Numerical Linear Algebra
Programming Assignment #02

2015-17231
박우정

Exercise 2.9.

MATLAB으로 $A = LU$ factorization with complete pivoting을 구현한 코드는 아래와 같다.

```
A=input('LU분해를 알고 싶은 정사각행렬을 입력하시오: ');
b=input('A의 size에 맞추어 Ax=b 항목의 b를 입력하시오: ');
[n,m]=size(A);
if(n~=m)
    disp('정사각행렬이 아닙니다.');
end
Q=cell(1,n-1); %Permutation matrix Q1 Q2... Q(n-1)
saveA=A;saveb=b;x=zeros(n,1);count=0;
for j=1:n-1
    [m,maxlocs]=max(abs(A(j:n,j:n)));  %maxlocs에 column 단위로 max값들의 위치 저장
    [m,maxloc]=max(m); %찾은 max값들 중에서 제일 큰 값과 그것의 위치 반환. 그 위치는 몇 번째 column인지 알려준다.
    maxc=maxloc; maxr=maxlocs(maxloc);
    maxr=maxr+j-1;
    maxc=maxc+j-1;
    Q{j}=eye(n); %Q 초기화
    A([j,maxr],:)=A([maxr,j],:); %행 교환
    b([j,maxr])=b([maxr,j]);
    A(:,[j,maxc])=A(:,[maxc,j]); %열 교환
    Q{j}(:,[j,maxc])=Q{j}(:,[maxc,j]);
    for i=j+1:n
        m=A(i,j)/A(j,j);
        A(i,j:n)=A(i,j:n)-m*A(j,j:n);
        b(i)=b(i)-m*b(j);
    end %Gauss elimination
end
disp("The result of LU decomposition is: ")
U=A
x(n,1)=b(n)/U(n,n);
for i=n-1:-1:1 %back substitution
    x(i)=(b(i)-dot(U(i,i+1:n),x(i+1:n)))/U(i,i);
end
y=zeros(n,1); %temporary variable
for j=n-1:-1:1
    for k=1:n
        y(k)=Q{j}(k,:)*x;
    end
    x=y;
end
%solution 검산
check1=zeros(n,1); check2=zeros(n,1);
for i=1:n
    check1(i,1)=saveA(i,:)*x;
    check2(i,1)=saveb(i);
end
for i=1:n
    if(abs(check1(i,1)-check2(i,1))>10^-13) %한계를 10^-13 으로 설정.
        count=count+1;
    end
end
if(count~=0)
    disp("This solution is incorrect")
else
    disp("This solution is correct")
end
```

실제로 어떤 행렬을 대입하여 계산해보면 다음과 같은 결과를 얻는다.

```
>> complete_pivot
LU분해를 알고 싶은 정사각행렬을 입력하시오:  [2 3 4:4 7 5:4 9 5]
A의 size에 맞추어 Ax=b 항목의 b를 입력하시오: [1 0 0]
The result of LU decomposition is:

U =

    9.0000    5.0000    4.0000
         0    2.3333    0.6667
         0         0    0.5714

This solution is correct
```

간단한 예시를 통해 decomposition의 결과와 solution의 정확도까지 알아낼 수 있었다.
한편, Gauss Elimination with Partial Pivoting을 위한 Matlab 코드는 다음과 같다.

```
%Partial pivoting
A=input('LU분해를 알고 싶은 정사각행렬을 입력하시오: ');
b=input('A의 size에 맞추어 Ax=b 항목의 b를 입력하시오: ');
saveA=A;saveb=b;x=1:n;count=0;
[n,m]=size(A);
if(n~=m)
    disp('정사각행렬이 아닙니다.');
    return;
end
for j=1:n
    [x,jsave]=max(abs(A(j:n,j))); %가장 큰 row의 위치 저장
    js=jsave+j-1; %실제 위치 반영
    A([j,js],:)=A([js,j],:); %행 교환
    b([j,js])=b([js,j]);
    for i=j+1:n %Gauss Elimination
        m=A(i,j)/A(j,j);
        A(i,:)=A(i,:)-m*A(j,:);
        b(i)=b(i)-m*b(j);
    end
end
x(n)=b(n)/A(n,n);
for i=n-1:-1:1 %back substitution
    x(i)=(b(i)-dot(A(i,i+1:n),x(i+1:n)))/A(i,i);
end
disp("pivoting하고 gauss elimination까지 한 결과")
disp("patial pivoting upper triangular is: ")
U=A
b
disp("the solution is: ")
x
%검산
check1=zeros(n,1); check2=zeros(n,1);
for i=1:n
    check1(i,1)=saveA(i,:)*x';
    check2(i,1)=saveb(i);
end
for i=1:n
    if(abs(check1(i,1)-check2(i,1))>10^-13) %한계를 10^-13 으로 설정.
        count=count+1;
    end
end
if(count~=0)
    disp("This solution is incorrect")
else
    disp("This solution is correct")
end
```

