Numerical Linear Algebra
Programming Assignment #05

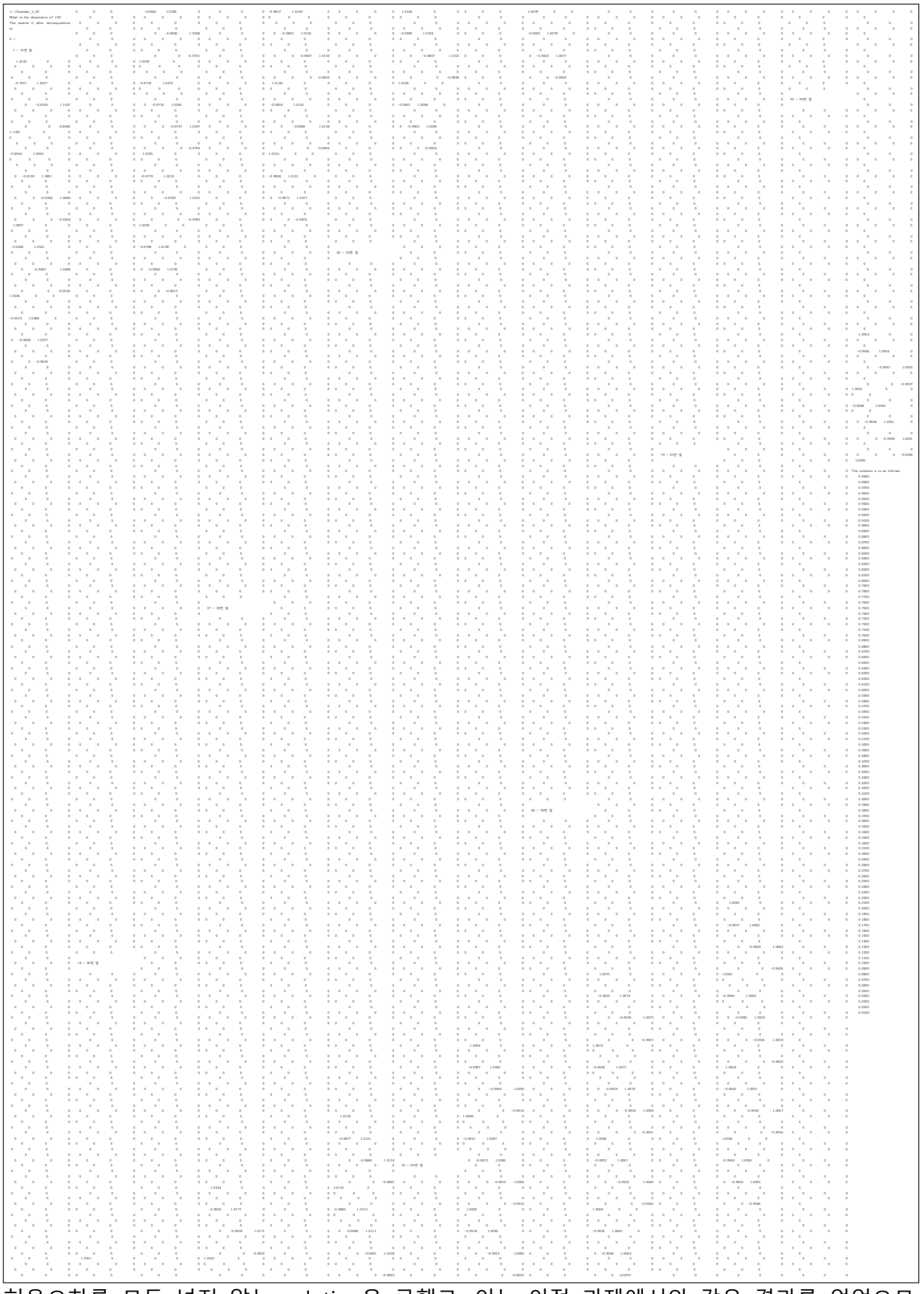
2015-17231
박우정

Exercise 2.20.

