FEM 计算二维传热问题

胡金山,朱青云,余治国
（西安空军工程大学工程学院,西安 710038）

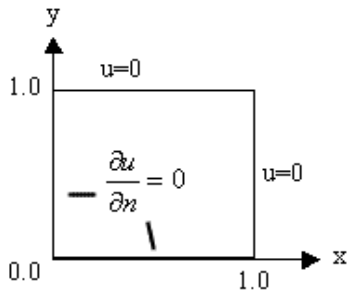
本文所涉及的有限元基本理论请参考章本照先生编著的<<流体力学中的有限元方法>>, PP.156-165。

一. 二维传热问题

1、在边长为 1 的正方形中，(1,0)—(1,1)边上和(0,1)—(1,1)边上，温度 u=0；在(1,0)—(0,0)边上和(0,1)—(0,0)边上，$\left(\frac{\partial u}{\partial n}\right)=0$，求解 Laplace 方程：$-\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=1$

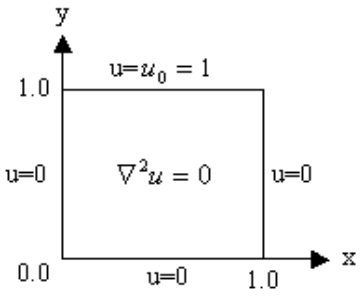
比较有限元解和精确解：

$$u(x,y)=\frac{1}{2}\left\{\left(1-y^2\right)+\frac{32}{\pi^2}\cdot\sum_{n=1}^{\infty}\frac{(-1)^n\cdot\cos\left[(2n-1)\pi y/2\right]\cdot\cosh\cdot\left[(2n-1)\pi x/2\right]}{(2n-1)^3\cdot\cosh\cdot(2n-1)\pi/2}\right\}$$



2、当 u(0,y) = u(1,y) = u(x,0) = 0 和 u(x,1) = u_0 时，在边长为 1 的正方形中，求解 Laplace 方程：$-\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=0$

比较有限元解和精确解：$u(x,y)=\frac{4}{\pi}\sum_{n=0}^{\infty}\frac{\sin\left[(2n+1)\pi x\right]\sinh\left[(2n+1)\pi y\right]}{(2n+1)\sinh\left[(2n+1)\pi\right]}$



单元剖分示意：（实际划分 24*24*2=1152 个单元，25*25＝625 个结点）

二. 解题过程

1、对结构进行离散化，将待分析的结构物从几何上用线或面划分为有限个单元，按结构物的不同和分析要求，选取不同形式的单元，在单元的边界上设置节点，并书写编号。计算节点坐标

2、单元分析：设法导出单元的结点位移和结点力之间的关系，建立单元刚度矩阵。

单元刚度矩阵的计算：

对于方程

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=p & (x,y)\in\Omega \\ u|_{\Gamma_1}=\bar{u} \\ \dfrac{\partial u}{\partial n}|_{\Gamma_2}=g \end{cases}$$

采用 Galerkin 弱解表达式

$$\iint_\Omega\left[\frac{\partial u}{\partial x}\frac{\partial(\delta uu)}{\partial x}+\frac{\partial u}{\partial y}\frac{\partial(\delta uu)}{\partial y}\right]d\Omega+\iint_\Omega p\,\delta uu\,d\Omega=\int_{\Gamma_2} g\,\delta uu\,d\Gamma \qquad (*)$$

这里采用三节点的三角形单元，单元的基函数共有三个，选用插值多项式

$$\Phi_i^{(e)}=a_i^{(e)}+b_i^{(e)}x+c_i^{(e)}y \qquad (i=1,2,3)$$

分别代入单元三个节点的坐标可解得

$$a_i^{(e)}=\frac{1}{D}\begin{vmatrix}1 & x_i^{(e)} & y_i^{(e)}\\ 0 & x_j^{(e)} & y_j^{(e)}\\ 0 & x_k^{(e)} & y_k^{(e)}\end{vmatrix}=\frac{1}{D}(x_j^{(e)}y_k^{(e)}-x_k^{(e)}y_j^{(e)})$$

$$b_i^{(e)}=\frac{1}{D}(y_j^{(e)}-y_k^{(e)})$$

$$c_i^{(e)}=\frac{1}{D}(x_k^{(e)}-x_j^{(e)})$$

其中

$$D=\begin{vmatrix}1 & x_i^{(e)} & y_i^{(e)}\\ 1 & x_j^{(e)} & y_j^{(e)}\\ 1 & x_k^{(e)} & y_k^{(e)}\end{vmatrix}=2A^{(e)}$$

$$A^{(e)}=\frac{1}{2}\left[(x_j^{(e)}-x_i^{(e)})(y_k^{(e)}-y_i^{(e)})-(y_j^{(e)}-y_i^{(e)})(x_k^{(e)}-x_i^{(e)})\right]$$

e 单元中的近似函数为

$$u^{(e)}=\sum u_i^{(e)}\Phi_i^{(e)} \qquad (**)$$

将式（*）中的积分区域取为 e 单元的区域$\Omega^{(e)}$，并将单元中的近似函数表达式（**）

代入，并注意到$\delta uu_i^{(e)}$的任意性，可得

$$u_j^{(e)}\iint_{\Omega^{(e)}}\left(\frac{\partial\Phi_i^{(e)}}{\partial x}\frac{\partial\Phi_j^{(e)}}{\partial x}+\frac{\partial\Phi_i^{(e)}}{\partial y}\frac{\partial\Phi_j^{(e)}}{\partial y}\right)dxdy$$

$$=-\iint_{\Omega^{(e)}}p\Phi_i^{(e)}dxdy+\int_{\Gamma_2^{(e)}}g\Phi_i^{(e)}d\Gamma$$

记 $A_{ij}^{(e)} = \iint_{\Omega^{(e)}} \left(\frac{\partial \Phi_i^{(e)}}{\partial x}\frac{\partial \Phi_j^{(e)}}{\partial x} + \frac{\partial \Phi_i^{(e)}}{\partial y}\frac{\partial \Phi_j^{(e)}}{\partial y}\right) dxdy$ （***）

$$f_i^{(e)} = -\iint_{\Omega^{(e)}} p\Phi_i^{(e)} dxdy + \int_{\Gamma_2^{(e)}} g\Phi_i^{(e)} d\Gamma \quad (****)$$

将单元基函数的具体表达式（*）代入（***）式中，可得

$$A_{ij}^{(e)} = (b_i^{(e)} b_j^{(e)} + c_i^{(e)} c_j^{(e)}) A^{(e)}$$

通过等参变换（具体见文献 1 第 201 页），可得

$$f_i^{(e)} = -p\int_{\Delta} \xi_1^l \xi_2^m \xi_3^n d\sigma = 2A\frac{l!m!n!}{(l+m+n+2)!}$$

