2003 年度情報処理 II　第 10 回

数学のためのコンピューター (2) 線形計算

桂田 祐史 (かつらだ まさし)

2003 年 6 月 19 日

「情報処理 II トップページ」[1]

前回は非線形方程式を解くための方法として、二分法と Newton 法の紹介を行ったが、今回は線形計算 (連立一次方程式や固有値問題など) のための方法とあるソフトウェア (MATLAB[2] 互換の Octave) について概説する。

1　連絡事項

1.1　実験のための設定

今日の実験には、**Octave** というソフトを使う (これは MATLAB 互換システムである。MATLAB については、付録 B.2 を見よ。)。このコマンドは /usr/meiji/gnu/bin/octave にあり、/usr/meiji/gnu/lib にあるダイナミック・ライブラリィを必要とするので、少々設定をする必要がある。

Octave を使えるようにするための設定 (一度やるだけで OK)

```
isc-xas05% cp ~re00018/syori2.add .                    最後の . を忘れない
isc-xas05% cat syori2.add                              中身を確認
set path=($path /usr/meiji/gnu/bin)
setenv LD_LIBRARY_PATH (途中略) ..:/usr/meiji/gnu/lib
isc-xas05% cat syori2.add >> .cshrc                    .cshrc に追加。
                                                       >> を間違えないように。
isc-xas05% source .cshrc
```

[1] http://www.math.meiji.ac.jp/~mk/syori2-2005/
[2] http://www.mathworks.com/

2　最初の例

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}, \quad x = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

に対して、 $b = Ax$ を計算してから、$A^{-1}b$ を計算し (もちろん x と等しくなるはず)、A^{-1} を計算して、$A^{-1}A = I$ を確かめ、最後に A の固有値 λ_1, λ_2 と、$P^{-1}AP = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$ となる P を求める。

Octave のコマンド・メモ (1)

$A^{-1}b$ を計算する	`a\b`
A^{-1} を計算する	`inv(a)`
A の固有値を計算する	`eig(a)`
A の固有値と固有ベクトルを計算する	`[p lambda]=eig(a)`
	p は固有ベクトルを並べた行列
Octave の終了	`quit`

```
samba03% octave
GNU Octave, version 2.0.17 (sparc-sun-solaris2.8).
Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2002 John W. Eaton.
This is free software with ABSOLUTELY NO WARRANTY.
For details, type 'warranty'.

octave:1> a=[1,2;2,3]
a =

  1  2
  2  3

octave:2> x=[1;-1]
x =

   1
  -1

octave:3> b=a*x
b =

  -1
  -1

octave:4> a\b
ans =

   1
  -1

octave:5> inv(a)
ans =
```

```
  -3   2
   2  -1

ctave:6> inv(a)*a
ans =

  1  0
  0  1

octave:7> eig(a)
ans =

  -0.23607
   4.23607

octave:8> [p lambda]=eig(a)
p =

  -0.85065   0.52573
   0.52573   0.85065

lambda =

  -0.23607   0.00000
   0.00000   4.23607

octave:9> inv(p)*a*p
ans =

   -2.3607e-01   -3.4694e-16
   -4.4409e-16    4.2361e+00

octave:10> quit
Samba03%
```

最後の $P^{-1}AP$ の計算は、丸め誤差の影響で、完全な対角行列にはなっていない (が、誤差は 10^{-16} のオーダーで十分小さい)。

3　行列が疎でも逆行列が疎とは限らない

微分方程式の数値シミュレーションに現われる連立 1 次方程式の係数行列は、ほとんどの場合、成分に 0 が多い。そのような行列を疎 (そ) (sparse) であるという。疎行列はその性質をうまく利用することで、効率的な計算が可能になり、大規模な問題が解けるようになる。行列が疎であっても、その逆行列は疎であるとは限らない (大抵の場合、疎ではない) ことに注意する。

Octave のコマンド・メモ (2)

1:n	[1 2 3 $\cdots$ n]
a'	a の Hermite 共役 (実行列、実ベクトルの場合転置)
a.'	a の転置
eye(m,n)	単位行列
zeros(m,n)	零行列
ones(m,n)	成分がすべて 1 の行列
rand(m,n)	成分が乱数の行列
diag()	対角行列 (を少しずらした行列)
命令 1 ; 命令 2	一行に複数の命令を書ける。";" があると表示が抑制される。