MATLAB으로 Cholesky decomposition을 실행하는 프로그래밍 코드는 다음과 같다.

```
n=input('What is the dimension n? ');
n=n-1;
A=zeros(n,n);b=zeros(n,1); G=zeros(n,n);
saveb=zeros(n,1);saveA=zeros(n,n); count=0;
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
b(1)=1;
saveA=A;
saveb=b;
for i=1:n %Cholesky decomposition
   G(i,i)=sqrt(A(i,i)-dot(G(i,1:i-1),G(i,1:i-1)));
   for j=(i+1):n
      G(j,i)=(A(j,i) -dot(G(j,1:i-1),G(i,1:i-1)))/G(i,i);
   end
end
disp("The matrix G after decomposition is: ")
G
for j=1:n %Solve Gy=b where y=G'x
    b(j)=(b(j)-dot(G(j,1:j-1),b(1:j-1)))/G(j,j); %forward substitution
end
G=G';
for j=n:-1:1 %Solve G'x=y
    b(j)=(b(j)-dot(G(j,j+1:n),b(j+1:n)))/G(j,j); %backward substitution overwriting.
end
for j=1:n %solution check
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)
```

허용오차는 5*10^-13으로 주었고, $n = 100$으로 input을 주어 실행하면 다음을 얻는다.



허용오차를 모두 넘지 않는 solution을 구했고, 이는 이전 과제에서와 같은 결과를 얻었으므로 알맞게 프로그래밍 되었다고 할 수 있다.

Exercise 2.23

프로그래밍으로 주어진 미분방정식을 풀어보자. 먼저

$$f(x) = -c'(x)\cos(\pi x) + \pi c(x)\sin(\pi x)$$

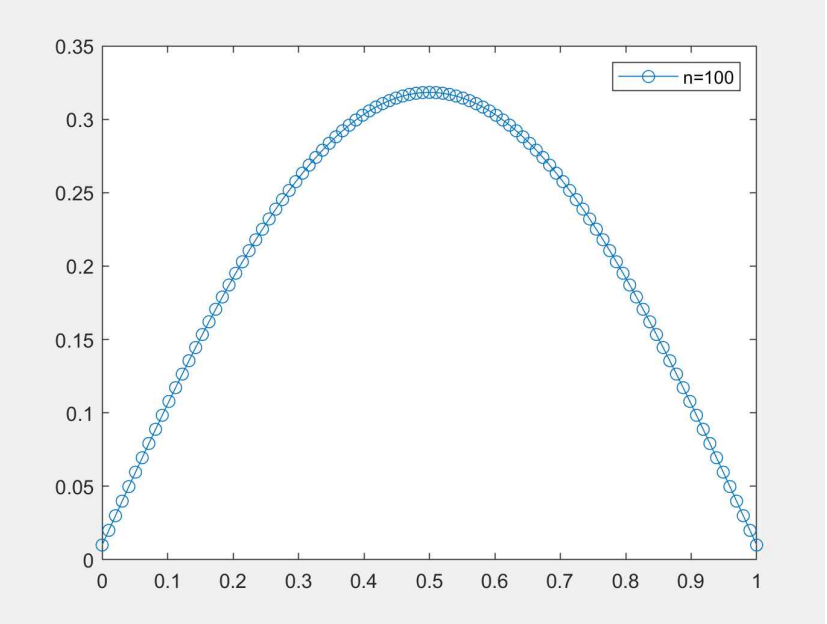
로 풀어 쓴 뒤에, 프로그래밍 구문에 적용해야 한다. 위 계산결과와 Exercise 2.22의 Matrix representation을 참고하여 MATLAB 구문을 작성하면 아래와 같다.

```
n=input('What is the dimension n? ');
h=1/n;n=n-1;
A=zeros(n,n);b=zeros(n,1);c=zeros(n+1,1);x=zeros(n,1); %c must have n+1 dimension.
saveb=zeros(n,1);saveA=zeros(n,n);savec=zeros(n+1,1);
for k=1:n
    b(k)=h^2*(-(pi)*cos(2*pi*h*k)+(2*pi)*sin(pi*h*k));
    c(k)=sin(pi*h*(2*k-1)/2)+2;
end
c(n+1)=sin(pi*h*(2*n+1)/2)+2; %We need one more dimension.
for j=1:n %Matrix for solving the equation.
    A(j,j)=c(j+1)+c(j);
end
for j=1:n-1
    A(j,j+1)=-c(j+1);
    A(j+1,j)=-c(j+1);
end
saveA=A;saveb=b;
for j=1:n-1 %Gauss elimination
    for k=j+1:n
        m=A(k,j)/A(j,j);
        A(k,j:n)=A(k,j:n)-m*A(j,j:n);
        b(k)=b(k)-m*b(j);
    end
end
for j=n:-1:1 %find solution(overwriting b) by back substitution.
    b(j)=(b(j)-dot(A(j,j+1:n),b(j+1:n)))/A(j,j);
end
for j=1:n
if(abs(saveb(j)-dot(saveA(j,1:n),b(1:n)))>5*10^-13)
    fprintf('The computed solution seems to be wrong at %f \n',j);
end
end
disp('The solution x is as follows: ')
disp(b)
range = linspace(0,1,n);
title('Exercise 2_23','fontsize',12,'fontname','arial');
xlabel('x');
ylabel('solution');
plot(range,b,'-o');
legend('n=100')
```

