連続系アルゴリズム演習 II

チャンヴァンサン

０５－１６１０２３

問題 A：LU の実装とテスト

実装コード：

コードの言語：GNU Octave

```
function [p, lu_mat, q] = my_lu(A, is_full)
        n = rows(A);
        if n <= 1
                lu_mat = A;
                p = ones(1,1);
                q = ones(1,1);
        else
                #find max
                [max_rows, max_rows_arg] = max(A(1:n,1:1));
                [max_cols, max_cols_arg] = max(A(1:1, 1:n));
                p = eye(n);
                q = eye(n);
                if max_rows > max_cols || !is_full
                        p(1,1) = 0;
                        p(1,max_rows_arg) = 1;
                        p(max_rows_arg, max_rows_arg) = 0;
                        p(max_rows_arg, 1) = 1;
                else
                        q(1,1) = 0;
                        q(max_cols_arg, 1) = 1;
                        q(max_cols_arg, max_cols_arg) = 0;
                        q(1, max_cols_arg) = 1;
                endif
                A = p * A * q;


                lu_mat = zeros(n,n);
                lu_mat(1,1:n) = A(1, 1:n);
                l = A(2:n,1) / A(1,1);
                lu_mat(2:n, 1) = l;
                A_sub = A(2:n, 2:n) - l * A(1,2:n);
                [sub_p, lu_mat(2:n,2:n), sub_q] = my_lu(A_sub, is_full);


                # transpose calculated rows and columns
                lu_mat(2:n,1) = sub_p * lu_mat(2:n, 1);
                lu_mat(1,2:n) = lu_mat(1, 2:n) * sub_q;
                new_p = eye(n);
                new_p(2:n,2:n) = sub_p;
                p = p * new_p;
                new_q = eye(n);
                new_q(2:n,2:n) = sub_q;
        q = new_q * q;
    endif
endfunction

function x = solve_low(lu_mat, b)
    n = rows(b);
    x = [b(1)];
    for i = 2:n
        x = [x; b(i) - lu_mat(i, 1:i-1) * x];
    endfor
endfunction


function x = solve_up(lu_mat, b)
    n = rows(b);
    x = [b(n) / lu_mat(n,n)];
    for i = n - 1:-1:1
        x = [(b(i) - lu_mat(i, i + 1:n) * x)/lu_mat(i,i) ;x];
    endfor
endfunction


function x = lu_solve(lu_mat, b)
    #solve low left
    z = solve_low(lu_mat, b);
    x = solve_up(lu_mat, z);
endfunction


function x = solve(A, b)
    [p, lu_mat, q] = my_lu (A, false);
    y = lu_solve(lu_mat, p * b);
    x = q * y;
endfunction


function [A, p, l, u, q ] = solve_6()
    n = 6;
    A = zeros(n,n);
    for i = 1:n
        for j = 1:n
            A(i,j) = 1 / (i + j -1);
        endfor
    endfor
    [p,lu_mat,q]=my_lu(A, false);
    l = tril(lu_mat, -1) + eye(n);
    u = triu(lu_mat,0);
endfunction
```

```
function easy_check()
    #generate data
    display("Generating data ...");
    A = [[1 4]; [2 3]];
    b = [11; 12];
    display("A = ");
    display(A);
    display("b = ");
    display(b);

    display("Solving ...");
    x = solve(A, b);

    display("x = ");
    display(x);
endfunction


function check_6(is_full)
    display("Check with A_6 = ");
    if is_full
        display("Full pivoting ...");
    else
        display("Partial pivoting ...");
    endif
    [A, p, l, u,q ] = solve_6();
    display("p * A * q = l * u");
    display(p);
    display("*");
    display(A);
    display("*");
    display(q);
    display(" = ");
    display(l);
    display("*");
    display(u);
    display("<=>");
    display(p * A * q);
    display("=");
    display(l * u);
endfunction


function main()
    easy_check();
    check_6(false);
    check_6(true);
    display("Done!");
```

```
endfunction

main()
```

