1 网格的图示与标记

1.1 网格与解的图示

基本数据结构

采用陈龙编写的 iFEM 工具箱 [1] 中的数据结构, 用 `node` 表示节点坐标, `elem` 表示单元的连通性, 即单元顶点编号. 考虑下图中 L 形区域的一个简单剖分, iFEM 的网页说明链接如下:

https://www.math.uci.edu/~chenlong/ifemdoc/mesh/meshbasicdoc.html

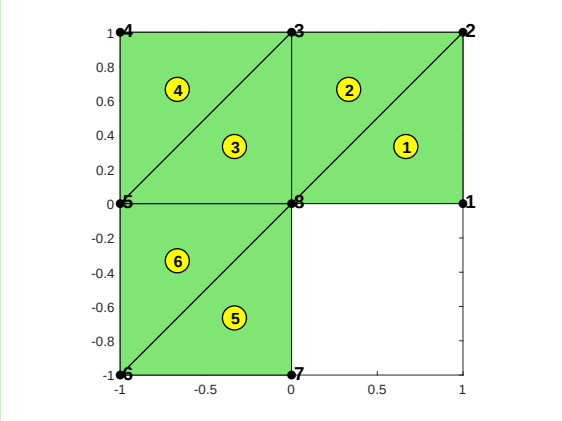


图 1. L 形区域的剖分

1. 矩阵 `node`: 节点坐标

   编程中需要每个节点的坐标, 用 `node` 记录, 它是两列的矩阵, 第一列表示各节点的横坐标, 第二列表示各节点的纵坐标, 行的索引对应节点编号. 图中给出的顶点坐标信息如下

8x2 double

	1	2
1	1	0
2	1	1
3	0	1
4	-1	1
5	-1	0
6	-1	-1
7	0	-1
8	0	0

   这里左侧的序号对应节点的整体编号.

2. 矩阵 `elem`: 连通性

   `elem` 给出每个三角形的顶点编号, 它给出的是单元的连通性信息, 每行对应一个单元.

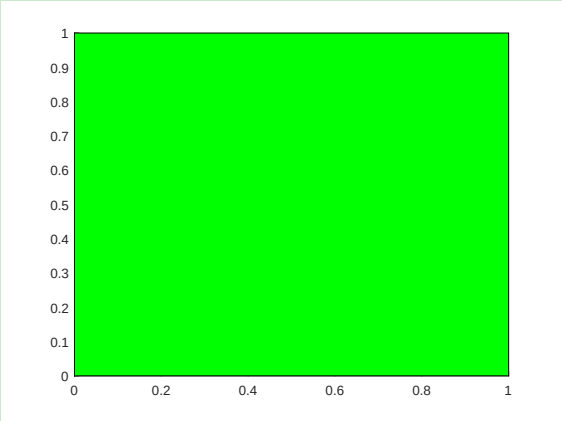
6x3 double

	1	2	3
1	1	2	8
2	3	8	2
3	8	3	5
4	4	5	3
5	7	8	6
6	5	6	8

图中第 i 列表示所有三角形的第 i 个点的编号, 其中 $i = 1, 2, 3$. 注意三角形顶点的顺序符合逆时针定向. `elem` 是有限元编程装配过程中 `P1`-元的局部整体对应.

补片函数 patch

MATLAB 中采用补片函数 `patch` 绘制多边形, 其内置的三角剖分画图函数 `trisurf` 也是如此. 以下考虑二维多角形剖分的图示, 命名为 showmesh.m. 一个简单的例子如下图



可如下编程

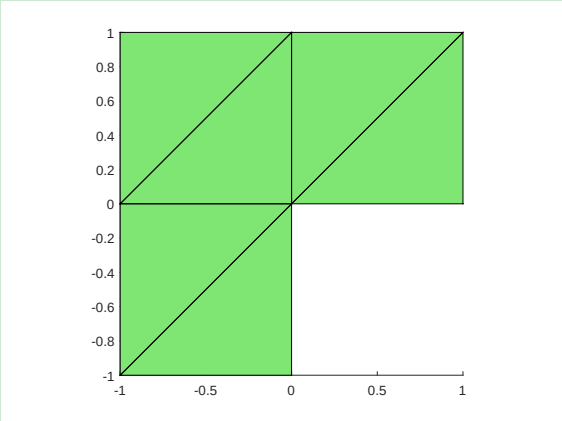
```
node = [0 0; 1 0; 1 1; 0 1];
elem = [1 2 3 4];
patch('Faces',elem,'Vertices',node,'FaceColor','g');
```

