RECTANGULAR ELEMENTS SOLUTION FOR POISSON'S EQUATION

Myat Moe Khaing, Ni Ni Win

Abstract— In this paper, firstly introduced the finite element method especially for Poisson's equations are thoroughly discussed with boundary conditions. It is the purpose herein to present brief derivations of the finite element approximation describing a discrete model and computer program using C++ have been written rectangular elements solution is calculated for Poisson's equation.

Key Words: Rectangular Elements Method, Partial Differential Equation, Variational Principle, C++ program

1 INTRODUCTION

Let us consider the field problem Lϕ = f in R, with B ϕ = g on C, the bounding R, L and B are differential operators. The finite element method seeks on approximations,

(x, y), to the exact solution, ϕ (x, y) in a piecewise manner, the approximations being sought in each of total E elements. This in the general element an approximation $\emptyset^e$ (x, y) is sought in such a manner that outside e, $\emptyset^e$(x, y) = 0, e = 1, 2,…, E and it follow that the approximate solution may be written as $\phi(x,y) = \sum_e \phi^e(x,y)$ where the summation is taken over all the elements

2 RECTANGULAR ELEMENT DISCRETIZATION

A simplest rectangular element is one with just four nodes, one at each corner. Choose local coordinates (ξ, η) as shown is fig1. Since there are four nodes with one degree of freedom at each node, the displacement variation throughout the element is of the following bilinear form,

$$\emptyset^e(x,y) = C_0 + C_{1x} + C_{2y} + C_{3xy}$$

- *Myat Moe Khaing is currently working as a Lecturer at Faculty of Computing Department in Myanmar Institute of Information Technology, Myanmar, PH-09422481777 .E-mail: myat_moe_khaing@miit.edu.mm*
- *Ni Ni Win is currently working as a Professor at Faculty of Computing Department in Myanmar Institute of Information Technology, Myanmar, PH-09797486626*

(1)

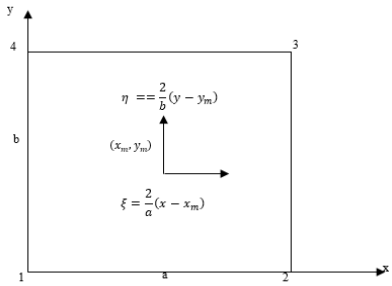


Fig. 1. The four-node rectangle(x_m,y_m) are the coordinates of the mid-point of the rectangle.

Using the Lagrangian interpolation polynomials, the shape functions are obtained as follows:

$$N_1 = \frac{\xi - 1}{-1-1}\frac{\eta - 1}{-1-1} = \frac{(1-\xi)(1-\eta)}{4}$$

Similarly

$$N_2 = \frac{(1-\xi)(1-\eta)}{4}$$

$$N_3 = \frac{(1+\xi)(1+\eta)}{4}$$

$$N_4 = \frac{(1-\xi)(1+\eta)}{4}$$

Then

$$\phi^e(x,y) = N^e\delta^e = N_{1\emptyset1} + N_{2\emptyset2} + N_{3\emptyset3} + N_{4\emptyset4}$$

$$\phi^e(x,y) =$$

$$\frac{1}{4}[(1-(1-\xi))(1-\eta) \quad (1+\xi)(1-\eta) \quad (1+\xi)(1+\eta) \quad (1-\xi)(1-\eta)]\begin{bmatrix}\emptyset_1\\ \emptyset_2\\ \emptyset_3\\ \emptyset_4\end{bmatrix}$$

(2) Now $\frac{\partial}{\partial x} = \frac{2}{a}\frac{\partial}{\partial \xi}$ *and* $\frac{\partial}{\partial y} = \frac{2}{b}\frac{\partial}{\partial \eta}$Thus

$$\alpha = \frac{1}{2}\begin{bmatrix}\frac{-1(1-\eta)}{a} & \frac{(1-\eta))}{a} & \frac{(1+\eta)}{a} & \frac{-(1+\eta)}{a}\\ \frac{-(1-\xi)}{b} & \frac{-(1+\xi)}{b} & \frac{(1+\xi)}{b} & \frac{(1-\xi)}{b}\end{bmatrix}$$

$$k^e = \int_{-}^{1}\int_{-1}^{1} k\alpha^t\alpha\frac{a}{2}d\xi\frac{b}{2}d\eta$$

