Contents


```
% HW 4
% finite element - body
node = load( 'npeltr4.dat' );
elem = load( 'epeltr4.dat' );

bc3 = load( 'bpeltr4.dat' );
ht_rate = load( 'ppelt4.dat' );

% gauss points
zeta = [-.57735, .57735, .57735, -.57735];
eta = [-.57735, -.57735, .57335, .57335];
% materials
k = [0, 0, .210, .642, .436, .561, .515, .642];
m = [0, 0, 200, 2001.4, 0, 0, 1482.5, 0];
```

element loop

```
A = zeros( size(node,1) );
B = zeros( size(node,1), 1 );

for i=1:size(elem,1)
    % begin triangular element
    if elem(i,4) == elem(i,5)
        n1 = elem(i,2);
        n2 = elem(i,3);
        n3 = elem(i,4);

        % assign coordinate to each node
        x1 = node(n1,2);
        y1 = node(n1,3);
        x2 = node(n2,2);
        y2 = node(n2,3);
        x3 = node(n3,2);
        y3 = node(n3,3);

        area = ( x1*(y2-y3) + x2*(y3-y1) + x3*(y1-y2) ) / 2;

        km = k(elem(i,6));
        mm = m(elem(i,6));

        dp_dy_1 = -(x2-x3)/(2*area);
        dp_dy_2 = -(x3-x1)/(2*area);
        dp_dy_3 = -(x1-x2)/(2*area);

        dp_dx_1 = (y2-y3)/(2*area);
        dp_dx_2 = (y3-y1)/(2*area);
```

```
        dp_dx_3 = (y1-y2)/(2*area);

        A_e(1,1) = -km * (dp_dx_1 * dp_dx_1 + dp_dy_1 * dp_dy_1 )  - mm * (area / 6);
        A_e(1,2) = -km * (dp_dx_1 * dp_dx_2 + dp_dy_1 * dp_dy_2 )  - mm * (area / 12);
        A_e(1,3) = -km * (dp_dx_1 * dp_dx_3 + dp_dy_1 * dp_dy_3 )  - mm * (area / 12);
        A_e(2,1) = -km * (dp_dx_2 * dp_dx_1 + dp_dy_2 * dp_dy_1 )  - mm * (area / 12);
        A_e(2,2) = -km * (dp_dx_2 * dp_dx_2 + dp_dy_2 * dp_dy_2 )  - mm * (area / 6);
        A_e(2,3) = -km * (dp_dx_2 * dp_dx_3 + dp_dy_2 * dp_dy_3 )  - mm * (area / 12);
        A_e(3,1) = -km * (dp_dx_3 * dp_dx_1 + dp_dy_3 * dp_dy_1 )  - mm * (area / 12);
        A_e(3,2) = -km * (dp_dx_3 * dp_dx_2 + dp_dy_3 * dp_dy_2 )  - mm * (area / 12);
        A_e(3,3) = -km * (dp_dx_3 * dp_dx_3 + dp_dy_3 * dp_dy_3 )  - mm * (area / 6);
        B_e(1)   = - ht_rate(n1,3)* (area / 3);
        B_e(2)   = - ht_rate(n2,3)* (area / 3);
        B_e(3)   = - ht_rate(n3,3)* (area / 3);

        % assemble global matrix
        A(n1,n1) = A(n1,n1) + A_e(1,1);
                A(n1,n2) = A(n1,n2) + A_e(1,2);
                A(n1,n3) = A(n1,n3) + A_e(1,3);
                A(n2,n1) = A(n2,n1) + A_e(2,1);
                A(n2,n2) = A(n2,n2) + A_e(2,2);
                A(n2,n3) = A(n2,n3) + A_e(2,3);
                A(n3,n1) = A(n3,n1) + A_e(3,1);
                A(n3,n2) = A(n3,n2) + A_e(3,2);
                A(n3,n3) = A(n3,n3) + A_e(3,3);

        B(n1)    = B(n1) + B_e(1);
        B(n2)    = B(n2) + B_e(2);
        B(n3)    = B(n3) + B_e(3);

    % quadrilateral element
    else
        Ae = zeros( 4 );
        Be = zeros( 4, 1 );

        n1 = elem(i,2);
        n2 = elem(i,3);
        n3 = elem(i,4);
        n4 = elem(i,5);

        % assign coordinate to each node
        x1 = node(n1,2);
        y1 = node(n1,3);
        x2 = node(n2,2);
        y2 = node(n2,3);
        x3 = node(n3,2);
        y3 = node(n3,3);
        x4 = node(n4,2);
        y4 = node(n4,3);

        % gauss point loop
        for j = 1:4
            % construct phi
```

```
            phi(1) = ((1-zeta(j)) * (1-eta(j)))/4;
            phi(2) = ((1+zeta(j)) * (1-eta(j)))/4;
            phi(3) = ((1+zeta(j)) * (1+eta(j)))/4;
            phi(4) = ((1-zeta(j)) * (1+eta(j)))/4;

            dpz(1) = -(1-eta(j))/4;
            dpz(2) = (1-eta(j))/4;