이제 문제의 요구대로 $n = 100$을 input으로 주어 실행하면 다음 결과를 얻는다.

```
>                          >     0.1951    0.3177    0.2251
finite_difference_schem          0.2029    0.3182    0.2179
e_2_23                           0.2105    0.3183    0.2105
What is the dimension n?         0.2179    0.3182    0.2029
100                              0.2251    0.3177    0.1951
The  solution  x  is  as         0.2321    0.3169    0.1871
follows:                         0.2388    0.3158    0.1789
    0.0100                       0.2453    0.3144    0.1706
    0.0200                       0.2515    0.3127    0.1620
    0.0300                       0.2575    0.3107    0.1534
    0.0399                       0.2633    0.3083    0.1445
    0.0498                       0.2688    0.3057    0.1355
    0.0597                       0.2740    0.3028    0.1264
    0.0694                       0.2790    0.2995    0.1172
    0.0792                       0.2836    0.2960    0.1078
    0.0888                       0.2880    0.2922    0.0984
    0.0984                       0.2922    0.2880    0.0888
    0.1078                       0.2960    0.2836    0.0792
    0.1172                       0.2995    0.2790    0.0694
    0.1264                       0.3028    0.2740    0.0597
    0.1355                       0.3057    0.2688    0.0498
    0.1445                       0.3083    0.2633    0.0399
    0.1534                       0.3107    0.2575    0.0300
    0.1620                       0.3127    0.2515    0.0200
    0.1706                       0.3144    0.2453    0.0100
    0.1789                       0.3158    0.2388
    0.1871                       0.3169    0.2321
```

한편, 이 미분방정식은 Analytic solution $u(x)=\frac{1}{\pi}\sin(\pi x)$을 가진다는 사실을 쉽게 알 수 있고, 다음 그래프로부터 위 수치해가 잘 프로그래밍 되었음을 확인할 수 있다.