这里指 p 为常数的情况，A 为三角形单元的面积。

$$I_i = \int_{\Gamma_2^{(e)}} g\Phi_i^{(e)} d\Gamma$$

这里 g 均为 0，所以此项不用计算。

3、整体分析（以求结点力为例）

整体分析就是将各个单元组成结构整体进行分析。整体分析的目的在于导出整个结构结点位移与结点力之间的关系，建立整个结构的刚度方程。

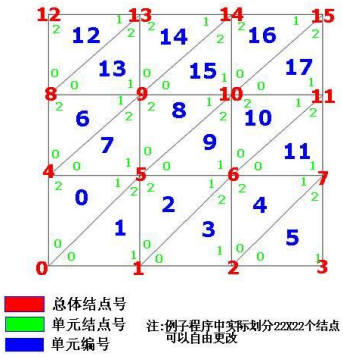
分析步骤：首先按着一定的集成规则，将各单元刚度矩阵集合成结构整体刚度矩阵，并将单元等效结点荷载集合成整体等效结点荷载列阵；然后引入结构的位移边界条件，求解整体平衡方程组，得出基本未知量——结点位移列阵。

4、用选定的算法语言编写出程序（C/C++），调试程序调用高斯消元法解方程的出结果。

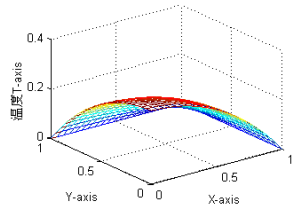
附件程序 Fem1.cpp 计算了积分值，Fem2.cpp 则采用了面积坐标下的插值函数，积分值取为三角形面积的三分之一。两者结果相同，但是后者更为通用，可以把程序用于其他形状的二维区域的有限元计算，Fem3.cpp 计算了题 2。

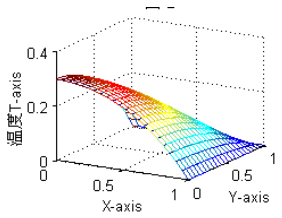
三．单元网格划分

四边形单元网格划分单元网格划分示意图 1



计算结果结果数据可视化如图 2，3。它们是题 1 分别用 Fem1.cpp 程序和 Fem2.cpp 程序计算结果的 Matlab 数据可视化图，它们表现的数据基本一致，观察视点不同。图 4 是题 2 的解。





