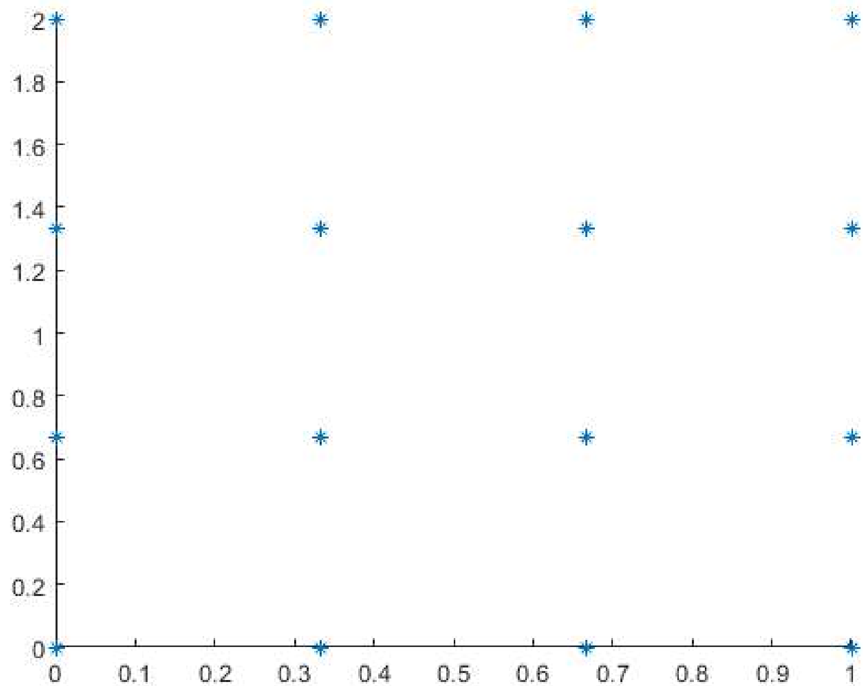
Contents

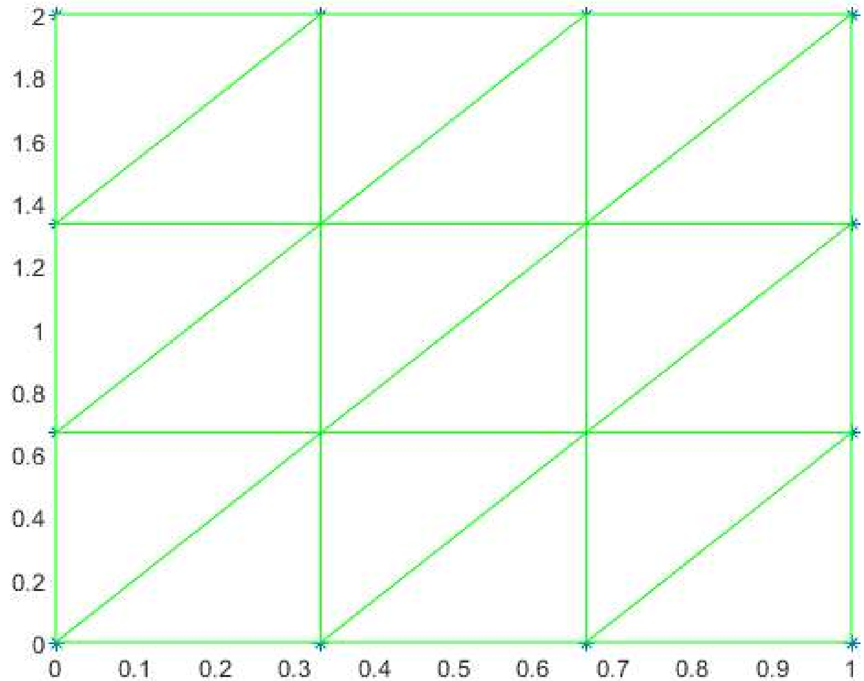
2D Triangular mesh

```
clc
close all
clear
%Geometric dimension of plate
a=1;
b=2;
Xp=4;
Yp=4;
k11=1;
k22=1;
k12=0;
k21=0;
f=10;
Nel=2*(Xp-1)*(Yp-1);
Nnode=Xp*Yp;
dx=a/(Xp-1);

dy=b/(Yp-1);
co=zeros((Xp*Yp),2);

% lets start by writing
% for co-ordinates of the nodes
index=0;
for i=1:Yp
    y=(i-1)*(dy);
    for j=1:Xp
        x=(j-1)*dx;
        index=index+1;
        co(index,1)=x;
        co(index,2)=y;
    end
end
hold on
plot(co(:,1),co(:,2),'*');
```



