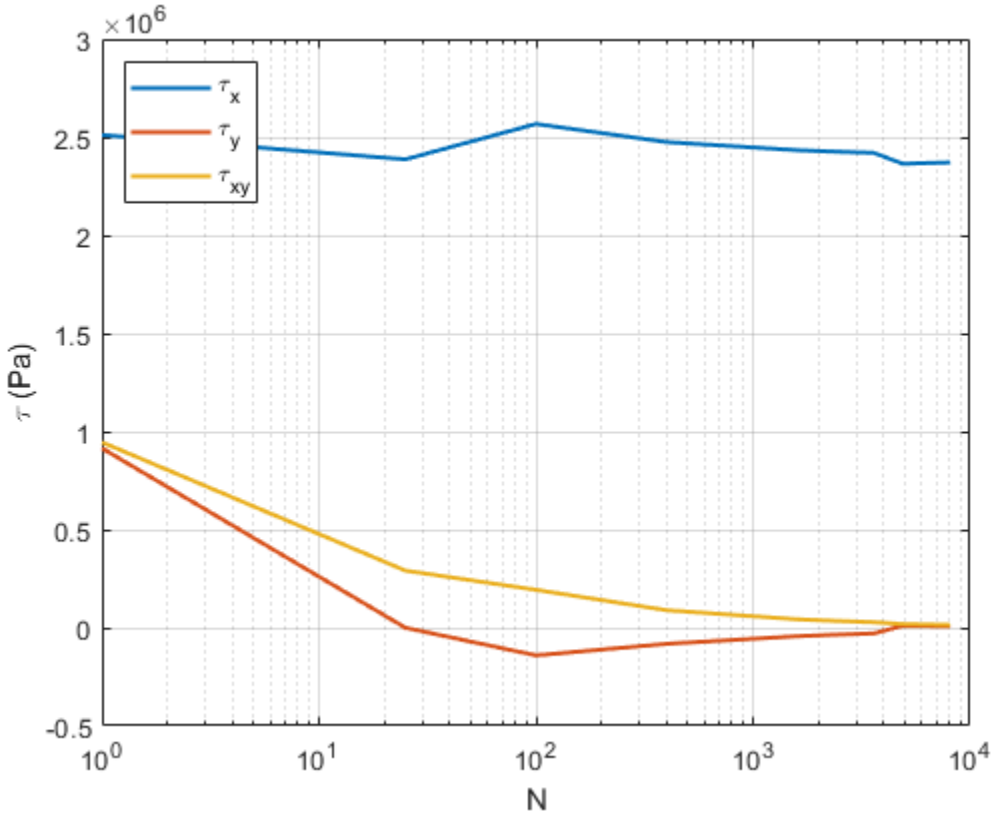
```
clear
close all

N_vec = [1,25,100,400,1600,50^2,60^2,70^2,80^2,90^2];
tau_vec = zeros(3,size(N_vec,2));
for count = 1:size(N_vec,2)
    N = N_vec(count);
    [co,e] = buildMesh(N);%write a mesh function: this is a one element mesh
 now for HW6

    Nel = size(e,1);%number of elements
    Nnodes = size(co,1); %number of nodes
    nne = 4; %number of nodes per element
    dof = 2; %degree of freedom per node
    %%%%%%%%%%%%PREPROCESSING END%%%%%%%%%%%
    %%%Generic block: Initializes global stiffness matrix 'K' and force vector
 'F'
    K = zeros(Nnodes*dof,Nnodes*dof);
    F = zeros(Nnodes*dof,1);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%Assemble Global system - generic FE code
    for A = 1:Nel
        coord = co(e(A,:),:);% get coord matrix for element A
        intpts = (1/sqrt(3))*[-1 -1;1 -1;1 1;-1 1]; %set of integration
 points, same for every element

        %local stiffness matrix and force vector
        localbodyf = localbody(intpts,coord); %hw6, p2

        tr_node = find(coord(:,1)==2);%find nodes (for which x = 2, HW6) where
 traction is applied
        localtracf = localtraction(tr_node,coord); % HW6, p3
        localforce = localbodyf + localtracf;

        localstiffness = localstiffnessmat(intpts,coord); %HW6, p4


        %DONT TOUCH BELOW BLOCK!! Assembles the global stiffness matrix,
 Generic
        %block which works for any element

        for B = 1: nne
            for i = 1: dof
                nK1 = (e(A, B)-1)*dof+i;
                nKe1 = (B-1)*dof+i;
                F(nK1) = F(nK1) + localforce(nKe1);
                    for C = 1: nne
                        for j = 1: dof
                            nK2 = (e(A, C)-1)*dof+j;
                            nKe2 = (C-1)*dof+j;
```

```
                            K(nK1, nK2) = K(nK1, nK2) + localstiffness(nKe1,
nKe2);
                        end
                    end
            end
        end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   end
   %%%%%%%%%%%%%%%%%%%BOUNDARY CONDITIONS%%%%%%%%%%%%%%%%%%%%%%%%

   tr_node = find(co(:,1)==0); %find nodes (for which x = 0) that need to be
fixed
   deletedofs = [2*tr_node-1;2*tr_node];%for these nodes, list of degrees of
freedom
   K(deletedofs,:) = [];
   K(:,deletedofs) = [];
   F(deletedofs,:) = [];
   %%%%%%%%%%%%%%%%%%%BOUNDARY CONDITIONS END%%%%%%%%%%%%%%%%%%%%%%%%%%

   %solve for displacement unknowns (uk)
   uk = K\F;
   %expand u to include deleted displacement bcs
   u = ones(Nnodes*dof,1);
   u(deletedofs) = 0;
   I = find(u == 1);
   u(I) = uk;


   %%%write a code for postprocessing the stress at 1,0

   %Find the element that contains the node
   bottom_co = co(1:sqrt(N)+1,:); %Obtaining coordinates of node at y = 0

   D = vecnorm([1,0] - bottom_co,2,2);
   [~,closest_n] = sort(D,'ascend');
   closest_n = closest_n(1:2);

   logi_1 = sum(e == closest_n(1),2);
   logi_2 = sum(e == closest_n(2),2);

   logi = logi_1 + logi_2;
   elem = find(logi == 2);

   node_elem = e(elem,:);

   node_elem_temp = node_elem(3);
   node_elem(3) = node_elem(4);
   node_elem(4) = node_elem_temp;
   xi = (abs(1-co(node_elem(1),1))./(abs(co(node_elem(1),1) -
co(node_elem(2),1)))).*2-1 %%THIS IS WRONG FIX IT
   eta = -1;

   E = 70e9;
   nu = 0.3;
```

```
    D = E/(1+nu)/(1-2*nu) * [ 1-nu nu 0 ; nu 1-nu 0 ; 0 0 1/2-nu ];

    quad_coord = co(node_elem,:);

    [~,~,B] = element(xi,eta,quad_coord);

    u_node = [2*node_elem-1;2*node_elem];

    q = u(u_node(:));

    tau = D*B*q;
    tau_vec(:,count) = tau;
end
```