以下の例では、

$$A = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

について、連立 1 次方程式を解いたり、逆行列を計算したりしている。

```
samba03% octave
GNU Octave, version 2.0.17 (sparc-sun-solaris2.8).
Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2002 John W. Eaton.
This is free software with ABSOLUTELY NO WARRANTY.
For details, type 'warranty'.

octave:1> n=4
n = 4
octave:2> 1:n
ans =

  1  2  3  4

octave:3> (1:n)'*(1:n)
ans =

   1   2   3   4
   2   4   6   8
   3   6   9  12
   4   8  12  16

octave:4> eye(n,n)
ans =

  1  0  0  0
  0  1  0  0
  0  0  1  0
  0  0  0  1

octave:5> zeros(n,n)
ans =
```

```
  0  0  0  0
  0  0  0  0
  0  0  0  0
  0  0  0  0

octave:6> ones(n-1,1)
ans =

  1
  1
  1

octave:7> diag(ones(n-1,1),1)
ans =

  0  1  0  0
  0  0  1  0
  0  0  0  1
  0  0  0  0

octave:8> diag(ones(n-1,1),-1)
ans =

  0  0  0  0
  1  0  0  0
  0  1  0  0
  0  0  1  0

octave:9> a=2*eye(n,n)-diag(ones(n-1,1),1)-diag(ones(n-1,1),-1)
a =

   2  -1   0   0
  -1   2  -1   0
   0  -1   2  -1
   0   0  -1   2

octave:10> x=(1:n)'
x =

  1
  2
  3
  4

octave:11> b=a*x
b =

  0
  0
  0
  5

octave:12> ia=inv(a)
ia =

  0.80000  0.60000  0.40000  0.20000
  0.60000  1.20000  0.80000  0.40000
  0.40000  0.80000  1.20000  0.60000
```

```
  0.20000  0.40000  0.60000  0.80000

octave:13> ia*a
ans =

   1.00000  -0.00000   0.00000   0.00000
   0.00000   1.00000   0.00000   0.00000
   0.00000   0.00000   1.00000   0.00000
   0.00000   0.00000   0.00000   1.00000

octave:14> n=8;a=2*eye(n,n)-diag(ones(n-1,1),1)-diag(ones(n-1,1),-1)
a =

   2  -1   0   0   0   0   0   0
  -1   2  -1   0   0   0   0   0
   0  -1   2  -1   0   0   0   0
   0   0  -1   2  -1   0   0   0
   0   0   0  -1   2  -1   0   0
   0   0   0   0  -1   2  -1   0
   0   0   0   0   0  -1   2  -1
   0   0   0   0   0   0  -1   2

octave:15> inv(a)
ans =

  0.88889  0.77778  0.66667  0.55556  0.44444  0.33333  0.22222  0.11111
  0.77778  1.55556  1.33333  1.11111  0.88889  0.66667  0.44444  0.22222
  0.66667  1.33333  2.00000  1.66667  1.33333  1.00000  0.66667  0.33333
  0.55556  1.11111  1.66667  2.22222  1.77778  1.33333  0.88889  0.44444
  0.44444  0.88889  1.33333  1.77778  2.22222  1.66667  1.11111  0.55556
  0.33333  0.66667  1.00000  1.33333  1.66667  2.00000  1.33333  0.66667
  0.22222  0.44444  0.66667  0.88889  1.11111  1.33333  1.55556  0.77778
  0.11111  0.22222  0.33333  0.44444  0.55556  0.66667  0.77778  0.88889

octave:16>
```

A が疎であっても、A^{-1} の成分には 1 つも 0 がないことを確認しよう。

任意の正則行列 A に対して、

$$PA = LU, \tag{1}$$

$$P = \text{置換行列}, \quad L = \text{下三角行列}, \quad U = \text{上三角行列} \tag{2}$$

を満たす P, L, U が存在する (付録 A.5 を見よ)。

$Ax = b$ であれば、$Pb = PAx = LUx$ であるから、

$$x = U^{-1}(L^{-1}(Pb))$$

となり[3]、L^{-1}, U^{-1} の掛け算は非常に簡単なので (付録 A.5.2 参照)、P, L, U が求まっていれば連立 1 次方程式は簡単に解ける。$(1), (2)$ を満たす P, L, U を求めることを A を LU 分解するという。

[3]ここで、$((U^{-1}L^{-1})P)b$ という順番にかけないのがミソである。$A^{-1} = (U^{-1}L^{-1})P$ を計算するのは計算の手間がかかる。特に A が疎行列の場合は、桁違いの差になるのが普通である。

Octave のコマンド・メモ (3)

