

Computer Implementation 5.2 (Matlab) BVP Using 8 node elements (p. 350)

The following functions implement the 8 node rectangular element in *Matlab*. The first function defines the element $\mathbf{k}_k + \mathbf{k}_p$ and \mathbf{r}_q vectors, the second function evaluates terms resulting from any specified natural boundary conditions, and the third function computes the element solution.

MatlabFiles\Chap5\BVPRect8Element.m

```
function [ke, rq] = BVPRect8Element(kx, ky, p, q, coord)
% [ke, rq] = BVPRect8Element(kx, ky, p, q, coord)
% 8 node rectangular 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);
x3=coord(3,1); y7=coord(7,2);
a = abs(x3 - x1)/2; b = abs(y7 - y1)/2;
kk = [(26*b*kx)/(45*a) + (26*a*ky)/(45*b), ...
      (-8*b*kx)/(9*a) + (a*ky)/(15*b), ...
      (14*b*kx)/(45*a) + (17*a*ky)/(90*b), ...
      -(b*kx)/(15*a) - (4*a*ky)/(9*b), ...
      (23*b*kx)/(90*a) + (23*a*ky)/(90*b), ...
      (-4*b*kx)/(9*a) - (a*ky)/(15*b), ...
      (17*b*kx)/(90*a) + (14*a*ky)/(45*b), ...
      (b*kx)/(15*a) - (8*a*ky)/(9*b);
      (-8*b*kx)/(9*a) + (a*ky)/(15*b), ...
      (16*b*kx)/(9*a) + (8*a*ky)/(15*b), ...
      (-8*b*kx)/(9*a) + (a*ky)/(15*b), 0, ...
      (-4*b*kx)/(9*a) - (a*ky)/(15*b), ...
      (8*b*kx)/(9*a) - (8*a*ky)/(15*b), ...
      (-4*b*kx)/(9*a) - (a*ky)/(15*b), 0;
      (14*b*kx)/(45*a) + (17*a*ky)/(90*b),...
      (-8*b*kx)/(9*a) + (a*ky)/(15*b), ...
      (26*b*kx)/(45*a) + (26*a*ky)/(45*b), ...
      (b*kx)/(15*a) - (8*a*ky)/(9*b), ...
      (17*b*kx)/(90*a) + (14*a*ky)/(45*b),...
      (-4*b*kx)/(9*a) - (a*ky)/(15*b), ...
      (23*b*kx)/(90*a) + (23*a*ky)/(90*b),...
      -(b*kx)/(15*a) - (4*a*ky)/(9*b);
      -(b*kx)/(15*a) - (4*a*ky)/(9*b), 0, ...
      (b*kx)/(15*a) - (8*a*ky)/(9*b), ...
      (8*b*kx)/(15*a) + (16*a*ky)/(9*b),...
      (b*kx)/(15*a) - (8*a*ky)/(9*b), 0,...
      -(b*kx)/(15*a) - (4*a*ky)/(9*b), ...
      (-8*b*kx)/(15*a) + (8*a*ky)/(9*b);

rq = [0, 0, 0, 0, 0, 0, 0, 0];
```