$A=\begin{pmatrix}1 & 10^{10} & 10^{10}\\ 1 & 10 & 1\\ 1 & 2 & 10^{10}\end{pmatrix}$, $b=\begin{pmatrix}1\\ 1\\ 10^{10}\end{pmatrix}$을 두 프로그램에 적용하면 각각 다른 결과를 얻을 수 있는데, 두 가지를 모두 직접 실행해본 결과가 아래와 같다.

```
>> partial_pivot
LU분해를 알고 싶은 정사각행렬을 입력하시오: [1
10^10 10^10;1 10 1; 1 2 10^10]
A의 size에 맞추어 Ax=b 항목의 b를 입력하시오:
[1 1 10^10]
pivoting하고 gauss elimination까지 한 결과
patial pivoting upper triangular is:

U =

   1.0e+10 *

    0.0000    1.0000    1.0000
         0   -1.0000         0
         0         0   -1.0000


b =

   1.0e+09 *

    0.0000   10.0000  -10.0000

the solution is:

x =

   10.0000   -1.0000    1.0000

This solution is incorrect
```

```
>> complete_pivot
LU분해를 알고 싶은 정사각행렬을 입력하시오: [1
10^10 10^10;1 10 1; 1 2 10^10]
A의 size에 맞추어 Ax=b 항목의 b를 입력하시오:
[1 1 10^10]
The result of LU decomposition is:

U =

   1.0e+10 *

    1.0000    1.0000    0.0000
         0    1.0000    0.0000
         0         0    0.0000

This solution is correct
```

따라서 pivoting의 방법에 따라 solution의 accuracy가 다르다는 것을 확인할 수 있다.

Exercise 2.10.

MATLAB으로 $A = LU$ factorization with *Rook's* pivoting을 구현한 코드는 다음과 같다.

```
A=input('LU분해를 알고 싶은 정사각행렬을 입력하시오: ');
b=input('A의 size에 맞추어 Ax=b 항목의 b를 입력하시오: ');
[n,m]=size(A);
if(n~=m)
    disp('정사각행렬이 아닙니다.');
    return;
end
Q=cell(1,n-1); %Permutation matrix Q1 Q2... Q(n-1)
saveA=A;saveb=b;x=zeros(n,1);count=0;
for j=1:n-1
    Q{j}=eye(n);
    [mx,js]=max(abs(A(j:n,j)));
    [mx,ks]=max(abs(A(j,j:n)));
    js=js+j-1;
    ks=ks+j-1;
    if (abs(A(js,j))>=abs(A(j,ks))) %row쪽이 큰 경우
        ks=j;
        A([j,js],:)=A([js,j],:);
        b([j,js])=b([js,j]);
    else %column이 큰 경우
        js=j;
        A(:,[j,ks])=A(:,[ks,j]);
        Q{j}(:,[j,ks])=Q{j}(:,[ks,j]);
    end
    for i=j+1:n
        m=A(i,j)/A(j,j);
        A(i,j:n)=A(i,j:n)-m*A(j,j:n);
        b(i)=b(i)-m*b(j);
    end %Gauss elimination
end
disp("The result of LU decomposition is: ")
U=A
x(n,1)=b(n)/U(n,n);
for i=n-1:-1:1
    x(i)=(b(i)-dot(U(i,i+1:n),x(i+1:n)))/U(i,i);
end
y=zeros(n,1); %temporary variable
for j=n-1:-1:1
    for k=1:n
        y(k)=Q{j}(k,:)*x;
    end
    x=y;
end
%solution 검산
check1=zeros(n,1); check2=zeros(n,1);
for i=1:n
    check1(i,1)=saveA(i,:)*x;
    check2(i,1)=saveb(i);
end
for i=1:n
    if(abs(check1(i,1)-check2(i,1))>10^-13)
        count=count+1;
    end
end
if(count~=0)
    disp("This solution is incorrect")
else
    disp("This solution is correct")
end
```