[L u p]=lu(a)　a を LU 分解する。pa = Lu を満たす p, L, u を求める。
u\(L\(p*b))　ax = b を解く。

```
octave:16> [L u p]=lu(a)
L =

   1.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
  -0.50000   1.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
   0.00000  -0.66667   1.00000   0.00000   0.00000   0.00000   0.00000   0.00000
   0.00000   0.00000  -0.75000   1.00000   0.00000   0.00000   0.00000   0.00000
   0.00000   0.00000   0.00000  -0.80000   1.00000   0.00000   0.00000   0.00000
   0.00000   0.00000   0.00000   0.00000  -0.83333   1.00000   0.00000   0.00000
   0.00000   0.00000   0.00000   0.00000   0.00000  -0.85714   1.00000   0.00000
   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000  -0.87500   1.00000

u =

   2.00000  -1.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000
   0.00000   1.50000  -1.00000   0.00000   0.00000   0.00000   0.00000   0.00000
   0.00000   0.00000   1.33333  -1.00000   0.00000   0.00000   0.00000   0.00000
   0.00000   0.00000   0.00000   1.25000  -1.00000   0.00000   0.00000   0.00000
   0.00000   0.00000   0.00000   0.00000   1.20000  -1.00000   0.00000   0.00000
   0.00000   0.00000   0.00000   0.00000   0.00000   1.16667  -1.00000   0.00000
   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   1.14286  -1.00000
   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000   1.12500

p =
  1  0  0  0  0  0  0  0
  0  1  0  0  0  0  0  0
  0  0  1  0  0  0  0  0
  0  0  0  1  0  0  0  0
  0  0  0  0  1  0  0  0
  0  0  0  0  0  1  0  0
  0  0  0  0  0  0  1  0
  0  0  0  0  0  0  0  1

octave:17> x=(1:n)'
x =

  1
  2
  3
  4
  5
  6
  7
  8

octave:18> b=a*x
b =

  0
  0
  0
  0
```

```
  0
  0
  0
  9

octave:19> u\(L\(p*b))
ans =

  1.0000
  2.0000
  3.0000
  4.0000
  5.0000
  6.0000
  7.0000
  8.0000

octave:20> quit
samba03%
```

A を LU 分解したときの因子 P, L, U が確かに置換行列、下三角行列、上三角行列であることを確かめよう (このケースでは、P は単位行列である)。L, U が疎行列であることも確かめよう。

今回は使わなかったが、よく使われるコマンドを紹介しておく。

Octave のコマンド・メモ (4)

`x(i)`	x の第 i 成分
`a(i,j)`	a の第 (i, j) 成分
`det(a)`	a の行列式
`y'*x`	縦ベクトル x, y の内積
`x`	x のノルム (成分の絶対値の二乗の和の平方根)
`tril(a)`	a の下三角部分
`triu(a)`	a の上三角部分
`tic; コマンド; toc`	コマンドの実行時間を計測
`a(i,:)`	a の第 i 行ベクトル
`a(:,j)`	a の第 j 列ベクトル
`a(i1:i2,j1:j2)`	a の第 i1 ~ i2 行、第 j1 ~ j2 列の部分のブロック

余裕があれば試してみよう

1. 行列 a が大きいとき、`inv(a)`, `det(a)`, `a\` の実行時間を測って比較する。
2. `a\` の実行時間は、未知数の個数 n につれどう変化するか。

```
n=10
for i=1:8  % この 8 をむやみに大きくしないこと
  a=rand(n,n); b=rand(n,1);
  tic; a\b; toc
  n=n*2
end
```

A　線形計算とは

A.1　線形代数の現状

現在の日本の大学の理工系のカリキュラムでは、1, 2 年次に「線形代数」(linear algebra) を学ぶことになっている[4]。その主たるテーマは有限次元の線型空間とその間の線型写像の理論であるが[5]、ここでは連立 1 次方程式 $Ax = b$, 固有値問題[6] $Ax = \lambda x$ の解法に焦点を当ててみよう。

(念のために注意しておくと、連立 1 次方程式と固有値問題だけが重要なのではない。ともすると試験問題の花形なのでそう思ってしまう人もいると思うが...)

連立 1 次方程式の「解き方」としては、現在普通の線形代数の本では、

(1) 逆行列を用いる (逆行列については、行列式を使った公式があげられている)

(2) Cramer の公式を用いる

(3) 「掃き出し法」[7]を用いる

などが説明されているが、これらは、計算の効率 (なるべく少ない計算量で計算する) と数値的安定性に問題がある。特に、

連立 1 次方程式を解くために逆行列を求めてはいけない！

また固有値の求め方としては、線形代数の本では

命題 **A.1** n 次正方行列 A の固有値は、λ に関する代数方程式

$$\det(\lambda I - A) = 0$$

の根である。

という定理を述べて解決ということにしてある。この方法 (固有多項式に帰着する方法) では、実際にはごくごく小規模な問題しか解くことができない。

行列の固有値を求めるために固有方程式を解いてはいけない！

---

[4]実は「線形 (型) 代数」という名前はそれほど古くからあるものではない。かつては、「代数と幾何」、「行列と行列式」、「線形数学」というような名前の本があったが (ちなみに、筆者が学生の時の講義名は「代数と幾何」だった)、現在では「線形 (型) 代数」に収束したようである。個人的には収束と同時に「幾何」の匂いが薄れてきたように感じている (みんな「代数」だと信じるようになったのかな — 例えば 2 次, 3 次の行列式が、それぞれ平行四辺形の符号付き面積、平行六面体の符号付き体積を表わす (もちろん、これらはさらに一般の次元に拡張される) ということを書いていない本が存在するが、証明しなくても事実は教えておいた方が良いのにと思う)。解析屋としては、計量 (内積やノルム) の話があまり出て来ないことが残念である。

[5]ちなみに、解析的にこのテーマを取り扱った “Finite dimensional vector spaces” というタイトルの有名な面白い本がある。

[6]応用上は、一般化固有値問題という、行列 A, B が与えられたときに、$Ax = \lambda Bx$, $x \neq 0$ を満たす λ, x を求める問題も重要 (分野によっては、標準固有値問題なんて目にしないことも) である。