実装結果：

```
Generating data ...

A =

   1   4
   2   3

b =

   11
   12

Solving ...

x =

   3
   2

Check with A_6 =

Partial pivoting ...

p * A * q = l * u

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

*

   1.000000   0.500000   0.333333   0.250000   0.200000   0.166667
   0.500000   0.333333   0.250000   0.200000   0.166667   0.142857
   0.333333   0.250000   0.200000   0.166667   0.142857   0.125000
   0.250000   0.200000   0.166667   0.142857   0.125000   0.111111
   0.200000   0.166667   0.142857   0.125000   0.111111   0.100000
   0.166667   0.142857   0.125000   0.111111   0.100000   0.090909

*

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

 =

   1.00000   0.00000   0.00000   0.00000   0.00000   0.00000
   0.25000   1.00000   0.00000   0.00000   0.00000   0.00000
   0.50000   0.90000   1.00000   0.00000   0.00000   0.00000
   0.16667   0.71429  -0.93333   1.00000   0.00000   0.00000
   0.20000   0.80000   0.85333  -0.14286   1.00000   0.00000
   0.33333   1.00000   0.56000  -0.21429   1.12500   1.00000

*

   1.00000   0.50000   0.33333   0.25000   0.20000   0.16667
   0.00000   0.08333   0.08889   0.08333   0.07619   0.06944
   0.00000   0.00000   0.00595   0.00992   0.01224   0.01353
   0.00000   0.00000   0.00000   0.00093   0.00190   0.00271
   0.00000   0.00000   0.00000   0.00000  -0.00002  -0.00005
   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000

<=>

   1.000000   0.500000   0.333333   0.250000   0.200000   0.166667
   0.250000   0.200000   0.166667   0.142857   0.125000   0.111111
   0.500000   0.333333   0.250000   0.200000   0.166667   0.142857
   0.166667   0.142857   0.125000   0.111111   0.100000   0.090909
   0.200000   0.166667   0.142857   0.125000   0.111111   0.100000
   0.333333   0.250000   0.200000   0.166667   0.142857   0.125000

=

   1.000000   0.500000   0.333333   0.250000   0.200000   0.166667
   0.250000   0.208333   0.172222   0.145833   0.126190   0.111111
   0.500000   0.325000   0.252619   0.209921   0.180816   0.159361
   0.166667   0.142857   0.113492   0.092857   0.078231   0.067460
   0.200000   0.166667   0.142857   0.125000   0.111111   0.100000
   0.333333   0.250000   0.203333   0.172024   0.149286   0.131944

Check with A_6 =

Full pivoting ...

p * A * q = l * u

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

*

   1.000000   0.500000   0.333333   0.250000   0.200000   0.166667
   0.500000   0.333333   0.250000   0.200000   0.166667   0.142857
   0.333333   0.250000   0.200000   0.166667   0.142857   0.125000
   0.250000   0.200000   0.166667   0.142857   0.125000   0.111111
   0.200000   0.166667   0.142857   0.125000   0.111111   0.100000
   0.166667   0.142857   0.125000   0.111111   0.100000   0.090909

*

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

 =

   1.00000   0.00000   0.00000   0.00000   0.00000   0.00000
   0.25000   1.00000   0.00000   0.00000   0.00000   0.00000
   0.50000   0.90000   1.00000   0.00000   0.00000   0.00000
   0.16667   0.71429  -0.93333   1.00000   0.00000   0.00000
   0.20000   0.80000   0.85333  -0.14286   1.00000   0.00000
   0.33333   1.00000   0.56000  -0.21429   1.12500   1.00000

*

   1.00000   0.50000   0.33333   0.25000   0.20000   0.16667
   0.00000   0.08333   0.08889   0.08333   0.07619   0.06944
   0.00000   0.00000   0.00595   0.00992   0.01224   0.01353
   0.00000   0.00000   0.00000   0.00093   0.00190   0.00271
   0.00000   0.00000   0.00000   0.00000  -0.00002  -0.00005
   0.00000   0.00000   0.00000   0.00000   0.00000   0.00000

<=>

   1.000000   0.500000   0.333333   0.250000   0.200000   0.166667
   0.250000   0.200000   0.166667   0.142857   0.125000   0.111111
   0.500000   0.333333   0.250000   0.200000   0.166667   0.142857
   0.166667   0.142857   0.125000   0.111111   0.100000   0.090909
   0.200000   0.166667   0.142857   0.125000   0.111111   0.100000
   0.333333   0.250000   0.200000   0.166667   0.142857   0.125000

=

   1.000000   0.500000   0.333333   0.250000   0.200000   0.166667
   0.250000   0.208333   0.172222   0.145833   0.126190   0.111111
   0.500000   0.325000   0.252619   0.209921   0.180816   0.159361
   0.166667   0.142857   0.113492   0.092857   0.078231   0.067460
   0.200000   0.166667   0.142857   0.125000   0.111111   0.100000
   0.333333   0.250000   0.203333   0.172024   0.149286   0.131944

Done!
```

課題 B：行列の条件数の計算

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} \Rightarrow A^{-1} = \begin{pmatrix} -3 & 2 \\ 2 & -1 \end{pmatrix}$$

$$|A - \lambda I| = 0 \Rightarrow (1-\lambda)(3-\lambda) - 4 = 0$$

$$\Rightarrow \lambda^2 - 4\lambda - 1 = 0$$

$$\Rightarrow \lambda_1 = 2 + \sqrt{5}$$

$$\lambda_2 = 2 - \sqrt{5}$$

$$\Rightarrow K(A) = \frac{|\lambda_{max}|}{|\lambda_{min}|} = \frac{2+\sqrt{5}}{2-\sqrt{5}} = -9 - 4\sqrt{5}$$

$$B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \Rightarrow B^{-1} = \begin{pmatrix} -2 & 1 \\ 3/2 & -1/2 \end{pmatrix}$$

$$\|B\|_2 = \sqrt{\max \lambda (B^* B)} = \sqrt{\max \lambda \begin{pmatrix} 10 & 14 \\ 14 & 20 \end{pmatrix}} = \sqrt{15 + \sqrt{221}}$$

$$\|B^{-1}\|_2 = \sqrt{\max \lambda (B^{-1*} B^{-1})} = \sqrt{\max \lambda \left( \frac{1}{4} \begin{pmatrix} 25 & -11 \\ 11 & 5 \end{pmatrix} \right)} = \sqrt{\frac{15}{2}}$$

$$\Rightarrow K(B) = \sqrt{\frac{15(15 + \sqrt{221})}{2}} \simeq 14.996$$