第１章　線形代数–逆行列 (LAMatrixInverse)

1.1　行列計算の概要

数値計算の中心課題の一つである，行列に関する演算について見ていく．多次元，大規模な行列に対する効率のよい計算法が多数開発されており，多くの既存のライブラリが用意されている．本章ではそれらの中心をなす，逆行列 (matrix inverse) と固有値 (Eigen values) に関して具体的な計算方法を示す．現実的な問題には既存のライブラリを使うのが上策であるが，それでも基礎となる原理の理解や，ちょっとした計算，ライブラリの結果の検証に使えるルーチンを示す．

逆行列は連立一次方程式を解くことと等価である．ルーチン的なやり方にガウスの消去法がある．これは上三角行列になれば代入を適宜おこなうことで解が容易に求まることを利用する．さらに，初期値から始めて次々に解に近づけていく反復法がある．この代表例である Jacobi(ヤコビ) 法と，収束性を高めた Gauss-Seidel(ガウス- ザイデル) 法を紹介する．

上記の手法をより高速にした修正コレスキー分解と共役傾斜 (共役勾配) 法があるが，少し複雑になるので割愛する．必要ならば NumRecipe を読め．

1.2　ガウス消去法による連立一次方程式の解

逆行列は連立一次方程式を解くことと等価である．すなわち，A を行列，x を未知数ベクトル，b を数値ベクトルとすると，

$$\begin{aligned} Ax &= b \\ A^{-1}x &= A^{-1}b \\ x &= A^{-1}b \end{aligned}$$

である．未知数の少ない連立一次方程式では，適当に組み合わせて未知数を消していけばいいが，未知数が多くなってしまうと破綻する．未知数の多い多元連立一次方程式で，ルーチン的に解を求めていく方法がガウス消去法で，前進消去と後退代入という 2 つの操作からなる．

後退代入 (Backward substitution) による解の求め方を先ず見よう．たとえば，

$$\begin{aligned} x + y - 2z &= -4 \\ -3y + 3z &= 9 \\ -z &= -2 \end{aligned}$$

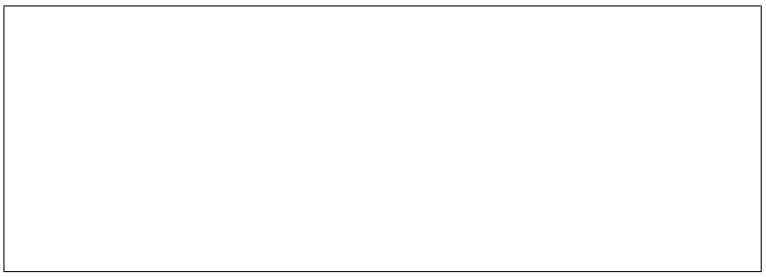
では，下から順番に $z \to y \to x$ と適宜代入することによって，簡単に解を求めることが出来る．係数で作る行列でこのような形をした上三角行列にする操作を前進消去あるいはガウスの消去法 (Gaussian elimination) という．下三角行列 L(lower triangular matrix) と上三角行列 U(upper triangular matrix) の積に分解する操作

$$A = L.U$$

を LU 分解 (LU decomposition) という．例えば先に示した上三角行列を係数とする連立方程式は，

$$\begin{aligned} x + y - 2z &= -4 \\ x - 2y + z &= 5 \\ 2x - 2y - z &= 2 \end{aligned}$$

を変形することで得られる．この変形を示せ．



1.3 Maple による LU 分解

係数行列 (coefficient matrix) と定数項 (b) との関係は以下の通りである．

```
> restart;
  A:=Matrix([[1,1,-2],[1,-2,1],[2,-2,-1]]):
  X:=Vector([x,y,z]):
  #X:=Vector([1,-1,2]):
  b:=Vector([-4,5,2]):
  A.X=b;
```

$$\begin{bmatrix} x+y-2z \\ x-2y+z \\ 2x-2y-z \end{bmatrix} = \begin{bmatrix} -4 \\ 5 \\ 2 \end{bmatrix}$$