이제 $A=\begin{pmatrix}1 & 1 & 10^{12}\\ 1 & 10 & 1\\ 1 & 1 & 1\end{pmatrix}, b=\begin{pmatrix}10^{10}\\ 1\\ 1\end{pmatrix}$을 partial pivoting program과 Rook's pivoting program에 대입해보면 다음과 같은 결과를 얻을 수 있다.

```
>> partial_pivot
LU분해를 알고 싶은 정사각행렬을 입력하시오: [1
1 10^12;1 10 1; 1 1 1]
A의 size에 맞추어 Ax=b 항목의 b를 입력하시오:
[10^10 1 1]

pivoting하고 gauss elimination까지 한 결과
patial pivoting upper triangular is:

U =

   1.0e+12 *

    0.0000    0.0000    1.0000
         0    0.0000   -1.0000
         0         0   -1.0000


b =

   1.0e+10 *

    1.0000   -1.0000   -1.0000

the solution is:

x =

    0.9900         0    0.0100

This solution is incorrect
```

```
>> Rook_pivoting
LU분해를 알고 싶은 정사각행렬을 입력하시오: [1
1 10^12;1 10 1; 1 1 1]
A의 size에 맞추어 Ax=b 항목의 b를 입력하시오:
[10^10 1 1]
The result of LU decomposition is:

U =

   1.0e+12 *

    1.0000    0.0000    0.0000
         0    0.0000    0.0000
         0         0    0.0000

This solution is correct
```

따라서, Partial Pivoting으로 발생하는 차이를 Rook's Pivoting이 보완해줄 수 있다.

Exercise 2.11.

MATLAB으로 Crout algorithm(for n*n, symmetric and positive-difinite matrix)을 구현한 코드는 다음과 같다.

```
n=input('What is the dimension n? ');
A=zeros(n,n);b=zeros(n,1);l=eye(n);u=eye(n);
saveb=zeros(n,1);
saveA=zeros(n,n);
for j=1:n %문제 조건을 만족하는 행렬 만들기
    A(j,j)=2;
end
for j=1:n-1
    A(j,j+1)=-1;
    A(j+1,j)=-1;
end
b(1)=1;
saveA=A;
saveb=b;
l(:,1)=A(:,1);
u(1,:)=A(1,:)/l(1,1);
u(1,1)=1;
for k=2:n %Crout Algorithm
for j=2:n
    for i=j:n
        l(i,j)=A(i,j)-dot(l(i,1:j-1),u(1:j-1,j));
    end
    u(k,j)=(A(k,j)-dot(l(k,1:k-1),u(1:k-1,j)))/l(k,k);
end
end
% decomposition 확인
check=zeros(n,n);
count = 0;
for i = 1 : n
    for j = 1 : n
        check(i,j)=l(i,:)*u(:,j);
        if(abs(saveA(i,j) - check(i,j)) >5*10^-13)
            count = count + 1;
        end
    end
end
if(count ~= 0)
    disp("incorrect LU decomposition");
else
    disp("correct LU decomposition");
end
for j=1:n
    b(j)=(b(j)-dot(l(j,1:j-1),b(1:j-1)))/l(j,j); %forward substitution
end
for j=n:-1:1
    b(j)=(b(j)-dot(u(j,j+1:n),b(j+1:n)))/u(j,j); %backward substitution
end
for j=1:n
if(abs(saveb(j)-dot(saveA(j,1:n),b(1:n)))>5*10^-13)
    count=count+1;
end
end
if(count~=0)
    fprintf('The computed solution seems to be wrong at %f \n',j);
end
disp('The solution x is as follows: ')
disp(b)
```