[7]数値線形代数では、**Jordan** (ヨルダン) の消去法と呼ぶ。

A.2　連立 1 次方程式の解法概説

コンピューターで行われる数値計算の時間の 90% は連立 1 次方程式を解いている時間だという説があるくらい、連立 1 次方程式は頻出する問題である。例えば微分方程式の問題も、離散化 (有限次元近似) された後は連立 1 次方程式を解くことに帰着されることが多い[8]。ところがこのように工学的な応用で、連立 1 次方程式を解くために実際に利用されている方法は (ほとんどの) 線形代数の本には載っていない。

実際に利用される解法は色々あるが、大きく次の二つに分類される (直接法以外知らない、見当もつかない人が多いと思われるが、それで構わない)。以下では直接法について、やや詳しく説明する。

直接法 (exact method) もし個々の演算に丸め誤差がなければ、有限回の演算で解が得られる方法

- Gauss の消去法 (LU 分解), 対称な問題に対する Cholesky 分解法, QR 分解に基づく方法などある。

反復法 (iterative method) 有限回の演算では真の解が得られないかもしれないが、反復するごとに近似解の精度を上げて行き、十分な精度が得られたところで打ち切る方法

- 古くからある定常反復法 (Jacobi 法, Gauss-Seidel 法, SOR 法など) と非定常反復法 (CG 法とその改良版, 親類) に分類される[9]。
- 計算のためには行列 A の成分そのものを知る必要はなく、任意に与えたベクトル x との積 Ax が分かれば良いという特徴を持ち、行列の疎性[10]を利用しやすい。

A.3　固有値問題の解法概説

(授業では説明する時間的余裕はないと思うが...)

固有値を求めるという問題は代数方程式に帰着されるわけだが、その逆に任意の代数方程式の問題は行列の固有値を求めるという問題に帰着できる (この事実はしばしば常微分方程式の本に書いてある)。

それゆえ、ある意味で固有値問題は代数方程式と等価であり、例えば「5 次以上の代数方程式の、四則とべき根を有限回用いた根の公式は存在しない」という有名な定理から、5 次以上の行列の固有値問題は、有限回の四則演算とべき乗演算では解けないことが分かる。それゆえ連立 1 次方程式の解法と対比してみると、(5 次以上の) 固有値問題には直接法は存在せず、反復法を使うしかないことが分かる。

もう一つ重要なのは、行列から固有多項式を作ると、もともとの行列が持っていた情報の多くが消えてしまうということである。例えば対称な固有値問題は、そうでない問題よりもうまく (速く安定に) 解けるのだが、それを固有多項式にしてしまうと、対称性のうま味がなくなってしまう。その他にも理由があって、結局は

---

[8]昨年書かれた文章によると、$n = 10^8$ (1 億) の問題が解かれているとか。

[9]今のところ、対称な問題に対しては、CG 法の系統の解法がほとんど決定版とみなされている。

[10]行列の成分に 0 が多いとき、行列は疎 (sparse) であるという。微分方程式の離散化で現われる連立 1 次方程式の係数行列は、ほぼ例外無く疎行列である。

固有値を求めるのに固有方程式を解こうとしてはいけない！

それでは実際にどうするか？現在では次の 2 ステップで攻略するのが良いと言われている。

(1) 与えられた行列 A を、実直交行列 U で相似変換して、Hessenberg 行列 (あるいは A が対称の場合は三重対角行列) $\widetilde{A}$ に変換する (つまり $\widetilde{A} = U^T A U$)。

(2) $\widetilde{A}$ に対して、

  (a) 冪乗法の系統の方法 (逆反復法, シフト法, 減次)

  (b) Strum の方法 (bisection method)

  (c) QR 法

  などの反復法を施して、固有値を求める。

$A = (a_{ij})$ が **Hessenberg** 行列とは、

$$i > j + 1 \quad \Longrightarrow \quad a_{ij} = 0$$

が成り立つこと (対角線より二つ以上下は 0) を言う。大ざっぱに言うと、後一歩で上三角行列になる行列ということである。

A.4 Gauss の消去法

Gauss の消去法というのは、要するに中学校で習った (未知数を一つ一つ消去して減らしていく) 消去法である。

例として次の方程式を取りあげて説明しよう。

$$\left\{ \begin{array}{rcrcrl} 2x_1 & + & 3x_2 & - & x_3 & = 5 \\ 4x_1 & + & 4x_2 & - & 3x_3 & = 3 \\ -2x_1 & + & 3x_2 & - & x_3 & = 1. \end{array} \right. \tag{3}$$

