平成15年度

卒業論文

# 連立1次方程式に対する $LU$ 分解法と $LDL^T$ 分解法の計算時間比較

平成１６年２月５日


指導教授： 山本　哲朗　教授

早稲田大学　理工学部　情報学科

1g00p013-5　井上　陽介

# 目 次

# 表 目 次

# 図 目 次

# 第 1 章

# 序論

## 1.1 背景

数値解析の基本的精神は、構造を利用していかに速くかつ精度良く問題を解くかということであり、数値線形代数において対称性を利用したりして能率化をはかることは意味があることである。私は、前者のいかに速く問題を解くかというところに着目した。ここで扱う連立1次方程式においては、複数の未知数をいかに効率よく導くかという問題になる。そこで、行列の構造に注目しそれに適合するアルゴリズムを用いることが重要になる。
構造に注目するということは、具体的に言えば、例えば後述する $LDL^T$ 分解法は行列 $A$ が対称行列でないと扱えないが、$A$ が対称行列ならば一般的な解法よりも効率がよくなるということである。実際、未知数が多い時の連立1次方程式にはただでさえ時間がかかるため、よりよい解決法が必須となる。
そこで次元が大きいときの連立1次方程式を解く方法として、$LU$ 分解法と $LDL^T$ 分解法をとりあげ、いろいろな次元の方程式に適用して両者の計算時間の比較を試みた。

## 1.2 本論文の目的

本論文の目的は連立 1 次方程式に対する $LU$ 分解法と $LDL^T$ 分解法での計算時間の比較である。次元が大きくなった場合にどのように変化をたどるのかを分析する。

## 1.3 本論文の構成

第 2 章では, 全ての考え方の基礎になる Gauss の消去法について説明する。

第 3 章では,Gauss 消去法の発展である $LU$ 分解法について説明する。

第 4 章では,$LDL^T$ 分解法のもとになっている Cholesky 法について説明する.

第 5 章では,$LDL^T$ 分解法について説明する。

第 6 章では, 今回の検証について方法について述べる。

第 7 章では, 結果として今回作成した 2 つのプログラムより計算時間を比較する。

第 8 章では, 結論について述べる.

# 第 2 章

# Gaussの消去法

## 2.1 はじめに

連立１次方程式を解くための代表的な方法として Gauss の消去法がある。この章ではこれについて説明する。この考え方の実用的なものが $LU$ 分解法になる。

## 2.2 Gauss の消去法の例

３個の未知数 $x_1, x_2, x_3$ についての連立方程式

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \tag{2.1}$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \tag{2.2}$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \tag{2.3}$$

を例にとって説明する。具体的に考えるために以下の数字で考えることにする。

$$2x_1 + 5x_2 + 7x_3 = 23 \tag{2.4}$$

$$4x_1 + 13x_2 + 20x_3 = 58 \tag{2.5}$$

$$8x_1 + 29x_2 + 50x_3 = 132 \tag{2.6}$$

これを Gauss 法の消去法で解くためには (2.4) の２倍を (2.5) から引き、１番目の式の４倍を (2.6) の式から引く。

$$2x_1 + 5x_2 + 7x_3 = 23 \tag{2.7}$$

$$13x_2 + 20x_3 = 12 \tag{2.8}$$

$$29x_2 + 50x_3 = 40 \tag{2.9}$$

次に (2.8) の３倍を (2.9) から引く.

$$2x_1 + 5x_2 + 7x_3 = 23 \tag{2.10}$$

$$3x_2 + 6x_3 = 12 \tag{2.11}$$

$$4x_3 = 4 \tag{2.12}$$

ここまでがGaussの消去法と呼ばれる部分である。後半は後進代入といって、下の式から順に次のように解いていく。まず (2.12) の式を両辺の４で割れば、

$$x_3 = 1$$

が得られる。次に、これを (2.11) の式に代入し、移項して解くと、

$$x_2 = (12 - 6x_3)/3 = (12 - 6 * 1)/3 = 2$$

が得られる。最後に、これらを (2.10) の式に代入し、移項して解くと、

$$x_1 = (23 - 5x_2 - 7x_3)/2 = (23 - 5 * 2 - 7 * 1)/2 = 3$$

が得られる。ピボット選択という操作を行わないと計算途中で係数行列の対角成分 $a_{ii}$ が 0 に近くなったとき不都合がおきてしまう。次のセクションでピボット選択について説明する。

## 2.3 ピボット選択

例えばGauss消去での一番最初の式、$a_{11}$ が 0 だったとしよう。そうすると、後で説明する前進消去の際の割り算ができなくなる。そこで次のように行を交換してみる。

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$
$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

第１行目と２行目を入れ替えた。もし $a_{21}$ も 0 であるならば、別の行 (例えば $a_{31}$) と交換すれば良いだけである。これで前進消去の際の割算が可能になり、問題が解決する。このように行だけを交換するピボット選択を部分ピボット選択と言う。
係数行列が正則であれば、部分ピボット選択を行うガウスの消去法で必ず連立一次方程式の近似解を計算できることが知られている。部分ピボット選択をしても解が求まらない場合、その連立一次方程式は「解なし」あるいは「解が唯一に定まらない」ということにな

る。

通常、ピボット選択は対角要素が 0 でなくとも行う意味がある。ピボット選択をすることで、浮動小数点数の丸め誤差を抑えることができるからである。上の例で考えてみると、$a_{11}$ が 0 に近い微小な数値だとすると、$x_1$ の項を消す際に、$a_{12}$,$\cdots$,$a_{1n}$ を拡大することになる。もし $a_{12}$,$\cdots$,$a_{1n}$ が丸め誤差を含んでいるとすると、この拡大により、その誤差も拡大することになる。逆に $a_{11}$ が大きな数値であれば、$a_{12}$,$\cdots$,$a_{1n}$ の誤差は縮小する。このような理由により、$a_{11}$ の位置に来る数値は、なるべく絶対値の大きな数が望ましいことがわかる。従って、ここになるべく絶対値の大きい数値が来るように部分ピボット選択を行う。