For the special case k=1,

$$k^e = \frac{1}{6ab}\begin{bmatrix} 2(r+1/r) & r-2/r & -r-1/r & 1/r-2r \\ r-2/r & 2(r+1/r) & 1/(r-2r) & -r-1/r \\ 1/r-2r & r-1/r & r-2/r & 2(r+1/r) \\ 1/r-2r & r-1/r & r-2/r & 2(r+1/r) \end{bmatrix}$$

(3)

where $r = a/b$ is the aspect ratio of the element and

$$f^e = \int_{-1}^{1}\int_{-1}^{1} f(x,y)\, N^t \frac{a}{2}\frac{b}{2} d\xi d\eta$$

(4)

If the element is boundary element and a non-homogeneous mixed boundary condition holds there, then additions are needed to the stiffness and force matrices "(1)". On each side the arc length s is such that

$$ds = -dx = -\frac{a}{2}d\xi$$

$$\bar{k}^e = \int_{-1}^{1} \sigma(s)\frac{1}{16}\begin{bmatrix} 0 \\ 0 \\ 2(1+\xi) \\ 2(1-\xi) \end{bmatrix}[0 \quad 0 \quad 2(1+\xi) \quad 2(1-\xi)]\left(-\frac{a}{2}d\xi\right)$$

(5)

$$\bar{f}^e = \int_{-1}^{1} h\frac{1}{4}\{0 \quad 0 \quad 2(1+\xi) \quad 2(1-\xi)\}\left(-\frac{a}{2}\right)d\xi$$

(6)

There results will now be used to obtain one-element solution of the boundary-value problem. It is possible to set up the element matrices using element solution using a single rectangular element.

3 RECTANGUALR ELEMENTS FOR POISSON'S EQUATION

Let us consider the problem $-\nabla^2\phi = 2(x+y)-4$ in the square whose vertices are at (0,0), (1,0) , (1,1) , (0,1) . The boundary condition are $\phi(0,y) = y^2$, $\phi(1,y) = 1-y$, $\phi(x,0) = x^2$, $\phi(x,1) = 1-x$.

Suppose that the square is divided into four square elements as shown in fig.2.

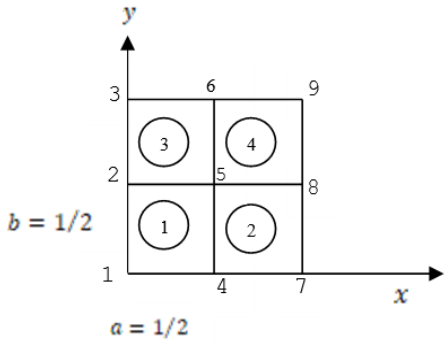


Fig. 2. Four element discretization

Using equations (3) and (4), the element stiffness matrices and force vectors are

$$k^1 = \frac{2}{3}\begin{bmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -1 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{bmatrix}\begin{matrix}1\\4\\5\\2\end{matrix}$$

$$k^2 = \frac{2}{3}\begin{bmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -1 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{bmatrix}\begin{matrix}4\\7\\8\\5\end{matrix}$$

$$k^3 = \frac{2}{3}\begin{bmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -1 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{bmatrix}\begin{matrix}2\\5\\6\\3\end{matrix}$$

$$k^4 = \frac{2}{3}\begin{bmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -1 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{bmatrix}\begin{matrix}5\\8\\9\\6\end{matrix}$$

and the overall stiffness matrix $K = k^1 + k^2 + k^3 + k^4$.

