Computer Implementation 6.6 ***(Matlab)*** *4 Node Quadrilateral Element for 2D BVP (p. 442)*

The analysis of two dimensional bounadry value problems using rectangular elements can be performed conveniently by writing three *Matlab* functions, one for defining the element $\boldsymbol{k}_k + \boldsymbol{k}_p$and $\boldsymbol{r}_q$ vectors, one for evaluating natural boundary terms, and the third for computing the element solution. The BVPQuad4Element employs 2×2 integration. The BVPQuad4NBCTerm employs two point integration. The functions can be modified easily to use any other integration formula. The BVPQuad4Results function computes results at the point of the element that corresponds to the origin of the master element ($s = t = 0$). The function returns the location of this point (in terms of x, y coordinates) and the solution and its x and y derivatives at this point. Obviously the function can be modified to compute results at any other point.

MatlabFiles\Chap6\BVPQuad4Element.m

```
function [ke, rq] = BVPQuad4Element(kx, ky, p, q, coord)
% [ke, rq] = BVPQuad4Element(kx, ky, p, q, coord)
% Generates for a 4 node quadrilateral element for 2d BVP
% kx, ky, p, q = parameters defining the BVP
% coord = coordinates at the element ends

% Use 2x2 integration. Gauss point locations and weights
pt=1/sqrt(3);
gpLocs = [-pt,-pt; -pt,pt; pt,-pt; pt,pt];
gpWts = [1,1,1,1];
kk=zeros(4); kp=zeros(4); rq=zeros(4,1);
for i=1:length(gpWts)
   s = gpLocs(i, 1); t = gpLocs(i, 2); w = gpWts(i);
   n = [(1/4)*(1 - s)*(1 - t), (1/4)*(s + 1)*(1 -t), ...
          (1/4)*(s + 1)*(t + 1), (1/4)*(1 - s)*(t + 1)];
   dns=[(-1 + t)/4, (1 - t)/4, (1 + t)/4, (-1 - t)/4];
   dnt=[(-1 + s)/4, (-1 - s)/4, (1 + s)/4, (1 - s)/4];
   x = n*coord(:,1); y = n*coord(:,2);
   dxs = dns*coord(:,1); dxt = dnt*coord(:,1);
   dys = dns*coord(:,2); dyt = dnt*coord(:,2);
   J = [dxs, dxt; dys, dyt]; detJ = det(J);
   bx = (J(2, 2)*dns - J(2, 1)*dnt)/detJ;
   by = (-J(1, 2)*dns + J(1, 1)*dnt)/detJ;
   b = [bx; by];
   c = [kx, 0; 0, ky];
   kk = kk + detJ*w* b'*c*b;
   kp = kp - detJ*w*p * n'*n;
   rq = rq + detJ*w*q * n';
end
ke=kk+kp;
```

MatlabFiles\Chap6\BVPQuad4NBCTerm.m

```
function [ka, rb] = BVPQuad4NBCTerm(side, alpha, beta, coord)
% [ka, rb] = BVPQuad4NBCTerm(side, alpha, beta, coord)
% Generates kalpha and rbeta when NBC is specified along a side
% side = side over which the NBC is specified
% alpha and beta = coefficients specifying the NBC
% coord = coordinates at the element ends

% Use 2 point integration. Gauss point locations and weights
pt=-1/sqrt(3);
gpLocs = [-pt, pt];
gpWts = [1,1];
ka=zeros(4); rb=zeros(4,1);
for i=1:length(gpWts)
    a = gpLocs(i); w = gpWts(i);
    switch (side)
    case 1
        n = [(1 - a)/2, (1 + a)/2, 0, 0];
        dna = [-1/2, 1/2, 0, 0];
    case 2
        n = [0, (1 - a)/2, (1 + a)/2, 0];
        dna = [0, -1/2, 1/2, 0];
    case 3
        n = [0, 0, (1 - a)/2, (1 + a)/2];
        dna = [0, 0, -1/2, 1/2];
    case 4
        n = [(1 + a)/2, 0, 0, (1 - a)/2];
        dna = [1/2, 0, 0, -1/2];
    end
    dxa = dna*coord(:,1); dya = dna*coord(:,2);
    Jc=sqrt(dxa^2 + dya^2);
    ka = ka - alpha*Jc*w*n'*n;
    rb = rb + beta*Jc*w*n';
end
```

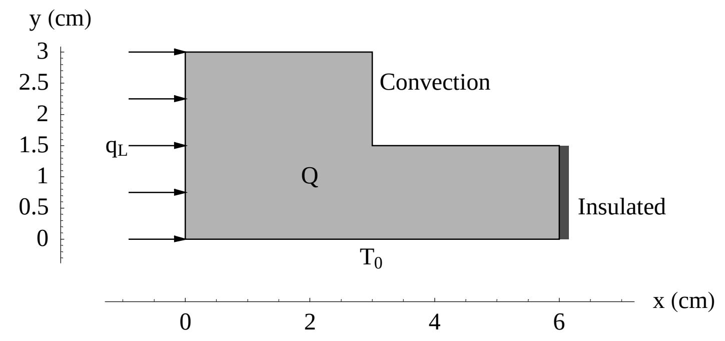
MatlabFiles\Chap6\BVPQuad4Results.m

```
function results = BVPQuad4Results(coord, dn)
% results = BVPQuad4Results(coord, dn)
% Computes element solution for a quadrilateral element for 2D BVP
% coord = nodal coordinates
% dn = nodal solution
% The solution is computed at the element center
% The output variables are loc, u and its x and y derivatives
```