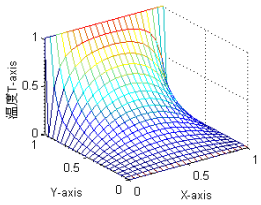


图 2　　图 3　　图 4

利用此程序的基本框架，我们还成功地解算了三角形、椭圆形区域的有限元问题。

参考文献：

1. 章本照. 流体力学中的有限元方法[M]. 机械工业出版社, 1986

Fem1.cpp

```
// Finite Element Method solve the Poisson formula
// If this program works, it was written by hujinshan@2003.5.9
// If not, I don't know who wrote it :)
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

int Gauss(double a[],double b[],int n);      //全选主元高斯消去法

double GaussIntegral(int n,int js[],    //计算多重积分的高斯方法
                     double(*fn)(int n,double x[]),
                     void(*fnGaussLimit)(int j, int n,double x[],double y[]));

struct ETNode{                                  //单元结点结构体
      double x,y;                               //单元结点坐标
      int number;                               //单元结点在总体区域划分中的编号
                  };

struct ElementTriangle{                         //三角形单元结构体
      ETNode nd[3];                             //存储相对应的总体结点号
      double   a[3],b[3],c[3];                  //基函数的系数
      double   A;                               //单元面积
      double   Aij[3][3];                       //单元有限元特征式系数矩阵
      double   fi[3];                           //单元上的积分值
                        };

//-------------   全局变量   --------------------------
      int   i,j,k,                              //循环指标
            id;                                 //单元的循环指标
      const int nx=21,ny=21;       //x,y 方向划分网格数,三角形单元个数=nx*ny*2
      const double Lx=1,Ly=1;                   //矩形区域的两边长
      double   D;                               //单元矩阵行列式值
```

```
91          const int iNode=(nx+1)*(ny+1);          //结点个数
92          double* pMatrix;                        //总体矩阵指针
93          double* pMf;                            //f 向量指针
94          ElementTriangle* pE;                    //单元三角形结构体数组指针
95          double ai,bi,ci;                        //基函数的系数
96      //-----------------------------------------------------
97
98      double fn(int n,double x[2])                //被积函数,高斯积分函数的参数
99      {
100         return(ai+bi*x[0]+ci*x[1]);
101     }
102
103     void fnGaussLimit(int jFlag,int n,      double x[],double y[2])
104     {//积分上下限函数,高斯积分函数的参数//矩形中第一个三角形单元积分上下限计算
105         switch(jFlag)
106         {
107         case 0:{
108             y[0]=(Lx/nx)*i;
109             y[1]=(Lx/nx)*(i+1);
110             break;
111                 }
112         case 1:{
113             y[0]=(Ly/ny)*j+ x[0]-(Lx/nx)*i;
114             y[1]=(Ly/ny)*(j+1);
115             break;
116                 }
117         default:
118             break;
119         }
120     }
121
122     void fnGaussLimit2(int jFlag,int n, double x[],double y[2])
123     {     //积分上下限函数,高斯积分函数的参数//矩形中第二个三角形单元积分上下限计算
124         switch(jFlag)
125         {
126         case 0:{
127             y[0]=(Lx/nx)*i;
128             y[1]=(Lx/nx)*(i+1);
129             break;
130                 }
131         case 1:{
132             y[0]=(Ly/ny)*j;
133             y[1]=(Ly/ny)*j+ x[0]-(Lx/nx)*i;
134             break;
```

```
                }
        default:
                break;
        }
}

//--------------------  主程序  -------------------------
//有限元理论请参考章本照先生编著的<<流体力学中的有限元方法>>, PP.156-165
//机械工业出版社出版,1986
void main(void)
{
        //为总体矩阵,三角形单元数组,f 函数向量分配存储内存
        pMatrix=(double*)malloc(iNode*iNode*sizeof(double));
        pE=(ElementTriangle*)malloc(nx*ny*2*sizeof(ElementTriangle));
        pMf=(double*)malloc(iNode*sizeof(double));
        //初始化值为 0,因为下面要累加总体矩阵
        for(i=0;i<iNode*iNode;i++)
                pMatrix[i]=0;
        for(i=0;i<iNode;i++)
                pMf[i]=0;

try{
        //------    计算得到网格的信息  -----------
        for(j=0;j<nx;j++)
        for(i=0;i<ny;i++)
        {
                //for the first triangle in the rectangle
                pE[i*2+j*ny*2].nd[0].x=(Lx/nx)*i;
                pE[i*2+j*ny*2].nd[0].y=(Ly/ny)*j;
                pE[i*2+j*ny*2].nd[0].number=i+j*(nx+1);                 //NO.0
                pE[i*2+j*ny*2].nd[1].x=(Lx/nx)*(i+1);
                pE[i*2+j*ny*2].nd[1].y=(Ly/ny)*(j+1);
                pE[i*2+j*ny*2].nd[1].number=i+1+(nx+1)*(j+1);   //NO.1
                pE[i*2+j*ny*2].nd[2].x=(Lx/nx)*i;
                pE[i*2+j*ny*2].nd[2].y=(Ly/ny)*(j+1);
                pE[i*2+j*ny*2].nd[2].number=i+(nx+1)*(j+1);       //NO.2
                //for the second triangle in the rectangle
                pE[i*2+j*ny*2+1].nd[0].x=(Lx/nx)*i;
                pE[i*2+j*ny*2+1].nd[0].y=(Ly/ny)*j;
                pE[i*2+j*ny*2+1].nd[0].number=i+j*(nx+1);         //NO.0
                pE[i*2+j*ny*2+1].nd[1].x=(Lx/nx)*(i+1);
                pE[i*2+j*ny*2+1].nd[1].y=(Ly/ny)*j;
                pE[i*2+j*ny*2+1].nd[1].number=i+j*(nx+1)+1;           //NO.1
                pE[i*2+j*ny*2+1].nd[2].x=(Lx/nx)*(i+1);
```

```
        pE[i*2+j*ny*2+1].nd[2].y=(Ly/ny)*(j+1);
        pE[i*2+j*ny*2+1].nd[2].number=i+1+(nx+1)*(j+1);       //NO.2
    }
    //---------------------------------------------------------
    //please turn to page 158 for more details
    printf("计算基函数系数值...\n");
    for(id=0;id<nx*ny*2;id++)
    {
        for(i=0;i<3;i++)
        {
            if(i==0)        j=1,k=2;
            else if(i==1)   j=2,k=0;
            else if(i==2)   j=0,k=1;

            pE[id].A=( (pE[id].nd[j].x-pE[id].nd[i].x)*(pE[id].nd[k].y-pE[id].nd[i].y)-
                (pE[id].nd[j].y-pE[id].nd[i].y)*(pE[id].nd[k].x-pE[id].nd[i].x) )/2.0;
            D=2.0*pE[id].A;
            pE[id].a[i]=( pE[id].nd[j].x*pE[id].nd[k].y- pE[id].nd[k].x*pE[id].nd[j].y )/D;
            pE[id].b[i]=( pE[id].nd[j].y-pE[id].nd[k].y )/D;
            pE[id].c[i]=( pE[id].nd[k].x-pE[id].nd[j].x )/D;
        }
    }printf("OK!\n");
    printf("计算单元有限元特征式系数矩阵...\n");
    int l,m;
    for(i=0;i<nx;i++)                   //计算单元有限元特征式系数矩阵
    for(j=0;j<ny;j++)
    {
        for(l=0;l<3;l++)                //for the first triangle in the rectangle
            for(m=0;m<3;m++)
            {
                pE[i*2+j*ny*2].Aij[l][m]=( pE[i*2+j*ny*2].b[l]*pE[i*2+j*ny*2].b[m] +
                    pE[i*2+j*ny*2].c[l]*pE[i*2+j*ny*2].c[m] ) * pE[i*2+j*ny*2].A;
            }
        for(l=0;l<3;l++)                //for the second triangle in the rectangle
            for(m=0;m<3;m++)
            {
    pE[i*2+j*ny*2+1].Aij[l][m]=( pE[i*2+j*ny*2+1].b[l]*pE[i*2+j*ny*2+1].b[m] +
        pE[i*2+j*ny*2+1].c[l]*pE[i*2+j*ny*2+1].c[m] ) * pE[i*2+j*ny*2+1].A;
            }
    }printf("OK!\n");
    printf("计算积分值,填充到 f 函数向量数组...\n");
    static int js[2]={4,4};             //每一层积分区间均分为 4 个子区间
    int idx=0;
    for(i=0;i<nx;i++)                   //计算积分值,填充到 f 函数向量数组
```

```
223             for(j=0;j<ny;j++)
224             {
225                 for(idx=0;idx<3;idx++)       //for the first triangle in the rectangle
226                 {
227                     ai=pE[i*2+j*ny*2].a[idx];
228                     bi=pE[i*2+j*ny*2].b[idx];
229                     ci=pE[i*2+j*ny*2].c[idx];
230                     pE[i*2+j*ny*2].fi[idx]=GaussIntegral(2,js,fn,fnGaussLimit);
231                 }
232                 for(idx=0;idx<3;idx++)       //for the second triangle in the rectangle
233                 {
234                     ai=pE[i*2+j*ny*2+1].a[idx];
235                     bi=pE[i*2+j*ny*2+1].b[idx];
236                     ci=pE[i*2+j*ny*2+1].c[idx];
237                     pE[i*2+j*ny*2+1].fi[idx]=GaussIntegral(2,js,fn,fnGaussLimit2);
238                 }
239             }printf("OK!\n");
240
241         //单元矩阵元素累加到总体矩阵相应的位置上
242         printf("单元矩阵元素累加到总体矩阵相应的位置上...\n");
243         for(idx=0;idx<nx*ny*2;idx++)
244             for(i=0;i<3;i++)
245             {
246                 for(j=0;j<3;j++)
247             pMatrix[pE[idx].nd[i].number*iNode+pE[idx ].nd[j].number] +=pE[idx].Aij[i][j];
248             pMf[ pE[idx].nd[i].number ]+=pE[idx].fi[i];
249             }
250         printf("OK!\n");
251
252         double dBig=pow(10,20);                    //边界条件对角线扩大法处理所用的大数
253         double Ur=1.0;          //边界条件 1(边界条件 2 通过 Galerkin 弱解表达式自动满足)
254         for(i=0;i<nx+1;i++)
255         {   j=nx+1;
256             pMatrix[(j*nx+i)*iNode+(j*nx+i)]*=dBig;
257             pMf[(j*nx+i)]*=dBig*Ur;
258         }
259         for(j=0;j<nx+1;j++)
260         {   i=(nx+1)*(j+1)-1;
261             pMatrix[i*iNode+i]*=dBig;
262             pMf[i]*=dBig*Ur;
263         }
264         printf("调用全选主元高斯消去法函数解方程组...\n");
265         Gauss(pMatrix,pMf,iNode);           //调用全选主元高斯消去法函数解方程组
266         printf("OK!\n");
```

```
267         printf("写计算结果数据到文件...\n");
268         FILE *wfp;                                  //文件操作
269         if((wfp=fopen("dat.txt","w+"))==NULL)
270             printf("Cann't open the file... ");
271         //fprintf(wfp,"计算得各结点上的温度值为:\n");
272         //for(i=0;i<iNode;i++)fprintf(wfp,"%d     %f\n",i,pMf[i]);
273     //--------输出为 Matlab 数组格式------------------
274         for(i=0;i<iNode;i++)fprintf(wfp,"%f\n",pMf[i]);
275     /*  fprintf(wfp,"[");
276         for(i=0;i<nx+1;i++)
277         {
278             for(j=0;j<nx+1;j++)
279                 fprintf(wfp,"%f ",pMf[i]);
280             fprintf(wfp,";\n");
281
282         }
283         fprintf(wfp,"]");*/
284         //-----------------------------------------
285         printf("OK!\n");
286         fclose(wfp);
287     }
288     catch(...)
289     {
290         printf("Error occured...\n");
291     }
292         //释放总体矩阵和三角形单元数组占用内存
293         free(pMf);     free(pE); free(pMatrix);
294         printf("Please press Enter to exit...");
295         getchar();
296     }
297     //----------   全选主元高斯消去法   ----------------------------
298     //    a 体积为 n*n 的双精度实型二维数组，方程组系数矩阵，返回时将被破坏
299     //    b 长度为 n 的双精度实型一维数组，方程组右端的常数向量，返回方程组的解向量
300     //    n 整型变量，方程组的阶数
301     //-------------------------------------------------------------
302     int Gauss(double a[],double b[],int n)
303     {
304         int *js,l,k,i,j,is,p,q;
305         double d,t;
306         js=(int*)malloc(n*sizeof(int));
307         l=1;
308         for(k=0;k<=n-2;k++)
309         {
310             d=0.0;
```

```
311             for(i=k;i<=n-1;i++)
312                 for(j=k;j<=n-1;j++)
313                 {
314                     t=fabs(a[i*n+j]);
315                     if(t>d) { d=t; js[k]=j; is=i;}
316                 }
317                 if(d+1.0==1.0) l=0;
318                 else
319                 {
320                     if(js[k]!=k)
321                         for(i=0;i<=n-1;i++)
322                         {
323                             p=i*n+k; q=i*n+js[k];
324                             t=a[p]; a[p]=a[q]; a[q]=t;
325                         }
326                         if(is!=k)
327                         {
328                             for(j=k;j<=n-1;j++)
329                             {
330                                 p=k*n+j; q=is*n+j;
331                                 t=a[p]; a[p]=a[q]; a[q]=t;
332                             }
333                             t=b[k]; b[k]=b[is]; b[is]=t;
334                         }
335                 }
336                 if(l==0)
337                 {
338                     free(js);
339                     printf("Gauss funtion failed 1...\n");
340                     return(0);
341                 }
342                 d=a[k*n+k];
343                 for(j=k+1;j<=n-1;j++)
344                 {
345                     p=k*n+j; a[p]=a[p]/d;
346                 }
347                 b[k]=b[k]/d;
348                 for(i=k+1;i<=n-1;i++)
349                 {
350                     for(j=k+1;j<=n-1;j++)
351                     {
352                         p=i*n+j;
353                         a[p]=a[p]-a[i*n+k]*a[k*n+j];
354                     }
```

```
                    b[i]=b[i]-a[i*n+k]*b[k];
                }
        }
        d=a[(n-1)*n+n-1];
        if(fabs(d)+1.0==1.0)
        {
            free(js);
            printf("Gauss funtion failed 2...\n");
            return(0);
        }
        b[n-1]=b[n-1]/d;
        for(i=n-2;i>=0;i--)
        {
            t=0.0;
            for(j=i+1;j<=n-1;j++)
                t=t+a[i*n+j]*b[j];
            b[i]=b[i]-t;
        }
        js[n-1]=n-1;
        for(k=n-1;k>=0;k--)
            if(js[k]!=k)
            {
                t=b[k]; b[k]=b[js[k]]; b[js[k]]=t;
            }
        free(js);
        return(1);
}
//------------  计算多重积分的高斯方法  ------------------------
//    n  整型变量，积分重数
//    js  整型一维数组，长度为 n，
//        其 js(i) (i=0,1,...,n-1)表示第 i 层积分区间所划分的子区间个数
//    fn()  被积函数(函数指针)
//    fnGaussLimit()  函数(函数指针)计算各层积分上下限值(上限>下限)
//-------------------------------------------------------------
double GaussIntegral(int n,int js[],double (*fn)(int n,double x[]),
void (*fnGaussLimit)(int j,int n,double x[],double y[]))
{
        int m,j,k,q,*is;
        double y[2],p,s,*x,*a,*b;
        static double t[5]={-0.9061798459,-0.5384693101,0.0,0.5384693101,0.9061798459};
        static double
c[5]={0.2369268851,0.4786286705,0.5688888889,0.4786286705,0.2369268851};
        is=(int*)malloc(2*(n+1)*sizeof(int));
        x=(double*)malloc(n*sizeof(double));
```

```
399         a=(double*)malloc(2*(n+1)*sizeof(double));
400         b=(double*)malloc((n+1)*sizeof(double));
401         m=1;a[n]=1.0; a[2*n+1]=1.0;
402         while(true)
403         {
404             for(j=m;j<=n;j++)
405             {
406                 fnGaussLimit(j-1,n,x,y);
407                 a[j-1]=0.5*(y[1]-y[0])/js[j-1];
408                 b[j-1]=a[j-1]+y[0];
409                 x[j-1]=a[j-1]*t[0]+b[j-1];
410                 a[n+j]=0.0;    is[j-1]=1;is[n+j]=1;
411             }
412             j=n; q=1;
413             while(q==1)
414             {
415                 k=is[j-1];
416                 if(j==n) p=fn(n,x);
417                 else p=1.0;
418                 a[n+j]=a[n+j+1]*a[j]*p*c[k-1]+a[n+j];
419                 is[j-1]=is[j-1]+1;
420                 if(is[j-1]>5)
421                 {
422                     if(is[n+j]>=js[j-1])
423                     {
424                         j=j-1; q=1;
425                         if(j==0)
426                         {
427                             s=a[n+1]*a[0];
428                             free(is); free(x);free(a); free(b);
429                             return(s);
430                         }
431                     }
432                     else
433                     {
434                         is[n+j]=is[n+j]+1;
435                         b[j-1]=b[j-1]+a[j-1]*2.0;
436                         is[j-1]=1; k=is[j-1];
437                         x[j-1]=a[j-1]*t[k-1]+b[j-1];
438                         if(j==n)
439                             q=1;
440                         else
441                             q=0;
442                     }
```

```
                }
                else
                {
                    k=is[j-1];
                    x[j-1]=a[j-1]*t[k-1]+b[j-1];
                    if(j==n)
                        q=1;
                    else
                        q=0;
                }
            }
            m=j+1;
        }
    return 0;
}
//-------------------------------------------------------------
```