線形代数で習う掃き出し法では、係数行列と右辺のベクトルを並べた行列を作り、それに

(1) ある行に 0 でない定数をかける。

(2) 二つの行を入れ換える。

(3) ある行に別の行を加える。

のような操作 — 行に関する基本変形と呼ぶ — をほどこして、連立方程式の係数行列に相当する部分を単位行列にするのであった。

$$\begin{pmatrix} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ -2 & 3 & -1 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & \frac{3}{2} & -\frac{1}{2} & \frac{5}{2} \\ 4 & 4 & -3 & 3 \\ -2 & 3 & -1 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & \frac{3}{2} & -\frac{1}{2} & \frac{5}{2} \\ 0 & -2 & -1 & -7 \\ 0 & 6 & -2 & 6 \end{pmatrix} \to$$

$$\to \begin{pmatrix} 1 & \frac{3}{2} & -\frac{1}{2} & \frac{5}{2} \\ 0 & 1 & \frac{1}{2} & \frac{7}{2} \\ 0 & 3 & -1 & 3 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & -\frac{5}{4} & -\frac{11}{4} \\ 0 & 1 & \frac{1}{2} & \frac{7}{2} \\ 0 & 0 & -\frac{5}{2} & -\frac{15}{2} \end{pmatrix} \to \begin{pmatrix} 1 & 0 & -\frac{5}{4} & -\frac{11}{4} \\ 0 & 1 & \frac{1}{2} & \frac{7}{2} \\ 0 & 0 & 1 & 3 \end{pmatrix} \to$$

$$\to \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{pmatrix}, \quad \text{ゆえに} \begin{pmatrix} x_2 \\ x_3 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

ガウスの消去法も、前半の段階はこの方法に似ていて、同様の変形を用いて掃き出しを行なうのだが、以下のように対角線の下側だけを 0 にする。

$$\begin{pmatrix} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ -2 & 3 & -1 & 1 \end{pmatrix} \to \begin{pmatrix} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & 6 & -2 & 6 \end{pmatrix} \to \begin{pmatrix} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & 0 & -5 & -15 \end{pmatrix}.$$

最後の行列は

$$2x_1 + 3x_2 - x_3 = 5, \quad -2x_2 - x_3 = -7, \quad -5x_3 = -15$$

ということを表しているので、後の方から順に

$$x_3 = \frac{-15}{-5} = 3, \quad x_2 = \frac{-7 + x_3}{-2} = 2, \quad x_1 = \frac{5 - 3x_2 + x_3}{2} = \frac{5 - 3 \times 2 + 3}{2} = 1$$

と解くことが出来る。前半の対角線の下側を 0 にする掃き出しの操作を前進消去 (forward elimination)、後半の代入により解の値を求める操作を後退代入 (backward substitution) と呼ぶ。

A.5 行列の分解について

A.2 で行列の色々な「分解」が出て来た。

A.5.1 LU 分解

最初に二つの言葉を定義する。

行列 $L = (\ell_{ij})$ が下三角行列 (lower triangluar matrix) であるとは、対角線の上にある成分がすべて 0 である、つまり

$$i > j \quad \Longrightarrow \quad \ell_{ij} = 0$$

が成り立つことと定義する。

同様に、$U = (u_{ij})$ が上三角行列 (upper triangular matrix) であるとは、対角線の下にある成分がすべて 0 である、つまり

$$i < j \quad \Longrightarrow \quad u_{ij} = 0$$

が成り立つことと定義する。

目で見えるように書くと

$$L = \begin{pmatrix} \ell_{11} & 0 & \cdots & \cdots & 0 \\ \ell_{21} & \ell_{22} & 0 & \cdots & 0 \\ \vdots & & \ddots & & \\ \ell_{n1} & \cdots & \cdots & \ell_{nn} & \end{pmatrix}, \quad U = \begin{pmatrix} u_{11} & & & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & 0 & \ddots & \\ 0 & 0 & & u_{nn} \end{pmatrix}$$

ということである。

LU 分解

行列 A に対して、

$$A = LU$$

を満たす下三角行列 L と上三角行列 U を求めることを (これはいつもできるとは限らない — 後述)、A を **LU 分解**すると呼ぶ。

Gauss の消去法の前進消去段階では、係数行列に行に関する基本変形を施して上三角行列 (それを U とおく) に変形したが、行に関する基本変形は、基本行列を左からかけることで表現できる。Gauss の消去法では、これら基本行列はすべて正則な下三角行列である。そこで、この変形は

$$L_k \cdots L_2 L_1 A = U \quad (L_j \text{ は正則な下三角行列}, U \text{ は上三角行列})$$

とかける。これから

$$A = {L_1}^{-1} {L_2}^{-1} \cdots {L_k}^{-1} U$$

となるが、$L := {L_1}^{-1} {L_2}^{-1} \cdots {L_k}^{-1}$ は実は下三角行列である。これは次の補題から分かる。

補題 A.1 (1) 下三角行列全体は和と積に関して閉じている。

(2) 正則な下三角行列の逆行列は下三角行列である (ゆえに正則な下三角行列全体は乗法について群をなす)。

さて、それでは Gauss の消去法はいつでも出来るかというと、(掃き出し法との類推ですぐ分かるように) 対角成分に 0 が現われたらダメになるわけである。この場合は、行を適当に交換することで消去が出来るようになる。