## 2.4 Gauss の消去法のアルゴリズム

上では

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

を例にとり考えてみたがもっと一般的な場合について考えてみる。以下の操作において $k = 1, 2, ..., n-1$ の順に第 $k$ 行で第 $i$ 行 $(i = k+1, ..., n)$ を掃き出すという操作を続けていく。$(A\ \ b) = (A^{(1)}b^{(1)})$ と置いて、

$$(A^{(1)}b^{(1)}) \to (A^{(2)}b^{(2)}) \to \cdots \to (A^{(k)}b^{(k)}) \to (A^{(k+1)}b^{(k+1)}) \to \cdots \to (A^{(n)}b^{(n)})$$

$$A^{(n)} = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ 0 & & & u_{nn} \end{pmatrix}, b(n) = y$$

そして、$A^{(k)} = (a_{ij}^{(k)}), b^{(k)} = (b_i^{(k)})$ とおいて、以下の説明を続ける。Gauss の消去法の演算を全てまとめると、次の形にまとめることができる。

Gauss の消去法の演算方法

$k = 1, 2, ..., n-1$

$i = k+1, k+2, ..., n$

$m_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)}$

$j = k+1, k+2, ..., n$

$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}$

$b_i^{(k+1)} = b_i^{(k)} - m_{ik} b_k^{(k)}$

最終的には左辺が上三角形をした次の連立 1 次方程式に帰着される。

$$\begin{aligned} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \cdots + a_{1,n-1}^{(1)}x_{n-1} + a_{1n}^{(1)}x_n &= b_1^{(1)} \\ a_{22}^{(2)}x_2 + \cdots + a_{2,n-1}^{(2)}x_{n-1} + a_{2n}^{(2)}x_n &= b_2^{(2)} \\ &\vdots \\ a_{n-1,n-1}^{(n-1)}x_{n-1} + a_{n-1,n}^{(n-1)}x_n &= b_{n-1}^{(n-1)} \\ a_{nn}^{(n)}x_n &= b_n^{(n)} \end{aligned}$$

このようにして得られた上三角形の方程式の最後の方程式から

$$x_n = \frac{b_n^{(n)}}{a_{nn}^{(n)}} \tag{2.13}$$

が求められる。これを最後から 2 番目の式に代入してそれを右辺に移項することで $x_{n-1}$ が求められる。一般に

$$x_k = \frac{b_k^{(k)} - a_{k,k+1}^{(k)}x_{k+1} - a_{k,k+2}^{(k)}x_{k+2} - \cdots - a_{kn}^{(k)}x_n}{a_{kk}^{(k)}} \tag{2.14}$$

によって順次それまで得られた解を代入していくことにより $x_1$ までの全ての解が求められる。こうしてできたアルゴリズムは C 言語で表せば以下のようになる。

Gauss の消去法の C 言語プログラム

```
for (i=1;i<=n;i++){
   pivot = a[i][i];
      for (j=i+1;j<=n;j++){
          p=a[j][i]/pivot;
          a[j][i]=0.0;
          for (k=i+1;k<=n;k++)
              a[j][k] -= p*a[i][k];
          b[j] -= p*b[i];
      }
  }
 x[n] = b[n]/a[n][n];
  for (i=n-1;i>=1;i--){
      x[i]=b[i];
      for (j=i+1;j<=n;j++)
          x[i] -= a[i][j]*x[j];
      x[i] /= a[i][i];
  }
```

このプログラムに対して掛け算/割り算の回数は、

$$(n-i)+(n-i)(n-i+1)+(n-i)+1=(n-i)(n-i+3)+1$$

であり、足し算/引き算の回数は

$$(n-i)(n-i+1)+1+(n-i-1)=(n-i)(n-i+2)$$

である。この結果から掛け算/割り算の合計回数は

$$1+\sum_{i=1}^{n-1}[(n-i)(n-i+3)+1] \;=\; 1+\sum_{i=1}^{n-1}(n^2-2ni+i^2+3n-3i+1)$$

$$
\begin{aligned}
&= 1 + (n^2+3n+1)\sum_{i=1}^{n-1} 1 - (2n+3)\sum_{i=1}^{n-1} i + \sum_{i=1}^{n-1} i^2 \\
&= 1 + (n^2+3n-1)(n-1) - (2n+3)\frac{(n-1)n}{2} \\
&+ \frac{(n-1)n(2n-1)}{6} = \frac{2n^3+6n^2-3n}{6}
\end{aligned}
$$

であり、足し算/引き算の合計回数は

$$
\begin{aligned}
\sum_{i=1}^{n-1}(n-i)(n-i+2) &= \sum_{i=1}^{n-1}(n^2-2ni+i^2+2n-2i) \\
&= (n^2+2n)\sum_{i=1}^{n-1} 1 - (2n+2)\sum_{i=1}^{n-1} i + \sum_{i=1}^{n-1} i^2 \\
&= (n^2+2n)(n-1) - (2n+2)\frac{(n-1)n}{2} + \frac{(n-1)n(2n-1)}{6} \\
&= \frac{2n^3+3n^2-5n}{6}
\end{aligned}
$$

であるとわかる。nが大きいときの乗除演算回数は約 $n^3/3$ で加減演算回数も，$n^3/3$ である。したがって，次の連立 1 次方程式を Gauss の消去法で解くには，$O(n^3)$ 回の演算が必要である。

## 2.5 むすび

この章では連立 1 次方程式を解くための代表的な方法である Gauss の消去法を説明した。次章ではこの応用となる $LU$ 分解法の説明をする。

# 第 3 章

# $LU$ 分解法

## 3.1 はじめに

前章で説明したGaussの消去法の応用となるものが$LU$分解法である。この章では$LU$分解法について説明する。

## 3.2 $LU$分解法の定義

連立1次方程式$Ax = b$を解く際に、係数行列Aが同じ連立方程式を幾組みも解くなら、まずAの逆行列を求めて

$$x = A^{-1}b$$

とすることも考えられるが、それよりGauss法の前半部分のAについて行ってその結果を残しておき、連立1次方程式の右辺$b$が与えられるごとにGauss法の残りの部分を行う方が速い。まず、

$$M_1 = \begin{pmatrix} 1 & & & & \\ -m_{21} & 1 & & 0 & \\ -m_{31} & 0 & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ -m_{n1} & 0 & 0 & \cdots & 1 \end{pmatrix}$$