对多个相同类型的单元, showmesh.m 如下编写:

```
function showmesh(node,elem)
h = patch('Faces',elem, 'Vertices', node);
set(h,'facecolor',[0.5 0.9 0.45],'edgecolor','k');
axis equal; axis tight;
```

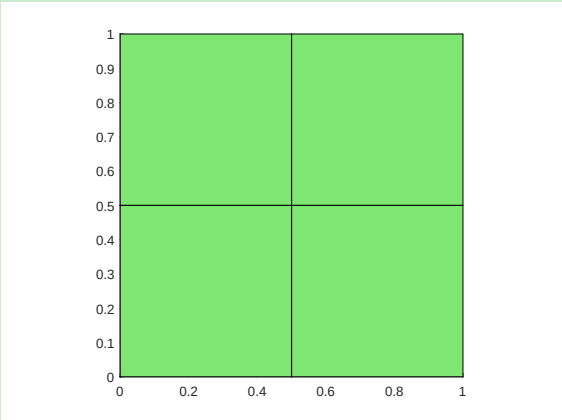
例 1.1 (三角剖分) 对前面的梯形区域, 如下调用 showmesh 函数

```
node = [1,0; 1,1; 0,1; -1,1; -1,0; -1,-1; 0,-1; 0,0];
elem = [1,2,8; 3,8,2; 8,3,5; 4,5,3; 7,8,6; 5,6,8];
showmesh(node,elem);
```



例 1.2 (四边形剖分)　对矩形区域的四边形剖分, 如下调用 showmesh 函数

```
[X,Y] = ndgrid(0:0.5:1,0:0.5:1);
node = [X(:), Y(:)];
elem = [1 2 5 4; 2 3 6 5; 4 5 8 7; 5 6 9 8];
showmesh(node,elem)
```



操作 cell 数组的 cellfun 函数

对含有不同多角形单元的区域, 因每个单元顶点数不同, `elem` 一般以 cell 数组存储. 为了使用 `patch` 画图 (避免循环语句), 需要将 `elem` 的每个 cell 用 `NaN` 填充成相同维数的向量, 该填充不会影响画图. 先介绍 MATLAB 中操作 cell 数组的函数 cellfun.m. 例如, 考虑下面的例子.

例 1.3　计算 cell 数组中元素的平均值和维数

```
C = {1:10, [2; 4; 6], []};
averages = cellfun(@mean, C)
[nrows, ncols] = cellfun(@size, C)

% 结果为 averages =  5.5000    4.0000       NaN
%        nrows =  1  3  0,    ncols = 10  1  0
```

cellfun 的直接输出规定为数值数组, 如果希望输出的是多种类型的元素, 那么需要指定 `UniformOutput` 为 `false`, 例如

例 1.4　对字符进行缩写

```
days = {'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'};
```

```
2 abbrev = cellfun(@(x) x(1:3), days, 'UniformOutput', false)
```

这里的 UniformOutput 也可简写为 un, 当然 false 也可写为 0. 正因为此时输出类型可以任意, MATLAB 默认仍保存为 cell 类型. 上面的结果为

```
abbrev =
  1×5 cell array
    {'Mon'}    {'Tue'}    {'Wed'}    {'Thu'}    {'Fri'}
```

showmesh 函数的建立

现在考虑图 2 所示的剖分

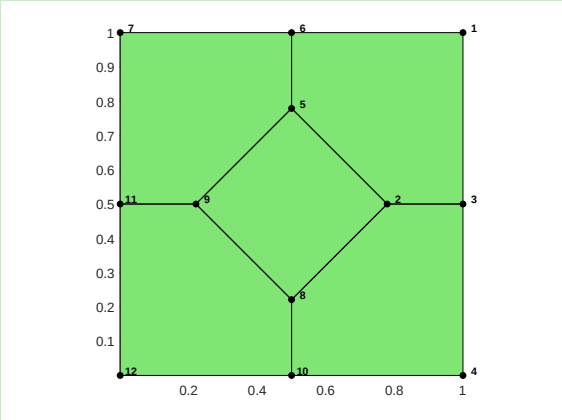


图 2. 多角形网格

相关的网格数据保存在 meshex1.mat 中. 程序如下

```
1 load('meshex1.mat'); % node, elem
2
3 max_n_vertices = max(cellfun(@length, elem));
4 % function to pad the vacancies ( 横向拼接 )
5 padding_func = @(vertex_ind) [vertex_ind,...
6       NaN(1,max_n_vertices-length(vertex_ind))];
7 tpad = cellfun(padding_func, elem, 'UniformOutput', false);
8 tpad = vertcat(tpad{:});
9 h = patch('Faces', tpad,'Vertices', node);
10 set(h,'facecolor',[0.5 0.9 0.45],'edgecolor','k');
11 axis equal; axis tight;
```

