**Functions:**

**Shape\_tri.m:**

function [N,dNx, dNy, detJ] = shape\_tri(x\_bar, y\_bar, xn, ien, nen)

N= zeros(1, nen);

dNx = zeros(1,nen);

dNy = zeros(1,nen);

x = zeros(1,nen);

y = zeros(1,nen);

for i= 1:nen

x(i) = xn(1, ien(i));

y(i) = xn(2, ien(i));

end

A = (1/2)\*(x(2)\*y(3)+x(1)\*y(2)+x(3)\*y(1)-x(2)\*y(1)-x(1)\*y(3)-x(3)\*y(2));

% N1 = (1/(2\*A))\*((x(2)\*y(3)-x(3)\*y(2))+(y(2)-y(3))\*x\_bar+(x(3)-x(2))\*y\_bar);

% N2 = (1/(2\*A))\*((x(3)\*y(1)-x(1)\*y(3))+(y(3)-y(1))\*x\_bar+(x(1)-x(3))\*y\_bar);

% N3 = (1/(2\*A))\*((x(1)\*y(2)-x(2)\*y(1))+(y(1)-y(2))\*x\_bar+(x(2)-x(1))\*y\_bar);

N1x = (1/(2\*A))\*(y(2)-y(3)); %y(2)-y(1)

N2x = (1/(2\*A))\*(y(3)-y(1));

N3x = (1/(2\*A))\*(y(1)-y(2));

N1y = (1/(2\*A))\*(x(3)-x(2));

N2y = (1/(2\*A))\*(x(1)-x(3));

N3y = (1/(2\*A))\*(x(2)-x(1));

detJ =2\*A;

dNx(1) = N1x;

dNx(2) = N2x;

dNx(3) = N3x;

dNy(1) = N1y;

dNy(2) = N2y;

dNy(3) = N3y;

end

**Ke\_heat\_tri.m:**

function [Ke] = Ke\_heat\_tri(D, xn, ien, t, nen, nsd)

Ke = zeros(nen);

x\_bar = 0;

y\_bar = 0;

for i = 1:nen

x\_bar = x\_bar + 1/3\*xn(1,ien(i));

y\_bar = y\_bar + 1/3\*xn(2,ien(i));

end;

[N, dNx, dNy, detJ] = shape\_tri(x\_bar, y\_bar, xn, ien, nen);

B = B\_heat\_tri(dNx, dNy, nen, nsd);

Ke = (t/2)\*detJ\*B'\*D\*B;

end

**B\_heat\_tri.m:**

function [B] = B\_heat\_tri(dNx, dNy, nen, nsd)

B = zeros(nsd,nen);

for i = 1:nen

B(1,i) = dNx(i);

end

for i = 1:nen

B(2,i) = dNy(i);

end

end

**Heat\_v2\_problem\_1\_mesh\_1:**

clear; % removes all variables from the workspace.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DATA %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%

% Conductivity %

%%%%%%%%%%%%

k = 0.05;

D = k\*eye(2);

t = 10;

%%%%%%%%

% Mesh %

%%%%%%%%

nsd=2; % number of space dimension

ndf=1; % number of freedom per node

nen=3; % number of element nodes

nel=4; % number of elements/triangle pieces

nnp=6; % number of nodal points

%%%%%%%%%%%%%%%%%%%%%

% Nodal coordinates %

%%%%%%%%%%%%%%%%%%%%%

% xn(i,N):= coordinate i for node N

% N=1,...,nnp

% i=1,...,nsd

xn=zeros(nsd,nnp);

xn(1,1)=0; xn(2,1)=0;

xn(1,2)=100; xn(2,2)=0;

xn(1,3)=200; xn(2,3)=0;

xn(1,4)=200; xn(2,4)=50;

xn(1,5)=100; xn(2,5)=75;

xn(1,6)=0; xn(2,6)=100;

%%%%%%%%%%%%%%%%

% Connectivity %

%%%%%%%%%%%%%%%%

% ien(a,e)=N

% N: global node number - N=1,...,nnp

% e: element number - e=1,...,nel

% a: local node number - a=1,...,nen

ien=zeros(nen,nel);

%mat=zeros(nel);

ien(1,1)=1; ien(2,1)=2; ien(3,1)=6;

ien(1,2)=2; ien(2,2)=5; ien(3,2)=6;

ien(1,3)=2; ien(2,3)=3; ien(3,3)=5;

ien(1,4)=3; ien(2,4)=4; ien(3,4)=5;