for conectivity information

```
cn=zeros(Nel,3);  % initiation of variable for connectivity information
% cn(element,:)
index=0;
for i=1:Xp-1
    for j=1:Yp-1
        id1=j+(i-1)*Xp;
        id2=id1+1;
        id3=id1+Xp;
        id4=id2+Xp;
        index=index+1;
        cn(index,:)=[id1,id2,id4];
        index=index+1;
        cn(index,:)=[id4,id3,id1];
    end
end

 patch('faces',cn,'Vertices',co,'facecolor','w','edgecolor','g')
```



for jacobian calculation of every element

```
J=zeros(2,2,Nel);    %Initialisation for jacobian
I_J=zeros(2,2,Nel);
d_J=zeros(Nel,1);

for i=1:Nel

    ld=cn(1,:);
    lnd1=ld(1,1); % local node point 1
    lnd2=ld(1,2); % local node point 2
    lnd3=ld(1,3); % local node point 3
    j11=(co(lnd2,1)-co(lnd1,1));
    j12=(co(lnd3,1)-co(lnd1,1));
    j21=(co(lnd2,2)-co(lnd1,2));
    j22=(co(lnd3,2)-co(lnd1,2));
    dummy=[j11,j12;j21,j22];
    J(:,:,i)=dummy;                 % jacobian for a perticular element
    I_J(:,:,i)=inv(dummy);          % Inverse of jacobian for that element
    d_J(i)=det(dummy);              % Determinant of jacobian for each element

end
```

shape function for element

```
K_l=zeros(3,3,Nel);
F_l=zeros(3,1,Nel);

% defining the derevatives of shapefunctions in principal Co-ordinate
% systems
% as Np_1=1-jai-eta;
% Np_2=jai;
% Np_3=eta;
% del(N1^)/del(jai)=-1
```

syms jai eta

```
syms jai eta

Np(1,1)=1-jai-eta;
Np(1,2)=jai;
Np(1,3)=eta;

% similer for others, hence creating a list
rNp_jai=[-1,1,0];
rNp_eta=[-1,0,1];

% Now defining the partial derevatives of Shape functions in physical
% Co-ordinate system
% rN_x(i,1)=del(N_i)/del(x),rN_x(j,1)=del(N_j)/del(x)
% rN_y(i,1)=del(N_i)/del(y),rN_y(j,1)=del(N_j)/del(y)

for k=1:Nel          % loop for iteration over each element
    for i=1:3          %

        F_l(i,1,k)=f*d_J(k)/6;

        rN_x(i,k)=rNp_jai(1,i)*I_J(1,1,k)+rNp_eta(1,i)*I_J(2,1,k);
        rN_y(i,k)=rNp_jai(1,i)*I_J(1,2,k)+rNp_eta(1,i)*I_J(2,2,k);
        for j=1:3      % loops for iteration inside an element
            rN_x(j,k)=rNp_jai(1,j)*I_J(1,1,k)+rNp_eta(1,j)*I_J(2,1,k);
            rN_y(j,k)=rNp_jai(1,j)*I_J(1,2,k)+rNp_eta(1,j)*I_J(2,2,k);

            % stiffness matrix for each element
            K_l(i,j,k)=0.5*(d_J(k))*(rN_x(i,k)*(k11*rN_x(j,k)+k12*rN_y(j,k))+rN_y(i,k)*(k21*rN_x(j,k)+k22*rN_y(j,k)));

        end
    end
end
```

