学号：S201840001

姓名：胡军锋

**问题：**

 (1)

其中，

真解为 

**变分:**

**** 上述方程两边同乘以，利用格林公式以及边界条件，易得：

 (2)

令

 (3)

原问题等价于变分问题：求 使得

**问题转化**

一般的变分问题，描述为：

 (4)

构造V的有限维子空间 考虑如下的离散问题：

 (5)

用问题(5)近似问题(4),后者的解逼近前者的解，故问题(5)的解可作为问题

(4)的近似解。

设空间中的一组基为 将代入，并依次分别取为每

个基函数，可以得到：

 (6)

上述方程(6)可看作线性方程组 其中



因此，对于一般的变分问题，可以通过离散化，求解方程(6)来近似。那么问题本质上就转化为了找一个合适的空间逼近，在这个空间中求基函数和基函数之间的相互作用以及基函数和f之间的作用，构建出刚度矩阵，形成方程组，最后求解，得到逼近阶关于基函数的组合系数，最后得到原问题的一个逼近的数值解。

接下来考虑单元上的插值问题。对于问题(1),采用三角形分割，假设两个方向上是等距分割，即三角形是等边直角三角形。如下图所示：

21 22 23 24 25

1 2 3 4 5

|  |  |  |  |
| --- | --- | --- | --- |
| T26  T25 | T28  T27 | T30  T29 | 16 17 18 19 20  T32  T31 |
| T18  T17 | T20  T19 | T22  T21 | 11 12 13 14 15  T24  T23 |
| T10  T9 | 6 7 8 9 10  T12  T11 | T14  T13 | T16  T15 |
| T2  T1 | T4  T3 | T6  T5 | T8  T7 |

在每个单元上，采用面积坐标。分割单元上的形函数空间为多项式集合其中单元自由度为3。

且



其中T表示三角形面积。

**问题求解**

单刚组总刚：在每个单元上计算基函数，基函数之间的相互作用，以及基函数和右端项的作用，并分配到总刚度矩阵的相应位置，并进行相应的边值条件处理进行求解。

误差及收敛阶如下表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | err\_H1 | 收敛阶 | err\_L2 | 收敛阶 | Time(s) |
| 2 | 1.2372 |  | 0.2922 |  | 0.8507 |
| 3 | 0.8497 | 0.9266 | 0.1547 | 1.5687 | 1.8688 |
| 4 | 0.6432 | 0.9681 | 0.0929 | 1.7728 | 3.3152 |
| 5 | 0.5166 | 0.9819 | 0.0614 | 1.8586 | 4.9867 |
| 6 | 0.4314 | 0.9883 | 0.0434 | 1.9034 | 7.2802 |
| 7 | 0.3703 | 0.9917 | 0.0322 | 1.9299 | 9.6220 |
| 8 | 0.3242 | 0.9939 | 0.0248 | 1.9468 | 12.6862 |
| 9 | 0.2884 | 0.9952 | 0.0197 | 1.9583 | 16.1913 |
| 10 | 0.2596 | 0.9962 | 0.0160 | 1.9664 | 19.3560 |
| 20 | 0.1300 | 0.9982 | 0.0041 | 1.9835 | 76.4583 |
| 100 | 0.0260 | 0.9997 | 0.0001 | 1.9977 | 1925 |

代码详见附录

**附录**

function [err\_global\_H1, err\_global\_L2] = main(N)

%% 二维有限元程序

tic;

clc;

clear;

close all;

L\_x = 1; %x方向区间长度

L\_y = 1; %y方向区间长度

%N = 1;