            dpz(3) = (1+eta(j))/4;
            dpz(4) = -(1+eta(j))/4;

            dpe(1) = -(1-zeta(j))/4;
            dpe(2) = -(1+zeta(j))/4;
            dpe(3) = (1+zeta(j))/4;
            dpe(4) = (1-zeta(j))/4;

            % construct Jacobian
            dxz = x1*dpz(1) + x2*dpz(2) + x3*dpz(3) + x4*dpz(4);
            dxe = x1*dpe(1) + x2*dpe(2) + x3*dpe(3) + x4*dpe(4);
            dyz = y1*dpz(1) + y2*dpz(2) + y3*dpz(3) + y4*dpz(4);
            dye = y1*dpe(1) + y2*dpe(2) + y3*dpe(3) + y4*dpe(4);


            dj = dxz * dye - dyz * dxe;

            % derivatives
            dpx(1) = (dye*dpz(1) - dyz*dpe(1))/dj;
            dpx(2) = (dye*dpz(2) - dyz*dpe(2))/dj;
            dpx(3) = (dye*dpz(3) - dyz*dpe(3))/dj;
            dpx(4) = (dye*dpz(4) - dyz*dpe(4))/dj;

            dpy(1) = (-dxe*dpz(1) + dxz*dpe(1))/dj;
            dpy(2) = (-dxe*dpz(2) + dxz*dpe(2))/dj;
            dpy(3) = (-dxe*dpz(3) + dxz*dpe(3))/dj;
            dpy(4) = (-dxe*dpz(4) + dxz*dpe(4))/dj;

            % assemble coefficients
            km = k(elem(i,6))*phi(1) + k(elem(i,6))*phi(2) + k(elem(i,6))*phi(3) + k(elem(i,6)
)*phi(4);
            mm = m(elem(i,6))*phi(1) + m(elem(i,6))*phi(2) + m(elem(i,6))*phi(3) + m(elem(i,6)
)*phi(4);

            % loop over rows
            for ii = 1:4
                % loop over cols
                for jj = 1:4
                    Ae(ii,jj) = Ae(ii,jj) + dj * ( -km*( dpx(ii)*dpx(jj) + dpy(ii)*dpy(jj) ) -
 mm*phi(jj)*phi(ii) );
                end
                Be(ii) = Be(ii) + dj * ( -ht_rate(i,3)*phi(ii) );
            end
        end

        % assemble global matrix
```

```
        A(n1,n1) = A(n1,n1) + Ae(1,1);
                A(n1,n2) = A(n1,n2) + Ae(1,2);
                A(n1,n3) = A(n1,n3) + Ae(1,3);
        A(n1,n4) = A(n1,n4) + Ae(1,4);
                A(n2,n1) = A(n2,n1) + Ae(2,1);
                A(n2,n2) = A(n2,n2) + Ae(2,2);
                A(n2,n3) = A(n2,n3) + Ae(2,3);
        A(n2,n4) = A(n2,n4) + Ae(2,4);
                A(n3,n1) = A(n3,n1) + Ae(3,1);
                A(n3,n2) = A(n3,n2) + Ae(3,2);
                A(n3,n3) = A(n3,n3) + Ae(3,3);
        A(n3,n4) = A(n3,n4) + Ae(3,4);
        A(n4,n1) = A(n4,n1) + Ae(4,1);
                A(n4,n2) = A(n4,n2) + Ae(4,2);
                A(n4,n3) = A(n4,n3) + Ae(4,3);
        A(n4,n4) = A(n4,n4) + Ae(4,4);

        B(n1)    = B(n1) + Be(1);
        B(n2)    = B(n2) + Be(2);
        B(n3)    = B(n3) + Be(3);
        B(n4)    = B(n4) + Be(4);
    end
end

% apply BCs
for i=1:size(bc3,1)
    n_b = bc3(i,2);
    n_c = bc3(i,4);
    n_d = bc3(i,5);

    L_I = sqrt( (node(n_b,2)-node(n_c,2))^2 + (node(n_b,3)-node(n_c,3))^2 );
    L_II = sqrt( (node(n_b,2)-node(n_d,2))^2 + (node(n_b,3)-node(n_d,3))^2 );

    A(n_b,n_b) = A(n_b,n_b) - bc3(i,6)*( L_I + L_II ) / 3;
    A(n_b,n_c) = A(n_b,n_c) - bc3(i,6)*( L_I ) / 6;
    A(n_b,n_d) = A(n_b,n_d) - bc3(i,6)*( L_II ) / 6;
    B(n_b)     = B(n_b) - bc3(i,6)*bc3(i,7)*(L_I+L_II) / 2;
end


T = linsolve( A, B );

% figure(1);
% for i = 1:size(elem,1)
%     if elem(i,4) == elem(i,5)
%         patch( node( elem(i,2:4), 2 ), node( elem(i,2:4), 3 ), T(elem(i,2:4)), 'FaceColor','
interp' );
%         hold on;
%     else
%         patch( node( elem(i,2:5), 2 ), node( elem(i,2:5), 3 ), T(elem(i,2:5)), 'FaceColor','
interp' );
%         hold on;
%     end
```

```
% end
% title('Baseline - Tissue Heat Transfer');
% colorbar;
```

Published with MATLAB® R2016a