とおく。各成分の右上に添字を付したのに対応して

$$A^{(1)} = A$$

と書いておくことにする。消去の第1段は

$$M_1 A^{(1)} = A^{(2)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix}$$

という演算を行うことと等しい。これは積 $M_1A^{(1)}$ の各行を取り出し実際の消去の演算と比較することにより容易に確かめられる。このようにすれば一般に消去の第 k 段は

$$M_K = \begin{pmatrix} 1 & & & & & & & \\ 0 & 1 & & & & & & \\ \vdots & & \ddots & & & 0 & & \\ 0 & 0 & \cdots & 1 & & & & \\ \vdots & \vdots & & -m_{k+1,k} & 1 & & & \\ \vdots & \vdots & & -m_{k+2,k} & 0 & \ddots & & \\ \vdots & \vdots & & \vdots & \vdots & & \ddots & \\ 0 & 0 & \cdots & -m_{n,k} & 0 & \cdots & \cdots & 1 \end{pmatrix}, m_{jk} = a_{jk}^{(k)}/a_{kk}^{(k)} \tag{3.1}$$

とおくと、

Gauss の消去法での演算方法

$k = 1, 2, ..., n-1$

$i = k+1, k+2, ..., n$

$m_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}$

$j = k+1, k+2, ..., n$

$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}$

$b_i^{(k+1)} = b_i^{(k)} - m_{ik}b_k^{(k)}$

より

$$M_kA^{(k)} = A^{(k+1)} \tag{3.2}$$

と表すことができる。最終段まで進めば結局

$$M_{n-1}M_{n-2}...M_2M_1A = A^{(n)}$$

$$= U = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & a_{33}^{(1)} & \cdots & a_{3n}^{(3)} \\ & 0 & & \ddots & \vdots \\ & & & & a_{nn}^{(n)} \end{pmatrix} \tag{3.3}$$

となる。この行列 $U$ のように対角線より左下にある成分が全て 0 の行列を上三角行列をいう。右上のベクトルにも同様の操作

$$M_{n-1}M_{n-2}...M_1 b = b^{(n)} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(n)} \end{pmatrix} \tag{3.4}$$

を行えば、方程式 $Ax = b$ は（3.3）の U を用いて次の形になる。

$$Ux = b^{(n)} \tag{3.5}$$

一方、先ほど求めた $M_k$ の逆行列は

$$M_k^{-1} = \begin{pmatrix} 1 & & & & & & & \\ 0 & 1 & & & & & & \\ \vdots & & \ddots & & & 0 & & \\ 0 & 0 & \cdots & 1 & & & & \\ \vdots & \vdots & & m_{k+1,k} & 1 & & & \\ \vdots & \vdots & & m_{k+2,k} & 0 & \ddots & & \\ \vdots & \vdots & & \vdots & \vdots & & \ddots & \\ 0 & 0 & \cdots & m_{n,k} & 0 & \cdots & \cdots & 1 \end{pmatrix}, m_{jk} = a_{jk}^{(k)}/a_{kk}^{(k)} \tag{3.6}$$

で与えられ、実際に積 $M_k^{-1}M_k$ を実際に作れば単位行列 $I$ に等しくなる。さらに行列 $L$ を

$$L = M_1^{-1}M_2^{-1}...M_{n-1}^{-1} \tag{3.7}$$

と定義すれば

$$L = \begin{pmatrix} 1 & & & & & \\ m_{21} & 1 & & & 0 & \\ m_{31} & m_{32} & 1 & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ \vdots & \vdots & \vdots & & 1 & \\ m_{n1} & m_{n2} & m_{n3} & \cdots & m_{n,n-1} & 1 \end{pmatrix}, m_{jk} = a_{jk}^{(k)}/a_{kk}^{(k)} \tag{3.8}$$

となる。この行列 $L$ のように対角線より右上にある成分が全て 0 の行列を下三角行列という。こうして（3.3）と（3.7）より Gauss の消去法よりはじめの行列 $A$ は

$$A = LU \tag{3.9}$$

の形に下三角行列 $L$ と上三角行列 $U$ の形に分解されたことになる。これを行列 $A$ の $LU$ 分解という。方程式を解く際に行列 $A$ を $LU$ 分解しておけば $L$,$U$ ともに三角行列なので方程式 $Ax = b$ を解くことは簡単である。方程式 $Ax = b$ は

$$LUx = b$$

と書くことができる。まず

$$Ly = b \tag{3.10}$$

の解 $y$ を求め、次に

$$Ux = y \tag{3.11}$$

を解いて $x$ を求めればよい。方程式 $Ly = b$ は $m_{kk} = 1$ に注意すれば

前進代入

$$y_1 = b_1$$

$$k = 2, 3, ..., n$$

$$y_k = b_k - \sum_{j=1}^{k-1} m_{kj} y_j$$

によって解くことができる。

次に方程式 $Ux = y$ は後ろから順に

後進代入

$$x_n = y_n / a_{nn}^{(n)}$$

$$k = n-1, n-2, ..., 1$$

$$x_k = \left( y_k - \sum_{j=k+1}^{n} a_{kj}^{(k)} x_j \right) / a_{kk}^{(k)}$$

によって解くことができる。

## 3.3 $LU$ 分解法の具体的な説明

例として3×3の行列 $A$ を

$$A = \begin{pmatrix} 2 & 5 & 7 \\ 4 & 13 & 20 \\ 8 & 29 & 50 \end{pmatrix}$$

とする。Gauss 法ではこの係数行列の行をある一定の順序で２倍、４倍、３倍して他の行から引いて、

$$\begin{pmatrix} 2 & 5 & 7 \\ 0 & 3 & 6 \\ 0 & 0 & 4 \end{pmatrix}$$

のように左下部分を全て0にするのである。この左下部分の0はもう不要であるから、ここに先ほどの２倍、４倍、３倍という情報を

$$A' = \begin{pmatrix} 2 & 5 & 7 \\ 2 & 3 & 6 \\ 4 & 3 & 4 \end{pmatrix}$$

のように入れておけば、右辺 $b$ が与えられたとき、簡単に連立方程式を解くことができる。ちなみにこうして求めたA'を左下部分 $L$（対角成分は１）と右上部分 $U$ に分けると、元の

