Computer Implementation 5.3 ***(Matlab)*** *Torsion using triangular elements (p. 371)*

The analysis of two dimensional bounadry value problems using triangular elements can be performed conveniently by writing three *Matlab* functions, one for defining the element $\boldsymbol{k}_k + k_p$ and $\boldsymbol{r}_q$ vectors, one for evaluating natural boundary terms, and the third for computing the element solution.

MatlabFiles\Chap5\BVPTriElement.m

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

x1=coord(1,1); y1=coord(1,2);
x2=coord(2,1); y2=coord(2,2);
x3=coord(3,1); y3=coord(3,2);
b1 = y2 - y3; b2 = y3 - y1; b3 = y1 - y2;
c1 = x3 - x2; c2 = x1 - x3; c3 = x2 - x1;
f1 = x2*y3 - x3*y2; f2 = x3*y1 - x1*y3; f3 = x1*y2 - x2*y1;
A = (f1 + f2 + f3)/2;
kxx = 1/(4*A)*kx*[b1^2, b1*b2, b1*b3;
   b1*b2, b2^2, b2*b3; b1*b3, b2*b3, b3^2];
kyy = 1/(4*A)*ky*[c1^2, c1*c2, c1*c3;
   c1*c2, c2^2, c2*c3; c1*c3, c2*c3, c3^2];
kp = -(p*A/12)*[2, 1, 1; 1, 2, 1; 1, 1, 2];
ke = kxx + kyy + kp;
rq = 1/3*q*A*[1; 1; 1];
```

MatlabFiles\Chap5\BVPTriNBCTerm.m

```
function [ka, rb] = BVPTriNBCTerm(side, alpha, beta, coord)
% [ka, rb] = BVPTriNBCTerm(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

x1=coord(1,1); y1=coord(1,2);
x2=coord(2,1); y2=coord(2,2);
x3=coord(3,1); y3=coord(3,2);
switch (side)
case 1
   L = sqrt((x2-x1)^2+(y2-y1)^2);
   ka = -alpha*L/6 * [2,1,0; 1,2,0; 0,0,0];
   rb = beta *L/2 * [1; 1; 0];
```

```
case 2
   L = sqrt((x2-x3)^2+(y2-y3)^2);
   ka = -alpha*L/6 * [0,0,0; 0,2,1; 0,1,2];
   rb = beta *L/2 * [0; 1; 1];
case 3
   L = sqrt((x3-x1)^2+(y3-y1)^2);
   ka = -alpha*L/6 * [2,0,1; 0,0,0; 1,0,2];
   rb = beta *L/2 * [1; 0; 1];
end
```

MatlabFiles\Chap5\BVPTriResults.m

```
function results = BVPTriResults(coord, dn)
% results = BVPTriResults(coord, dn)
% Computes element solution for a triangular element for 2D BVP
% coord = nodal coordinates
% dn = nodal solution
% The results are computed at the element center
% The first three output variables are u and its x and y derivatives
% The last output variable is integral of solution over the element
x1=coord(1,1); y1=coord(1,2);
x2=coord(2,1); y2=coord(2,2);
x3=coord(3,1); y3=coord(3,2);
x=(x1+x2+x3)/3; y=(y1+y2+y3)/3;
b1 = y2 - y3; b2 = y3 - y1; b3 = y1 - y2;
c1 = x3 - x2; c2 = x1 - x3; c3 = x2 - x1;
f1 = x2*y3 - x3*y2; f2 = x3*y1 - x1*y3; f3 = x1*y2 - x2*y1;
A = (f1 + f2 + f3)/2;
n = [f1 + x*b1 + y*c1, f2 + x*b2 + y*c2, f3 + x*b3 + y*c3]/(2*A);
u = n*dn; dudx=[b1, b2, b3]*dn/(2*A); dudy=[c1, c2, c3]*dn/(2*A);
ui = sum(dn)*A/3;
results=[u, dudx, dudy, ui];
```

Using these functions now we consider solution of the torsion problem. The steps are exactly those used in other *Matlab* implementations.

MatlabFiles\Chap5\CShapeTorsionEx.m

```
% Torsion of a C Shaped Section
kx=1; ky=1; q=2; p=0;
nodes=[0,0; 0,6; 0.255,0;0.255,5.749499999999999;
   0.51,0; 0.51,5.499; 3.17,5.499; 3.17,5.749499999999999; 3.17,6];
lmm = [1,3,4; 4,2,1; 3,5,6; 6,4,3; 6,7,8; 8,4,6; 4,8,9; 9,2,4];
debc = [1,2,5,6,7,8,9]; ebcVals=zeros(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] = BVPTriElement(kx, ky, p, q, nodes(lm,:));
   K(lm, lm) = K(lm, lm) + k;
   R(lm) = R(lm) + r;
end

% Nodal solution
d = NodalSoln(K, R, debc, ebcVals)
results=[];
for i=1:elems
   results = [results; BVPTriResults(nodes(lmm(i,:),:), d(lmm(i,:)))];
end
results
% Torsional constant calculations
J = 4*sum(results(:,4))
```

```
>> CShapeTorsionEx

d =

         0
         0
    0.0640
    0.0663
         0
         0
         0
         0
         0


results =

    0.0434    0.2511    0.0004    0.0318
    0.0221    0.2598         0    0.0169
    0.0213   -0.2511         0    0.0150
    0.0434   -0.2595    0.0004    0.0318
         0         0         0         0
    0.0221   -0.0227    0.2414    0.0081
    0.0221   -0.0227         0    0.0081
    0.0221         0   -0.2645    0.0088


J =
```

0.4817

To get torsional constant we add all integrals over the elements (the fourth item in each element results). Multiplying this sum by 2 gives the total torque. Since we are modeling half of the C shape, the torque for the entire section is twice this value. Since $T = J\,G\,\theta$ and we have used $G = \theta = 1$, the computations show that the torsional constant J for the section is 0.482 in^4.