FEM2.cpp

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

int Gauss(double a[],double b[],int n);      //全选主元高斯消去法

/*double GaussIntegral(int n,int js[], //计算多重积分的高斯方法
                     double(*fn)(int n,double x[]),
                     void(*fnGaussLimit)(int j, int n,double x[],double y[]));    */

struct ETNode{                                  //单元结点结构体
    double x,y;                                 //单元结点坐标
    int number;                                 //单元结点在总体区域划分中的编号
};

struct ElementTriangle{                         //三角形单元结构体
    ETNode nd[3];                               //存储相对应的总体结点号
    double  a[3],b[3],c[3];                 //基函数的系数
    double  A;                                  //单元面积
    double  Aij[3][3];                      //单元有限元特征式系数矩阵
    double  fi[3];                              //单元上的积分值
};

//--------------   全局变量   --------------------------
    int  i,j,k,                             //循环指标
         id;                                    //单元的循环指标
```

```
    const int nx=21,ny=21;                  //x,y 方向划分网格数,三角形单元个数=nx*ny*2
    const double Lx=1,Ly=1;                     //矩形区域的两边长
    double    D;                                //单元矩阵行列式值
    const int iNode=(nx+1)*(nx+1);          //结点个数
    double* pMatrix;                        //总体矩阵指针
    double* pMf;                            //f 向量指针
    ElementTriangle* pE;                    //单元三角形结构体数组指针
    double ai,bi,ci;                        //基函数的系数
//-----------------------------------------------------

double fn(int n,double x[2])                //被积函数,高斯积分函数的参数
{
    return(ai+bi*x[0]+ci*x[1]);
}