$A$ はちょうど $L$ と $U$ をかけたものになっている。

$$\begin{pmatrix} 2 & 5 & 7 \\ 4 & 13 & 20 \\ 8 & 29 & 50 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 4 & 3 & 1 \end{pmatrix} \begin{pmatrix} 2 & 5 & 7 \\ 0 & 3 & 6 \\ 0 & 0 & 4 \end{pmatrix} \tag{3.12}$$

とこのように $A$ を $L$ と $U$ に具体的に $LU$ 分解することができる。

上で述べた $LU$ 分解のアルゴリズムはつぎのように書ける。

$LU$ 分解法の C 言語プログラム

```
for( i = 1; i <= n; i++ ){
    for( j = 1; j <= n; j++ ){
        if( i == j ){ l[i][j] = 1; }
        else{ l[i][j] = 0; }
        u[i][j] = a[i][j];
    }
}
for( i = 1; i <= n; i++ ){
    for( j = i+1; j <= n; j++ ){
        l[j][i] = u[j][i] / u[i][i];
        for( k = i; k <= n; k++ ){
            u[j][k] = u[j][k] - u[i][k] * l[j][i];
        }
    }
}
```

Gauss 法の節で説明したピボット選択をもう一度おさらいすると、このプログラムの部分では a[1][1] が 0 なので計算できなくなる。そこで a[1][1] と a[2][1] の行を交換すると解けるというものだ。対角成分 $a_{kk}$、プログラムでは a[k][k] が例え 0 に等しくなくても 0 に近いと解の誤差が増えてしまう。そこで先ほども述べたように $a_{kk}$ になるべく絶対値の大きなも

のがくるように行を交換しながら解くとよい。

## 3.4 むすび

この章では Gauss の消去法の応用である $LU$ 分解について説明した。次章では、また異なるタイプの解法として Cholesky 法を説明する。

# 第 4 章

# Cholesky法

## 4.1 はじめに

前章では $LU$ 分解法を説明したが、この章では行列 $A$ が特別な場合についての解法、Cholesky 法について説明する。

## 4.2 Cholesky 法の定義

行列 $A$ が正定値対称行列のとき、$A = S^T S$ と分解され、この分解を利用して連立 1 次方程式の一つの解法を示すことができる。また、後に述べる $A$ が対称行列である $LDL^T$ 分解は修正コレスキー法とも呼ばれているので、先にこの Cholesky 法を説明する。今行列 $A$ は正定値対称行列であるから、$a_{kk}^{(k)} > 0$ である。したがって、

$$D^{\frac{1}{2}} = \begin{pmatrix} \sqrt{a_{11}^{(1)}} & & & 0 \\ & \sqrt{a_{22}^{(2)}} & & \\ & & \ddots & \\ 0 & & & \sqrt{a_{nn}^{(n)}} \end{pmatrix} \tag{4.1}$$

と書いて積 $S = D^{\frac{1}{2}} L^T$ を作れば $A$ は次のように分解される。

$$A = S^T S \tag{4.2}$$

この分解を正定値対称行列 $A$ のコレスキー分解という。この行列 $S$ は上三角形である。

$$S = \begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1n} \\ & s_{22} & \cdots & s_{2n} \\ & & \ddots & \vdots \\ 0 & & & s_{nn} \end{pmatrix} \tag{4.3}$$

いま形式的に積 $S^T S$ の第 $ij$ 成分を作り、それを $a_{ij}$ と等置すると、

$$s_{1i}s_{ij} + s_{2i}s_{2j} + \cdots + s_{ii}s_{ij} = a_{ij}, (i \leq j) \tag{4.4}$$

となり、これから次の関係式がえられる。

$$\begin{aligned} s_{ii} &= \sqrt{a_{ii} - \sum_{k=1}^{i-1} s_{ki}^2}, (s_{11} = \sqrt{a_{11}}) \\ s_{ij} &= (a_{ij} - \sum_{k=1}^{i-1} s_{ki} s_{kj})/s_{ii}, (s_{1j} = a_{1j}/s_{11}) \end{aligned} \tag{4.5}$$

この式によって $s_{11} = \sqrt{a_{11}}$ を出発点として $s_{12}, s_{13}, ..., s_{1n}, s_{22}, s_{23}, ...$ の順に $S$ の成分を求めることができる。$S$ が求められれば、三角行列 $S^T y = b, Sx = y$ を解くことにより $Ax = b$ の解が得られる。これをコレスキー法という。この方法は一番初めに示したように行列 $A$ が正定値対称の場合に限り使うことができる。

## 4.3 Cholesky 法のアルゴリズム

Cholesky 法は、上で述べた

$$\begin{aligned} s_{ii} &= \sqrt{a_{ii} - \sum_{k=1}^{i-1} s_{ki}^2}, (s_{11} = \sqrt{a_{11}}) \\ s_{ij} &= (a_{ij} - \sum_{k=1}^{i-1} s_{ki} s_{kj})/s_{ii}, (s_{1j} = a_{1j}/s_{11}) \end{aligned} \tag{4.6}$$

を使えば実装することができる。実際にその部分を C 言語で実装したものが下の部分的プログラムである。

Cholesky 法の C 言語プログラム

```
for (i = 1; i <= n; i++) {
   s = a[i][i];
   for (k = 1; k < i; k++)
        s -= sqr(L[i][k]);
            if (s < 0) {
                  exit(0);
            }
          L[i][i] = sqrt(s);
          for (j =i + 1; j <= n; j++) {
                     s = a[j][i];
                     for (k = 1; k < i; k++)
                           s -= L[j][k] * L[i][k];
                           L[j][i] = s / L[i][i];
               }
         }
  }
```