Post Processing

```
plot(N_vec,tau_vec,'LineWidth',1.5)
set(gca,'Xscale','log')
grid on
legend('\tau_x','\tau_y','\tau_{xy}','Location','NorthWest')
xlabel('N')
ylabel('\tau (Pa)')
```




```
tau_x = tau_vec(1,:)';
tau_y = tau_vec(2,:)';
tau_xy= tau_vec(3,:)';
```

```
N = N_vec';

table(N,tau_x,tau_y,tau_xy)
```

ans =

10×4 table

N	tau_x	tau_y	tau_xy
1	2.5113e+06	9.1464e+05	9.4546e+05
25	2.3869e+06	2.3358	2.9126e+05
100	2.5674e+06	-1.4083e+05	1.9325e+05
400	2.4751e+06	-81985	89199
1600	2.4337e+06	-43059	42971
2500	2.4257e+06	-34752	34120
3600	2.4203e+06	-29126	28290
4900	2.3645e+06	9569.7	18560
6400	2.3682e+06	8363.6	16286
8100	2.3711e+06	7426.9	14508

Functions Declared

```
function [N, J, B] = element(xi, eta, coords) %hw6, p1
    Ni = 0.25*[(1-xi)*(1-eta), (1+xi)*(1-eta), (1+xi)*(1+eta), (1-
xi)*(1+eta)];
    N = [Ni(1) 0 Ni(2) 0 Ni(3) 0 Ni(4) 0
         0 Ni(1) 0 Ni(2) 0 Ni(3) 0 Ni(4) ];
    dNdxi = 0.25*[ eta-1 1-eta 1+eta -1-eta ; xi-1 -1-xi xi+1 1-xi ];
    J = dNdxi*coords;
    dN = J \ dNdxi;
    B = [dN(1, 1) 0 dN(1, 2) 0 dN(1, 3) 0 dN(1, 4) 0
        0 dN(2, 1) 0 dN(2, 2) 0 dN(2, 3) 0 dN(2, 4)
        dN(2, 1) dN(1, 1) dN(2, 2) dN(1, 2) dN(2, 3) dN(1, 3) dN(2, 4) dN(1,
 4)
    ];
end

function f = localbody(intpts,coords)
    f = zeros(8,1);
    t = 1;
    for i = 1:size(intpts,1)
        [N, J, B] = element(intpts(i,1), intpts(i,2), coords);
        f = f + N'*[0;1E6]*det(J)*t;
    end
end

function k = localstiffnessmat(intpts,coords)
    k = zeros(8,8);
    t = 1;
```

```
    E = 70e9;
    nu = 0.3;
    D = E/(1+nu)/(1-2*nu) * [ 1-nu nu 0 ; nu 1-nu 0 ; 0 0 1/2-nu ];
    for i = 1:size(intpts,1)
        [N, J, B] = element(intpts(i,1), intpts(i,2), coords);
        k = k + B'*D*B*det(J)*t;
    end
end

function f = localtraction(tr_node,coords)
    f = zeros(8,1);
    if length(tr_node) > 0 %if nodes have traction
        t = 1;
        intpts = [1 1/sqrt(3);1 -1/sqrt(3)];
        for i = 1:size(intpts,1)
            [N, J, B] = element(intpts(i,1), intpts(i,2), coords);
            detJstar = sqrt(J(2,1)^2 + J(2,2)^2);
            f = f + N'*[1e6;0]*detJstar*t;
        end
    end
end
% function [co,e,elem,loc] = genmesh(nelem)
%     %for HW7, you need to write a meshing code
%     %make sure the elements are numbered ccw such that the face on which
 traction
%     is applied has the second and third nodes of the element
%     %elem and loc are the element number containing (1,0) and the
 integration
%     %point at that location
%     co = [ 0 0
%            2 0
%            0 2
%            2 1];
%     e = [ 1 2 4 3];
%     elem = 1;
%     loc = [0 -1];
% end


xi =

     0


xi =

     0


xi =

     1
```

```
xi =

     1

xi =

     1

xi =

     1

xi =

     1

xi =

    -1

xi =

    -1

xi =

    -1
```

Published with MATLAB® R2022a