%%%%%%%%%%%%%%%%%%%%%%%

% Boundary conditions %

%%%%%%%%%%%%%%%%%%%%%%%

% prescribed displacement (essential boundary condition)

%

% idb(i,N)=1 if the degree of freedom i of the node N is prescribed

% =0 otherwise

%

% 1) initialize idb to 0

idb=zeros(ndf,nnp);

% 2) enter the flag for prescribed displacement boundary conditions

idb(1,1)=1; idb(1,2)=1; idb(1,3)=1; idb(1,4)=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% prescribed nodal displacement boundary conditions %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% g(i,N): prescribed displacement for the dof i of node N

% initialize g

g=zeros(ndf,nnp);

% enter the values

g(1,1)=10; g(1,2)=10; g(1,3)=10; g(1,4)=20;

%%%%%%%%%%%%%%%%%%%%%%%%%%%

% prescribed nodal fluxes %

%%%%%%%%%%%%%%%%%%%%%%%%%%%

% f(i,N): prescribed flux for the dof i of node N

% initialize f

f=zeros(ndf,nnp);

% enter the values

f(1,5) = 5;

f(1,6) = 2.5;

%---------------------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% number the equations; build the id table %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[id,neq]=number\_eq(idb,nnp,ndf)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Compute the elemental quantities in the elemental coordinate system %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for e=1:nel

[Ke(:,:,e)]=Ke\_heat\_quad(D,xn,ien(:,e),t,nen,nsd)

end;

% Contribution of the prescribed displacements to the elemental force vector

% fe=fe-Ke\*Ue

fe=zeros(ndf\*nen,nel); % fe may be non zero in general

Ue=zeros(ndf\*nen,nel);

for e=1:nel

for n=1:nen

for i=1:ndf

Ue(i+(n-1)\*ndf,e)=g(i,ien(n,e));

end

end

fe(:,e)=fe(:,e)-Ke(:,:,e)\*Ue(:,e);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Assembly operation %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%----------------

% build K and F

%----------------

K=zeros(neq,neq);

F=zeros(neq,1);

% input the prescribed nodal forces in F

for N=1:nnp

for i=1:ndf

if (id(i,N) > 0)

P=id(i,N);

F(P)=f(i,N);

end

end

end

% compute global K and F

if (neq > 0)

for e=1:nel

K = addstiff(K,id,Ke(:,:,e),ien(:,e),nen,ndf);

F = addforce(F,id,fe(:,e),ien(:,e),nen,ndf);

end

end

% Solve the system

if (neq > 0)

U=K\F;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% post processing %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%

% complete U %

%%%%%%%%%%%%%%

Ucomp=zeros(ndf,nnp);

for N=1:nnp

for i=1:ndf

if (id(i,N) == 0)

Ucomp(i,N)=g(i,N);

else

P=id(i,N);

Ucomp(i,N)=U(P);

end

end

end

% print results

disp('Nodal Displacements:')

disp(' node d1 d2')

for N=1:nnp

disp(sprintf('%5d %7g %7g',N,Ucomp(:,N)))

end

disp(' ')

%%%%%%%%%%%%%

% REACTIONS %

%%%%%%%%%%%%%

% build the idb table; overwrite original idb table

% idb(i,N): equation number associated with dof i of node N

ineq=0; % number of equations

%for i=1:ndf

%for N=1:nnp

% if (idb(i,N) > 0) % assign an equation number to all prescribed nodes

% ineq=ineq+1;

% idb(i,N)=ineq;

% end;

% end;

%end;

% Contribution of the displacement to the elemental force vector

% fe=Ke\*Ue

for e=1:nel

Ue(:,e)=zeros(ndf\*nen,1);

for n=1:nen

for i=1:ndf

Ue(i+(n-1)\*ndf,e)=Ucomp(i,ien(n,e));

end

end

fe(:,e)=Ke(:,:,e)\*Ue(:,e);

end;

% compute reactions R %

%R=zeros(ineq,1);

%for e=1:nel

% R = addforce(R,idb,fe(:,e),ien(:,e),nen,ndf);

%end

% Collect reactions

%Rcomp=zeros(ndf,nnp);

%for N=1:nnp

% for i=1:ndf

% if (idb(i,N) > 0)

% Rcomp(i,N)=R(idb(i,N));

% end

% end

%end

% print results

%disp('Nodal Reactions')

%disp(' node R1 R2')