命題 A.2 A を n 次正則行列とする。

(1) A が LU 分解できるための必要十分条件は、A のすべての主座小行列式が 0 でないことである。特に任意の正値対称行列は LU 分解可能である。

(2) 適当な置換行列 P が存在して、PA は LU 分解できる。すなわち適当な下三角行列 L, 上三角行列 U が存在して

$$PA = LU$$

が成り立つ。

正則行列 A の LU 分解はたとえ存在しても一意ではないが、A を

$A = LDU$　(L は対角成分が 1 の下三角行列、U は対角成分が 1 の上三角行列、D は対角行列)

の形に分解する **LDU** 分解は一意である。

A.5.2　三角行列係数の連立 1 次方程式

三角行列係数の連立 1 次方程式は非常に簡単に解くことができる。これを上三角行列の場合に見てみよう。

$$Ux = b$$

において $U = (u_{ij})$ が上三角行列とする。

$$\begin{array}{ccccccc} u_{11}x_1 & + & u_{12}x_2 & + & \cdots & + & u_{1n}x_n = b_1 \\ & & u_{22}x_2 & + & \cdots & + & u_{2n}x_n = b_2 \\ & & & & \ddots & & \vdots \\ & & & & & & u_{nn}x_n = b_n \end{array}$$

は下の方程式から順に

$$\begin{aligned} x_n &= b_n/u_{nn}, \\ x_{n-1} &= (b_{n-1} - u_{n-1,n}x_n)/u_{n-1,n-1}, \\ &\vdots \\ x_i &= \left(b_i - \sum_{j=i+1}^{n} u_{i,j}x_j\right)/u_{ii} \\ &\vdots \\ x_1 &= \left(b_1 - \sum_{j=2}^{n} u_{1,j}x_j\right)/u_{11} \end{aligned}$$

と解ける。要するにこれは Gauss の消去法の後退代入過程と同じである。この計算は数値的安定性に関しても申し分のないものになっている。

行列 A の LU 分解 $A = LU$ があるとき、$Ax = b$ の解は

$$x = U^{-1}(L^{-1}b)$$

と書けるので、三角行列係数の方程式

$$Ly = b,$$

$$Ux = y$$

を順に解けば得られることが分かる。よって、これは非常に簡単に計算できる。

A.5.3　Cholesky 分解

A が正値対称行列である場合、

$$A = LL^T$$

を満たす下三角行列 L が存在する。これを **Cholesky** 分解と呼ぶ。L の対角成分は正であるように取ることができるが、そういうものに限ると分解は一意的である。

L^T は上三角行列であるから、Cholesky 分解は一種の LU 分解である。

Cholesky 分解は、通常の LU 分解の半分程度の計算量で計算できる (具体的な計算法は省略)。

A.5.4　QR 分解

A を実正則行列とするとき、実直交行列 Q と、上三角行列 R で

$$A = QR$$

を満たすものが存在する。特に R の対角成分は正であるように取ることができ、そういうものに限ると分解は一意的である。これを A の **QR** 分解と呼ぶ[11]。

$A = (a_1\ a_2 \cdots a_n)$ とするとき、$a_1, \cdots, a_n$ から、Gram-Schmidt の直交化を行って正規直交基底 $q_1, \cdots, q_n$ を作る計算は、A の QR 分解を求めていることになる。

しかし QR 分解を求める場合、この素朴な Gram-Schmidt の直交化法を適用することはない[12]。

LU 分解と同様に QR 分解があれば連立 1 次方程式は簡単に解ける。例えば $Ax = b$ を解きたいときに、$A = QR$ という QR 分解が得られたとしよう。

$$x = R^{-1}(Q^{-1}b) = R^{-1}(Q^Tb)$$

であるから、x の計算は簡単である (つまり、実直交行列の逆行列はもとの行列の転置行列に他ならないから、計算するまでもなく分かっているわけ)。

A.5.5　連立 1 次方程式に対する直接法についてのまとめ

色々なことを書いたが、要するに、

与えられた行列 A を逆行列の計算の簡単な行列の積に分解 (**factorize**) することが要点

A.6　なぜ逆行列を計算してはいけないか

- 逆行列の計算には、Gauss の消去法に基づく LU 分解の計算よりも多くの計算量 (約 3 倍) が必要である。

---

[11] この分解について言及してある線形代数の教科書は結構ある。Schmidt 分解とか Gram-Schmidt 分解と呼ばれることが多いが、数値線形代数の世界ではもっぱら QR 分解と呼ばれる。

[12] 修正 Gram-Schmidt 法や、基本的な直交変換 (例えば超平面に関する対称移動を表わす Householder 行列や、二次元平面の回転を表わす Givens 行列) を次々にかけていく方法が使われる。

- 元の係数行列 A が疎である場合、逆行列は (ほとんどすべての場合に) 疎ではないが、LU 分解したときの因子 L, U は疎性を保っている (実例を見せる)。そのため、係数行列が疎である場合には、計算量に大差 (次数 n のベキが異なることもあるので、「桁違い」の差になるのが普通) が生じる。

A.7 固有値問題の解法を理解するための緒命題

与えられた行列をまず直交行列による相似変換で「簡単な形」に変形していく。もしも上三角行列に出来ればめでたしめでたしであるが...