$$\begin{aligned}
& (23*b*kx)/(90*a) + (23*a*ky)/(90*b), \dots \\
& (-4*b*kx)/(9*a) - (a*ky)/(15*b), \dots \\
& (17*b*kx)/(90*a) + (14*a*ky)/(45*b), \dots \\
& (b*kx)/(15*a) - (8*a*ky)/(9*b), \dots \\
& (26*b*kx)/(45*a) + (26*a*ky)/(45*b), \dots \\
& (-8*b*kx)/(9*a) + (a*ky)/(15*b), \dots \\
& (14*b*kx)/(45*a) + (17*a*ky)/(90*b), \dots \\
& -(b*kx)/(15*a) - (4*a*ky)/(9*b); \\
& (-4*b*kx)/(9*a) - (a*ky)/(15*b), \dots \\
& (8*b*kx)/(9*a) - (8*a*ky)/(15*b), \dots \\
& (-4*b*kx)/(9*a) - (a*ky)/(15*b), 0, \dots \\
& (-8*b*kx)/(9*a) + (a*ky)/(15*b), \dots \\
& (16*b*kx)/(9*a) + (8*a*ky)/(15*b), \dots \\
& (-8*b*kx)/(9*a) + (a*ky)/(15*b), 0; \\
& (17*b*kx)/(90*a) + (14*a*ky)/(45*b), \dots \\
& (-4*b*kx)/(9*a) - (a*ky)/(15*b), \dots \\
& (23*b*kx)/(90*a) + (23*a*ky)/(90*b), \dots \\
& -(b*kx)/(15*a) - (4*a*ky)/(9*b), \dots \\
& (14*b*kx)/(45*a) + (17*a*ky)/(90*b), \dots \\
& (-8*b*kx)/(9*a) + (a*ky)/(15*b), \dots \\
& (26*b*kx)/(45*a) + (26*a*ky)/(45*b), \dots \\
& (b*kx)/(15*a) - (8*a*ky)/(9*b); \\
& (b*kx)/(15*a) - (8*a*ky)/(9*b), 0, \dots \\
& -(b*kx)/(15*a) - (4*a*ky)/(9*b), \dots \\
& (-8*b*kx)/(15*a) + (8*a*ky)/(9*b), \dots \\
& -(b*kx)/(15*a) - (4*a*ky)/(9*b), 0, \dots \\
& (b*kx)/(15*a) - (8*a*ky)/(9*b), (8*b*kx)/(15*a) + (16*a*ky)/(9*b)]; \\
kp = [& (-2/15)) * a * b * p, (2 * a * b * p) / 15, (-2/45)) * a * b * p, (8 * a * b * p) / 45, \dots \\
& (-1/15)) * a * b * p, (8 * a * b * p) / 45, (-2/45)) * a * b * p, (2 * a * b * p) / 15; \\
& (2 * a * b * p) / 15, (-32/45)) * a * b * p, (2 * a * b * p) / 15, (-4/9)) * a * b * p, \dots \\
& (8 * a * b * p) / 45, (-16/45)) * a * b * p, (8 * a * b * p) / 45, (-4/9)) * a * b * p; \\
& (-2/45)) * a * b * p, (2 * a * b * p) / 15, (-2/15)) * a * b * p, (2 * a * b * p) / 15, \dots \\
& (-2/45)) * a * b * p, (8 * a * b * p) / 45, (-1/15)) * a * b * p, (8 * a * b * p) / 45; \\
& (8 * a * b * p) / 45, (-4/9)) * a * b * p, (2 * a * b * p) / 15, (-32/45)) * a * b * p, \dots \\
& (2 * a * b * p) / 15, (-4/9)) * a * b * p, (8 * a * b * p) / 45, (-16/45)) * a * b * p; \\
& (-1/15)) * a * b * p, (8 * a * b * p) / 45, (-2/45)) * a * b * p, (2 * a * b * p) / 15, \dots \\
& (-2/15)) * a * b * p, (2 * a * b * p) / 15, (-2/45)) * a * b * p, (8 * a * b * p) / 45; \\
& (8 * a * b * p) / 45, (-16/45)) * a * b * p, (8 * a * b * p) / 45, (-4/9)) * a * b * p, \dots \\
& (2 * a * b * p) / 15, (-32/45)) * a * b * p, (2 * a * b * p) / 15, (-4/9)) * a * b * p; \\
& (-2/45)) * a * b * p, (8 * a * b * p) / 45, (-1/15)) * a * b * p, (8 * a * b * p) / 45, \dots \\
& (-2/45)) * a * b * p, (2 * a * b * p) / 15, (-2/15)) * a * b * p, (2 * a * b * p) / 15; \\
& (2 * a * b * p) / 15, (-4/9)) * a * b * p, (8 * a * b * p) / 45, (-16/45)) * a * b * p, \dots \\
& (8 * a * b * p) / 45, (-4/9)) * a * b * p, (2 * a * b * p) / 15, (-32/45)) * a * b * p]; \\
ke = kk + kp; \\
rq = [& (-1/3)) * a * b * q; (4 * a * b * q) / 3; (-1/3)) * a * b * q;
\end{aligned}$$

```
(4*a*b*q)/3; -(1/3))*a*b*q; (4*a*b*q)/3;
-(1/3))*a*b*q; (4*a*b*q)/3];
```

