電気電子計算工学及演習

三軒家 佑將（さんげんや ゆうすけ）
3 回生
1026-26-5817
a0146089

1 前進代入

1.1 採用したアルゴリズム

$$y_i = b_i - \sum_{k=0}^{i-1} l_{ik} y_k$$

として、$i = 0, 1, 2$ の順に y_i を求めた。

1.2 プログラムに関する情報

ファイル名

- 1.go
- forward.go
- print.go

コンパイルコマンド

go run 1.go print.go forward.go

作成した主な関数

PrintVector
: ベクトル (1 次元配列) を表示する関数。引数としてベクトルを渡す。

PrintMatrix
: 行列 (2 次元配列) を表示する関数。引数としてベクトルを渡す。

Forward
: 前進代入法によって方程式の解を求める関数。第一引数として行列（下三角行列）を、第二引数としてベクトル（$b = Lx$ のときの b）を渡すと、方程式の解をベクトルとして返す。

1.3 結果

手計算の結果は、

$$\mathbf{y} = \begin{pmatrix} 9 \\ 8 \\ -4 \end{pmatrix}$$

であった。また、プログラムによる数値解は、

$$\mathbf{y} = \begin{pmatrix} 9.000 \\ 8.000 \\ -4.000 \end{pmatrix}$$

であった。

1.4 考察

手計算による解とプログラムによる数値解は一致していた。

2 後退代入

2.1 採用したアルゴリズム

$$x_i = \frac{1}{u_{ii}} \left(c_i - \sum_{k=i+1}^{n-1} u_{ik} x_k \right)$$

として、$i = 2, 1, 0$ の順に x_i を求めた。

2.2 プログラムに関する情報

ファイル名

- 2.go
- backward.go

コンパイルコマンド

go run 2.go print.go backward.go

作成した主な関数

Backward

後退代入法によって方程式の解を求める関数。第一引数として行列（上三角行列）を、第二引数としてベクトル（$b = Ux$ のときの b）を渡すと、方程式の解をベクトルとして返す。

2.3 結果

手計算の結果は、

$$\mathbf{y} = \begin{pmatrix} 2 \\ -3 \\ -2 \end{pmatrix}$$

であった。また、プログラムによる数値解は、

$$\mathbf{y} = \begin{pmatrix} 2.000 \\ -3.000 \\ -2.000 \end{pmatrix}$$

であった。

2.4 考察

手計算による解とプログラムによる数値解は一致していた。

3 行列の計算

3.1 結果

手計算により、

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} 3 & 1 & -3 \\ -3 & -3 & 2 \\ -6 & -8 & 5 \end{pmatrix} \tag{1}$$

と求められた。

4 LU 分解

4.1 採用したアルゴリズム

$$u_{ij} = a_{ij} - \sum_{k=0}^{i-1}$$
$$l_{ij} = \frac{1}{u_{jj}} \left(a_{ij} - \sum_{k=0}^{j-1} l_{jk} ukj \right)$$

として、u_{ij}, l_{ij} を求めた。

このとき、問に与えられた順序に従って計算するのは面倒なので、添字から u_{ij}, l_{ij} を計算する関数をそれぞれ用意し、内部的に再帰呼び出しするように実装した。

4.2 プログラムに関する情報

ファイル名

- 4.go
- lu.go

コンパイルコマンド

go run 4.go print.go lu.go

作成した主な関数

Decomp

matrix.Decomp のように呼び出すと、上三角行列と下三角行列を返す関数。内部的に lower 関数と upper 関数を呼び出し、2次元配列に入れて、それを返すだけの関数。

upper

matrix.upper(i,j) のように呼び出すと、添字（i と j）に対応する上三角行列の要素を返す関数。内部的に lower 関数を呼び出している。

lower

matrix.upper(i,j) のように呼び出すと、添字（i と j）に対応する下三角行列の要素を返す関数。内部的に upper 関数を呼び出している。

4.3 結果

式（**??**）を Decomp 関数により LU 分解した結果、

$$L = \begin{pmatrix} 1.000 & 0.000 & 0.000 \\ -1.000 & 1.000 & 0.000 \\ -2.000 & 3.000 & 1.000 \end{pmatrix}$$

$$U = \begin{pmatrix} 3.000 & 1.000 & -3.000 \\ 0.000 & -2.000 & -1.000 \\ 0.000 & 0.000 & 2.000 \end{pmatrix}$$

