**Functions:**

**Jacobian\_2d:**

function [J] = jacobian\_2d(r, s, x, y, nen)

J = zeros(2);

J(1,1) = 1/4\*((1-s)\*(x(2)-x(1)) + (1+s)\*(x(3)-x(4)));

J(1,2) = 1/4\*((1-s)\*(y(2)-y(1)) + (1+s)\*(y(3)-y(4)));

J(2,1) = 1/4\*((1-r)\*(x(4)-x(1)) + (1+r)\*(x(3)-x(2)));

J(2,2) = 1/4\*((1-r)\*(y(4)-y(1)) + (1+r)\*(y(3)-y(2)));

end

**Ke\_heat\_quad:**

function [Ke] = Ke\_heat\_quad(D, xn, ien, t, nen, point, weight, nglx, ngly)

Ke = 0;

%need to find r and s

r = zeros(4,1);

s = zeros(4,1);

r(1) = point(1,1);

r(2) = point(2,1);

r(3) = point(2,2);

r(4) = point(1,2);

s(1) = point(1,1);

s(2) = point(1,2);

s(3) = point(2,1);

s(4) = point(2,2);

Ketemp = zeros(2);

for i = 1:4

[N, B, detJ] = shape\_quad(r(i), s(i), xn, ien, nen);

Ketemp = t\*detJ\*B'\*D\*B\*weight(i);

Ketemp

r(i)

s(i)

Ke = Ke+Ketemp;

end

Ke

end

**Be\_heat\_quad:**

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

**shape\_quad:**

function [N, B, detJ] = shape\_quad(r, s, xn, ien, nen)

x = zeros(1,nen);

y = zeros(1,nen);

for i= 1:nen

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

y(1,i) = xn(2, ien(i));

end

[N, Nr, Ns] = shape2\_quad(r,s);

[J] = jacobian\_2d(r,s, x, y, nen);

detJ = det(J);

invJ = (inv(J));

B = zeros(nen,2);

Nsr= zeros(2,4);

for i = 1:nen

Nsr(1,i) = Nr(1,i);

Nsr(2,i) = Ns(1,i);

end

B = (invJ)\*(Nsr);

B

end

**Hw8\_problem\_1:**

clear; % removes all variables from the workspace.

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

% DATA %

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

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

% Conductivity %

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

k = .05;

D = k\*eye(2);

t = 10;

%%%%%%%%

% Mesh %

%%%%%%%%

nsd=2; % number of space dimension

ndf=1; % number of freedom per node

nen=4; % number of element nodes

nel=2; % number of elements/triangle pieces

nnp=6; % number of nodal points

nglx = 2;

ngly = 2;

nglz = 0;

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

% 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)=5; ien(4,1)=6;

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

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

% Gaussian integration %

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

[point,weight]=gauss(nglx,ngly,nglz,nsd);

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

% Compute the elemental quantities in the elemental coordinate system %

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

for e=1:nel

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

F

% 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);

**Hw8\_problem\_2:**

nsd=2; % number of space dimension

ndf=1; % number of freedom per node

nen=4; % number of element nodes

nel=16; % number of elements/triangle pieces

nnp=25; % number of nodal points

nglx = 2;

ngly = 2;

nglz = 0;

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

% 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)=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;

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

% 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) = 7; ien(4,1) = 6;

ien(1,2)=2; ien(2,2)=3; ien(3,2) = 8; ien(4,2) = 7;

ien(1,3)=3; ien(2,3)=4; ien(3,3) = 9; ien(4,3) = 8;

ien(1,4)=4; ien(2,4)=5; ien(3,4) = 10; ien(4,4) = 9;

ien(1,5)=6; ien(2,5)=7; ien(3,5) = 12; ien(4,5) = 11;

ien(1,6)=7; ien(2,6)=8; ien(3,6) = 13; ien(4,6) = 12;

ien(1,7)=8; ien(2,7)=9; ien(3,7) = 14; ien(4,7) = 13;

ien(1,8)=9; ien(2,8)=10; ien(3,8) = 15; ien(4,8) = 14;

ien(1,9)=11; ien(2,9)=12; ien(3,9) = 17; ien(4,9) = 16;

ien(1,10)=12; ien(2,10)=13; ien(3,10) = 18; ien(4,10) = 17;

ien(1,11)=13; ien(2,11)=14; ien(3,11) = 19; ien(4,11) = 18;

ien(1,12)=14; ien(2,12)=15; ien(3,12) = 20; ien(4,12) = 19;

ien(1,13)=16; ien(2,13)=17; ien(3,13) = 22; ien(4,13) = 21;

ien(1,14)=17; ien(2,14)=18; ien(3,14) = 23; ien(4,14) = 22;

ien(1,15)=18; ien(2,15)=19; ien(3,15) = 24; ien(4,15) = 23;

ien(1,16)=19; ien(2,16)=20; ien(3,16) = 25; ien(4,16) = 24;

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

% 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; idb(1,5)=1;

idb(1,6)=1; idb(1,11)=1; idb(1,16)=1; idb(1,21)=1;

idb(1,10)=1; idb(1,15)=1; idb(1,20)=1; idb(1,25)=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(1,25)=100; g(1,20)=92.3880; g(1,15)=70.7107; g(1,10)=38.6283;

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

% prescribed nodal fluxes %

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

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

% initialize f

f=zeros(ndf,nnp);

% enter the values

% NONE

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

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

% number the equations; build the id table %

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

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

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

% Gaussian integration %

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

[point,weight]=gauss(nglx,ngly,nglz,nsd)

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

% plot the results %

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

RComp=1; % Dummy value

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

**Discussion:**

**For both homework 7 and 8, I commented out the axial force calculation processes since they were not needed. Additionally, for homework 8, since problem 2 is the same as problem 1’s mechanics, the only thing that was changed was the connectivity and the removal of the flux.**