## 4.4 むすび

この章では行列 $A$ が正定値対称行列である場合についての解法、Cholesky 法について説明した。次章ではこの修正版といえる $LDL^T$ 分解法について説明する。

# 第 5 章

# $LDL^T$ 分解法

## 5.1 はじめに

この章では行列 $A$ が対称行列である場合についての解法、Cholesky 法の修正版といえる $LDL^T$ 分解法について説明する。

## 5.2 $LDL^T$ 分解法の定義

行列 $A$ を $LU$ 分解して得られる先ほどの右上三角行列 $U$ の各行を対角成分で割ることにより、$U$ を次の形に分解することができる。

$$U = DV \tag{5.1}$$

$$D = \begin{pmatrix} a_{11}^{(1)} & & & & \\ & a_{22}^{(2)} & & & 0 \\ & & a_{33}^{(3)} & & \\ & & & \ddots & \\ 0 & & & & a_{nn}^{(n)} \end{pmatrix} \tag{5.2}$$

$$V = \begin{pmatrix} 1 & v_{12} & v_{13} & \cdots & v_{1n} \\ & 1 & v_{23} & \cdots & v_{2n} \\ & & 1 & \cdots & v_{3n} \\ & 0 & & \ddots & \vdots \\ & & & & 1 \end{pmatrix}, v_{kj} = a_{kj}^{(k)} / a_{kk}^{(k)}, k < j \tag{5.3}$$

ここで $A$ が対称行列 $A^t = A$ であり、かつ消去の過程で行の入れ換えを行わない場合のガウス消去を考えてみる。このとき、第 k 段のいまだ消去されていない $(n-k) \times (n-k)$ ブロックの部分に関して対称性が保存される。Gauss 消去で第 $\mathrm{k}-1$ 段まで消去が進んだ時

の方程式を使い、ここで対象にする小行列を

$$A_s^{(k)} = \begin{pmatrix} a_{kk}^{(k)} & a_{k,k+1}^{(k)} & \cdots & a_{kn}^{(k)} \\ a_{k+1,k}^{(k)} & a_{k+1,k+1}^{(k)} & \cdots & a_{k+1,n}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{nk}^{(k)} & a_{n,k+1}^{(k)} & \cdots & a_{nn}^{(k)} \end{pmatrix} \tag{5.4}$$

とおく。このとき

$$A_s^{(k)} = A_s^{(k)T} \tag{5.5}$$

これを成分ごとに書けば

$$a_{ij}^{(k)} = a_{ji}^{(k)}, k \leq i, j \leq n \tag{5.6}$$

が成立する。なぜならば、第 $k-1$ 段目の消去でなされる演算は

$$a_{ij} = a_{ij}^{(k-1)} - \frac{a_{i,k-1}^{(k-1)}}{a_{k-1,k-1}^{(k-1)}} a_{k-1,j}^{(k-1)} \tag{5.7}$$

であるが、もし $a_{\alpha\beta}^{(k-1)} = a_{\beta\alpha}^{(k-1)}$ であれば $a_{ij}^{(k)} = a_{ji}^{(k)}$ となる。初期設定で $A = A^T$、すなわち $a_{\alpha\beta}^{(1)} = a_{\beta\alpha}^{(1)}$ であるから以下全ての段階でこれが成立する。
以上から $A$ が対称行列のとき、

$$v_{kj} = \frac{a_{kj}^{(k)}}{a_{kk}^{(k)}} = \frac{a_{jk}^{(k)}}{a_{kk}^{(k)}} = m_{jk} \tag{5.8}$$

となる。したがって、V と $L$ の間に次の関係が成立する。

$$V = L^T \tag{5.9}$$

$A$ が対称行列ならば $A$ は次の形に分解できることがわかった。

$$A = LDL^T \tag{5.10}$$

これを行列 $A$ の $LDL^T$ 分解という。

## 5.3 修正コレスキー法

行列 $A$ が対称の場合、$LDL^T$ 分解から連立 1 次方程式を解くアルゴリズムを直接導くことができる。

行列 $D$ の ii 成分を $d_i = a_{ii}^{(i)}$、行列 $L$ の ij 成分を $l_{ij}$ と書き、$A = LDL^T$ の右辺の積 $LDL^T$ の成分を左辺の $A$ の対応する成分と倒置すれば、k=1,2,...,n に対して次の関係を得る。

$$\sum_{j=1}^{i} l_{kj} d_j l_{ij} = a_{ki} \quad i = 1, 2, ..., k-1$$
$$\sum_{i=1}^{k} l_{ki} d_i l_{ki} = a_{kk}$$

ここで $l_{ii} = 1$ に注意すると、これから次の手順が導かれる。

$LDL^T$ 分解法の演算方法

$d_1 = a_{11}$

$k = 2, 3, ..., n$

$i = 1, 2, ..., k-1$

$$l_{ki} = \left( a_{ki} - \sum_{j=1}^{i-1} l_{kj} l_{ij} d_j \right) / d_i$$

$$d_k = a_{kk} - \sum_{i=1}^{k-1} l_{ki}^2 d_i$$

ただし、和の記号 $\sum$ の上限が下限よりも小さい場合にはその和は 0 とする。この式に従い $d_1 = a_{11}$ から出発して $l_{21}, d_2, l_{31}, l_{32}, d_3, ...$ の順に $D$ と $L$ の成分を求めることができる。

$A$ が $LDL^T$ の形に分解されれば、$Ax = LDL^T x = Ly, y = DL^T x$ の関係から三角行列を係数に持つ二つの方程式 $Ly = b, DL^T x = y$ を解くことにより $Ax = b$ の解を求めることができる。この方法は修正コレスキー法という。