이제 $n = 100$을 대입해보면, 결과는 다음과 같다.

```
>> Crout_algorithm
What is the dimension n?
100
correct LU decomposition
The solution x is as
follows:
    0.9901
    0.9802
    0.9703
    0.9604
    0.9505
    0.9406
    0.9307
    0.9208
    0.9109
    0.9010
    0.8911
    0.8812
    0.8713
    0.8614
    0.8515
    0.8416
    0.8317
    0.8218
    0.8119
    0.8020
    0.7921
    0.7822
    0.7723
    0.7624
    0.7525
    0.7426
    0.7327
    0.7228
    0.7129
    0.7030
    0.6931
    0.6832
    0.6733
    0.6634
    0.6535
    0.6436
    0.6337
    0.6238
    0.6139
    0.6040
    0.5941
    0.5842
    0.5743
    0.5644
    0.5545
    0.5446
    0.5347
    0.5248
    0.5149
    0.5050
    0.4950
    0.4851
    0.4752
    0.4653
    0.4554
    0.4455
    0.4356
    0.4257
    0.4158
    0.4059
    0.3960
    0.3861
    0.3762
    0.3663
    0.3564
    0.3465
    0.3366
    0.3267
    0.3168
    0.3069
    0.2970
    0.2871
    0.2772
    0.2673
    0.2574
    0.2475
    0.2376
    0.2277
    0.2178
    0.2079
    0.1980
    0.1881
    0.1782
    0.1683
    0.1584
    0.1485
    0.1386
    0.1287
    0.1188
    0.1089
    0.0990
    0.0891
    0.0792
    0.0693
    0.0594
    0.0495
    0.0396
    0.0297
    0.0198
    0.0099
```

이전에 해결했던 Exercise 2.7.과 같은 결과를 얻었으므로, 알맞게 프로그래밍되었다고 할 수 있다.

Exercise 2.12.

MATLAB으로 Doolittle's algorithm을 구현한 코드는 다음과 같다.

```
h=input('how many meshes do you want? Please enter the # of partition h(=1/N): ');
n=1/h;n=n-1;%n등분하면 구간의 개수는 n-1개
A=zeros(n,n);b=zeros(n,1);l=eye(n);u=eye(n);
saveb=zeros(n,1);saveA=zeros(n,n);
for j=1:n %필요한 행렬 생성. 이 때, u(0)=0, u(1)=0을 가정하고, 후에 1을 더하여 해를 구한다.
    A(j,j)=2;
end
for j=1:n-1
    A(j,j+1)=-1;
    A(j+1,j)=-1;
end
for k=1:n
    b(k)=exp(sin(k/(n+1)));
end
saveA=A;
saveb=b;
for j=1:n %Doolittle_algorithm
    for k=j:n
    u(j,k)=A(j,k)-dot(l(j,1:j-1),u(1:j-1,k));
    end
    for k=j+1:n
    l(k,j)=(A(k,j)-dot(l(k,1:j-1),u(1:j-1,j)))/u(j,j);
    end
end
check=zeros(n,n); %decomposition 확인
count = 0;
for i = 1 : n
    for j = 1 : n
        check(i,j)=l(i,:)*u(:,j);
        if(abs(saveA(i,j) - check(i,j)) >5*10^-13)
            count = count + 1;
        end
    end
end
if(count ~= 0)
    disp("incorrect LU decomposition");
else
    disp("correct LU decomposition");
end
for j=1:n %해 구하기. 아래는 forward substitution
    b(j)=(b(j)-dot(l(j,1:j-1),b(1:j-1)))/l(j,j);
end
for j=n:-1:1 %back substitution
    b(j)=(b(j)-dot(u(j,j+1:n),b(j+1:n)))/u(j,j);
end
disp('The solution x is as follows: ')
b=b*(h^2)+1; %최종적으로 boundary condition을 만족시키기 위해 1을 더해야 한다.
disp(b)
```