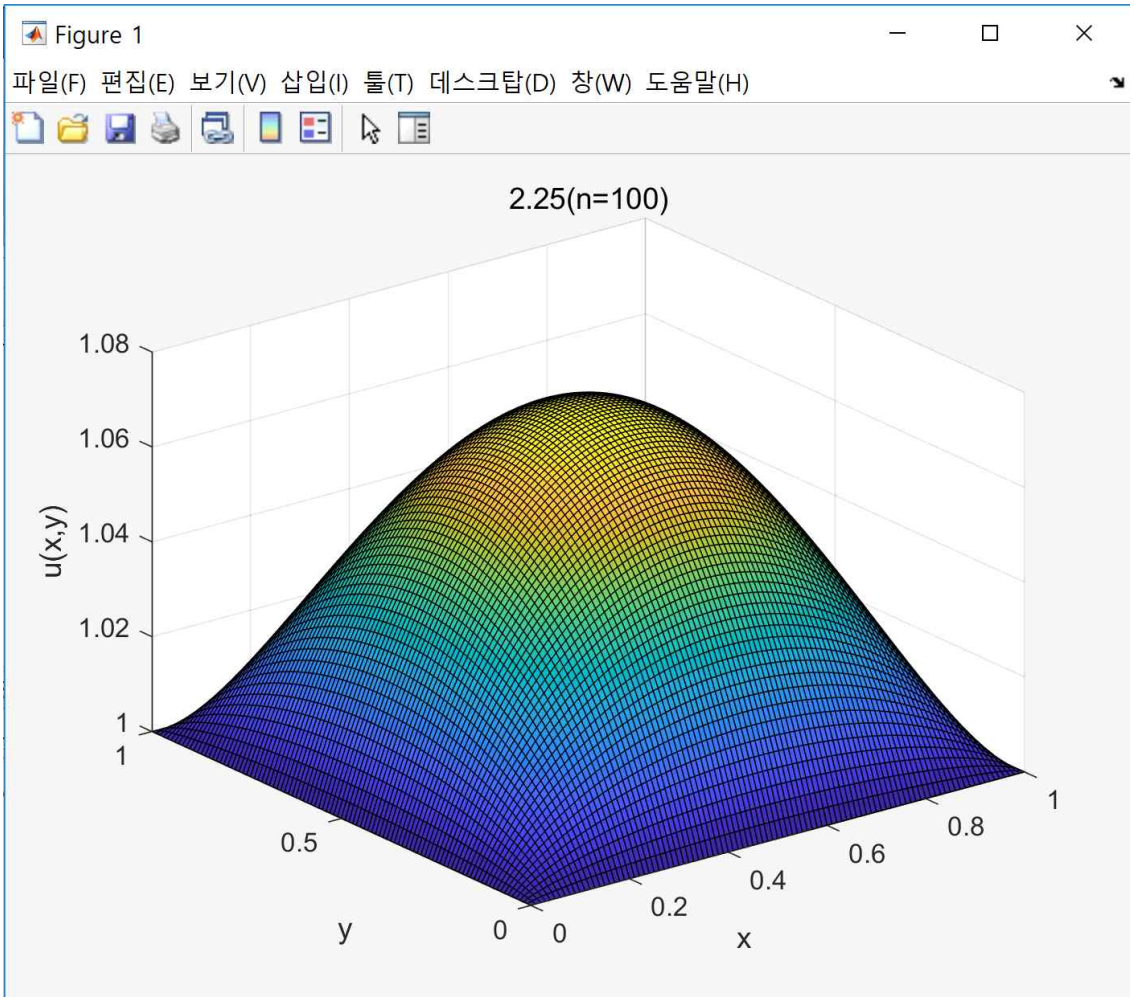


Exercise 2.25

주어진 방정식 2D-finite difference method를 LU분해로 풀어보자. 구문은 다음과 같다.

```
n=input('What is the dimension n? ');
h=1/n;n=n-1;nn=n^2;
A=zeros(nn,nn);L=eye(nn,nn);
b=zeros(nn,1);u=zeros(nn,1); %v=u-1, then the initial values corresponds to 1.
for i=1:nn
    A(i,i)=4;
    b(i)=1*h^2;
end
for i=0:n-1
    for j=1:n-1
        A(n*i+j,n*i+j+1)=-1;
        A(n*i+j+1,n*i+j)=-1;
    end
end
for i=1:nn-n
    A(i,i+n)=-1;
    A(i+n,i)=-1;
end
saveA=A;
saveb=b;
for j=1:nn-1 %LU-decomposition without pivoting(overwriting A=U)
    ml=min(j+n,nn);
    L(j+1:ml,j)=A(j+1:ml,j)/A(j,j);
    for k=j+1:ml
        mu=min(j+n,nn);
        A(k,j:mu)=A(k,j:mu)-L(k,j)*A(j,j:mu);
    end
end
disp("The LU decomposition is: ")
!disp(L);
!disp(A);
y=zeros(nn,1);%forward elimination for solving Ly=b
y(1)=b(1);
for j=2:nn
    k=max(1,j-n);
    y(j)=b(j)-dot(L(j,k:j-1),y(k:j-1,1));
end
x=zeros(nn,1);%back substitution Ux=y
for j=nn:-1:1
    k=min(j+n,nn);