命題 **A.3** (1) A を正則行列 P で相似変換した $\widetilde{A}$ (つまり $P^{-1}AP$) は、A と同じ固有値を持つ。

(2) 実直交行列 U の逆行列は U^T であるから、U による相似変換は $\widetilde{A} = U^T AU$ であるが、A が実対称である場合は $\widetilde{A}$ も実対称である。

(3) unitary 行列 U の逆行列は $U^* = \overline{U}^T$ であるから、U による相似変換は $\widetilde{A} = U^* AU$ であるが、A が Hermite 行列である場合は $\widetilde{A}$ も Hermite 行列である。

(4) 実直交行列の積は実直交行列であり、実直交行列による相似変換を有限回繰り返したものは、一つの実直交行列による相似変換に等しい。unitary 行列についても同様のことが成り立つ。

(5) 任意の正方行列 A に対して適当な unitary 行列 U が存在して、

$$U^* AU = R \quad (R \text{ は上三角行列})$$

となる (**Schur** 分解)。上三角行列の固有値はその対角成分に他ならないから、この分解が得られれば A の固有値が求められたことになる。

既に述べたように、固有値問題は代数方程式と等価であるから、有限回の四則演算と冪乗演算で Schur 分解を求めることはできない。しかし、上三角にすることをあきらめ、一歩手前の Hessenberg 行列で我慢することにすると、比較的計算量の少ない計算で済ませることが可能である。その詳細は省略するが、次のような基本的な直交行列による変換を繰り返すことで実現される。

(1) 超平面 $(u, x) = u_1 x_1 + u_2 x_2 + \cdots + u_n x_n = 0$ ($u = (u_i)$ は単位ベクトル) に関する対称移動 (鏡映、鏡像とも言われる) を表わす $H = I - 2u^T u$ (**Householder** 行列と呼ばれる)。

(2) $x_p x_q$ 平面内の角 $-\theta$ の回転を表わす $G = (g_{ij})$.

$$g_{ij} = \begin{cases} \cos\theta & ((i,j) = (p,p), (q,q)) \\ \sin\theta & ((i,j) = (p,q)) \\ -\sin\theta & ((i,j) = (q,p)) \\ 1 & (i = j \notin \{p,q\}) \\ 0 & (\text{それ以外}) \end{cases}$$

(**Givens** の回転行列と呼ばれる。)

B　線形計算ソフトウェアの発展 (駆け足説明)

B.1　線形計算ライブラリィの誕生と発達

1. サブルーチン[13](**subroutine**) の誕生、サブルーチン・ライブラリィの誕生
2. (汎用) プログラミング言語[14]の誕生 (**FORTRAN**[15], **LISP** などが最初の例)
3. 固有値計算ライブラリィ **EISPACK**
   (論文誌 "Numerische Mathematic" で発表されたアルゴリズムを元に最初は **ALGOL** で書かれ、後に FORTAN に移植される。主宰者は有名な数値解析学者である Wilkinson である。)
4. 連立 1 次方程式の解法ライブラリィ **LINPACK**
   途中から **BLAS** が生まれ、LINPACK は BLAS の上に構築される。
5. 線形計算ライブラリィ **LAPACK**
   (メモリー階層を考慮した BLAS を全面的に採用、EISPACK & LINPACK の現代化)
6. 他のプログラミング言語への移植 — **TNT**[16] (C++) など。

ここで名を紹介した EISPACK, LINPACK, LAPACK, TNT はいずれもソースが公開されているフリーソフトである[17]。

数値計算ライブラリィの採用で実現できること

(1) 高い生産性

(2) 高い信頼性 (バグが少ない、高精度、条件が悪い問題でも崩れないタフさ)

(3) 高い効率性 (速度、メモリー利用効率)

[13]機械語 (machine language) や、Fortran 言語における、あるまとまった処理をするプログラムの単位を呼ぶ言葉。C 言語における「関数」に相当すると考えて構わない。

[14]当時は「自動プログラミング言語」と呼ばれたそうである。

[15]FORTRAN は、IBM が線形計画法のプログラムを効率的に作成するために開発した言語であると言われている。

[16]C++ 向けには **LAPACK++** があったが、C++ 言語の ANSI 規格の進展に伴い、新しく設計し直されたのが TNT である。まだ発展途上で、LAPACK の機能のすべては実装されていない。

[17]このあたりに欧米文化の強さが感じられる。ここで紹介したソフトの中には、博士号クラスの研究者が数十人、何年も作業して始めて開発できたものもある。日本でも大学を中心に様々なライブラリィの開発がされたが、全面公開までこぎつけたものは少なく (途中で企業に売ってしまったものもある)、大変もったいない事態になっていると筆者は感じている。こうなってしまった背景には、ソフトウェアの開発を研究業績とは認めない風潮など、二三の理由が考えられる... (脱線)