이제 $h = \frac{1}{16}, \frac{1}{32}, \frac{1}{64}$ 그리고 $\frac{1}{128}$을 대입해보자. 실행한 결과는 다음과 같다.

```
>> Doolittle_algorithm_2_12
how many meshes do you want?
Please enter the # of partition
h(=1/N): 1/16
correct LU decomposition
The solution x is as follows:
    1.0419
    1.0797
    1.1130
    1.1416
    1.1653
    1.1836
    1.1963
    1.2030
    1.2034
    1.1971
    1.1839
    1.1632
    1.1349
    1.0984
    1.0536
```

```
>>                      1.1653
Doolittle_algorith      1.1751
m_2_12                  1.1836
how many                1.1907
meshes do you           1.1963
want? Please            1.2004
enter the # of          1.2030
partition               1.2040
h(=1/N): 1/32           1.2034
correct LU              1.2011
decomposition           1.1971
The solution x          1.1914
is as follows:          1.1839
    1.0215              1.1745
    1.0419              1.1632
    1.0613              1.1500
    1.0797              1.1349
    1.0969              1.1177
    1.1130              1.0984
    1.1279              1.0771
    1.1417              1.0536
    1.1541              1.0279
```

```
>>
Doolittle_algorithm
_2_12
how many meshes
do you want?
Please enter the #
of partition
h(=1/N): 1/64
correct LU
decomposition
The solution x is
as follows:
    1.0109
    1.0215
    1.0318
    1.0419
    1.0518
    1.0613
    1.0707
    1.0797
    1.0885
    1.0969
    1.1051
    1.1130
    1.1206
    1.1280
    1.1350
    1.1417
    1.1480
    1.1541
    1.1599
    1.1653
    1.1704
    1.1751
    1.1795
    1.1836
    1.1873
    1.1907
    1.1937
    1.1963
    1.1985
    1.2004
    1.2019
    1.2030
    1.2037
    1.2040
    1.2039
    1.2034
    1.2025
    1.2011
    1.1993
    1.1971
    1.1945
    1.1914
    1.1879
    1.1839
    1.1794
    1.1745
    1.1691
    1.1632
    1.1569
    1.1500
    1.1427
    1.1349
    1.1265
    1.1177
    1.1083
    1.0984
    1.0880
    1.0771
    1.0656
    1.0536
    1.0410
    1.0279
    1.0142
```

```
>>
Doolittle_algorithm
_2_12
how many meshes
do you want?
Please enter the #
of partition
h(=1/N): 1/128
correct LU
decomposition
The solution x is
as follows:
    1.0055
    1.0109
    1.0162
    1.0215
    1.0267
    1.0318
    1.0369
    1.0419
    1.0469
    1.0518
    1.0566
    1.0613
    1.0660
    1.0707
    1.0752
    1.0797
    1.0841
    1.0885
    1.0927
    1.0969
    1.1011
    1.1051
    1.1091
    1.1130
    1.1169
    1.1206
    1.1243
    1.1280
    1.1315
    1.1350
    1.1384
    1.1417
    1.1449
    1.1481
    1.1511
    1.1541
    1.1570
    1.1599
    1.1626
    1.1653
    1.1679
    1.1704
    1.1728
    1.1751
    1.1774
    1.1795
    1.1816
    1.1836
    1.1855
    1.1873
    1.1890
    1.1907
    1.1922
    1.1937
    1.1950
    1.1963
    1.1975
    1.1985
    1.1995
    1.2004
    1.2012
    1.2019
    1.2025
    1.2030
    1.2034
    1.2037
    1.2039
    1.2040
    1.2040
    1.2039
    1.2037
    1.2034
    1.2030
    1.2025
    1.2019
    1.2011
    1.2003
    1.1993
    1.1983
    1.1971
    1.1959
    1.1945
    1.1930
    1.1914
    1.1897
    1.1879
    1.1859
    1.1839
    1.1817
    1.1794
    1.1770
    1.1745
    1.1719
    1.1691
    1.1662
    1.1632
    1.1601
    1.1569
    1.1535
    1.1500
    1.1464
    1.1427
    1.1389
    1.1349
    1.1308
    1.1265
    1.1222
    1.1177
    1.1131
    1.1083
    1.1034
    1.0984
    1.0933
    1.0880
    1.0826
    1.0771
    1.0714
    1.0656
    1.0597
    1.0536
    1.0474
    1.0410
    1.0345
    1.0279
    1.0211
    1.0142
    1.0072
```