    x(j)=(y(j)-dot(A(j,j+1:k),x(j+1:k,1)))/A(j,j);
end
for j=1:nn %check the solution is correct;
if(abs(saveb(j)-dot(saveA(j,:),x(:)))>5*10^-13)
    fprintf('The computed solution seems to be wrong at %f \n',j);
end
end
x=x+1; %At last, for finding u=x+1, where v is the solution we find under the assumption. So, we add 1 !
n=n+2; %When we divide with meshes n, we get n+1 points. Don't forget that we started this program with n=n-1.
u=zeros(n);
u(1,:)=1;u(n,:)=1;u(:,1)=1;u(:,n)=1; %boundary condition
for i=2:n-1
    for j=2:n-1
      u(i,j)=x((n-2)*(i-2)+j-1); %rearrange the solution from vector x to matrix u(i,j)
    end
end
%Let's check plot!
[X,Y]=meshgrid(0:h:1);
surf(X,Y,u);
xlabel('x');
ylabel('y');
zlabel('u(x,y)');
title('2.25(n=100)');
```

특히 boundary 조건에 맞추어 [0,1]*[0,1] 정사각형의 테두리에 값을 1로 갖도록 해준다. 이제, $n = 100$으로 input을 준 뒤, 실행하면 다음과 3차원 그래프를 얻을 수 있다. decomposition의 형태와 solution의 경우는 개수가 매우 많으므로 생략한다.



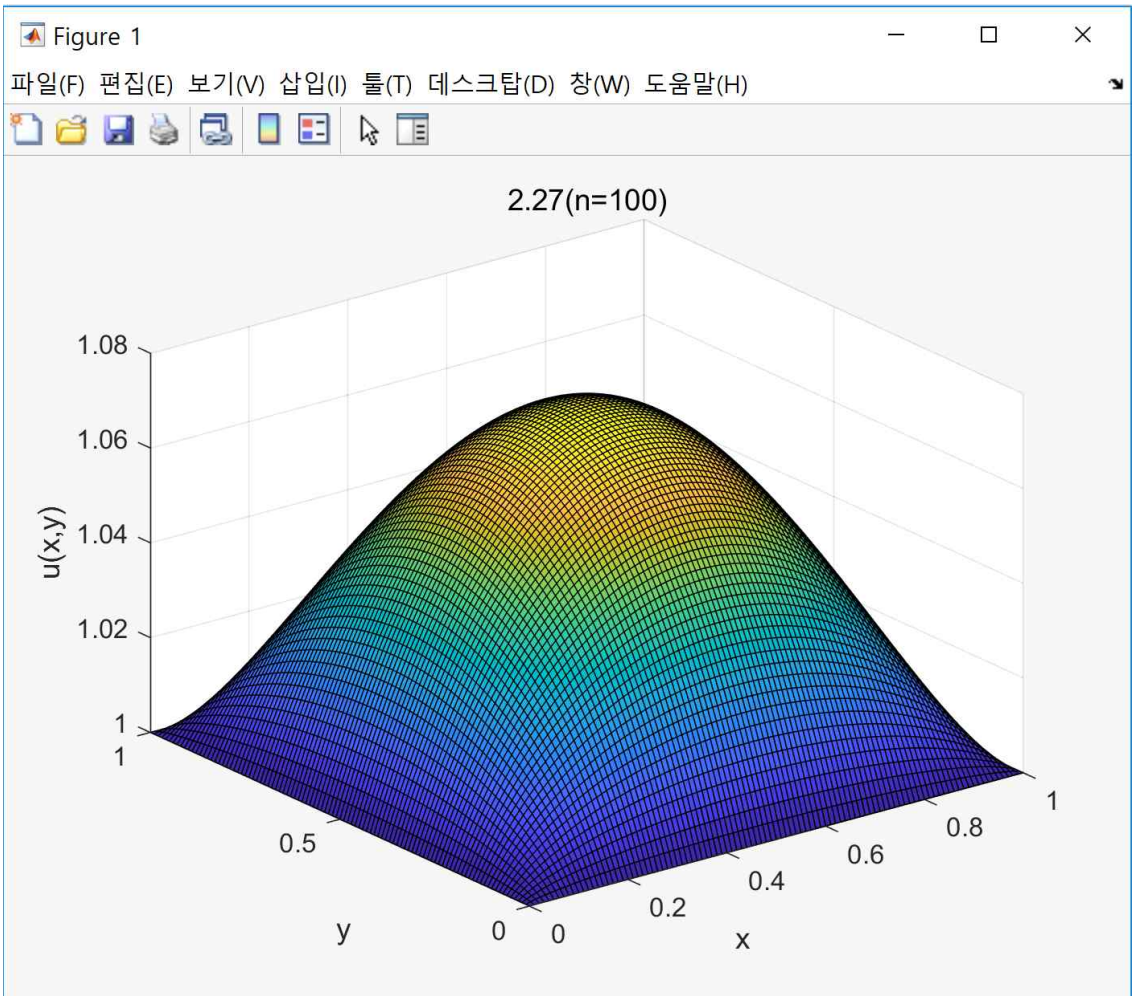
허용오차를 넘는 solution이 print되지 않았고, 따라서 올바르게 프로그래밍되었다고 할 수 있다.