となった。

4.4 考察

LU 分解の結果は、式（1）の計算前の式と一致しているため、妥当であると考えられる。

今回作成したプログラムは、lower 関数と upper 関数が呼び出されるたびに、大量の計算が発生する。$N = 11$ 程度まではすぐに計算が終了したが、それより N が大きくなると、プログラムはなかなか終了しなくなった。$N = 12$ 以上のケースに対応するには、メモ化が必要である。

この場合は、lower 関数と upper 関数の計算結果を、それぞれ「添字→計算結果」のハッシュに格納し、すでに計算済みの要素に関しては、ハッシュに格納されている値を返すようにする。これにより、一度計算した要素を計算し直すことがなくなり、計算量は大幅に減る。

実際、問 8 のプログラムの実行に、メモ化なしの場合は $N = 12$ のとき 25 秒ほどかかっていたが、メモ化をした場合は $N = 50$ のときでも 10 秒ほどで終了した。

5 n 元連立一次方程式の解法

5.1 採用したアルゴリズム

以下の手順により、n 元連立 1 次方程式 $\mathbf{Ax} = \mathbf{b}$ の解 $\mathbf{x}$ を求める。

まず、

$$\mathbf{A} = \mathbf{LU}$$

のように LU 分解する。

次に、

$$\mathbf{Ux} = \mathbf{y} \tag{2}$$

と置く。

これにより、

$$\mathbf{Ly} = \mathbf{b}$$

が成立する。これを、前進代入法によって、$\mathbf{y}$ について解く。

最後に、この $\mathbf{y}$ の値を用いて、式（2）を $\mathbf{x}$ について解く。

5.2 プログラムに関する情報

ファイル名

- 5.go
- solve.go

コンパイルコマンド

go run 5.go print.go solve.go lu.go forward.go backward.go

作成した主な関数

Solve

第一引数として行列 $\mathbf{A}$ を、第二引数としてベクトル $\mathbf{b}$ を渡すと、$\mathbf{Ax} = \mathbf{b}$ の解ベクトル $\mathbf{x}$ を返す関数。内部では、問 4 にて作成した Decomp 関数、問 1 で作成した Forward 関数、問 2 で作成した Backward 関数を利用している。

5.3 結果

以下の 3 元連立一次方程式の解をプログラムにより計算した。

$$\begin{pmatrix} 3 & 1 & -3 \\ -3 & -3 & 2 \\ -6 & -8 & 5 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 9.0 \\ -1.0 \\ 2.0 \end{pmatrix}$$

その結果は、

$$\mathbf{x} = \begin{pmatrix} 2.000 \\ -3.000 \\ -2.000 \end{pmatrix}$$

となった。

5.4 考察

上記で求めた解を、問の式に代入して計算すると、

$$\begin{pmatrix} 3 & 1 & -3 \\ -3 & -3 & 2 \\ -6 & -8 & 5 \end{pmatrix} \begin{pmatrix} 2.000 \\ -3.000 \\ -2.000 \end{pmatrix} = \begin{pmatrix} 9.0 \\ -1.0 \\ 2.0 \end{pmatrix}$$

となり、確かに解である事がわかる。

6 n 元連立一次方程式の解を求める

6.1 採用したアルゴリズム

問 5 と同様のアルゴリズムを採用する。

6.2 プログラムに関する情報

6_1.go 及び 6_2.go は N の値を与えているだけである。本質的な実装は 6.go にある。

ファイル名

- 6_1.go
- 6_2.go
- 6.go

コンパイルコマンド

$N = 3$ **の場合**	go run 6_1.go print.go 6.go solve.go lu.go forward.go backward.go
$N = 6$ **の場合**	go run 6_2.go print.go 6.go solve.go lu.go forward.go backward.go