$$K = \frac{2}{3}\begin{bmatrix} 4 & -1 & 0 & -1 & -2 & 0 & 0 & 0 & 0 \\ -1 & 8 & -1 & -2 & -2 & -2 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & -2 & -1 & 0 & 0 & 0 \\ -1 & -2 & 0 & 8 & -2 & 0 & -1 & -2 & 0 \\ -2 & -2 & -2 & -2 & 16 & -2 & -2 & -2 & -2 \\ 0 & -2 & -1 & 0 & -2 & 8 & 0 & -2 & -1 \\ 0 & 0 & 0 & -1 & -2 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & -2 & -2 & -2 & -1 & -8 & -1 \\ 0 & 0 & 0 & 0 & -2 & -1 & 0 & -1 & 4 \end{bmatrix}\begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\\9\end{matrix}$$

$$f^e = \frac{1}{8}(x_m y_m - 2)\{1, \quad 1, \quad 1, \quad 1\} + \frac{y_m}{96}\{-1, \quad 1, \quad 1, \quad -1\} + \frac{x_m}{96}\{-1, \quad -1, \quad 1, \quad 1\} + \frac{1}{1152}\{1, \quad -1, \quad 1, \quad -1\}$$

Now node 5 is the only where a Dirichlet boundary condition does not act; thus only the contributions to the equation corresponding to node 5 need by assembled. Row five of K is

$$\frac{2}{3}[-2 \quad -2 \quad -2 \quad -2 \quad 16 \quad -2 \quad -2 \quad -2 \quad -2]$$

Also $F_5 = f_3^1 + f_4^2 + f_2^3 + f_1^4$

where subscripts refer to the local node numbering given in Fig.1. and

$$f_3^1 = \frac{1}{8}\left(\frac{1}{16} - 2\right) + \frac{1}{192} + \frac{1}{1152}$$

$$f_4^2 = \frac{1}{8}\left(\frac{3}{16} - 2\right) + \frac{1}{192} - \frac{1}{1152}$$

$$f_3^2 = \frac{1}{8}\left(\frac{3}{16} - 2\right) + \frac{1}{192} - \frac{1}{1152}$$

$$f_1^4 = \frac{1}{8}\left(\frac{9}{16} - 2\right) - \frac{1}{192} + \frac{1}{1152}$$

thus $F_5 = -\frac{7}{8}$

since

$$\phi_1 = 0, \phi_2 = \frac{1}{4}, \phi_3 = 1, \phi_4 = \frac{1}{4}$$

$$\phi_6 = \frac{1}{2}, \phi_7 = 1, \phi_8 = \frac{1}{2}, \phi_9 = 0$$

it follows that the equation for ϕ_5 which gives $\phi_5 = 0.355$

The solution at $(\frac{1}{4}, \frac{1}{4})$, $(\frac{1}{4}, \frac{1}{4})$ and $(\frac{1}{4}, \frac{1}{4})$ is found by linear interpolation between nodes 1 and 4, 2 and 5, 4 and 6 respectively. The results are compared with the corresponding results using rectangular elements as shown in Table 1.

Table 1. Comparison of Solutions

(x, y)	**(1/4 , 1/4)**	**(1/4 , 1/4)**	**(1/4 , 1/4)**	**(1/4 , 1/4)**
Four rectan-gular elements	0.241	0.356	0.527	0.339
Exact Solutions	0.094	0.25	0.25	0.281

