INDIAN INSTITUTE OF TECHNOLOGY

KHARAGPUR

APPLIED COMPUTATIONAL METHODS LABORATORY

Lab-2

Rajiv Harlalka

20MA20073

Department of Mathematics

LU_PP Function

```
% Code for the algorithm 1

%LU Factorisation; GEPP Constructing, P,L,U such that PA = LU

function [L,P,U] = LU__PP(A)

[n,n] = size(A);

P = eye(n);

L = eye(n);

U = A;

for i=1: n-1

   [pivot, m] = max(abs(U(i:n, i)));

   m = m+i-1;

   if m~=i

      % swap rows m and i in P

      temp = P(i,:);

      P(i,:) = P(m,:);

      P(m,:) = temp;

      % swap rows m and i in P

      temp = P(i,:);

      P(i,:) = P(m,:);

      P(m,:) = temp;

      % swap rows m and i in P

         if i>=2

            temp = L(i,1:i-1);

            L(i,1:i-1) = L(m,1:i-1);

            L(m,1:i-1) = temp;

         end

   end

   L(i+1:n,i) = U(i+1:n,i)/U(i,i);

   U(i+1:n, i+1:n) = U(i+1:n, i+1:n) - L(i+1:n, i)*U(i,i+1:n);

   U(i+1:n,i) = 0;

End
```

GEPP 2D Poisson Function

```
% number of inner nodes

n=20;

% A, Af

a_amp = 12;

f_amp = 1;

x_0 = 0.5;

y_0 = 0.5;

c_x = 1;

c_y = 1;

h = 1/(n+1);



% Generating Matrix A

S = DiscretePoisson2D(n);


% LU Factorisation of A (S) with pivoting

[L,U,P] = LU__PP_function(S);


% Generate coefficient a((x_1)_i,(x_2)_j) = a(i*h,j*h)

C = zeros(n,n);

for i=1:n

   for j = 1:n

       C(i,j) = 1 + a_amp*exp(-((i*h-x_0)^2/(2*c_x^2)+(j*h-y_0)^2/(2*c_y^2)));

   end

end


% calculate the load vector

% if f is constant,  f = af_amp*ones(n^2,1);

% if f is  Gaussian function

f = zeros(n^2,1);

for i = 1:n

   for j = 1:n

       f(n*(i-1)+j) = f_amp*exp(-((i*h-x_0)^2/(2*c_x^2)+(j*h-y_0)^2/(2*c_y^2)));

   end

end
```

```
% solving LSE Au = (1/h^2)DLUu = f

% create diagonal matrix D from C

D = zeros(n^2,n^2);

for i = 1:n

   for j = 1:n

      D(j+n*(i-1),j+n*(i-1)) = C(i,j);

   end

end



% compute vector on RHS

%  b = D^(-1)*f given by b(i,j) = f(i,j)/a(i,j)


b = zeros(n^2,1);

for i = 1:n

   for j = 1:n

      b(n*(i-1)+j) = f(n*(i-1)+j)/C(i,j);

   end

end




% To solve 1/h^2Au = b, use first LU decomposition

% 1/h^2LUu = b, then compute v =  L^(-1)*b by forward substitution

v = ForwSub_function(L,P*b);


% Now solve 1/h^2Uu = v

% Compute w = U^(-1)*v by backward substitution

w = BackwSub_function(U,v);



%  4th step 1/h^2u=w

% compute final solution as u = h^2w


u = h^2*w;
```

```
%GEPP_2D_Poisson.m (main program)

%plot and figures

%sort the data in u into the mesh-grid

z=zeros(n+2,n+2);

for i=1:n

   for j=1:n

       z(i+1,j+1) = U(j+n*i-1);

   end

end


% Plotting

x1 = 0:h:1;

y1 = 0:h:1;

figure(1)

surf(x1,y1,z) % 3D plot of solution

view(2)

colorbar

xlabel('x_1')

ylabel('x_2')

zlabel('u(x_1,x_2)')

title(['u(x_1,x_2) with A = ',num2str(a_amp),', N = ',num2str(n)])


figure(2)

surf(x1,y1,z) % same plot

%view(z)

colorbar

xlabel('x_1')

ylabel('x_2')

zlabel('u(x_1,x_2)')

title(['u(x_1,x_2) with A = ',num2str(a_amp),', N = ',num2str(n)])


% Plotting a(x,y)

z_a = zeros(n+2);

for i = 1:(n+2)

   for j = 1:(n+2)

       z_a(i,j) = 1 + a_amp*exp(-((i*h-x_0)^2/(2*c_x^2)+(j*h-y_0)^2/(2*c_y^2)));

   end
```

```
end


figure(3)

surf(x1,y1,z_a) % 3D plot of a(x_1,x_2)

xlabel('x_1')

ylabel('x_2')

zlabel('a(x_1,x_2)')

title(['a(x_1,x_2) with A = ',num2str(a_amp)])


% plot of f(x,y)

z_f = zeros(n+2);

for i = 1:n

   for j = 1:n

       z_f(i,j) = f_amp*exp(-((x1(i)-x_0)^2/(2*c_x^2) + (y1(j)-y_0)^2/(2*c_y^2)));

   end

end


figure(4)

surf(x1,y1,z_f) % 3D plot of f(x_1,x_2)*a(x_1,x_2)

xlabel('x_1')

ylabel('x_2')

zlabel('f(x_1,x_2)')
```