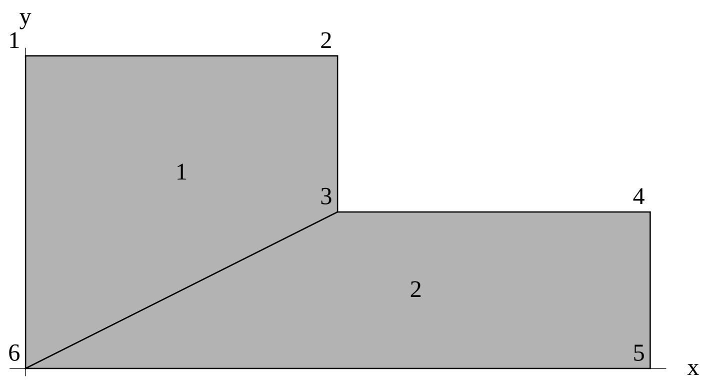
```
s = 0; t = 0;
n = [(1/4)*(1 - s)*(1 - t), (1/4)*(s + 1)*(1 -t), ...
        (1/4)*(s + 1)*(t + 1), (1/4)*(1 - s)*(t + 1)];
dns=[(-1 + t)/4, (1 - t)/4, (1 + t)/4, (-1 - t)/4];
dnt=[(-1 + s)/4, (-1 - s)/4, (1 + s)/4, (1 - s)/4];
x = n*coord(:,1); y = n*coord(:,2);
dxs = dns*coord(:,1); dxt = dnt*coord(:,1);
dys = dns*coord(:,2); dyt = dnt*coord(:,2);
J = [dxs, dxt; dys, dyt]; detJ = det(J);
bx = (J(2, 2)*dns - J(2, 1)*dnt)/detJ;
by = (-J(1, 2)*dns + J(1, 1)*dnt)/detJ;
b = [bx; by];
results=[x, y, n*dn, bx*dn, by*dn];
```

Heat flow in an L-shaped body using Quad4 elements

Consider two dimensional heat flow over an L-shaped body with thermal conductivity $k = 45\,W/m.$°C. The bottom is maintained at $T_0 = 110$ °C. Convection heat loss takes place on the top where the ambient air temperature is 20 °C and the convection heat transfer coefficient is $h = 55\,W/m^2.$°C. The right side is insulated. The left side is subjected to heat flux at a uniform rate of $q_L = 8000\,W/m^2$. Heat is generated in the body at a rate of $Q = 5\times10^6\ W/m^3$. Determine temperature distribution in the body.



To demonstrate the use of the functions presented in this section we obtain a finite element solution of the problem using only 2 quadrilateral elements. Obviously we don't expect very good results. The mesh is chosen to simplify calculations. The steps are exactly those used in other *Matlab* implementations.



MatlabFiles\Chap6\LHeatQuad4Ex.m

```
% Heat flow through an L-shaped body using quad4 elements
h = 55; tf = 20; htf = h*tf;
kx = 45;  ky = 45; Q = 5*10^6; ql = 8000; t0 = 110;
nodes = 1.5/100*[0, 2; 2, 2; 2, 1; 4, 1; 4, 0; 0, 0];
lmm = [6, 3, 2, 1; 6, 5, 4, 3];
debc = [5:6]; ebcVals=t0*ones(length(debc),1);
dof=length(nodes); elems=size(lmm,1);
K=zeros(dof); R = zeros(dof,1);
% Generate equations for each element and assemble them.
for i=1:elems
    lm = lmm(i,:);
    [k, r] = BVPQuad4Element(kx, ky, 0, Q, nodes(lm,:));
    K(lm, lm) = K(lm, lm) + k;
    R(lm) = R(lm) + r;
end
% Compute and assemble NBC contributions
lm = lmm(1,:);
[k, r] = BVPQuad4NBCTerm(4, 0, ql, nodes(lm,:));
K(lm, lm) = K(lm, lm) + k;
R(lm) = R(lm) + r;
[k, r] = BVPQuad4NBCTerm(2, -h, htf, nodes(lm,:));
K(lm, lm) = K(lm, lm) + k;
R(lm) = R(lm) + r;
[k, r] = BVPQuad4NBCTerm(3, -h, htf, nodes(lm,:));
K(lm, lm) = K(lm, lm) + k;
R(lm) = R(lm) + r;

lm = lmm(2,:);
```

```
[k, r] = BVPQuad4NBCTerm(3, -h, htf, nodes(lm,:));
K(lm, lm) = K(lm, lm) + k;
R(lm) = R(lm) + r;

% Nodal solution
d = NodalSoln(K, R, debc, ebcVals)
results=[];
for i=1:elems
    results = [results; BVPQuad4Results(nodes(lmm(i,:),:), ...
            d(lmm(i,:)))];
end
format short g
results
```

```
>> LHeatQuad4Ex

d =

  153.3936
  142.9067
  132.8533
  124.5394
  110.0000
  110.0000


results =

        0.015      0.01875       134.79       -90.82       1187.7
       0.0375       0.0075       119.35      -92.377       1338.8
```