単に逆行列を求める際は

```
> with(LinearAlgebra):
  MatrixInverse(A);
```

$$\begin{bmatrix} 4/3 & 5/3 & -1 \\ 1 & 1 & -1 \\ 2/3 & 4/3 & -1 \end{bmatrix}$$

である．Maple では行列を三角行列に分解するために，LUDecomposition コマンドが用意されている．

```
> P,L,U:=LUDecomposition(<A|b>);
```

$$P, L, U := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 2 & 4/3 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & -2 & -4 \\ 0 & -3 & 3 & 9 \\ 0 & 0 & -1 & -2 \end{bmatrix}$$

係数と定数項から作られる行列を拡大係数行列 (augmented matrix) といい，¡A—b¿とすると作られる．後退代入までおこなってもとめた連立方程式の解は以下の通り．

```
> LUDecomposition(<A|b>,output='R');
```

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

1.4 LU 分解のコード

LU 分解すれば線形方程式の解が容易に求まることは理解できると思う．具体的に A を LU 分解する行列 (消去行列と称す)T1,T2 の係数は次のようにして求められる．

```
> A0:=Matrix([[1,1,-2],[1,-2,1],[2,-2,-1]]):
  b0:=Vector([-4,5,2]):
  A:=Matrix(A0): B:=Vector(b0): n:=3:
  L:=Matrix(array(1..n,1..n,identity)):
  for i from 1 to n do #i 行目
    T[i]:=Matrix(array(1..n,1..n,identity)):
                              #i 番目の消去行列を作る
    for j from i+1 to n do
      am:=A[j,i]/A[i,i];      #i 行の要素を使って，i+1 行目の先頭を消す係数を求める
      T[i][j,i]:=-am;         #i 番目の消去行列に要素を入れる
      L[j,i]:=am;             #LTM の要素
      for k from 1 to n do
        A[j,k]:=A[j,k]-am*A[i,k]; #もとの行列を UTM にしていく
      end do;
      B[j]:=B[j]-B[i]*am;     #数値ベクトルも操作
    end do;
  end do:
```

上のコードによって得られた消去行列.

```
> T[1]; T[2];
```

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -4/3 & 1 \end{bmatrix}$$

これを実際に元の行列 $A0$ に作用させると，UTM が求められる．

```
> U:=T[2].T[1].A0;
```

$$U := \begin{bmatrix} 1 & 1 & -2 \\ 0 & -3 & 3 \\ 0 & 0 & -1 \end{bmatrix}$$

求められた LTM, UTM を掛けると

```
> L.U;
```

$$\begin{bmatrix} 1 & 1 & -2 \\ 1 & -2 & 1 \\ 2 & -2 & -1 \end{bmatrix}$$

元の行列を得られる．L,A に求めたい行列が入っていることを確認.

```
> L;A;
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 2 & 4/3 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & -2 \\ 0 & -3 & 3 \\ 0 & 0 & -1 \end{bmatrix}$$

数値ベクトルも期待通り変換されている．

```
> B;
```

$$\begin{bmatrix} -4 \\ 9 \\ -2 \end{bmatrix}$$

1.5　ピボット操作

ガウス消去法で困るのは，割ろうとした対角要素が 0 の場合である．しかし，この場合にも，方程式の順序を，行列の行と右辺の値をペアにして入れ替えれば解決する．この割るほうの要素をピボット要素あるいはピボット (pivot, バスケの軸足を動かさずにくるくる回すやつ) と呼ぶ．この操作は，変数の並びを変えたわけではなく，単に方程式の順番を変更する操作に相当する．

さらに対角要素の数値が厳密に 0 でなくとも，極端に 0 に近づいた場合にも，その数で割った数値が大きくなり他の数との差を取ると以前に示した情報落ちの可能性が出てくる．この現象を防ぐためには，絶対値が最大のピボットを選んで行の入れ替えを毎回おこなうといい結果が得られることが知られている．

Maple の LUDecomposition コマンドをこのような行列に適用すると，置換行列 (permutation matrix)P が単位行列ではなく，ピボット操作に対応した行列となる．P.A=L.U となることに注意．

1.6　反復法による連立方程式の解

以下のような連立方程式を

$$\begin{bmatrix} 5\,x+y+z+u \\ x+3\,y+z+u \\ x-2\,y-9\,z+u \\ x+3\,y-2\,z+5\,u \end{bmatrix} = \begin{bmatrix} -6 \\ 2 \\ -7 \\ 3 \end{bmatrix}$$

形式的に解くと

$$x = \frac{-6-(y+z=u)}{5}$$

となる．他の未知数も，

y=

z=

u=

となる．適当に初期値 (x_0, y_0, z_0, u_0) をとり，下側の方程式に代入すると，得られた出力 (x_1, y_1, z_1, u_1) はより正解に近い値となる．これを繰り返すことによって正解が得られる．これをヤコビ (Jacobi) 法と呼び，係数行列の対角要素が非対角要素にくらべて大きいときに適用できる．多くの現実の問題ではこの状況が成り立っている．

Gauss-Seidel 法は Jacobi 法の高速版である．n 番目の解の組が得られた後に一度に次の解の組に入れ替えるのではなく，得られた解を順次改良した解として使っていく．これにより，収束が早まる．以下にはヤコビ法のコードを示した．x1[i] の配列を変数に換えるだけで，Gauss-Seidel 法となる．