4 C++ PROGRAM OF RECTANGULAR ELEMENTS SOLUTION FOR POISSION'S EQUATION

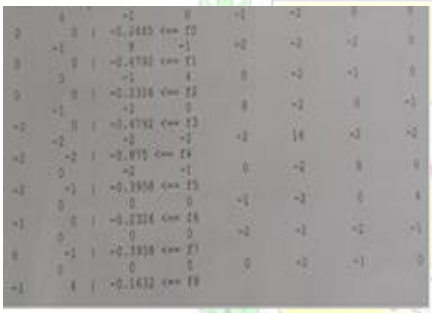
The C++ program for solving Poisson's equation, using four rectangular elements are described.

```
#include<iostream.h>
#include<iomanip.h>
#define ENODE 4
#define maxN 9
ostream &print1x4(float x[ENODE]);
ostream &print4x1(float x[ENODE]);
ostream &print4x4(float x[ENODE][ENODE]);
void MVM (float M[maxN][maxN], float V[maxN], int K, int L,float Y[maxN]);
void MIV(float M[maxN][maxN], int n);
void OutText(float M[maxN][maxN], float V[maxN], int r);
float Inter4(float s, float t, float f1, float f2, float f3, float f4);
void RectK(float K[ENODE][ENODE], float f[ENODE], float x[ENODE], float y[ENODE])
{
  float a,r,b,rpr,rmr,rm2r,xm,ym;
  float z[4][ENODE]= {{1,1,1,1},{-1,1,1,-1}, {-1,-1,1,1},{1,-1,1,-1}};
  int i,j,k;
  a=x[1]-x[0];
  b=y[2]-y[1];
  xm=(x[0]+x[1]+x[2]+x[3])/4;
  ym=(y[0]+y[1]+y[2]+y[3])/4;
  r=a/b;
  rpr=r+1/r;
  rmr=r-2/r;
  rm2r=1/r-2*r;
  a=1/(6*a*b);
  K[0][0]=K[1][1]=K[2][2]=K[3][3]=2*rpr*a;
  K[0][1]=K[1][0]=K[2][3]=K[3][2]=rmr*a;
  K[0][3]=K[1][2]=K[2][1]=K[3][0]=rm2r*a;
  K[0][2]=K[1][3]=K[3][1]=K[2][0]=-rpr*a;
  for(i=0;i<ENODE;i++){
    f[i]=1/8.0 * (xm*ym-2) * z[0][i] +
      ym/96.0 * z[1][i] +
        xm/96.0 * z[2][i] +
        1/1152.0 * z[3][i];}}
int main(){
float x[maxN]={0,0,0,0.5,0.5,0.5,1.0,1.0,1.0} ,xx[ENODE],
y[maxN]={0,0.5,1,0,0.5,1,0,0.5,1},yy[ENODE],f[ENODE],
```