%for N=1:nnp

% disp(sprintf('%5d %7g %7g',N,Rcomp(:,N)))

%end

%disp(' ')

%%%%%%%%%%%%%%%%%%%%%%%%%

% AXIAL FORCES/STRESSES %

%%%%%%%%%%%%%%%%%%%%%%%%%

%for e=1:nel

% Ue(:,e)=zeros(ndf\*nen,1);

% for n=1:nen

% for i=1:ndf

% Ue(i+(n-1)\*ndf,e)=Ucomp(i,ien(n,e));

% end

% end

% if (nsd > 1)

% axial(:,e)=ke(:,:,e)\*Qe(:,:,e)\*Ue(:,e);

% else

% axial(:,e)=ke(:,:,e)\*Qe(e)\*Ue(:,e);

% end;

% stress(e)=axial(2,e)/A(mat(e));

% strain(e)=stress(e)/E(mat(e));

%end;

% print results

%disp('Element Axial force/stress/strain')

%disp(' elem force stress strain')

%for e=1:nel

% disp(sprintf('%5d %7g %7g %7g',e,axial(2,e),stress(e),strain(e)))

%end

%disp(' ')

%%%%%%%%%%%%%%%%%%%%

% plot the results %

%%%%%%%%%%%%%%%%%%%%

RComp=1; % Dummy value

plot\_results\_heat('heat',xn,f,idb,Ucomp,RComp,ien,nel,nen,nsd,ndf,nnp);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Compute the elemental quantities in the elemental coordinate system %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for e=1:nel

[Ke(:,:,e)]=Ke\_heat\_quad(D,xn,ien(:,e),t,nen,nsd, point ,weight, nglx, ngly);

end;

% Contribution of the prescribed displacements to the elemental force vector

% fe=fe-Ke\*Ue

fe=zeros(ndf\*nen,nel); % fe may be non zero in general

Ue=zeros(ndf\*nen,nel);

for e=1:nel

for n=1:nen

for i=1:ndf

Ue(i+(n-1)\*ndf,e)=g(i,ien(n,e));

end

end

fe(:,e)=fe(:,e)-Ke(:,:,e)\*Ue(:,e);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Assembly operation %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%----------------

% build K and F

%----------------

K=zeros(neq,neq);

F=zeros(neq,1);

% input the prescribed nodal forces in F

for N=1:nnp

for i=1:ndf

if (id(i,N) > 0)

P=id(i,N);

F(P)=f(i,N);

end

end

end

% compute global K and F

if (neq > 0)

for e=1:nel

K = addstiff(K,id,Ke(:,:,e),ien(:,e),nen,ndf);

F = addforce(F,id,fe(:,e),ien(:,e),nen,ndf);

end

end

% Solve the system

if (neq > 0)

U=K\F;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% post processing %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%

% complete U %

%%%%%%%%%%%%%%

Ucomp=zeros(ndf,nnp);

for N=1:nnp

for i=1:ndf

if (id(i,N) == 0)

Ucomp(i,N)=g(i,N);

else

P=id(i,N);

Ucomp(i,N)=U(P);

end

end

end

% print results

disp('Nodal Displacements:')

disp(' node d1 d2')

for N=1:nnp

disp(sprintf('%5d %7g %7g',N,Ucomp(:,N)))

end

disp(' ')

%%%%%%%%%%%%%

% REACTIONS %

%%%%%%%%%%%%%

% build the idb table; overwrite original idb table

% idb(i,N): equation number associated with dof i of node N

ineq=0; % number of equations

%for i=1:ndf

%for N=1:nnp

% if (idb(i,N) > 0) % assign an equation number to all prescribed nodes

% ineq=ineq+1;

% idb(i,N)=ineq;

% end;

% end;

%end;

% Contribution of the displacement to the elemental force vector

% fe=Ke\*Ue

for e=1:nel

Ue(:,e)=zeros(ndf\*nen,1);

for n=1:nen

for i=1:ndf

Ue(i+(n-1)\*ndf,e)=Ucomp(i,ien(n,e));

end

end

fe(:,e)=Ke(:,:,e)\*Ue(:,e);

end;

% compute reactions R %

%R=zeros(ineq,1);

%for e=1:nel

% R = addforce(R,idb,fe(:,e),ien(:,e),nen,ndf);

%end

% Collect reactions

%Rcomp=zeros(ndf,nnp);

%for N=1:nnp

% for i=1:ndf

% if (idb(i,N) > 0)