Forward Sub Function

```
function x = ForwSub(L,b)

s = size(L);

n = s(1);

x = zeros(n,1);

x(1) = b(1)/L(1,1);

for i=2:n

    x(i) = (b(i) - L(i,1:i-1)*x(1:(i-1)))/L(i,i);

end

end
```

Backward Sub Function

```
function x = BackwSub(U,b)

s = size(U);
```

```
n = s(1);

x = zeros(n,1);

x(n) = b(n)/U(n,n);

for i=n-1:-1:1

   x(i) = (b(i) - U(i,(i+1):n)*x((i+1):n))/U(i,i);

end

end
```

Discrete Poisson 2D Function

```
% Constructs A for 2D discretization of - laplace operator

function A=DiscretePoisson2D(n)

% input parameter n, number of inner nodes, same in both directions

A=zeros(n*n,n*n);

% main diagonal

for i=1:n*n

A(i,i)=4;

end

% 1st and 2nd off-diagonals

for k=1:n % go through blocks 1 to n

   for i=1:(n-1)

   A(n*(k-1)+i,n*(k-1)+i+1)=-1;

   A(n*(k-1)+i+1,n*(k-1)+i)=-1;

   end

end

%3rd and 4th off-diagonals

for i=1:n*(n-1)

A(i,i+n)=-1;

A(i+n,i)=-1;

end

end
```

Output

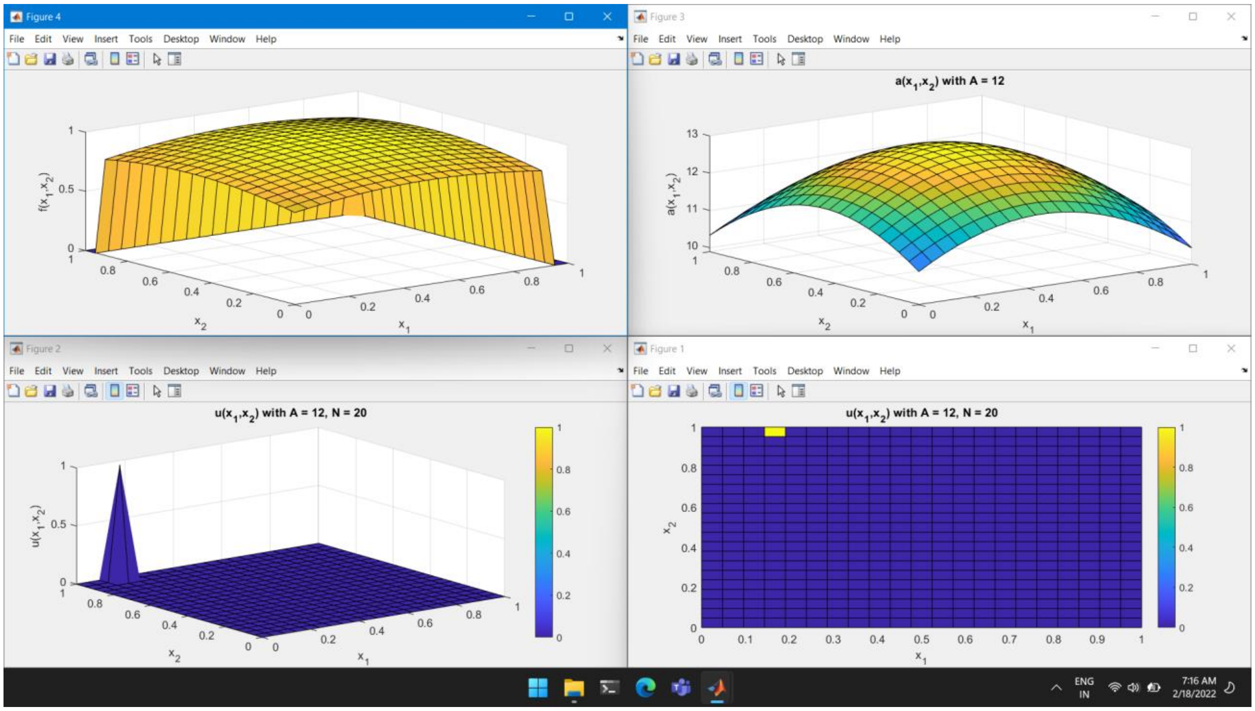

Figure 4
File Edit View Insert Tools Desktop Window Help
$f(x_1,x_2)$
x_2
x_1
Figure 3
$a(x_1,x_2)$ with A = 12
$a(x_1,x_2)$
Figure 2
$u(x_1,x_2)$ with A = 12, N = 20
$u(x_1,x_2)$
Figure 1
$u(x_1,x_2)$ with A = 12, N = 20
ENG
IN
7:16 AM
2/18/2022