最终给出的 showmesh 函数如下

CODE 1. showmesh.m (2D 网格画图)

```
1 function showmesh(node,elem)
2
3 dim = size(node,2);
4
5 % ------------------- Triangulation --------------------------
6 % 2D
7 if ¬iscell(elem) && dim==2
8     h = patch('Faces', elem, 'Vertices', node);
9 end
10
```

```
% ------------------- Polygonal mesh -----------------------
if iscell(elem)
    if iscell(elem{1}), elem = vertcat(elem{:}); end
    max_n_vertices = max(cellfun(@length, elem));
    padding_func = @(vertex_ind) [vertex_ind,...
        NaN(1,max_n_vertices-length(vertex_ind))];  % function to pad the vacancies
    tpad = cellfun(padding_func, elem, 'UniformOutput', false);
    face = vertcat(tpad{:});  % polygon
    h = patch('Faces', face, 'Vertices', node);
end

facecolor = [0.5 0.9 0.45];
set(h,'facecolor',facecolor,'edgecolor','k');
axis equal; axis tight;
```

showsolution 函数的建立

showsolution 函数绘制解的网格图, 程序如下

```
function showsolution(node,elem,u)

dim = size(node,2);

% ------------------- Triangulation --------------------------
data = [node,u];
if ¬iscell(elem) && dim==2
    patch('Faces', elem,...
        'Vertices', data,...
        'FaceColor', 'interp',...
        'CData', u / max(abs(u)) );
end

% ------------------- Polygonal mesh -----------------------
if iscell(elem)
    max_n_vertices = max(cellfun(@length, elem));
    padding_func = @(vertex_ind) [vertex_ind,...
        NaN(1,max_n_vertices-length(vertex_ind))];  % function to pad the vacancies
    tpad = cellfun(padding_func, elem, 'UniformOutput', false);
    tpad = vertcat(tpad{:});
    patch('Faces', tpad,...
        'Vertices', data,...
        'FaceColor', 'interp',...
        'CData', u / max(abs(u)) );
end
axis('square');
sh = 0;
xlim([min(node(:,1)) - sh, max(node(:,1)) + sh])
ylim([min(node(:,2)) - sh, max(node(:,2)) + sh])
zlim([min(u) - sh, max(u) + sh])
xlabel('x'); ylabel('y'); zlabel('u');

view(3); grid on; % view(150,30);
```

简单说明一下.

- patch 也可以画空间中的直面, 此时只要把 'Vertices' 处的数据换为三维的顶点坐标.

- 对解 u, 显然 `data = [node,u]` 就是画图的三维点坐标.
- `patch` 后的

  ```
  'FaceColor', 'interp', 'CData', u / max(abs(u))
  ```

  是三维图形的颜色, 它根据 'CData' 数据进行插值获得 (不对颜色进行设置, 默认为黑色). 也可以改为二维的

  ```
  set(h,'facecolor',[0.5 0.9 0.45],'edgecolor','k');
  ```

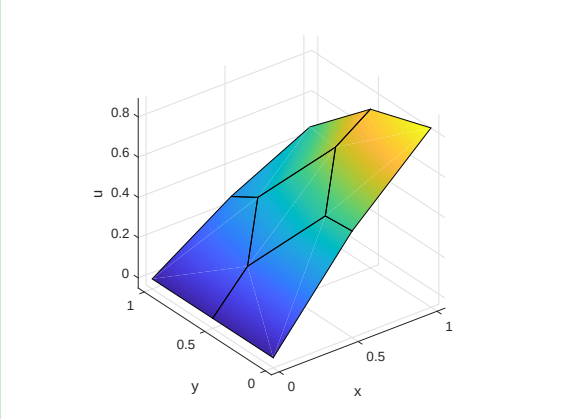
  此时显示的只是一种颜色, 但一般希望解有颜色的变化.
- 需要注意的是, 即便是三维数据, 若不加最后的

  ```
  view(3); grid on; %view(150,30);
  ```

  给出的也是二维图 (投影, 即二维剖分图).