n\_x = N; %x方向的矩形个数

n\_y = N; %y方向的矩形个数

h\_x = L\_x/n\_x; %x方向上的单元长度

h\_y = L\_y/n\_y; %y方向上的单元长度

num = 2\*n\_x\*n\_y; %三角形的个数

u\_b = zeros(2\*(n\_x+n\_y),1); %边界条件

node\_x = n\_x + 1; %x方向节点个数

node\_y = n\_y + 1; %y方向节点个数

node\_sum = node\_x\*node\_y; %总节点个数

nel = 3; %自由度

x = linspace(0,L\_x,node\_x)'; %等分节点的横坐标

y = linspace(0,L\_y,node\_y)'; %等分节点的纵坐标

[X, Y] = meshgrid(x,y); %张成网格，X和Y分别表示对应位置的横纵坐标

X = X'; Y = Y'; coord = [X(:) Y(:)]; %每一行是对应节点的坐标

lianjie\_mat = lianjie(node\_x,node\_y,nel); %连接矩阵，每个单元周围的节点编号

border = unique([1:node\_x node\_x\*n\_y+1:node\_x\*n\_y+node\_x ...%上下两边的边界

node\_x+1:node\_x:node\_x\*(n\_y-1)+1 2\*node\_x:node\_x:n\_y\*node\_x]);%左右两边边界

K = sparse(node\_sum, node\_sum); %刚度矩阵[K]，初始化为0，

F = sparse(node\_sum, 1); %右端项,初始化为0

%% 单刚组总刚

% 计算系数矩阵K和右端项f

for i = 1:num %同一维的情况，依然按单元来扫描

k = xishu(i,nel, h\_x, h\_y, coord, lianjie\_mat); %计算单元刚度矩阵

f = youduan(i, nel, h\_x, h\_y, coord, lianjie\_mat); %计算单元载荷向量

m = lianjie\_mat(i, :);

K(m, m) = K(m, m) + k;

F(m) = F(m) + f;

%full(K)

%full(F)

end

%full(K);

%% 边值条件处理

for i = 1:length(border)

n = border(i);

for j = 1:node\_sum

if (isempty(find(border == j, 1))) %第j个点若不是固定点

F(j) = F(j) - K(j,n)\*u\_b(i);

end

end

K(n, :) = 0.0;

K(:, n) = 0.0;

K(n, n) = 1.0;

F(n) = u\_b(i);

end

%full(F);

%full(K);

u = K\F;

%full(u);

%% 计算误差阶

e = sparse(num,2);%第一列存储H1半模局部误差，第二列存储L2局部误差

for i = 1:num

[e(i,1), e(i,2)] = err\_local(i, u, h\_x, h\_y, coord, lianjie\_mat);

end

err\_global\_H1 = sqrt(sum(e(:,1))) %H1半模

err\_global\_L2 = sqrt(sum(e(:,2))) %L2模

t=toc

%作图

u\_re = reshape(u, node\_x, node\_y);

u\_re = full(u\_re);

figure

mesh(x, y, u\_re);

title('Approximate Solutions');

% 求精确解

L =L\_x;

n\_exact = N;

x\_exact = linspace(0, L, n\_exact);

[X1, Y1] = meshgrid(x\_exact, x\_exact);

u\_exact = exactsolution(X1(:), Y1(:));

u\_exact\_re = reshape(u\_exact, n\_exact, n\_exact);

figure

mesh(x\_exact, x\_exact, u\_exact\_re);

title(' Exact Solutions');

u\_ex = exactsolution(coord(:, 1), coord(:, 2));

error\_L2 = sqrt(sum((u - u\_ex).^2)\*h\_x\*h\_y);

%=========== 连接矩阵 ============

function lianjie = lianjie(node\_x, node\_y, nel)

%输入横纵坐标的节点数目，和单元自由度

%输出连接矩阵，每个单元涉及的节点的编号

xn = 1: (node\_x\*node\_y); %编号一列排下来

A = reshape(xn, node\_x, node\_y); %同形状编号

for i = 1: (node\_x-1)\*(node\_y-1) %矩形单元

x = rem(i, node\_x-1); %表示单元为x方向数起第几个

if x == 0

x = node\_x-1;

end

y = ceil(i/(node\_x-1)); %向上取整,y方向数起第几个

a = A(x:x+1, y:y+1); %这个小矩阵，拉直了就是连接矩阵

a\_vec = a(:);

lianjie(2\*i-1:2\*i, 1:nel) = [a\_vec([1 4 3])'; a\_vec([4 1 2])'];

end

%========== 单元刚度矩阵 ============

function [k] = xishu(i, nel, h\_x, h\_y, coord, lianjie)

S = h\_x\*h\_y/2; %三角形面积

k = zeros(nel, nel);

nodes = lianjie(i, :); %相关节点编号

x = coord(nodes, :); %相关节点的坐标

xi1 = x(2, 1) - x(3, 1); xi2 = x(3, 1) - x(1, 1); xi3 = x(1, 1) - x(2, 1);

eta1 = x(2, 2) - x(3, 2); eta2 = x(3, 2) - x(1, 2); eta3 = x(1, 2) - x(2, 2);

auv11 = -((eta1\*xi2 - eta2\*xi1)\*(eta1^2 + eta2^2))/(8\*S^2);

auv12 = ((eta1\*xi1 + eta2\*xi2)\*(eta1\*xi2 - eta2\*xi1))/(8\*S^2);

auv13 = -((eta1\*xi2 - eta2\*xi1)\*(- eta1^2 + xi1\*eta1 - eta2^2 + xi2\*eta2))/(8\*S^2);

auv22 = -((xi1^2 + xi2^2)\*(eta1\*xi2 - eta2\*xi1))/(8\*S^2);

auv23 = -((eta1\*xi2 - eta2\*xi1)\*(- xi1^2 + eta1\*xi1 - xi2^2 + eta2\*xi2))/(8\*S^2);

auv33 = -((eta1\*xi2 - eta2\*xi1)\*(eta1^2 - 2\*eta1\*xi1 + eta2^2 - 2\*eta2\*xi2 + xi1^2 + xi2^2))/(8\*S^2);

k = k + [auv11 auv12 auv13;auv12 auv22 auv23;auv13 auv23 auv33];