MatlabFiles\Chap5\BVPRect8NBCTerm.m

```
function [ka, rb] = BVPRect8NBCTerm(side, alpha, beta, coord)
% [ka, rb] = BVPRect8NBCTerm(side, alpha, beta, coord)
% 8 node rectangular element for 2d BVP
% 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);
x3=coord(3,1); y7=coord(7,2);
a = abs(x3 - x1)/2; b = abs(y7 - y1)/2;
switch (side)
case 1
    ka = [-(4*a*alpha)/15, -(2*a*alpha)/15,...
          (a*alpha)/15, 0, 0, 0, 0, 0;
          -(2*a*alpha)/15, -(16*a*alpha)/15,...
          -(2*a*alpha)/15, 0, 0, 0, 0, 0;
          (a*alpha)/15, -(2*a*alpha)/15, ...
          -(4*a*alpha)/15, 0, 0, 0, 0, 0;
          0, 0, 0, 0, 0, 0, 0, 0;
          0, 0, 0, 0, 0, 0, 0, 0;
          0, 0, 0, 0, 0, 0, 0, 0;
          0, 0, 0, 0, 0, 0, 0, 0;
          0, 0, 0, 0, 0, 0, 0, 0];
    rb = [(a*beta)/3; (4*a*beta)/3; (a*beta)/3;
          0; 0; 0; 0; 0];
case 2
    ka = [0, 0, 0, 0, 0, 0, 0, 0;
          0, 0, 0, 0, 0, 0, 0, 0;
          0, 0, -(4*alpha*b)/15,...
          -(2*alpha*b)/15, (alpha*b)/15, 0, 0, 0;
          0, 0, -(2*alpha*b)/15, -(16*alpha*b)/15, ...
          -(2*alpha*b)/15, 0, 0, 0;
          0, 0, (alpha*b)/15, -(2*alpha*b)/15,...
          -(4*alpha*b)/15, 0, 0, 0;
          0, 0, 0, 0, 0, 0, 0, 0;
          0, 0, 0, 0, 0, 0, 0, 0;
          0, 0, 0, 0, 0, 0, 0, 0];
    rb = [0; 0; (b*beta)/3; (4*b*beta)/3; (b*beta)/3; 0;
          0; 0];
case 3
    ka = [0, 0, 0, 0, 0, 0, 0, 0;
```

```

0, 0, 0, 0, 0, 0, 0, 0;
0, 0, 0, 0, 0, 0, 0, 0;
0, 0, 0, 0, 0, 0, 0, 0;
0, 0, 0, 0,
-((4*a*alpha)/15), -(2*a*alpha)/15,...
(a*alpha)/15, 0;
0, 0, 0, 0, -(2*a*alpha)/15, -(16*a*alpha)/15, ...
-((2*a*alpha)/15), 0;
0, 0, 0, 0, (a*alpha)/15, -(2*a*alpha)/15,...
-((4*a*alpha)/15), 0;
0, 0, 0, 0, 0, 0, 0, 0];
rb = [0; 0; 0; 0; (a*beta)/3; (4*a*beta)/3;
(a*beta)/3; 0];
case 4
ka = [-((4*alpha*b)/15), 0, 0, 0, 0, 0, (alpha*b)/15,...
-((2*alpha*b)/15);
0, 0, 0, 0, 0, 0, 0, 0;
0, 0, 0, 0, 0, 0, 0, 0;
0, 0, 0, 0, 0, 0, 0, 0;
0, 0, 0, 0, 0, 0, 0, 0;
0, 0, 0, 0, 0, 0, 0, 0;
(alpha*b)/15, 0, 0, 0, 0, 0, 0,...
-((4*alpha*b)/15), -(2*alpha*b)/15);
-((2*alpha*b)/15), 0, 0, 0, 0, 0,...
-((2*alpha*b)/15), -(16*alpha*b)/15)];
rb = [(b*beta)/3; 0; 0; 0; 0; 0; (b*beta)/3; (4*b*beta)/3];
end

```

MatlabFiles\Chap5\BVPRect8Results.m

```

function results = BVPRect8Results(coord, dn)
% results = BVPRect8Results(coord, dn)
% 8 node rectangular element for 2d BVP
% Computes element solution
% coord = nodal coordinates
% dn = nodal solution
% Output variables are u and its x and y derivatives at element center
x1=coord(1,1); y1=coord(1,2);
x3=coord(3,1); y7=coord(7,2);
a = abs(x3 - x1)/2; b = abs(y7 - y1)/2;
s = 0; t = 0;
u = [-((a - s)*(b - t)*(b*s + a*(b + t)))/(4*a^2*b^2),...
((a^2 - s^2)*(b - t))/(2*a^2*b),...
-(((a + s)*(b - t)*(a*(b + t) - b*s))/(4*a^2*b^2)),...
((a + s)*(b^2 - t^2))/(2*a*b^2),...
-(((a + s)*(a*(b - t) - b*s)*(b + t))/(4*a^2*b^2)),...
((a^2 - s^2)*(b + t))/(2*a^2*b),...