## 5.4 $LDL^T$ 分解法の具体的な計算

$LDL^T$ 分解の具体的な計算をする際にも連立 1 次方程式を例にとって説明することにする。例えば次の連立 1 次方程式が与えられていたとする。

$$2x_1 + x_2 + x_3 = 8 \tag{5.11}$$

$$x_1 + 3x_2 + 2x_3 = 11 \tag{5.12}$$

$$x_1 + 2x_2 + 4x_3 = 16 \tag{5.13}$$

この式をマトリックス表示すれば、

$$\begin{pmatrix} 2 & 1 & 1 \\ 1 & 3 & 2 \\ 1 & 2 & 4 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \begin{pmatrix} 8 \\ 11 \\ 16 \end{pmatrix} \tag{5.14}$$

となる。左辺の係数行列Aは対称行列でなければならない。上式を

$$Ax = b \tag{5.15}$$

とおいた時、行列 $A$ を $LDL^T$ 分解すると、

$$A = LDL^T \tag{5.16}$$

$$= \begin{pmatrix} 1 & & \\ l_{21} & 1 & \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} d_{11} & & \\ & d_{22} & \\ & & d_{33} \end{pmatrix} \begin{pmatrix} 1 & l_{21} & l_{31} \\ & 1 & l_{32} \\ & & 1 \end{pmatrix} \tag{5.17}$$

$$= \begin{pmatrix} d_{11} & d_{11}l_{21} & d_{11}l_{31} \\ d_{11} & d_{11}l_{21}^2 + d_{22} & d_{11}l_{21}l_{31} + d_{22}l_{32} \\ d_{11} & d_{11}l_{21}l_{31} + d_{22}l_{32} & d_{11}l_{31}^2 + d_{22}l_{32}^2 + d_{33} \end{pmatrix} \tag{5.18}$$

$$= \begin{pmatrix} 1 & & \\ 0.5 & 1 & \\ 0.5 & 0.6 & 1 \end{pmatrix} \begin{pmatrix} 2 & & \\ & 2.5 & \\ & & 2.6 \end{pmatrix} \begin{pmatrix} 1 & 0.5 & 0.5 \\ & 1 & 0.6 \\ & & 1 \end{pmatrix} \tag{5.19}$$

$LU$ 分解と同様、$AX = b$ は $LDL^T x = b$ なので

$$LDL^T x = b \tag{5.20}$$

が言える。ここで

$$DL^T x = y \tag{5.21}$$

と置き換えれば、

$$Ly = b \tag{5.22}$$

が言える。上の式をマトリックス表示すれば、

$$\begin{pmatrix} 1 & & \\ 0.5 & 1 & \\ 0.5 & 0.6 & 1 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \begin{pmatrix} 8 \\ 11 \\ 16 \end{pmatrix} \tag{5.23}$$

となる。これより、

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 8 \\ 7 \\ 7.8 \end{pmatrix} \tag{5.24}$$

となり、この結果と $DL^T x = y$ より

$$\begin{pmatrix} 2 & & \\ & 2.5 & \\ & & 2.6 \end{pmatrix} \begin{pmatrix} 1 & 0.5 & 0.5 \\ & 1 & 0.6 \\ & & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 8 \\ 7 \\ 7.8 \end{pmatrix} \tag{5.25}$$

$$\begin{pmatrix} 2 & 1 & 1 \\ & 2.5 & 1.5 \\ & & 2.6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 8 \\ 7 \\ 7.8 \end{pmatrix} \tag{5.26}$$

後進代入より、

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix} \tag{5.27}$$

となり、解が求まる。ここで、$L$ と $D$ を一般化して求めるならば、

$$d_{11} = a_{11}, l_{j1} = a_{j1}/d_1 (j = 2, ..., n)$$

$$d_{22} = a_{22} - d_{11} l_{21}^2, l_{j2} = (a_{2j} - d_{11} l_{21} l_{j1}/d_{22}) (j = 3, ..., n)$$

$$d_{33} = a_{33} - d_{11} l_{31}^2 - d_{22} l_{32}^2, l_{j3} = (a_{3j} - d_{11} l_{31} l_{j1} - d_2 l_{32} l_{j2})/d_{33} (j = 4, ..., n)$$

となり、このアルゴリズムは次のように書き表せる。

$LDL^T$ 分解法の C 言語プログラム

```
d[1]=a[1][1];
for(k=2;k<=n;k++){
    s=0;t=0;
    for(i=1;i<=k-1;i++){
         for(j=1;j<=i-1;j++){
              s+=l[k][j]*l[i][j]*d[j];
        }
         l[k][i]=(a[k][i] - s)/d[i];
    }
    for(i=1;i<=k-1;i++){
          t+=l[k][i]*l[k][i]*d[i];
    }
     d[k]=a[k][k]-t ;
}
```

## 5.5 むすび

この章では行列 $A$ が対称行列である場合についての解法、$LDL^T$ 分解法について説明した。次章では $LU$ 分解法と $LDL^T$ 分解法の比較方法について述べる。

# 第 6 章

# 検証

## 6.1 はじめに

本章ではC言語でプログラミングした $LU$ 分解法と $LDL^T$ 分解法とのプログラムの計算時間の検証方法を説明する.

## 6.2 検証した計算機の環境

本論文でのプログラムの作成, 実行, 検証は全て以下に示す環境で行われている.

| | |
|---|---|
| CPU | Celeron 2.0GHz |
| memory | 512MB |
| OS | Windows XP |
| コンパイラ | cygwin+gcc |

表 6.1: 本論文における計算機の環境

## 6.3 検証したプログラムについて

### $LU$ 分解を用いた計算プログラム

以下の手順で計算した。

1. $LDL^T$ と条件を同じにするために、対称行列 $A$ を作る。

2. 対称行列 $A$ は $a_{11}$ からはじまり、それを囲うように2,3,...,nとする。

$$A = \begin{pmatrix} 1 & 2 & 3 & \cdots & n \\ 2 & 2 & 3 & \cdots & n \\ 3 & 3 & 3 & \cdots & n \\ \vdots & \vdots & \vdots & \ddots & n \\ n & n & n & n & n \end{pmatrix}$$