例 1.5 如下画 $u(x,y)=\sin x\cos y$ 的图像

```
load('meshex1.mat');
x = node(:,1); y = node(:,2); u = sin(x).*cos(y);
showsolution(node,elem,u);
```



1.2 网格的标记

节点标记

如下给出图 2 中的节点编号

```
load('meshex1.mat');
showmesh(node,elem);
findnode(node);
```

函数文件如下

CODE 2. findnode.m

```
function findnode(node,range)
%Findnode highlights nodes in certain range.

hold on
dotColor = 'k.';
```

```
if nargin==1
    range = (1:size(node,1))';
end
plot(node(range,1),node(range,2),dotColor, 'MarkerSize', 15);
shift = [0.015 0.015];
text(node(range,1)+shift(1),node(range,2)+shift(2),int2str(range), ...
     'FontSize',8,'FontWeight','bold'); % show index number
hold off
```

当然也可简单改动以适用于三维情形.

单元标记

现在标记单元, 为此需给出单元重心, 从而标记序号. 重心使用 polycentroid.m 计算.

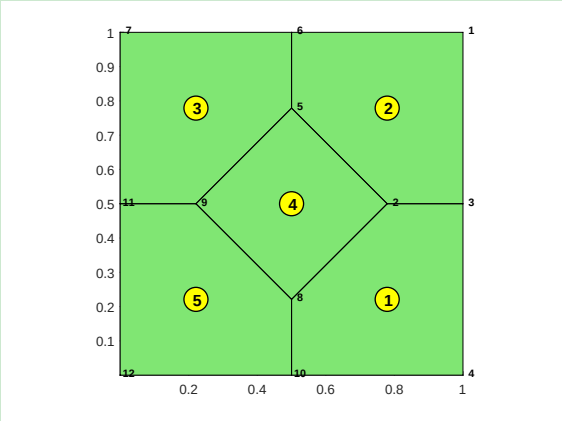


图 3. Polygonal mesh

主程序如下

```
load('meshex1.mat');
showmesh(node,elem);
findnode(node);
findelem(node,elem);
```

标记单元的函数如下

CODE 3. findelem.m

```
function findelem(node,elem,varargin)

hold on

NT = size(elem,1);
if ~iscell(elem) % transform to cell
    elem = mat2cell(elem,ones(NT,1),length(elem(1,:)));
end

range = 1:NT;
if nargin==3, range = unique(varargin{1}); end
range = range(:);

center = zeros(length(range),2);
s = 1;
for iel = range(:)' % only valid for row vector
```

```
    index = elem{iel};
    V = node(index, :);
    center(s,:) = polycentroid(V);
    s = s+1;
end
plot(center(:,1),center(:,2),'o','LineWidth',1,'MarkerEdgeColor','k',...
    'MarkerFaceColor','y','MarkerSize',18);
text(center(:,1)-0.01,center(:,2),int2str(range),'FontSize',12,...
    'FontWeight','bold','Color','k');

hold off

end
```

注 1.1 这里用圆圈标记单元, 对不同的剖分, 圆圈内的数字不一定在合适的位置, 需要手动调整偏移量. 为了方便, 可直接用红色数字标记单元序号.

edge 的生成

为了标记边, 先要生成边的数据结构 `edge`. iFEM 对应的介绍见如下网页

https://www.math.uci.edu/~chenlong/ifemdoc/mesh/auxstructuredoc.html

该网页给出了一些辅助网格数据结构 (三角剖分), 其中的 `edge` 记录每条边的顶点编号 (去除重复边). 以下设 `NT` 表示三角形单元的个数, `NE` 表示边的个数 (不重复), 并简单说明一下那里的思路.

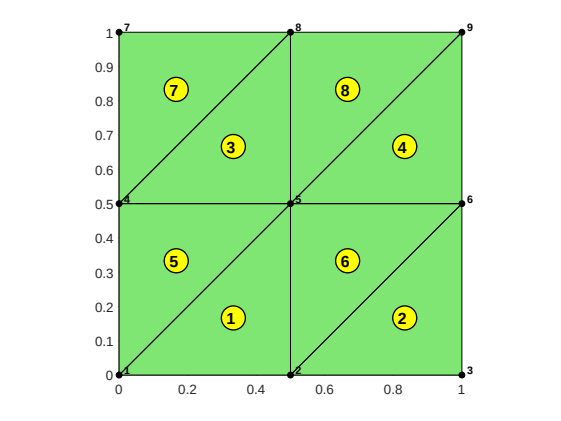
- 只要给出每条边两端的节点编号. 内部边在 `elem` 中会出现两次, 边界边只会出现一次, 为此可用 2 标记内部边, 1 标记边界边.
- 内部边在 `elem` 中会出现两次, 但它们是同一条边. 为了给定一致的标记, 规定每条边起点编号小于终点编号, 即 `edge(k,1)<edge(k,2)`.
- 对三角剖分, 通常规定三角形的第 i 条边对应第 i 个顶点. 例如, 设第 1 个三角形顶点顺序为 [1,4,5], 那么边的顺序应是 4-5, 5-1, 1-4. 在 MATLAB 中, 有如下对应

```
所有单元的第 1 条边: elem(:,[2,3]);  % NT * 2
所有单元的第 2 条边: elem(:,[3,1]);  % NT * 2
所有单元的第 3 条边: elem(:,[1,2]);  % NT * 2
```