作成した主な関数

main

内部で行列 $\mathbf{H}$ とベクトル $\mathbf{b}$ を計算している。

6.3　結果

$h_{ij} = 0.25^{|i-j|}$ を要素とした行列 $\mathbf{H}$ と、$b_i = 5.0 - 4.0^{i-n+1} - 4.0^{-i}$ を要素とするベクトル $\mathbf{b}$ を用いて、

$$\mathbf{Hx} = \mathbf{b} \tag{3}$$

と表される N 元連立一次方程式の数値解 $\mathbf{x}$ を、$N = 3, 6$ の場合についてそれぞれプログラムで求めた。

その結果は、$N = 3$ のとき、

$$\mathbf{x} = \begin{pmatrix} 3.000 \\ 3.000 \\ 3.000 \end{pmatrix}$$

となり、$N = 6$ のとき、

$$\mathbf{x} = \begin{pmatrix} 3.000 \\ 3.000 \\ 3.000 \\ 3.000 \\ 3.000 \\ 3.000 \end{pmatrix}$$

となった。

6.4　考察

式（3）の解は、要素がすべて 3 のベクトルになるので、得られた数値解はどちらも妥当であると考えられる。

7　逆行列を求めるアルゴリズム

7.1　採用したアルゴリズム

$$\mathbf{b_0} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\mathbf{b_1} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

$$\mathbf{b_2} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

として、

$$\mathbf{Ax_0} = \mathbf{b_0}$$
$$\mathbf{Ax_1} = \mathbf{b_1}$$
$$\mathbf{Ax_2} = \mathbf{b_2}$$

を満たす $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}$ を求める。このとき、

$$\mathbf{A^{-1}} = \begin{pmatrix} x_0 & x_1 & x_2 \end{pmatrix}$$

によって、A^{-1} を求める。

7.2 プログラムに関する情報

ファイル名

- inverse.go

作成した主な関数

Inverse

引数として行列を与えると、その逆行列を返す関数。

setCol

matrix.setCol(j, vector) などと実行する。

第一引数には列番号を、第二引数にはその列に代入する列ベクトルを与える。

この関数を実行すると、呼び出し元の行列 (上の例では matrix 変数) の、j 列目が、vector になる。

idMatrix

引数無しで実行され、N 行 N 列の単位行列を返す関数。

8 逆行列を計算する

8.1 採用したアルゴリズム

問 7 で示したアルゴリズムを採用して逆行列を求める。

8.2 プログラムに関する情報

8_1.go 及び 8_2.go は N の値を与えているだけである。本質的な実装は 8.go にある。

また、出力される行列はそれぞれ、

$$I1 = HH^{-1}$$
$$I2 = H^{-1}H$$

を示している。

ファイル名

- 8_1.go
- 8_2.go
- 8.go

コンパイルコマンド

$N = 3$ の場合

go run 8_1.go print.go 8.go inverse.go solve.go lu.go forward.go backward.go

$N = 6$ の場合

go run 8_2.go print.go 8.go inverse.go solve.go lu.go forward.go backward.go

作成した主な関数

MatMul

第一引数と第二引数に行列をとり、その２つの行列の行列積を返す関数。

8.3 結果

プログラムによる I1,I2 の計算結果は以下のようであった。

$N = 3$ のとき、

$$I1 = \begin{pmatrix} 1.000 & -0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{pmatrix}$$
$$I2 = \begin{pmatrix} 1.000 & 0.000 & 0.000 \\ -0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 1.000 \end{pmatrix}$$

$N = 6$ のとき、

$$I1 = \begin{pmatrix} 1.000 & -0.000 & -0.000 & -0.000 & -0.000 & 0.000 \\ 0.000 & 1.000 & -0.000 & -0.000 & -0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & -0.000 & -0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 & -0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 1.000 \end{pmatrix}$$

$$
I2 = \begin{pmatrix}
1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
-0.000 & 1.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
-0.000 & -0.000 & 1.000 & 0.000 & 0.000 & 0.000 \\
-0.000 & -0.000 & -0.000 & 1.000 & 0.000 & 0.000 \\
-0.000 & -0.000 & -0.000 & -0.000 & 1.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 1.000
\end{pmatrix}
$$

8.4 考察