/*
void fnGaussLimit(int jFlag,int n,              //积分上下限函数,高斯积分函数的参数
                  double x[],double y[2])
{                                                   //矩形中第一个三角形单元积分上下限
计算
    switch(jFlag)
    {
    case 0:{
        y[0]=(Lx/nx)*i;
        y[1]=(Lx/nx)*(i+1);

        break;
            }
    case 1:{
        y[0]=(Ly/ny)*j+ x[0]-(Lx/nx)*i;
        y[1]=(Ly/ny)*(j+1);

        break;
            }
    default:
        break;
    }
}

void fnGaussLimit2(int jFlag,int n, double x[],double y[2])
{//积分上下限函数,高斯积分函数的参数//矩形中第二个三角形单元积分上下限计算
    switch(jFlag)
    {
    case 0:{
```

```
            y[0]=(Lx/nx)*i;
            y[1]=(Lx/nx)*(i+1);
            break;
                }
        case 1:{
            y[0]=(Ly/ny)*j;
            y[1]=(Ly/ny)*j+ x[0]-(Lx/nx)*i;
            break;
                }
        default:
            break;
        }
    }
    */

    //--------------------   主程序   -------------------------
    //有限元理论请参考章本照先生编著的<<流体力学中的有限元方法>>, PP.156-165
    //机械工业出版社出版,1986

    void main(void)
    {
        //为总体矩阵,三角形单元数组,f 函数向量分配存储内存
        pMatrix=(double*)malloc(iNode*iNode*sizeof(double));
        pE=(ElementTriangle*)malloc(nx*ny*2*sizeof(ElementTriangle));
        pMf=(double*)malloc(iNode*sizeof(double));
        //初始化值为 0,因为下面要累加总体矩阵
        for(i=0;i<iNode*iNode;i++)
            pMatrix[i]=0;
        for(i=0;i<iNode;i++)
            pMf[i]=0;

    try{
        //------    计算得到网格的信息   -----------
        for(j=0;j<nx;j++)
        for(i=0;i<ny;i++)
        {
            //for the first triangle in the rectangle
            pE[i*2+j*ny*2].nd[0].x=(Lx/nx)*i;
            pE[i*2+j*ny*2].nd[0].y=(Ly/ny)*j;
            pE[i*2+j*ny*2].nd[0].number=i+j*(nx+1);                  //NO.0
            pE[i*2+j*ny*2].nd[1].x=(Lx/nx)*(i+1);
            pE[i*2+j*ny*2].nd[1].y=(Ly/ny)*(j+1);
            pE[i*2+j*ny*2].nd[1].number=i+1+(nx+1)*(j+1);    //NO.1
            pE[i*2+j*ny*2].nd[2].x=(Lx/nx)*i;
```

```
574             pE[i*2+j*ny*2].nd[2].y=(Ly/ny)*(j+1);
575             pE[i*2+j*ny*2].nd[2].number=i+(nx+1)*(j+1);       //NO.2
576             //for the second triangle in the rectangle
577             pE[i*2+j*ny*2+1].nd[0].x=(Lx/nx)*i;
578             pE[i*2+j*ny*2+1].nd[0].y=(Ly/ny)*j;
579             pE[i*2+j*ny*2+1].nd[0].number=i+j*(nx+1);         //NO.0
580             pE[i*2+j*ny*2+1].nd[1].x=(Lx/nx)*(i+1);
581             pE[i*2+j*ny*2+1].nd[1].y=(Ly/ny)*j;
582             pE[i*2+j*ny*2+1].nd[1].number=i+j*(nx+1)+1;           //NO.1
583             pE[i*2+j*ny*2+1].nd[2].x=(Lx/nx)*(i+1);
584             pE[i*2+j*ny*2+1].nd[2].y=(Ly/ny)*(j+1);
585             pE[i*2+j*ny*2+1].nd[2].number=i+1+(nx+1)*(j+1);       //NO.2
586         }
587         //-----------------------------------------------------------
588         //please turn to page 158 for more details
589         printf("计算基函数系数值...\n");
590         for(id=0;id<nx*ny*2;id++)
591         {
592             for(i=0;i<3;i++)
593             {
594                 if(i==0)        j=1,k=2;
595                 else if(i==1)   j=2,k=0;
596                 else if(i==2)   j=0,k=1;
597
598                 pE[id].A=( (pE[id].nd[j].x-pE[id].nd[i].x)*(pE[id].nd[k].y-pE[id].nd[i].y)-
599                     (pE[id].nd[j].y-pE[id].nd[i].y)*(pE[id].nd[k].x-pE[id].nd[i].x) )/2.0;
600                 D=2.0*pE[id].A;
601                 pE[id].a[i]=( pE[id].nd[j].x*pE[id].nd[k].y- pE[id].nd[k].x*pE[id].nd[j].y )/D;
602                 pE[id].b[i]=( pE[id].nd[j].y-pE[id].nd[k].y )/D;
603                 pE[id].c[i]=( pE[id].nd[k].x-pE[id].nd[j].x )/D;
604             }
605         }printf("OK!\n");
606         printf("计算单元有限元特征式系数矩阵...\n");
607
608     //-------------------------------------计算单元有限元特征式系数矩阵可以不再分两个三角形循环
609
610         int l,m;
611         for(i=0;i<nx*ny*2;i++)
612         {     for(l=0;l<3;l++)
613                 for(m=0;m<3;m++)                                        // Respaired
614                 {
615                     pE[i].Aij[l][m]=( pE[i].b[l]*pE[i].b[m] +
616                         pE[i].c[l]*pE[i].c[m] ) * pE[i].A;
617                 }
```

```
618         }
619     //-------------------------------------------------------------------------------------
620
621     /*    for(i=0;i<nx;i++)                         //计算单元有限元特征式系数矩阵
622           for(j=0;j<ny;j++)
623           {
624                 for(l=0;l<3;l++)                    //for the first triangle in the rectangle
625                       for(m=0;m<3;m++)
626                       {
627                             pE[i*2+j*ny*2].Aij[l][m]=( pE[i*2+j*ny*2].b[l]*pE[i*2+j*ny*2].b[m] +
628                                   pE[i*2+j*ny*2].c[l]*pE[i*2+j*ny*2].c[m] ) * pE[i*2+j*ny*2].A;
629                       }
630                 for(l=0;l<3;l++)                    //for the second triangle in the rectangle
631                       for(m=0;m<3;m++)
632                       {
633
634           pE[i*2+j*ny*2+1].Aij[l][m]=( pE[i*2+j*ny*2+1].b[l]*pE[i*2+j*ny*2+1].b[m] +
635                                   pE[i*2+j*ny*2+1].c[l]*pE[i*2+j*ny*2+1].c[m]          )          *
636     pE[i*2+j*ny*2+1].A;
637                       }
638           }    */
639           printf("OK!\n");
640
641           printf("计算积分值,填充到 f 函数向量数组...\n");
642
643     //---------------------------计算三角形单元 f 函数向量不用多重积分公式,直接代用单元三角形面
644     积的三分之一
645           int idx=0;
646           for(i=0;i<2*nx*ny;i++)
647           {     for(idx=0;idx<3;idx++)
648                 {
649                   pE[i].fi[idx]=(1.0/3.0)*(0.5)*(Lx/nx)*(Ly/ny);          // Respaired
650           }
651           }
652      /*     printf("pE[0].fi[0]=%f\n",pE[0].fi[0]);
653           printf("pE[0].fi[1]=%f\n",pE[0].fi[1]);
654           printf("pE[0].fi[2]=%f\n",pE[0].fi[2]); */
655
656     //-------------------------------------------------------------------------------------
657
658     /*    static int js[2]={4,4};                   //每一层积分区间均分为 4 个子区间
659           int idx=0;
660           for(i=0;i<nx;i++)                         //计算积分值,填充到 f 函数向量数组
661                 for(j=0;j<ny;j++)
```