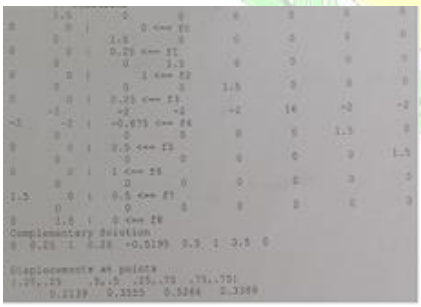
```
u[maxN][2]={{0,0},{0,0.25}, {0,1.0}, {0,0.25}, {1,0}, {0,0.5}, {0,1},{0,0.5},{0,0}},K[ENODE][ENODE], Ks[maxN][maxN],Fs[maxN],PFs[maxN],Un[maxN];

int elem[4][ENODE]= {{0,3,4,1},{3,6,7,4},{1,4,5,2},{4,7,8,5}};

int n=maxN,en=4;int i,j,k; //clear Assemblage Matrices

   for(i=0;i<n;i++){for(j=0;j<n;j++)

   Ks[i][j]=Fs[j]=0;for(i=0;i<en;i++)

  {                for(j=0;j<ENODE;j++)

  {                xx[j]=x[elem[i][j]];

yy[j]=y[elem[i][j]];}

                   RectK(K,f,xx,yy);

                   print1x4(f)<<endl;

                   print4x4(K)<<endl;

          for(j=0;j<ENODE;j++)

  {       for(k=0;k<ENODE;k++)

          Ks[elem[i][j]][elem[i][k]]+=K[j][k];

                        Fs[elem[i][j]]+=f[j];}  }

cout<<"Assembled equations (Before Boundary Condition)" << endl;

OutText(Ks,Fs,n); for(i=0;i<n;i++){

                   if(u[i][0]==0){for(j=0;j<n;j++)

                        Ks[i][j]=0;

                        Ks[i][i]=1;

                        Fs[i]=u[i][1];}

                        PFs[i]=Fs[i];}

          cout<<"Assembled equations "<<endl;

          OutText(Ks,Fs,n);

          MIV(Ks,n);

          MVM(Ks,Fs,n,n,Un);

          for(i=0;i<n;i++){cout<<Un[i]<< " ";}

          Un[4]-=Fs[4];

          cout<< endl;

          cout<< "(.25,.25,.5,.5,.25,.75,.75,.75)" << endl;

cout<<setw(9)<<(Inter4(0,0,Un[0],Un[3],Un[4],Un[1]))
  <<setw(9)<<Un[4]<<setw(9)<<(Inter4(0,0,Un[1],Un[4],Un[5],Un[2]))<<setw(9)<<(Inter4(0,0,Un[4],Un[7],Un[8],Un[5]))<<endl;

          return 1;}}

  ostream &print1x4(float x[ENODE]){int i;

          cout<<setprecision(3);

  for(i=0;i<ENODE;i++)cout<<setw(8)<<x[i]<<" ";

          return cout;}

  ostream &print4x1(float x[ENODE]){int i;

  cout<<setprecision(3);for(i=0;i<ENODE;i++)

          cout<<setw(8)<<x[i]<<endl;

          return cout;}

  ostream &print4x4(float x[ENODE][ENODE]){

  int i;for(i=0;i<ENODE;i++)print1x4(x[i])<<endl;

          return cout;}

void OutText(float M[maxN][maxN], float V[maxN], int r)

{    int i,j;

          cout<<setiosflags(ios::fixed);

          cout<<setprecision(4);

          cout<<endl;

          for(i=0;i<r;i++){for(j=0;j<r;j++)

          cout<<setw(8)<<M[i][j]<<" ";

cout<<" | "<<setw(8) << V[i] << " <== f" <<i<<endl;}}

float Inter4(float s, float t, float f1, float f2, float f3, float f4)

{ float f;

  f=(1-s)*(1-t)*f1+ (1+s)*(1-t)*f2+(1+s)*(1+t)*f3+

          (1-s)*(1+t)*f4;return f/4;}

void MVM(float M[maxN][maxN], float V[maxN], int K,      int L ,float Y[maxN])

{

          float X; int i,j;

          for(i=0;i<K;i++){X=0.0;

                    for(j=0;j<L;j++)

                              X +=M[i][j] * V[j];

                    Y[i]=X;}}
```

```
void MIV(float M[maxN][maxN], int n)
{
#define FORJN for(j=0;j<n;j++)
#define FORKN for(k=0;k<n;k++)
#define FORIN for(i=0;i<n;i++)
#define M(i,j) M[(i)][(j)]int i,j,k; float KII;
  FORKN{KII=M(k,k);M(k,k)=1;
  FORJN
        M(k,j)/=KII;
        FORIN{if(i!=k){KII=M(i,k);M(i,k)=0;
                FORJN
        M(i,j)-=-M(k,j)*KII;}}}}
```

Assembled equations (Before boundary Conditions) are



Out put of the Assembled equations are



5 CONCLUSION

In the above mentioned results, rectangular elements solution and C++ program solution are the same for field problems. Finite element approximation for field problems improvements are made by refining the finite element mesh. Higher-order element may be introduced to get a better polynomial approximation but integrands involved would be necessary to use numerical integration. Using this program, we can easy to know the solution of the other points of this problem.

It is reasonable to expect that the proposed finite elements model is a very good approximation of the continuum. This method is now very widely used, and forms the basis of most calculations.

ACKNOWLEDGMENT

I would like to take this opportunity to express great appreciation to Rector, Kalay Technological University for giving me the chance to take part in this journal. I would like to include my acknowledgements by expressing great appreciation for Professor Dr. Ni Ni Win, Head of Faculty of Computing, Myanmar Institute of Information Technology, Mandalay. Finally, I would like to acknowledgement my thanks to minister, teacher, colleagues and friends.

REFERENCES

[1] Davies. Alan, J. (1980) "The Finite Element method, A First Approach". Oxford University Press, New York.

[2] Gurtin, M.E. (1964) "Variational Principles for Linear Initial Value Problems".

[3] Hall, C.A and Heinrich, J. (1978) "A finite element that satisfies not Ural boundary conditions exactly". J. Inst Math's Apples 21,237-50.

[4] Mrrchell, A.R and Warr.R (1977) "The Finite Element Method in Partial Differential Equations" Wiley.

[5] Smith G.D. (1978) "Numerical solution of partial differential equations: finite difference methods". Oxford University Press.