```

```

-(((a - s)*(b*s + a*(b - t))*(b + t))/(4*a^2*b^2)),...
((a - s)*(b^2 - t^2))/(2*a*b^2)]*dn;
dudx= [((b - t)*(2*b*s + a*t))/(4*a^2*b^2), ...
(s*(t - b))/(a^2*b), -(((b - t)*(a*t - 2*b*s))/(4*a^2*b^2)),...
(b^2 - t^2)/(2*a*b^2),...
((b + t)*(2*b*s + a*t))/(4*a^2*b^2), ...
-((s*(b + t))/(a^2*b)), ...
-(((b + t)*(a*t - 2*b*s))/(4*a^2*b^2)), ...
(t^2 - b^2)/(2*a*b^2)]*dn;
dudy = [((a - s)*(b*s + 2*a*t))/(4*a^2*b^2),...
(s^2 - a^2)/(2*a^2*b),...
((a + s)*(2*a*t - b*s))/(4*a^2*b^2),...
-(((a + s)*t)/(a*b^2)),...
((a + s)*(b*s + 2*a*t))/(4*a^2*b^2),...
(a^2 - s^2)/(2*a^2*b), ...
((a - s)*(2*a*t - b*s))/(4*a^2*b^2),...
((s - a)*t)/(a*b^2)]*dn;
results=[u, dudx, dudy];

```

As an example we use these functions to obtain a finite element solution of the Laplace equation considered in Example 5.1. We take advantage of symmetry and model the left-half of the domain using two 8 node elements as shown in Figure 5.16. (The results should obviously be identical to those obtained by using 4 square elements for the entire domain.) There are 13 nodes and thus we have a total of 13 degrees of freedom.

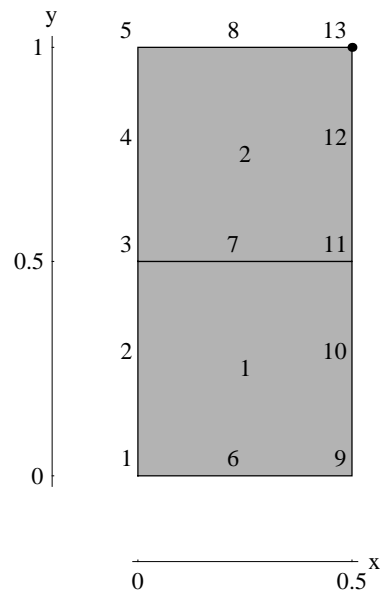


Figure 5.16. Two element model for half of solution domain using 8 node elements

Along line of symmetry (nodes 9 through 13), the normal derivative is 0. From the given essential boundary conditions the following nodal values are known.

Essential boundary conditions {node, value}:

$\{1, 0\} \{2, 0\} \{3, 0\} \{4, 0\} \{5, 0\} \{6, \frac{3}{16}\} \{8, 0\} \{9, \frac{1}{4}\} \{13, 0\}$

MatlabFiles\Chap5\PoissonEqnEx.m

```
% Solution of poisson equation example using
% two 8 node elements (symmetric left half model)
kx = 1; ky = 1; p=0; q=0;
nodes=[0,0; 0,1/4; 0,1/2; 0,3/4;
        0,1; 1/4,0; 1/4,1/2; 1/4,1; 1/2,0; 1/2,1/4;
        1/2,1/2; 1/2,3/4; 1/2,1];
Imm=[1,6,9,10,11,7,3,2;
```

```

    3,7,11,12,13,8,5,4];
debc=[1,2,3,4,5,6,8,9,13]';
ebcVals=[0,0,0,0,0,3/16,0,1/4,0]';
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] = BVPRect8Element(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; BVPRect8Results(nodes(lmm(i,:),:), d(lmm(i,:)))];
end
results

>> PoissonEqnEx

d =

    0
    0
    0
    0
    0
    0.1875
    0.0337
    0
    0.2500
    0.1094
    0.0576
    0.0205
    0

results =

    0.0884    0.2189   -0.3076
    0.0127    0.0410   -0.0674

```

In Figure 5.17 the finite element solution at the element centers is compared with the exact solution with five terms in the sum. The solution obtained earlier in Example 5.1 using 4 node elements is also shown for

comparison. Clearly the 8 node element solution is much closer to the exact solution than that obtained using 4 node elements. The solution can obviously be further improved by using more elements.

Even though both finite element models employ the same number of elements, it obviously takes more effort to obtain solution using higher-order (8 node) elements. In fact the computational effort using eight 4 node elements (15 nodes total) is comparable to that of using two 8 node elements (13 nodes total). Since it is easier to derive and program equations for low-order elements, in practice, use of low-order elements is quite common. One simply uses more elements to improve accuracy. However when more accurate solution derivatives are desired generally higher-order elements are preferred over their low-order counterparts. The reason being that the higher-order elements are based on assumed solutions that involve quadratic and higher-order terms and therefore solution derivatives obtained from these elements may be more accurate even with coarse meshes.

8 node elements	4 node elements	Exact Solution
12.7012	7.8125	13.7286
88.4224	70.3125	83.201

Figure 5.17. Comparison of 4 node and 8 node element solutions with the exact solution ($\times 10^{-3}$)
