 反復法の数理, 朝倉書店, 1996.