Exercise 2.27

Cholesky decomposition을 이용하여 2.25의 문제를 다시 풀어보자. 필요한 MATLAB 구문은 아래와 같다.

```
n=input('What is the dimension n? ');
h=1/n;n=n-1;nn=n^2;
A=zeros(nn,nn);L=eye(nn,nn);
b=zeros(nn,1);u=zeros(nn,1); %v=u-1, then the initial values corresponds to 1.
for i=1:nn
    A(i,i)=4;
    b(i)=1*h^2;
end
for i=0:n-1
    for j=1:n-1
        A(n*i+j,n*i+j+1)=-1;
        A(n*i+j+1,n*i+j)=-1;
    end
end
for i=1:nn-n
    A(i,i+n)=-1;
    A(i+n,i)=-1;
end
saveA=A;
saveb=b;
G=zeros(nn);
for j=1:nn %Cholesky decomposition
    mj=max(j-n,1);
    G(j,j)=sqrt(A(j,j)-dot(G(j,mj:j-1),G(j,mj:j-1)));
   for k=j+1:min(nn,j+n)
      mk=max(k-n,1);
      G(k,j)=(A(k,j)-dot(G(k,mk:j-1),G(j,mk:j-1)))/G(j,j);
   end
end
y=zeros(nn,1);%forward elimination for solving Gy=b
y(1)=b(1)/G(1,1);
for j=2:nn
    k=max(1,j-n);
    y(j)=(b(j)-dot(G(j,k:j-1),y(k:j-1,1)))/G(j,j);
end
x=zeros(nn,1);%back substitution G'x=y
H=G';
x(nn)=y(nn)/H(nn,nn);
for j=(nn-1):-1:1
    k=min(j+n,nn);
    x(j)=(y(j)-dot(H(j,j+1:k),x(j+1:k,1)))/H(j,j);
end
for j=1:nn %check the solution is correct;
if(abs(saveb(j)-dot(saveA(j,:),x(:)))>5*10^-13)
    fprintf('The computed solution seems to be wrong at %f \n',j);
end
end
x=x+1; %At last, for finding u=x+1, where v is the solution we find under the assumption. So, we add 1 !
n=n+2; %When we divide with meshes n, we get n+1 points. Don't forget that we started this program with n=n-1.
u=zeros(n);
u(1,:)=1;u(n,:)=1;u(:,1)=1;u(:,n)=1; %boundary condition
for i=2:n-1
    for j=2:n-1
       u(i,j)=x((n-2)*(i-2)+j-1); %rearrange the solution from vector x to matrix u(i,j)
    end
end
%Let's check plot!
[X,Y]=meshgrid(0:h:1);
surf(X,Y,u);
xlabel('x');
ylabel('y');
zlabel('u(x,y)');
title('2.27(n=100)');
```