```
662             {
663                 for(idx=0;idx<3;idx++)       //for the first triangle in the rectangle
664                 {
665                     ai=pE[i*2+j*ny*2].a[idx];
666                     bi=pE[i*2+j*ny*2].b[idx];
667                     ci=pE[i*2+j*ny*2].c[idx];
668                     pE[i*2+j*ny*2].fi[idx]=GaussIntegral(2,js,fn,fnGaussLimit);
669                 }
670                 for(idx=0;idx<3;idx++)       //for the second triangle in the rectangle
671                 {
672                     ai=pE[i*2+j*ny*2+1].a[idx];
673                     bi=pE[i*2+j*ny*2+1].b[idx];
674                     ci=pE[i*2+j*ny*2+1].c[idx];
675                     pE[i*2+j*ny*2+1].fi[idx]=GaussIntegral(2,js,fn,fnGaussLimit2);
676                 }
677             }   */
678
679             printf("OK!\n");
680
681         //单元矩阵元素累加到总体矩阵相应的位置上
682         printf("单元矩阵元素累加到总体矩阵相应的位置上...\n");
683         for(idx=0;idx<nx*ny*2;idx++)
684             for(i=0;i<3;i++)
685             {
686                 for(j=0;j<3;j++)
687                     pMatrix[pE[idx].nd[i].number*iNode+pE[idx                ].nd[j].number]
688     +=pE[idx].Aij[i][j];
689
690                 pMf[ pE[idx].nd[i].number ]+=pE[idx].fi[i];
691             }
692         printf("OK!\n");
693
694         double dBig=pow(10,20);                  //边界条件对角线扩大法处理所用的大数
695         double Ur=0.0;                           //边界条件 1(边界条件 2 通过 Calerkin 弱解表
696     达式自动满足)
697         for(i=0;i<nx+1;i++)
698         {     j=nx+1;
699             pMatrix[(j*nx+i)*iNode+(j*nx+i)]*=dBig;
700             pMf[(j*nx+i)]=pMatrix[(j*nx+i)*iNode+(j*nx+i)]*Ur;
701         }
702         for(j=0;j<nx+1;j++)
703         {     i=(nx+1)*(j+1)-1;
704             pMatrix[i*iNode+i]*=dBig;
705             pMf[i]=pMatrix[i*iNode+i]*Ur;
```

```
706         }
707
708         printf("调用全选主元高斯消去法函数解方程组...\n");
709         Gauss(pMatrix,pMf,iNode);              //调用全选主元高斯消去法函数解方程组
710         printf("OK!\n");
711         printf("写计算结果数据到文件...\n");
712         FILE *wfp;                                    //文件操作
713         if((wfp=fopen("dat.txt","w+"))==NULL)
714             printf("Cann't open the file... ");
715         //fprintf(wfp,"计算得各结点上的温度值为:\n");
716         for(i=0;i<iNode;i++)
717             fprintf(wfp,"%f\n", pMf[i]);
718             //fprintf(wfp,"%d     %f\n",i+1,pMf[i]);
719         printf("OK!\n");
720
721         fclose(wfp);
722     }
723     catch(...)
724     {
725         printf("Error occured...\n");
726     }
727         //释放总体矩阵和三角形单元数组占用内存
728         free(pMf);     free(pE); free(pMatrix);
729
730         printf("Please press Enter to exit...");
731         getchar();
732     }
733
734     //----------   全选主元高斯消去法   ------------------------------
735     //    a  体积为 n*n 的双精度实型二维数组，方程组系数矩阵，返回时将被破坏
736     //    b  长度为 n 的双精度实型一维数组，方程组右端的常数向量，返回方程组的解向量
737     //    n  整型变量，方程组的阶数
738     //-------------------------------------------------------------
739     int Gauss(double a[],double b[],int n)
740     {
741         int *js,l,k,i,j,is,p,q;
742         double d,t;
743         js=(int*)malloc(n*sizeof(int));
744         l=1;
745         for(k=0;k<=n-2;k++)
746         {
747             d=0.0;
748             for(i=k;i<=n-1;i++)
749                 for(j=k;j<=n-1;j++)
```

```
{
    t=fabs(a[i*n+j]);
    if(t>d) { d=t; js[k]=j; is=i;}
}
if(d+1.0==1.0) l=0;
else
{
    if(js[k]!=k)
        for(i=0;i<=n-1;i++)
        {
            p=i*n+k; q=i*n+js[k];
            t=a[p]; a[p]=a[q]; a[q]=t;
        }
        if(is!=k)
        {
            for(j=k;j<=n-1;j++)
            {
                p=k*n+j; q=is*n+j;
                t=a[p]; a[p]=a[q]; a[q]=t;
            }
            t=b[k]; b[k]=b[is]; b[is]=t;
        }
}
if(l==0)
{
    free(js);
    printf("Gauss funtion failed 1...\n");
    return(0);
}
d=a[k*n+k];
for(j=k+1;j<=n-1;j++)
{
    p=k*n+j; a[p]=a[p]/d;
}
b[k]=b[k]/d;
for(i=k+1;i<=n-1;i++)
{
    for(j=k+1;j<=n-1;j++)
    {
        p=i*n+j;
        a[p]=a[p]-a[i*n+k]*a[k*n+j];
    }
    b[i]=b[i]-a[i*n+k]*b[k];
}
```

```
794         }
795         d=a[(n-1)*n+n-1];
796         if(fabs(d)+1.0==1.0)
797         {
798             free(js);
799             printf("Gauss funtion failed 2...\n");
800             return(0);
801         }
802         b[n-1]=b[n-1]/d;
803         for(i=n-2;i>=0;i--)
804         {
805             t=0.0;
806             for(j=i+1;j<=n-1;j++)
807                 t=t+a[i*n+j]*b[j];
808             b[i]=b[i]-t;
809         }
810         js[n-1]=n-1;
811         for(k=n-1;k>=0;k--)
812             if(js[k]!=k)
813             {
814                 t=b[k]; b[k]=b[js[k]]; b[js[k]]=t;
815             }
816         free(js);
817         return(1);
818     }
819
820     /*
821     //------------   计算多重积分的高斯方法   ------------------------
822     //    n  整型变量，积分重数
823     //    js  整型一维数组，长度为 n，
824     //          其 js(i) (i=0,1,...,n-1)表示第 i 层积分区间所划分的子区间个数
825     //    fn()  被积函数(函数指针)
826     //    fnGaussLimit()  函数(函数指针)计算各层积分上下限值(上限>下限)
827     //-------------------------------------------------------------
828     double GaussIntegral(int n,int js[],double (*fn)(int n,double x[]),
829             void (*fnGaussLimit)(int j,int n,double x[],double y[]))
830     {
831         int m,j,k,q,*is;
832         double y[2],p,s,*x,*a,*b;
833         static double t[5]={-0.9061798459,-0.5384693101,0.0,0.5384693101,0.9061798459};
834         static double
835     c[5]={0.2369268851,0.4786286705,0.5688888889,0.4786286705,0.2369268851};
836         is=(int*)malloc(2*(n+1)*sizeof(int));
837         x=(double*)malloc(n*sizeof(double));
```