Assembly process

```
p=1;
pl=(p+1)*(p+2)/2;
K=zeros(Nnode,Nnode);
F=zeros(Nnode,1);

for k=1:Nel
    for i=1:pl
        I=cn(k,i);
        F(I,1)=F(I,1)+F_l(i,1,k);
        for j=1:pl
            J=cn(k,j);
            K(I,J)=K(I,J)+K_l(i,j,k);
        end
    end
end
```

Tempering Matrix for dirichlet conditions

edge info

```
fdof=[1;2;3;4;5;8;9;12;13;14;15;16];
[num,c]=size(fdof);
for i=1:num
    namedof=fdof(i);
    val=0;
    K(namedof,namedof)=1;
    F(namedof,1)=val;
    for j=1:Nnode
```

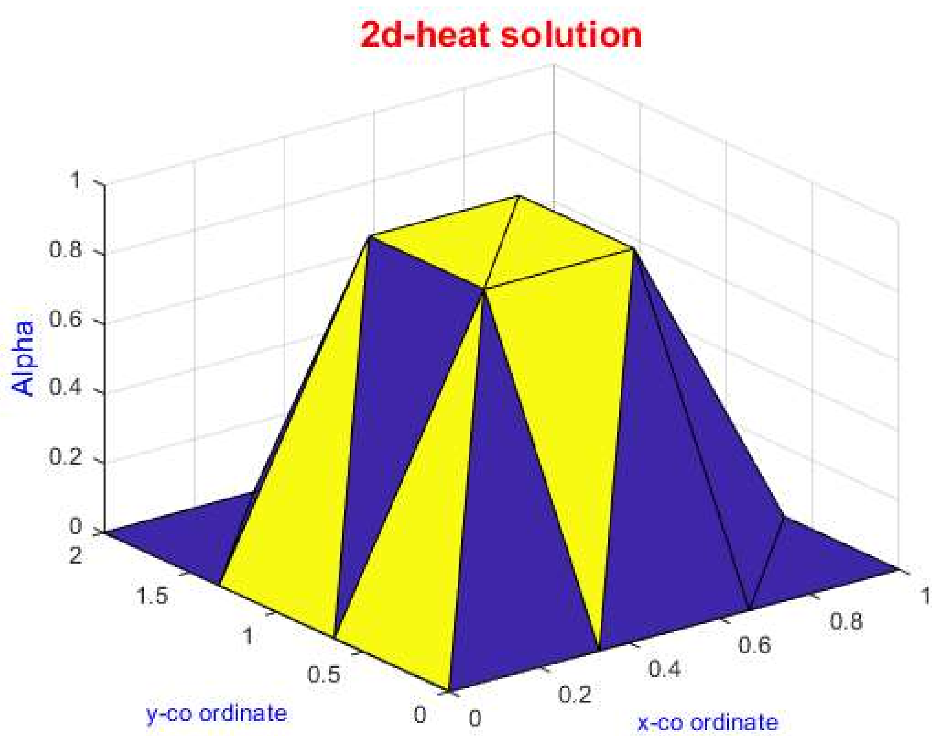
```
        if j==namedof
            K(namedof,j)=1;
        else
            K(namedof,j)=0;
        end

        F(j,1)=F(j,1)-K(j,namedof)*val;
        if j==namedof
            K(j,namedof)=K(j,namedof);
        else
            K(j,namedof)=0;
        end
    end
end
```

Generating Alphas

```
alpha=K\F;
```

plot

```
figure
trisurf(cn,co(:,1),co(:,2),alpha);
xlabel('x-co ordinate','FontSize',10,'Color','b');
ylabel('y-co ordinate','FontSize',10,'Color','b');
zlabel('Alpha',FontSize=12,Color='b')
title("2d-heat solution","FontSize",15,"Color",'r')
```



Published with MATLAB® R2022a