Numerical Linear Algebra
Programming Assignment #04

2015-17231
박우정

Exercise 2.17.

MATLAB으로 문제에서 요구하는 프로그램을 짠 코드는 다음과 같다.

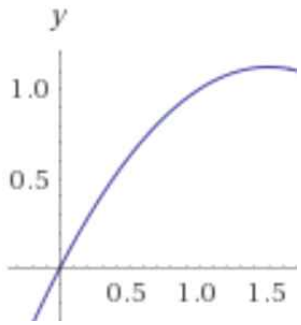
```
n=input('What is the dimension n? ');
A=zeros(n,n);D=zeros(n,n);L=zeros(n);M=zeros(n);
r=zeros(1,n);w=zeros(1,n);%For reducing flops into half, I use Algorithm 6.2. the reduced version of
decomposition. r saves D(p,p)*M(j,p), and w saves L(j,p)*D(p,p) where p=1, 2, 3, ... , j-1.
check1=zeros(n,n);check2=zeros(n,n);
for j=1:n
    A(j,j)=2;
end
for j=1:n-1
    A(j,j+1)=-1;
    A(j+1,j)=-1;
end
saveA=A;
for j=1:n %A=LDM^t decomposition.
    for p=1:j-1
        r(p)=D(p,p)*M(j,p);
        w(p)=L(j,p)*D(p,p);
    end
    D(j,j)=A(j,j)-dot(w(1:j-1),M(j,1:j-1));
    for k=j+1:n
        L(k,j)=(A(k,j)-dot(L(k,1:j-1),r(1:j-1)))/D(j,j);
        M(k,j)=(A(k,j)-dot(w(1:j-1),M(k,1:j-1)))/D(j,j);
    end
end
for i=1:n
    L(i,i)=1;
    M(i,i)=1;
end
disp("The LDM^t decomposition result is: ")
L=L
D=D
M=M
for i=1:n %check whether the inverse is correct within the tolerance 10e-10
    for j=1:n
        check1(i,j)=L(i,:)*D(:,j);
    end
end
M=M';
for i=1:n
    for j=1:n
        check2(i,j)=check1(i,:)*M(:,j);
        if(abs(check2(i,j)-saveA(i,j))>10^-10)
            disp("This inverse is incorrect")
        end
    end
end
```

허용오차는 10^-10으로 주었고, $n = 100$으로 input을 주어 실행하면 다음을 얻는다.

(dimension이 너무 크므로 상세한 모습은 생략) 허용오차를 넘는 값이 전혀 출력되지 않았으므로, 알맞게 프로그래밍되었다고 할 수 있다.

Exercise 2.19

MATLAB을 이용하여 방정식 $-u''(x)=1, x \in (0,1); u(0)=0, u(1)=1$의 Numerical solution을 구해보자. 사실 이 differential equation은 analytic solution을 바로 구할 수 있고, 그것은 $u(x)=-\frac{1}{2}x^2+\frac{3}{2}x$이다. 따라서 우리는 다음과 같은 수치해를 얻기를 기대한다.



전체적인 solution을 구하는 과정을 다음과 같이 요약할 수 있다.

step 1.

Initial value가 $u(0)=0, u(1)=1$이므로, initial value가 일치하게 하는 '좋은' 함수를 먼저 찾는다. $y(x)=u(x)-x$라 하면, $y(0)=y(1)=0$이므로, 원하는 '좋은' 함수가 된다.

step 2.

$LDM^t y=b$를 해결하기 위해 다음과 같은 순서를 지키도록 한다.

a) $DM^t y=y_1$이라 정의하고, $Ly_1=b$를 만족하는 y_1을 구한다.

b) $M^t y=y_2$라 정의하고, $Dy_2=y_1$을 만족하는 y_2를 구한다.

c) 최종적으로 $M^t y=y_2$를 만족하는 y를 구한다.

step 3.

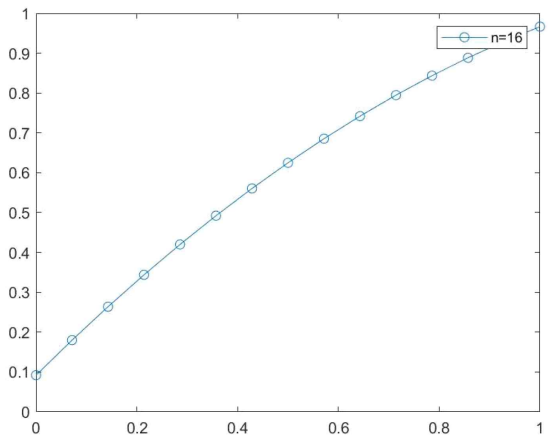
step 2에서 y를 구했으므로, $u(x)=y(x)+x$를 구한다.