% Rcomp(i,N)=R(idb(i,N));

% end

% end

%end

% print results

%disp('Nodal Reactions')

%disp(' node R1 R2')

%for N=1:nnp

% disp(sprintf('%5d %7g %7g',N,Rcomp(:,N)))

%end

%disp(' ')

%%%%%%%%%%%%%%%%%%%%%%%%%

% AXIAL FORCES/STRESSES %

%%%%%%%%%%%%%%%%%%%%%%%%%

%for e=1:nel

% Ue(:,e)=zeros(ndf\*nen,1);

% for n=1:nen

% for i=1:ndf

% Ue(i+(n-1)\*ndf,e)=Ucomp(i,ien(n,e));

% end

% end

% if (nsd > 1)

% axial(:,e)=ke(:,:,e)\*Qe(:,:,e)\*Ue(:,e);

% else

% axial(:,e)=ke(:,:,e)\*Qe(e)\*Ue(:,e);

% end;

% stress(e)=axial(2,e)/A(mat(e));

% strain(e)=stress(e)/E(mat(e));

%end;

% print results

%disp('Element Axial force/stress/strain')

%disp(' elem force stress strain')

%for e=1:nel

% disp(sprintf('%5d %7g %7g %7g',e,axial(2,e),stress(e),strain(e)))

%end

%disp(' ')

%%%%%%%%%%%%%%%%%%%%

% plot the results %

%%%%%%%%%%%%%%%%%%%%

RComp=1; % Dummy value

plot\_results\_heat('heat',xn,f,idb,Ucomp,RComp,ien,nel,nen,nsd,ndf,nnp);

**Heat\_v2\_problem\_1\_mesh\_2:**

xn=zeros(nsd,nnp);

xn(1,1)=0; xn(2,1)=0;

xn(1,2)=100; xn(2,2)=0;

xn(1,3)=200; xn(2,3)=0;

xn(1,4)=200; xn(2,4)=50;

xn(1,5)=100; xn(2,5)=75;

xn(1,6)=0; xn(2,6)=100;

ien=zeros(nen,nel);

%mat=zeros(nel);

ien(1,1)=1; ien(2,1)=5; ien(3,1)=6;

ien(1,2)=1; ien(2,2)=2; ien(3,2)=5;

ien(1,3)=2; ien(2,3)=4; ien(3,3)=5;

ien(1,4)=2; ien(2,4)=3; ien(3,4)=4;

**(In addition to the changes described above (the code is essentially the same for both meshes except for the connectivity.)**

**Heat\_v2\_problem\_2:**

nsd=2; % number of space dimension

ndf=1; % number of freedom per node

nen=3; % number of element nodes

nel=32; % number of elements/triangle pieces

nnp=25; % number of nodal points

xn=zeros(nsd,nnp);

xn(1,1)=0; xn(2,1)=0;

xn(1,2)=2.5; xn(2,2)=0;

xn(1,3)=5; xn(2,3)=0;

xn(1,4)=7.5; xn(2,4)=0;

xn(1,5)=10; xn(2,5)=0;

xn(1,6)=0; xn(2,6)=1.25;

xn(1,7)=2.5; xn(2,7)=1.25;

xn(1,8)=5; xn(2,8)=1.25;

xn(1,9)=7.5; xn(2,9)=1.25;

xn(1,10)=10; xn(2,10)=1.25;

xn(1,11)=0; xn(2,11)=2.5;

xn(1,12)=2.5; xn(2,12)=2.5;

xn(1,13)=5; xn(2,13)=2.5;

xn(1,14)=7.5; xn(2,14)=2.5;

xn(1,15)=10; xn(2,15)=2.5;

xn(1,16)=0; xn(2,16)=3.75;

xn(1,17)=2.5; xn(2,17)=3.75;

xn(1,18)=5; xn(2,18)=3.75;

xn(1,19)=7.5; xn(2,19)=3.75;

xn(1,20)=10; xn(2,20)=3.75;

xn(1,21)=0; xn(2,21)=5;

xn(1,22)=2.5; xn(2,22)=5;

xn(1,23)=5; xn(2,23)=5;

xn(1,24)=7.5; xn(2,24)=5;

xn(1,25)=10; xn(2,25)=5;

**For problem 2, we only have the connectivity that is different in addition to the absence of a external applied q. We produce a larger geometry that has no fluxes applied to it and observe the temperature distribution by having a few prescribed temperatures at points on the mesh.**