exercise 2.25와 조금 더 빠른 속도로, 같은 결과를 얻어낼 수 있었다. 다음 3차원 그래프를 보면 그 결과가 완전히 일치함을 확인할 수 있다.



Exercise 2.28

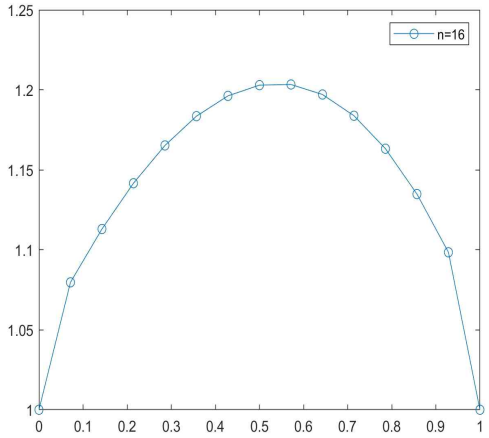
Exercise 2.12를 tridiagonal matrix를 푸는 데 매우 효율적인 알고리즘인 Thomas algorithm을 이용하여 compact program을 작성해보자. 필요한 MATLAB 구문은 아래와 같다.

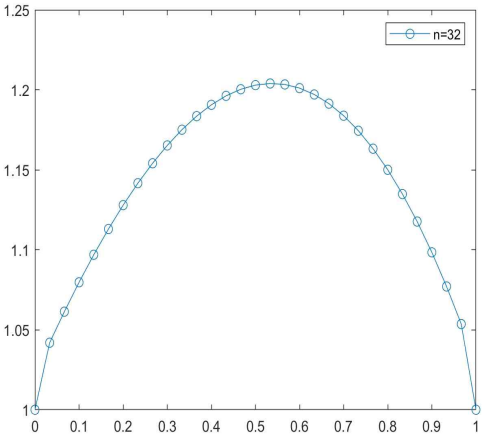
```
h=input('how many meshes do you want? Please enter the # of partition h(=1/N): ');
n=1/h;n=n-1;
A=zeros(n,n);b=zeros(n,1);al=zeros(n,1);au=zeros(n,1);ad=zeros(n,1);
saveb=zeros(n,1);saveA=zeros(n,n);
for j=1:n %필요한 행렬 생성. 이 때, u(0)=0, u(1)=0을 가정하고, 후에 1을 더하여 해를 구한다.
    A(j,j)=2;
end
for j=1:n-1
    A(j,j+1)=-1;
    A(j+1,j)=-1;
end
for k=1:n
    b(k)=h^2*exp(sin(k/(n+1)));
end
saveA=A;
saveb=b;
for j=2:n %Thomas algorithm setting
    al(j)=-1;
    au(j-1)=-1;
    ad(j)=2;
end
ad(1)=2;
for j=2:n %Thomas algorithm
    m=al(j)/ad(j-1); al(j)=m;
    ad(j)=ad(j)-m*au(j-1);
    b(j)=b(j)-m*b(j-1);
end
b(n)=b(n)/ad(n);%back substitution, overwriting on b
for j=n-1:-1:1
    b(j)=(b(j)-au(j)*b(j+1))/ad(j);
end
for j=1:n %check the solution is correct;
    if(abs(saveb(j)-dot(saveA(j,:),b(:)))>5*10^-13)
    fprintf('The computed solution seems to be wrong at %f \n',j);
    end
end
disp('The solution x is as follows: ')
b=b+1; %최종적으로 boundary condition을 만족시키기 위해 1을 더한다. 초기값 설정
b(1)=1;b(n)=1;
disp(b)

range = linspace(0,1,n);
title('Exercise 2_28','fontsize',12,'fontname','arial');
xlabel('x');
ylabel('solution');
plot(range,b,'-o');
legend('n=128')
```