というように対称行列 $A$ を作成する。

3. $A$ の決定後、ベクトル $x$ を作成（簡単のため、$x_i = i, (i = 1, ..., n)$ とする。）
4. $Ax = b$ より、ベクトル $b$ が決定。
5. 対称行列 $A$ を $LU$ 分解。
6. $Ly = b$ の解 $y$ を求める（前進代入）。
7. $Ux = y$ の解 $x$ を求める（後進代入）。
8. 5 から 7 までの時間を測定する。

## $LDL^T$ 分解を用いた計算プログラム

以下の手順で計算した。

1. 行列 $A$, ベクトル $x, b$ の作成については $LU$ 分解と同じ。
2. 対称行列 $A$ を $LDL^T$ 分解。
3. $Ly = b$ の解 $y$ を求める（前進代入）。
4. $DL^T x = y$ の解 $x$ を求める（後進代入）。
5. 2 から 4 までの時間を測定する。

## 6.4 むすび

本章ではC言語でプログラミングした $LU$ 分解法と $LDL^T$ 分解法とのプログラムの計算時間の検証方法を説明した。次章ではこの結果について解説する。

# 第 7 章

# 結果

## 7.1 はじめに

本章ではC言語でプログラミングした $LU$ 分解法と $LDL^T$ 分解法とのプログラムの計算時間を比較する。

## 7.2 結果

実験では行列の次元nを100から1000までに変化させてその時間の推移を2つのプログラムで比べた。結果のグラフは下のようになった。

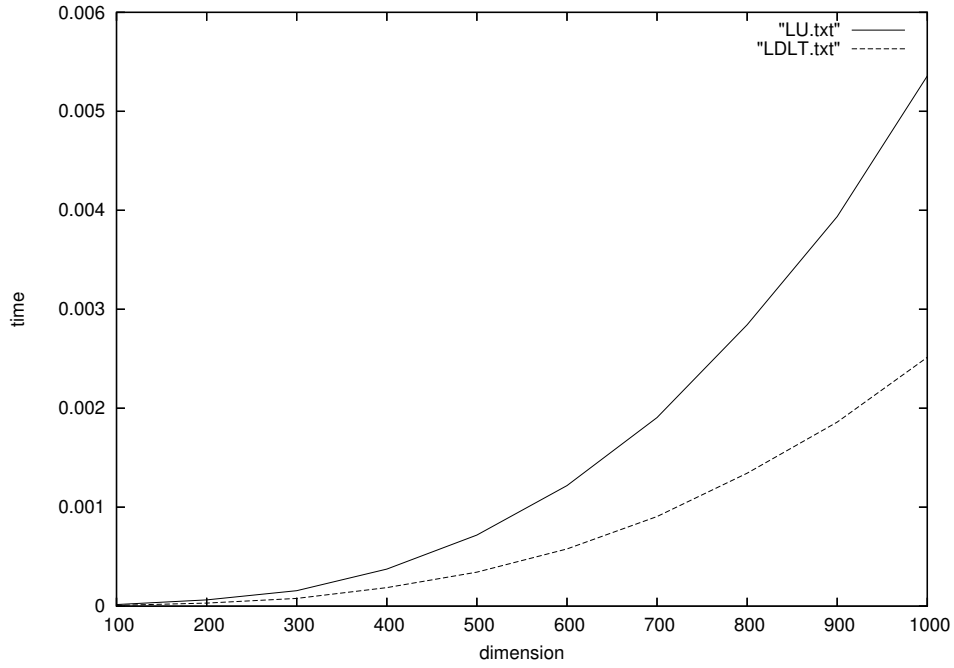


図 7.1: $LU$ 分解法と $LDL^T$ 分解法の実行時間比較

実験時、100次元の時点で,$LDL^T$ 分解法の実行時間は0.000008秒で $LU$ 分解法の実行時間は0.000015秒となった。この結果が次元を大きくしたときにも1000次元までほぼ比例していった。アルゴリズムを比較しても、$LU$ 分解法の計算量は $n^3/3+O(n^2)$ であり、$LDL^T$ 分解法の計算量は、$n^3/6+O(n^2)$ であるからこれにも適合している。

## 7.3 むすび

この章ではC言語でプログラミングした$LU$分解法と$LDL^T$分解法とのプログラムの計算時間の検証結果を説明した。次章ではこの結果をもとに考察していく。

# 第 8 章

# 統括

## 8.1 はじめに

本章では，本論文の結びとして，本論文の統括を述べる.

## 8.2 統括

本論文では，第 2 章では, 全ての考え方の基礎になる Gauss の消去法について述べた。
第 3 章では,Gauss 消去法の発展である $LU$ 分解法について述べた。
第 4 章では,$LDL^T$ 分解法のもとになっている Cholesky 法について述べた.
第 5 章では,$LDL^T$ 分解法について述べた。
第 6 章では, 今回の検証について方法について述べた。
第 7 章では, 結果として今回作成した 2 つのプログラムより検証時間を比較した。

本論文の構成は以上のようであった.

## 8.3 考察

行列 $A$ が対称行列ならば、$LDL^T$ 分解法は $LU$ 分解法の約半分の計算時間で納まるということが前章のグラフより明らかになった。実験時にはほんのわずかではあるが誤差がでたがそこは約半分の範囲に入れても構わないものであると思っている。正確に半分にならないことに関しては、計算機の環境がいつも同じであるとは限らないということが挙げられるだろう。とはいえ、約半分であることを確認するのには十分な結果を得られたのではないか。

## 8.4 むすび

本論文の目的は $LU$ 分解法と $LDL^T$ 分解法の実行時間を比較することであった。アルゴリズムを比較しても、$LU$ 分解法の計算量は $n^3/3 + O(n^2)$ であり、$LDL^T$ 分解法の計算量は、$n^3/6 + O(n^2)$ であるから、プログラムを走らせた実行時間もこれに適合しているといえるだろう。数値計算において時間計算量というものは、環境の違い等によりあまり好まれるものではないかもしれないが、実際にプログラムを走らせることにより、四則演算を数えて導く計算量を、実際に身をもって体験することができた。$LDL^T$ 分解法が対称行列においては $LU$ 分解法をはるかにしのぐ性能を持っていることが次元を増やした今回の実験で確かに認められたという結果をもって本論文のむすびとしたい。

# 謝辞

本研究を進めるに当たり, 終始丁寧な御指導及び御激励を賜り, その他多くの面でも色々と御面倒を見て下さり御助言を与えて下さいました山本 哲朗教授に深く感謝いたします.