```
a=(double*)malloc(2*(n+1)*sizeof(double));
b=(double*)malloc((n+1)*sizeof(double));
m=1;a[n]=1.0; a[2*n+1]=1.0;
while(true)
{
    for(j=m;j<=n;j++)
    {
        fnGaussLimit(j-1,n,x,y);
        a[j-1]=0.5*(y[1]-y[0])/js[j-1];
        b[j-1]=a[j-1]+y[0];
        x[j-1]=a[j-1]*t[0]+b[j-1];
        a[n+j]=0.0;    is[j-1]=1;is[n+j]=1;
    }
    j=n; q=1;
    while(q==1)
    {
        k=is[j-1];
        if(j==n) p=fn(n,x);
        else p=1.0;
        a[n+j]=a[n+j+1]*a[j]*p*c[k-1]+a[n+j];
        is[j-1]=is[j-1]+1;
        if(is[j-1]>5)
        {
            if(is[n+j]>=js[j-1])
            {
                j=j-1; q=1;
                if(j==0)
                {
                    s=a[n+1]*a[0];
                    free(is); free(x);free(a); free(b);
                    return(s);
                }
            }
            else
            {
                is[n+j]=is[n+j]+1;
                b[j-1]=b[j-1]+a[j-1]*2.0;
                is[j-1]=1; k=is[j-1];
                x[j-1]=a[j-1]*t[k-1]+b[j-1];
                if(j==n)
                    q=1;
                else
                    q=0;
            }
```