%============ 右端向量 =============

function [f] = youduan(i, nel, h\_x, h\_y, coord, lianjie)

nodes = lianjie(i, :); %自由度编号

xe = coord(nodes, :); %单元自由节点坐标

xi1 = xe(2, 1) - xe(3, 1); xi2 = xe(3, 1) - xe(1, 1); xi3 = xe(1, 1) - xe(2, 1);

eta1 = xe(2, 2) - xe(3, 2); eta2 = xe(3, 2) - xe(1, 2); eta3 = xe(1, 2) - xe(2, 2);

detJ = eta2\*xi1 - eta1\*xi2;

g1 = @(lam1, lam2) fun(xe(3, 1) + lam1\*(xi2) + lam2\*(xi1), xe(3, 2) + lam1\*(-eta2) + lam2\*(eta1)).\*lam1\*detJ;

g2 = @(lam1, lam2) fun(xe(3, 1) + lam1\*(xi2) + lam2\*(xi1), xe(3, 2) + lam1\*(-eta2) + lam2\*(eta1)).\*lam2\*detJ;

g3 = @(lam1, lam2) fun(xe(3, 1) + lam1\*(xi2) + lam2\*(xi1), xe(3, 2) + lam1\*(-eta2) + lam2\*(eta1)).\*(1-lam1-lam2)\*detJ;

lammax = @(lam1) 1 - lam1;

gx(1) = integral2(g1, 0, 1, 0, lammax);

gx(2) = integral2(g2, 0, 1, 0, lammax);

gx(3) = integral2(g3, 0, 1, 0, lammax);

f = [gx(1); gx(2); gx(3)];

%========= f ===========

function bx = fun(x, y)

bx = (2\*pi^2)\*sin(pi\*x).\*sin(pi\*y);

%====== exactsolution ========

function uexact = exactsolution(x, y)

uexact = sin(pi\*x).\*sin(pi\*y);

%========= error ==========

function [err\_H1, err\_L2] = err\_local(i, u, h\_x, h\_y, coord, lianjie)

syms x y

S = h\_x\*h\_y/2; %三角形面积

nodes = lianjie(i, :); %相关节点编号

xe = coord(nodes, :); %相关节点的坐标

ui = u(nodes);

xi1 = xe(2, 1) - xe(3, 1); xi2 = xe(3, 1) - xe(1, 1); xi3 = xe(1, 1) - xe(2, 1);

eta1 = xe(2, 2) - xe(3, 2); eta2 = xe(3, 2) - xe(1, 2); eta3 = xe(1, 2) - xe(2, 2);

w1 = xe(2, 1)\*xe(3, 2)-xe(3, 1)\*xe(2, 2); w2 = xe(3, 1)\*xe(1, 2)-xe(1, 1)\*xe(3, 2);

w3 = xe(1, 1)\*xe(2, 2)-xe(2, 1)\*xe(1, 2);

%基函数

lamd1 = (eta1\*x-xi1\*y+w1)/(2\*S);

lamd2 = (eta2\*x-xi2\*y+w2)/(2\*S);

lamd3 = (eta3\*x-xi3\*y+w3)/(2\*S);

u\_local = ui(1, 1)\*lamd1+ui(2, 1)\*lamd2+ui(3, 1)\*lamd3;

u\_grad = gradient(u\_local, [x ,y])-gradient(sin(pi\*x).\*sin(pi\*y),[x ,y]);

u\_ex = sin(pi\*x).\*sin(pi\*y); %精确解

u\_in1 = sum((u\_grad).^2); %H1半模

u\_in2 = sum((u\_local - u\_ex).^2);%L2模

x\_average = (xe(1, 1)+xe(2, 1)+xe(3, 1))/3;

y\_average = (xe(1, 2)+xe(2, 2)+xe(3, 2))/3;

err\_H1 = vpa(S\*(subs(subs(u\_in1, x, x\_average), y, y\_average)), 10);

err\_L2 = vpa(S\*(subs(subs(u\_in2, x, x\_average), y, y\_average)), 10);