なぜ高い実行効率が得られるか

速度をあげるために考えねばならないこととして、講義では計算量を説明した。これらのソフトウェアでは計算量の観点から無駄がないことはもちろんであるが、それ以外にも重要な要素がある。

1. ループのアンロールなどのテクニック
2. メモリー階層を考慮したプログラミング

これらを追求すると、利用するコンピューター・システムに合わせたプログラムのチューニングが必要になり、プログラムの汎用性が低くなる恐れがある。しかし、システムごとにチューニングが必要な部分を小さな部分に凝縮し、他と分離することにより、この問題をある程度解決できる。LAPACK については、BLAS がこのチューニングが必要な部分を担当している。LAPACK に付属する BLAS は “reference (参考) BLAS” と呼ばれ、FORTRAN で書かれているので移植性があるが、高速な計算を望む場合は最適化された BLAS に差し換えることになる。例えば Sun Workstation の場合、Sun Microsystem 製ソフトウェア開発環境 Sun Workshop では、コンパイラーと一緒に BLAS が (と実は LAPACK も) 提供されている)。Windows や Intel CPU 向けの Linux の場合は Intel から BLAS が提供されている (無償)。これらは人手で最適化されたものである (と思われる) が、色々なパラメーターを変化させながら性能を測定することで、最適なパラメーターを実験的に「算出」し、その結果を元に BLAS プログラムを自動生成するソフトウェアもある (例えばフリーソフトの **ATLAS**)。

B.2 MATLAB とその互換システムの登場

MATLAB は EISPACK の開発にも関わった C.Moler が作成したシステムで、彼が創立した MathWorks 社から販売されている。

MATLAB の特徴

- インタープリター型言語である。そのため[a]、
  - 対話的で使いやすいシステムになっている。
  - 注意深く利用しないと実行効率が低くなる[b](個々の命令の実行時に命令解釈のコストが必要なため、繰り返し処理を多用すると計算時間が長くなりがちである)。
- LAPACK などの各種数値計算ライブラリィを内蔵している (これらのライブラリィ群へのインターフェイスであると理解すべきかもしれない)。
- ベクトル、行列などのデータの型が始めから定義されているので、命令が簡潔になっていて、プログラミングも楽になった[c]。

[a] ここで指摘することは、例えば Mathematica, Maple, REDUCE のような数式処理系にも当てはまる。

[b] ここで述べたような注意は、かつてはパソコン上で BASIC 言語を使ってプログラムを開発する際の常識であったのだが、今ではあまり知られていないことなのだろう。

[c] オブジェクト指向であり、データ構造が隠蔽されていると言って良いかもしれない。LAPACK などの利用で面倒な点の一つに、プログラマーにライブラリィ中で定義されたデータ構造を正しくなぞったプログラムを書く努力が要求されるというものがあるが、MATLAB ではこれがなくなっている。この点は C++ で書かれたライブラリィでも期待できることであるが。

MATLAB をいかに評価するかであるが、筆者は最近

案外大したものではないか、ひょっとするとコロンブスの卵で大発明？

と考えるようになった。このようなシステムを作るのは実は簡単で (実際、以下紹介するように「真似」がたくさん出て来た)、しかし使ってみると分るが、非常に便利である。日本の数学界ではあまり人気がない (というか知られていない) ようであるが、工学の世界では浸透している。

MATLAB は現在も改良が続けられていて、行列計算関係では、疎行列向けの処理や反復法なども採り入れられている。簡単な偏微分方程式のシミュレーションへの応用も十分可能なレベルに成長した。

MATLAB を後を追ったシステムがたくさん開発されたが、MATLAB の言語仕様は「準標準」となっている。MATLAB と似たシステムとして、Octave を紹介する。これは MATLAB との互換性が高く、残念ながら疎行列専用の処理が用意されていないが、入門には十分であるし、用途を選べば実用性も高い。

C

ここでは、実例で Octave の使い方を紹介する。

マニュアル: `http://www.math.meiji.ac.jp/~mk/labo/howto/octave.pdf`

また自分用のメモを

`http://www.math.meiji.ac.jp/~mk/labo/text/private-matlab-notebook/` 『MATLAB 手習い』

を公開する。

Octave, Scilab については、WWW 上に解説文書があふれているが、日本語で読める書籍としては 大石 [3] がある。

参考文献

[1] 有木進, 工学のための線形代数, 日本評論社 (200?).

[2] 伊理正夫, 線形代数 I, 岩波講座 応用数学, 岩波書店 (1993).

[3] 大石進一, Linux 数値計算ツール, コロナ社 (2000).

[4] 大石進一, MATLAB による数値計算, 培風館 (2001).

[5] 小国力/Dongarra, Jack J., MATLAB による線形計算ソフトウェア, 丸善 (1998).

[6] 小国力, MATLAB と利用の実際 — 現代の応用数学と CG —, サイエンス社 (1995).

[7] 菊地 文雄, 山本 昌宏, 微分方程式と計算機演習, 山海堂 (1991).