为了满足 `edge(k,1)<edge(k,2)`, 只需将每行的两个元素排序. 在 MATLAB 中用 `sort(A,2)` 实现. 把这些边逐行排在一起, 则所有的边 (包含重复) 为

```
totalEdge = sort([elem(:,[2,3]); elem(:,[3,1]); elem(:,[1,2])],2);
```

它是 `3NT * 2` 的矩阵. `totalEdge` 见下面的右图.



	1	2
1	1	5
2	2	6
3	4	8
4	5	9
5	1	5
6	2	6
7	4	8
8	5	9
9	1	2
10	2	3
11	4	5
12	5	6
13	4	5
14	5	6
15	7	8
16	8	9
17	2	5
18	3	6
19	5	8
20	6	9
21	1	4
22	2	5
23	4	7
24	5	8

- 设 e_{ij} $(i<j)$ 表示起点 i 终点 j 的边的重数, 对应的矩阵可用 sparse 函数如下生成

```
sparse(totalEdge(:,1),totalEdge(:,2),1)
```

注意, 在 MATLAB 中, sparse 有一个特殊的性质 (summation property), 即当某个位置指标出现两次, 则相应的值会相加.

- 显然, sparse 命令产生的矩阵, 第一行对应起点 1 的边的重数, 第二行对应起点 2 的边的重数, 等等. 希望按下述方式排列边: 先排所有起点为 1 的边, 再排所有起点为 2 的边, 等等. 由于 find 是按列找非零元素, 因此要把矩阵进行转置, 即

```
sparse(totalEdge(:,2),totalEdge(:,1),1)
```

这样, 第一列对应的是起点为 1 的边, 第二列对应的是起点为 2 的边.

- edge 如下给出

```
[i,j,s] = find(sparse(totalEdge(:,2),totalEdge(:,1),1));
edge = [j,i];
```

注意因为前面进行了转置, 所以 `edge = [j,i]`. 边界边为

```
bdEdge = edge(s==1,:);
```

- 也可直接去重复实现, 即

```
edge = unique(totalEdge,'rows');
```

上面的思想也可用于多角形剖分, 只不过因边数不同, 要逐个单元存储每条边. 图 3 中第 1 个单元的顶点顺序为 [8,10,4,3,2], 按顺序 8-10, 10-4, 4-3, 3-2, 2-8 给出单元的边的标记. 所有边的起点是 elem 中元素按列拉直给出的结果, 而终点是对 [10,4,3,2,8] 的拉直. 如下实现

```
% the starting points of edges
v0 = horzcat(elem{:})';
% the ending points of edges
shiftfun = @(verts) [verts(2:end),verts(1)];
T1 = cellfun(shiftfun, elem, 'UniformOutput', false);
v1 = horzcat(elem{:})';
```

其他过程与三角剖分一致.

边的标记

有了 edge, 边可用中点进行标记.

```
midEdge = (node(edge(range,1),:)+node(edge(range,2),:))/2;
plot(midEdge(:,1),midEdge(:,2),'s','LineWidth',1,'MarkerEdgeColor','k',...
        'MarkerFaceColor',[0.6 0.5 0.8],'MarkerSize',20);
    text(midEdge(:,1)-0.015,midEdge(:,2),int2str(range), ...
        'FontSize',12,'FontWeight','bold','Color','k');
```

上面的过程用函数 findedge.m 实现, 如

```
load('meshex1.mat');
figure,
showmesh(node,elem);
bdInd = 1;
findedge(node,elem,bdInd);% only show boundary edges
figure,
showmesh(node,elem);
findedge(node,elem); % show all edges
```

注 1.2 为了统一虚拟元的编号规则, 三角形的第一条边就是第一个顶点连接的“右侧边”.