```
> AA:=Matrix([[5,1,1,1],[1,3,1,1],[1,-2,-9,1],[1,3,-2,5]]):
  b:=Vector([-6,2,-7,3]): n:=4;
  x0:=[0,0,0,0]: x1:=[0,0,0,0]:
  for iter from 1 to 20 do
    for i from 1 to n do
      x1[i]:=b[i];
      for j from 1 to n do
        x1[i]:=x1[i]-AA[i,j]*x0[j];
      end do:
      x1[i]:=x1[i]+AA[i,i]*x0[i];
      x1[i]:=x1[i]/AA[i,i];
    end do:
    x0:=evalf(x1);
    print(iter,x0);
  end do:
```

$$4$$

```
1, [-1.200000000, 0.6666666667, 0.7777777778, 0.6000000000]
2, [-1.608888889, 0.6074074073, 0.5629629630, 0.7511111112]
3, [-1.584296296, 0.7649382717, 0.5474897119, 0.7825185186]
4, [-1.618989300, 0.7514293553, 0.5187050756, 0.6768921810]
5, [-1.589405322, 0.8077973477, 0.5061160189, 0.6804222770]
6, [-1.598867129, 0.8009556753, 0.4972691400, 0.6356490634]
7, [-1.586774776, 0.8219829753, 0.4927633981, 0.6381076766]
8, [-1.590570810, 0.8186345670, 0.4897074389, 0.6212705292]
9, [-1.585922507, 0.8265309473, 0.4881589539, 0.6228163974]
10, [-1.587501260, 0.8249823853, 0.4870924439, 0.6165295146]
11, [-1.585720869, 0.8279597673, 0.4865626093, 0.6173477984]
12, [-1.586374035, 0.8272701537, 0.4861897104, 0.6149933572]
13, [-1.585690644, 0.8283969890, 0.4860087794, 0.6153885990]
14, [-1.585958873, 0.8280977553, 0.4858782197, 0.6145034472]
15, [-1.585695884, 0.8285257353, 0.4858165626, 0.6146844092]
16, [-1.585805341, 0.8283983040, 0.4857707838, 0.6143503606]
17, [-1.585703890, 0.8285613990, 0.4857498236, 0.6144303994]
18, [-1.585748324, 0.8285078890, 0.4857337457, 0.6143038680]
19, [-1.585709101, 0.8285702367, 0.4857266407, 0.6143384296]
20, [-1.585727061, 0.8285480103, 0.4857209840, 0.6142903344]
```

1.7 課題

1. 後退代入法で解を求めよ．(2005 年度期末類題)

$$\begin{aligned} x + 4y - 3z &= 1 \\ -6y + 4z &= 1 \\ -\tfrac{5}{3}z &= \tfrac{1}{3} \end{aligned}$$

2. 次の行列 A を LU 分解せよ．