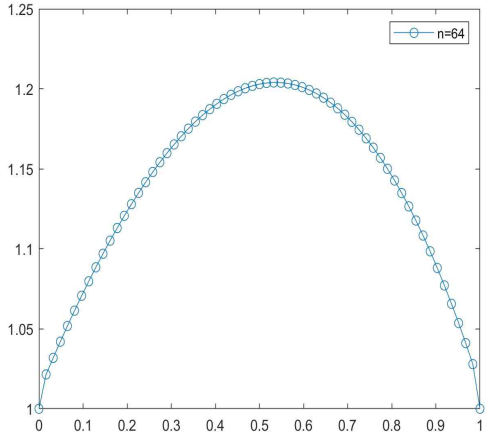
이제 문제에서 요구하는 대로 $n = 16, 32, 64, 128$을 대입하여 각각의 결과를 구해보면 다음을 얻는다.

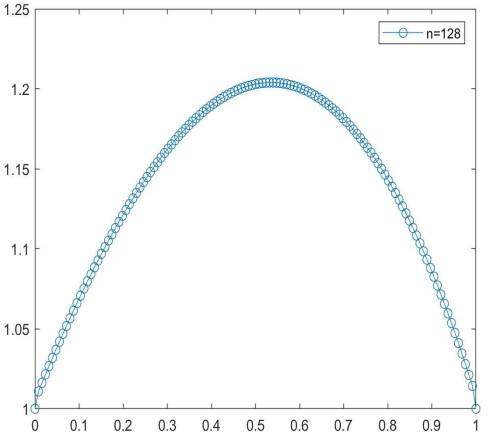
```
> > exercise_2_12_cpt_ver
how many meshes do you want?
Please enter the # of partition
h(=1/N): 1/16
The solution x is as follows:
    1.0000
    1.0797
    1.1130
    1.1416
    1.1653
    1.1836
    1.1963
    1.2030
    1.2034
    1.1971
    1.1839
    1.1632
    1.1349
    1.0984
    1.0000

> > exercise_2_12_cpt_ver
how many meshes do you want?
Please enter the # of partition
h(=1/N): 1/32
The solution x is as follows:
    1.0000
    1.0419
    1.0613
    1.0797
    1.0969
    1.1130
    1.1279
    1.1417
    1.1541
    1.1653
    1.1751
    1.1836
    1.1907
    1.1963
    1.2004
    1.2030
    1.2040
    1.2034
    1.1971
    1.1839
    1.1632
    1.1349
    1.0984
    1.0536
    1.0000

> > exercise_2_12_cpt_ver
how many meshes do you want?
Please enter the # of partition
h(=1/N): 1/64
The solution x is as follows:
    1.0000
    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
    1.0000

> > exercise_2_12_cpt_ver
how many meshes do you want?
Please enter the # of partition
h(=1/N): 1/128
The solution x is as follows:
    1.0000
    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
    1.0000
```

단 양 끝값에서의 경우, boundary condition을 만족시키기 위해 강제로 대입하여 맞춰주었다. 위 solution들은 Exercise 2.12의 Doolittle's algorithm으로 구한 해와 같고, 그림으로 비교해보아도 그 결과는 명백하다.


n=16
1.25
1.2
1.15
1.1
1.05
1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1


n=32
1.25
1.2
1.15
1.1
1.05
1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1


n=64
1.25
1.2
1.15
1.1
1.05
1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1


n=128
1.25
1.2
1.15
1.1
1.05
1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1