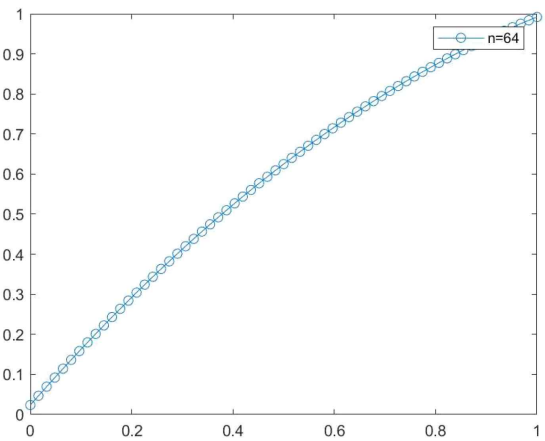
이제 위 순서에 따라 MATLAB으로 프로그래밍을 짜면 다음과 같다.

```
n=input('What is the dimension n? ');
%WANT : solve -u''(x)=1
%When considering the initial value, u(0)=0, u(1)=1, we may start to solve
%-y''(x)=1 when y(x)=u(x)-x. Then initial values of y is : y(0)=y(1)=0;
h=1/n;n=n-1;y=zeros(n,1);b=ones(n,1);
for i=1:n
    b(i,1)=h^2*b(i,1);
end
%Decompose tridiagonal matrix to LDM^t
A=zeros(n,n);D=zeros(n,n);L=zeros(n);M=zeros(n);
r=zeros(1,n);w=zeros(1,n);
for j=1:n
    A(j,j)=2;
end
for j=1:n-1
    A(j,j+1)=-1;
    A(j+1,j)=-1;
end
saveA=A;
for j=1:n
    for p=1:j-1
        r(p)=D(p,p)*M(j,p);
        w(p)=L(j,p)*D(p,p);
    end
    D(j,j)=A(j,j)-dot(w(1:j-1),M(j,1:j-1));
    for k=j+1:n
        L(k,j)=(A(k,j)-dot(L(k,1:j-1),r(1:j-1)))/D(j,j);
        M(k,j)=(A(k,j)-dot(w(1:j-1),M(k,1:j-1)))/D(j,j);
    end
end
for i=1:n
    L(i,i)=1;
    M(i,i)=1;
end
%Now, let's find the solution. Put (D M^t y)=y1, then the equation is equivalent to Ly1=b. Find y1 by forward
substitution.
y1=zeros(n,1);
y1(1)= b(1)/L(1,1);
for j = 2 : n
y1(j) = (b(j) - dot(L(j,1:j-1), y1(1:j-1,1)))/L(j,j);
end
%Next, put (M^t y)=y2, then the equation is: Dy2=y1. Find y2. It is very
%easy since D is diagonal matrix.
y2=zeros(n,1);
for j=1:n
    y2(j)=y1(j)/D(j,j);
end
%At last, the equation is M^t * y=y2, and M^t is upper triangular matrix, so we must find y using back substitution.
M=M'
y(n) = y2(n) %M(n,n)=1, so we don't need to divide it.
for j = (n-1) : -1 : 1
y(j) = (y2(j) - dot(M(j,j+1:n), y(j+1:n,1)))/M(j,j);
end
%The final solution is u(x)=y(x)+x, and each y(i) in matlab means y(i/n)
%where n is the dimension. For n=n-1 above, we should code like;
u=zeros(n,1);
for i=1:n
    u(i)=y(i)+i/(n+1);
end
u
%Note that if n is sufficiently large, u(0) is almost 0 and u(1) is almost 1.
range = linspace(0,1,n);
title('Exercise 2_19','fontsize',12,'fontname','arial');
xlabel('x');
ylabel('solution');
plot(range,u,'-o');
```

이제, $n = 16, 64$로 input을 준 뒤, 실행하면 각각 다음과 같은 결과를 얻을 수 있다.

```
>> LDMt_equation_2_19
What is the dimension n? 16

u =

    0.0918
    0.1797
    0.2637
    0.3438
    0.4199
    0.4922
    0.5605
    0.6250
    0.6855
    0.7422
    0.7949
    0.8438
    0.8887
    0.9297
    0.9668

>> LDMt_equation_2_19
What is the dimension n? 64

u =

    0.0233
    0.0464
    0.0692
    0.0918
    0.1141
    0.1362
    0.1581
    0.1797
    0.2010
    0.2222
    0.2430
    0.2637
    0.2841
    0.3042
    0.3241
    0.3437
    0.3632
    0.3823
    0.4012
    0.4199
    0.4384
    0.4565
    0.4745
    0.4922
    0.5096
    0.5269
    0.5438
    0.5605
    0.5770
    0.5933
    0.6093
    0.6250
    0.6405
    0.6558
    0.6708
    0.6855
    0.7001
    0.7144
    0.7284
    0.7422
    0.7557
    0.7690
    0.7821
    0.7949
    0.8075
    0.8198
    0.8319
    0.8437
    0.8553
    0.8667
    0.8778
    0.8887
    0.8993
    0.9097
    0.9198
    0.9297
    0.9393
    0.9487
    0.9579
    0.9668
    0.9755
    0.9839
    0.9921
```





Numerical solution이 원하는 바와 유사하므로, 알맞게 프로그래밍되었다고 할 수 있다.