また，日常生活において色々とお世話になりました，山本研究室修士課程２年阿口 誠司氏に深く感謝いたします.

最後に， 同じ研究室として意見の交換や協力などをして下さいました山本研究室４年 藤田 祐作氏, 嶋田 陽介氏, 佐藤 裕介氏 に深く感謝いたします.

# 参考文献

[1] 山本哲朗:”数値解析入門（増訂版）”, サイエンス社, 2003 年,

[2] 大石進一:”数値計算”, 裳華房，1999 年,

[3] 森正武:”数値解析（改訂版）”, 共立出版, 2002 年,

[4] G.H.Golub and C.F.Van Loan:”Matrix Computations”,Johns Hopkins,1996 年

# Appendix A

# プログラム

```
///////////////LDL^T_Factorization///////////////
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define N  1000 /*次元の数*/

int main(void);
  int main(void){

   int i,j,k;
   double s;
   double t;
   double sum;

   static double a[N+1][N+1];  /*a_{11}を a[1][1] と対応させるため*/
   static double b[N+1];
   static double w[N+1][N+1];
   static double x[N+1];
   static double y[N+1];
   static double l[N+1][N+1];
   static double d[N+1];
   static double ld[N+1][N+1];
   unsigned long t1,t2 ;

     /*A 対称行列を作成*/
     for(i=1;i<=N;i++){
          for(j=i;j<=N;j++){
               a[i][j]=j;
          }
          for(j=i-1;j>=1;j--){
               a[i][j]=a[j][i];
          }
     }

     /*ランダムな X を決定する*/
```

```
for(i=1;i<=N;i++){
     x[i]=i;
}

/*Ax=b よりベクトル B が決定*/
for(i=1;i<=N;i++){
     for(k=1,sum=0;k<=N;k++){
          sum+=a[i][k]*x[k];
     }
     b[i] = sum;
}

/*Xをリセット*/
for(i=1;i<=N;i++){
     x[i]=0;
}
t1=clock();

/*↓これより LDL^T 分解*/
/*L をリセット、L行列の対角成分は1*/
for(i=1;i<=N;i++){
     for(j=1;j<=N;j++){
          if(i==j){
               l[i][j]=1;
          }
          else{
               l[i][j]=0;
          }
     }
}

d[1]=a[1][1];
     for(k=2;k<=N;k++){
          for(i=1;i<=k-1;i++){
               for(j=1,s=0;j<=i-1;j++){
```

```
                    s+=l[k][j]*l[i][j]*d[j];
               }
               l[k][i]=(a[k][i] - s)/d[i];
          }

          for(i=1,t=0;i<=k-1;i++){
               t+=l[k][i]*l[k][i]*d[i];
          }
          d[k]=a[k][k]-t ;
     }
/*ここまでが LDL^T 分解*/

for(i=1;i<=N;i++){
     y[i]=b[i];
     for(j=1;j<i;j++){
          y[i]-=l[i][j]*y[j];
     }
}

/* DL^T*x=y 解く */
/*DL^T を計算*/
for(i=1;i<=N;i++){
     for(j=1;j<=N;j++){
          w[i][j]=d[i]*l[j][i];    /*l[j][i] はL^T*/
     }
}
for(i=N;i>=1;i--){
     x[i]=y[i];
     for(j=N;j>i;j--){
          x[i]-=w[i][j]*x[j];
     }
     x[i]/=w[i][i];
}
t2=clock();
```

```
     printf("      ; %f[ms]\n", (t2-t1)/1000000.0 );
     return(0);
}

///////////////LU_Factorization///////////////
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define N 1000

int main(void);
  int main(void){

   int i,j,k;
   double s,t,sum;
   static double a[N+1][N+1];  /*a_{11}を a[1][1] と対応させるため*/
   static double b[N+1];
   static double u[N+1][N+1];
   static double w[N+1][N+1];
   static double x[N+1];
   static double y[N+1];
   static double l[N+1][N+1];
   static double d[N+1];
   static double ld[N+1][N+1];
   unsigned long t1,t2 ;

     /*A 対称行列を作成*/
     for(i=1;i<=N;i++){
          for(j=i;j<=N;j++){
               a[i][j]=j;
          }
          for(j=i-1;j>=1;j--){
               a[i][j]=a[j][i];
          }
     }
```

```
 /*ランダムな X を決定する*/
 for(i=1;i<=N;i++){
      x[i]=i;
 }

 /*Ax=b よりベクトル B が決定*/
 for(i=1;i<=N;i++){
      for(k=1,sum=0;k<=N;k++){
           sum+=a[i][k]*x[k];
      }
      b[i] = sum;
 }

 /*Xをリセット*/
 for(i=1;i<=N;i++){
      x[i]=0;
 }
 t1=clock();

 /*ここより LU 分解*/
for( i = 1; i <= N; i++ ){
    for( j = 1; j <= N; j++ ){
        if( i == j ){ l[i][j] = 1; }
        else{ l[i][j] = 0; }
        u[i][j] = a[i][j];
    }
}

for( i = 1; i <= N; i++ ){
    for( j = i+1; j <= N; j++ ){
        l[j][i] = u[j][i] / u[i][i];
        for( k = i; k <= N; k++ ){
            u[j][k] = u[j][k] - u[i][k] * l[j][i];
        }
```

```
      }
  }

   for(i=1;i<=N;i++){
        y[i]=b[i];
        for(j=1;j<i;j++){
             y[i]-=l[i][j]*y[j];
        }
   }
   for(i=N;i>=1;i--){
        x[i]=y[i];
        for(j=N;j>i;j--){
             x[i]-=u[i][j]*x[j];
        }
        x[i]/=u[i][i];
   }
   t2=clock();

   printf("      ; %f[ms]\n", (t2-t1)/1000000.0 );
   return(0);
}
```