```
                }
                else
                {
                    k=is[j-1];
                    x[j-1]=a[j-1]*t[k-1]+b[j-1];
                    if(j==n)
                        q=1;
                    else
                        q=0;
                }
            }
            m=j+1;
        }
    return 0;
}
//-------------------------------------------------------------*/
```

FEM3.cpp

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

int Gauss(double a[],double b[],int n);      //全选主元高斯消去法
struct ETNode{                                  //单元结点结构体
    double x,y;                                 //单元结点坐标
    int number;                                 //单元结点在总体区域划分中的编号
};

struct ElementTriangle{                         //三角形单元结构体
    ETNode nd[3];                               //存储相对应的单元结点号
    double   a[3],b[3],c[3];                    //基函数的系数
    double   A;                                 //单元面积
    double   Aij[3][3];                         //单元有限元特征式系数矩阵
};
//--------------   全局变量   ---------------------------
    int   i,j,k,                                //循环指标
        id;                                     //单元的循环指标
    const int nx=21,ny=21;                     //x,y 方向划分网格数,三角形单元个数=nx*ny*2
                                               //    (即 24*24*2=1152 个单元)
    const double Lx=1,Ly=1;                   //矩形区域的两边长
    double   D;                                 //单元矩阵行列式值
    const int iNode=(nx+1)*(ny+1);             //结点个数=(nx+1)*(ny+1) (即 25*25=625 个节点)
    double* pMatrix;                            //总体矩阵指针
    double* pMf;                                //f 向量指针
```

```
925         ElementTriangle* pE;                    //单元三角形结构体数组指针
926         double ai,bi,ci;                        //基函数的系数
927
928
929     //--------------------   主程序   -------------------------//
930     void main(void)
931     {
932         //为总体矩阵,三角形单元数组,f 函数向量分配存储内存
933         pMatrix=(double*)malloc(iNode*iNode*sizeof(double));
934         pE=(ElementTriangle*)malloc(nx*ny*2*sizeof(ElementTriangle));
935         pMf=(double*)malloc(iNode*sizeof(double));
936         //初始化值为 0,因为下面要累加总体矩阵
937         for(i=0;i<iNode*iNode;i++)
938             pMatrix[i]=0;
939         for(i=0;i<iNode;i++)
940             pMf[i]=0;
941
942     try{
943         //------    计算得到网格的信息   -----------
944         for(j=0;j<nx;j++)
945         for(i=0;i<ny;i++)
946         {
947             //for the first triangle in the rectangle
948             pE[i*2+j*ny*2].nd[0].x=(Lx/nx)*i;
949             pE[i*2+j*ny*2].nd[0].y=(Ly/ny)*j;
950             pE[i*2+j*ny*2].nd[0].number=i+j*(nx+1);                 //NO.0
951             pE[i*2+j*ny*2].nd[1].x=(Lx/nx)*(i+1);
952             pE[i*2+j*ny*2].nd[1].y=(Ly/ny)*(j+1);
953             pE[i*2+j*ny*2].nd[1].number=i+1+(nx+1)*(j+1);   //NO.1
954             pE[i*2+j*ny*2].nd[2].x=(Lx/nx)*i;
955             pE[i*2+j*ny*2].nd[2].y=(Ly/ny)*(j+1);
956             pE[i*2+j*ny*2].nd[2].number=i+(nx+1)*(j+1);      //NO.2
957             //for the second triangle in the rectangle
958             pE[i*2+j*ny*2+1].nd[0].x=(Lx/nx)*i;
959             pE[i*2+j*ny*2+1].nd[0].y=(Ly/ny)*j;
960             pE[i*2+j*ny*2+1].nd[0].number=i+j*(nx+1);        //NO.0
961             pE[i*2+j*ny*2+1].nd[1].x=(Lx/nx)*(i+1);
962             pE[i*2+j*ny*2+1].nd[1].y=(Ly/ny)*j;
963             pE[i*2+j*ny*2+1].nd[1].number=i+j*(nx+1)+1;          //NO.1
964             pE[i*2+j*ny*2+1].nd[2].x=(Lx/nx)*(i+1);
965             pE[i*2+j*ny*2+1].nd[2].y=(Ly/ny)*(j+1);
966             pE[i*2+j*ny*2+1].nd[2].number=i+1+(nx+1)*(j+1);      //NO.2
967         }
968
```

```
969
970         printf("计算基函数系数值...\n");
971         for(id=0;id<nx*ny*2;id++)
972         {
973             for(i=0;i<3;i++)
974             {
975                 if(i==0)        j=1,k=2;
976                 else if(i==1)   j=2,k=0;
977                 else if(i==2)   j=0,k=1;
978
979                 pE[id].A=( (pE[id].nd[j].x-pE[id].nd[i].x)*(pE[id].nd[k].y-pE[id].nd[i].y)-
980                     (pE[id].nd[j].y-pE[id].nd[i].y)*(pE[id].nd[k].x-pE[id].nd[i].x) )/2.0;
981                 D=2.0*pE[id].A;
982                 pE[id].a[i]=( pE[id].nd[j].x*pE[id].nd[k].y- pE[id].nd[k].x*pE[id].nd[j].y )/D;
983                 pE[id].b[i]=( pE[id].nd[j].y-pE[id].nd[k].y )/D;
984                 pE[id].c[i]=( pE[id].nd[k].x-pE[id].nd[j].x )/D;
985
986             }
987         }
988         printf("OK!\n");
989         printf("计算单元有限元特征式系数矩阵...\n");
990         int l,m;
991         for(i=0;i<nx*ny*2;i++)
992         {
993             for(l=0;l<3;l++)
994                 for(m=0;m<3;m++)
995                 {
996                     pE[i].Aij[l][m]=( pE[i].b[l]*pE[i].b[m] +
997                         pE[i].c[l]*pE[i].c[m] ) * pE[i].A/3;
998                 }
999         }
1000        printf("OK!\n");
1001        //单元矩阵元素累加到总体矩阵相应的位置上
1002        printf("单元矩阵元素累加到总体矩阵相应的位置上...\n");
1003        int idx=0;
1004        for(idx=0;idx<nx*ny*2;idx++)
1005            for(i=0;i<3;i++)
1006            {
1007                for(j=0;j<3;j++)
1008            pMatrix[ pE[idx].nd[i].number*iNode+pE[idx ].nd[j].number ] +=pE[idx].Aij[i][j];
1009                pMf[ pE[idx].nd[i].number ]=0;
1010            }
1011        printf("OK!\n");
1012
```

```
1013    ////////////////////////////////// respaired by zqy 2003,6,20
1014
1015        double dBig=pow(10,20);          //边界条件对角线扩大法处理所用的大数
1016
1017        for(i=0;i<nx+1;i++)
1018        {     double Ur=1.0;                              //强制边界条件 1
1019              j=nx+1;
1020              pMatrix[(j*nx+i)*iNode+(j*nx+i)]*=dBig;
1021              pMf[(j*nx+i)]=pMatrix[(j*nx+i)*iNode+(j*nx+i)]*Ur;
1022        }
1023        double Ur=0.0;                                    //强制边界条件 2
1024        for(i=0;i<nx+1;i++)
1025        {     pMatrix[i*iNode+i]*=dBig;
1026              pMf[i]=0;
1027        }
1028        for(j=0;j<nx;j++)
1029        {     i=(nx+1)*(j+1)-1;
1030              pMatrix[i*iNode+i]*=dBig;
1031              pMf[i]=0;
1032        }
1033        for(j=0;j<nx;j++)
1034        {     i=(nx+1)*j;
1035              pMatrix[i*iNode+i]*=dBig;
1036              pMf[i]=0;
1037        }
1038
1039    ////////////////////////////////// respaired by zqy 2003,6,20
1040
1041        printf("调用全选主元高斯消去法函数解方程组...\n");
1042        Gauss(pMatrix,pMf,iNode);          //调用全选主元高斯消去法函数解方程组
1043        printf("OK!\n");
1044        printf("写计算结果数据到文件...\n");
1045        FILE *wfp;                                        //文件操作
1046        if((wfp=fopen("dat.txt","w+"))==NULL)
1047              printf("Cann't open the file... ");
1048        //fprintf(wfp,"计算得各结点上的温度值为:\n");
1049        for(i=0;i<iNode;i++)
1050        {
1051              fprintf(wfp,"%f\n",pMf[i]);
1052        }
1053        printf("OK!\n");
1054
1055        fclose(wfp);
1056    }
```

```
1057    catch(...)
1058    {
1059        printf("Error occured...\n");
1060    }
1061        //释放总体矩阵和三角形单元数组占用内存
1062        free(pMf);     free(pE); free(pMatrix);
1063
1064        printf("Please press Enter to exit...");
1065        getchar();
1066    }
1067    //----------  全选主元高斯消去法  ------------------------------
1068    //    a 体积为 n*n 的双精度实型二维数组，方程组系数矩阵，返回时将被破坏
1069    //    b 长度为 n 的双精度实型一维数组，方程组右端的常数向量，返回方程组的解向量
1070    //    n 整型变量，方程组的阶数
1071    //-----------------------------------------------------------------
1072    int Gauss(double a[],double b[],int n)
1073    {
1074        int *js,l,k,i,j,is,p,q;
1075        double d,t;
1076        js=(int*)malloc(n*sizeof(int));
1077        l=1;
1078        for(k=0;k<=n-2;k++)
1079        {
1080            d=0.0;
1081            for(i=k;i<=n-1;i++)
1082                for(j=k;j<=n-1;j++)
1083                {
1084                    t=fabs(a[i*n+j]);
1085                    if(t>d) { d=t; js[k]=j; is=i;}
1086                }
1087                if(d+1.0==1.0) l=0;
1088                else
1089                {
1090                    if(js[k]!=k)
1091                        for(i=0;i<=n-1;i++)
1092                        {
1093                            p=i*n+k; q=i*n+js[k];
1094                            t=a[p]; a[p]=a[q]; a[q]=t;
1095                        }
1096                        if(is!=k)
1097                        {
1098                            for(j=k;j<=n-1;j++)
1099                            {
1100                                p=k*n+j; q=is*n+j;
```

```
                                    t=a[p]; a[p]=a[q]; a[q]=t;
                                }
                                t=b[k]; b[k]=b[is]; b[is]=t;
                            }
                        }
                        if(l==0)
                        {
                            free(js);
                            printf("Gauss funtion failed 1...\n");
                            return(0);
                        }
                        d=a[k*n+k];
                        for(j=k+1;j<=n-1;j++)
                        {
                            p=k*n+j; a[p]=a[p]/d;
                        }
                        b[k]=b[k]/d;
                        for(i=k+1;i<=n-1;i++)
                        {
                            for(j=k+1;j<=n-1;j++)
                            {
                                p=i*n+j;
                                a[p]=a[p]-a[i*n+k]*a[k*n+j];
                            }
                            b[i]=b[i]-a[i*n+k]*b[k];
                        }
                }
                d=a[(n-1)*n+n-1];
                if(fabs(d)+1.0==1.0)
                {
                    free(js);
                    printf("Gauss funtion failed 2...\n");
                    return(0);
                }
                b[n-1]=b[n-1]/d;
                for(i=n-2;i>=0;i--)
                {
                    t=0.0;
                    for(j=i+1;j<=n-1;j++)
                        t=t+a[i*n+j]*b[j];
                    b[i]=b[i]-t;
                }
                js[n-1]=n-1;
                for(k=n-1;k>=0;k--)
```

```
            if(js[k]!=k)
            {
                t=b[k]; b[k]=b[js[k]]; b[js[k]]=t;
            }
        free(js);
        return(1);
}
```

%matlab FemReadDat

```
N=22;
str='dat.txt';
fid=fopen(str,'r');

M=fscanf(fid,'%f',[N N]);
fclose(fid);

nx=N-1;ny=N-1;
dx=1/nx;
dy=1/ny;
x=0:(1/nx):1;
y=0:(1/ny):1;
[X,Y]=meshgrid(x,y);
subplot(2,2,1)
Mesh(X,Y,M)
xlabel('X-axis'),ylabel('Y-axis'),zlabel('温度 T-axis')
title('有限元法解二维热传导问题计算结果数据可视化 图 1')
subplot(2,2,2)
Mesh(X,Y,M)
xlabel('X-axis'),ylabel('Y-axis'),zlabel('温度 T-axis')
title('图 2')
view(30,20)
subplot(2,2,3)
Mesh(X,Y,M)
xlabel('X-axis'),ylabel('Y-axis'),zlabel('温度 T-axis')
title('图 3')
view(-30,45)
subplot(2,2,4)
surfl(X,Y,M)
view(-60,45)
shading interp
xlabel('X-axis'),ylabel('Y-axis'),zlabel('温度 T-axis')
title('图 4')
```