```
> A:=Matrix([[1,4,3],[1,-2,1],[2,-2,-1]]);
```

$$\begin{bmatrix} 1 & 4 & 3 \\ 1 & -2 & 1 \\ 2 & -2 & -1 \end{bmatrix}$$

3. 次の連立方程式の係数行列を LU 分解し，上・下三角行列を求めよ．さらに連立方程式の解を求めよ.(2005 年度期末試験)

$$\begin{bmatrix} x_1 + 3\,x_2 + 4\,x_3 + 3\,x_4 \\ -2\,x_1 + 5\,x_2 + 3\,x_3 - 3\,x_4 \\ x_1 + 3\,x_2 - 2\,x_3 + 3\,x_4 \\ 3\,x_1 - 2\,x_2 + x_3 + 4\,x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ -2 \\ 3 \end{bmatrix}$$

4. Jacobi 法のプログラムを参照して Gauss-Seidel 法のプログラムを作れ．Jacobi 法と収束性を比べよ．

5. 次の連立方程式の解を求めよ．ただし，pivot 操作が必要となる．

```
> with(LinearAlgebra):
  A:=Matrix([[3,2,2,1],[3,2,3,1],[1,-2,-3,1],[5,3,-2,5]]):
  X:=Vector([w,x,y,z]):
  b:=Vector([-6,2,-9,2]):
  A.X=b;
```

$$\begin{bmatrix} 3\,w + 2\,x + 2\,y + z \\ 3\,w + 2\,x + 3\,y + z \\ w - 2\,x - 3\,y + z \\ 5\,w + 3\,x - 2\,y + 5\,z \end{bmatrix} = \begin{bmatrix} -6 \\ 2 \\ -9 \\ 2 \end{bmatrix}$$

6. (おまけ) pivot 操作を含めた LU 分解のプログラムを作成せよ．上の問題を解き，その L, U 行列および $L^{-1}.b$ ベクトルを求めよ．

1.8 解答例

4. Jacobi 法のプログラムを参照して Gauss-Seidel 法のプログラムを作れ．Jacobi 法と収束性を比べよ．

解答例

```
#Gauss-Seidel
AA:=Matrix([[5,1,1,1],[1,3,1,1],[1,-2,-9,1],[1,3,-2,5]]):
b:=Vector([-6,2,-7,3]):
n:=4;
x0:=[0,0,0,0]:
x1:=[0,0,0,0]:
for iter from 1 to 20 do
for i from 1 to n do
  x1[i]:=b[i];
  for j from 1 to n do
    x1[i]:=x1[i]-AA[i,j]*x0[j];
  end do:
```

```
  x1[i]:=x1[i]+AA[i,i]*x0[i];
  x1[i]:=x1[i]/AA[i,i];
  x0:=evalf(x1);  #change here from ...
end do:
print(iter,x0);
end do:
```

```
                              4
   1, [-1.200000000, 1.066666667, 0.4074074073, 0.3629629628]
  2, [-1.567407407, 0.9323456790, 0.4367626887, 0.5287791494]
  3, [-1.579577503, 0.8713452217, 0.4673901337, 0.5800644210]
  4, [-1.583759955, 0.8454351333, 0.4783815777, 0.6008435420]
  5, [-1.584932051, 0.8352356437, 0.4828266893, 0.6089756998]
  6, [-1.585407607, 0.8312017393, 0.4845738460, 0.6121900162]
  7, [-1.585593120, 0.8296097527, 0.4852641546, 0.6134584342]
  8, [-1.585666468, 0.8289812930, 0.4855365978, 0.6139591570]
  9, [-1.585695410, 0.8287332183, 0.4856441456, 0.6141568092]
  10, [-1.585706835, 0.8286352933, 0.4856865986, 0.6142348304]
  11, [-1.585711344, 0.8285966383, 0.4857033566, 0.6142656284]
  12, [-1.585713125, 0.8285813800, 0.4857099714, 0.6142777856]
  13, [-1.585713827, 0.8285753567, 0.4857125829, 0.6142825846]
  14, [-1.585714105, 0.8285729793, 0.4857136134, 0.6142844788]
  15, [-1.585714214, 0.8285720407, 0.4857140204, 0.6142852266]
  16, [-1.585714258, 0.8285716703, 0.4857141809, 0.6142855218]
  17, [-1.585714275, 0.8285715240, 0.4857142443, 0.6142856384]
  18, [-1.585714281, 0.8285714660, 0.4857142694, 0.6142856844]
  19, [-1.585714284, 0.8285714433, 0.4857142792, 0.6142857024]
  20, [-1.585714285, 0.8285714343, 0.4857142831, 0.6142857096]
```