有限元分析及应用大作业

2019 年 12 月 2 日

Email: chunyu2018@foxmail.com

目 录

1 研究背景

本作业以大作业试题的第 16 题为主，基于 MATLAB 编写三节点三角形平面弹性力学的有限元分析程序，并以第 1 题为实例对自编的有限元程序进行测试。最后与商业有限元程序计算的结果进行对比。

1.1 问题描述

某水坝横截面为直角三角形，如图 1-1 所示。水坝宽 6 m，高 10 m。受齐顶的水压力作用。水坝由混凝土构成，假设其具有各向同性，弹性模量为 3×10^4 MPa，泊松比为 0.3。试对该水坝进行有限元分析。

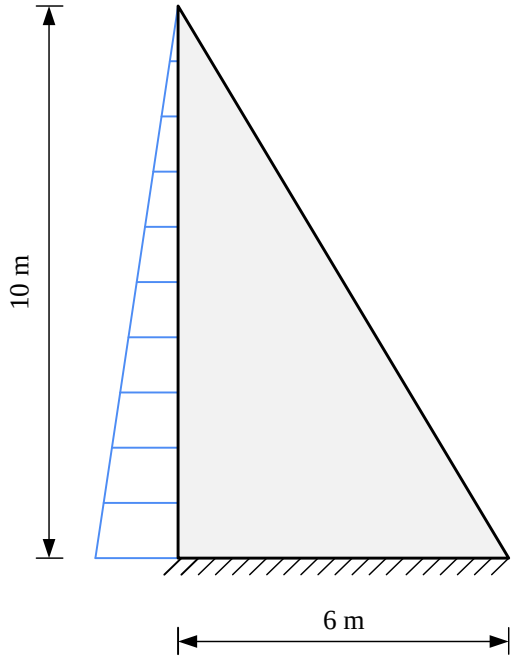


图 1-1 水坝横截面示意图

1.2 模型分析

该模型研究对象为水坝，通常情况下，水坝的长度方向的尺寸远大于横截面尺寸，可近似为无限长，因而每个横截面都可以近似为对称面，因而纵向应变近似为零；另一方面，水坝端部通常是结构的加强部位，端部位移被约束为零，据此也可以假设任意横截面的纵向应变为零。综合这两方面考虑，该问题可简化为

平面应变问题。

进一步，考虑水坝横截面为三角形，为了使其能够充分划分为多个有限元分析的基本单元，可选取基本单元为三角形单元。可行的网格划分如图 1-2 所示。

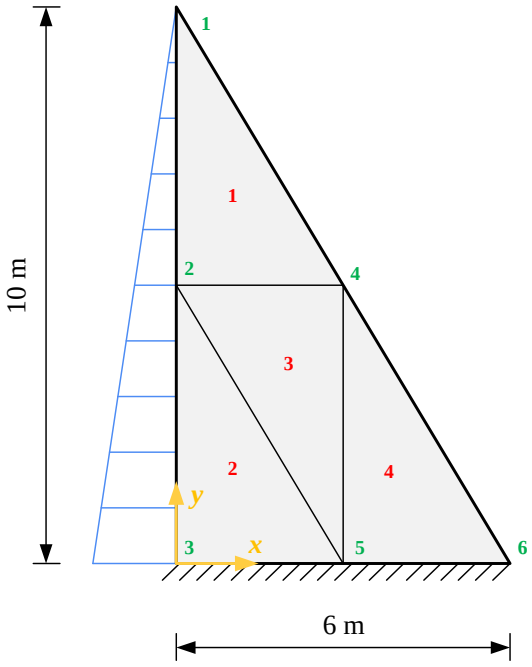


图 1-2 三角形单元划分示例

三角形单元进一步分为三节点三角形单元和六节点三角形单元，前者属于常应变单元，精度较低；后者虽具有更高的精度，但阶数较高、计算复杂。考虑到本算例中结构不存在容易产生应力集中的部位，且受力较为单一，整体结构较为简单。故选用三节点三角形单元。为提高计算精度，可通过增加单元数量的方式实现。

为计算的方便，模型计算时采用位移法。

2 自编有限元程序

MATLAB 是一款专注科学计算的软件，其对矩阵计算做了充分的优化。为了减小编程中矩阵计算的压力，基于该软件编写本作业相关的有限元程序。

2.1 程序顶层设计

根据有限元的基本思路，为实现结构分析，程序应当具备以下几个功能：

1）结构定义：即结构几何、力学等参数的输入端口；

2）单元划分：即划分网格，定义节点编号与单元编号；

3）刚度矩阵计算：包括单元刚度矩阵的计算和总体刚度矩阵的组装；

4）边界的引入：包括荷载分布为主的力边界条件和位移边界条件；

5）节点位移的计算：实际上是线性方程组的求解；

6）计算结果的整理：各节点应力应变分析。

其中，由于单元划分的方式众多，且易受边界形状约束，自动划分网格的功能已超出本人当前的能力范围。考虑到本题结构简单，网格数量不需要太多，因此采用手动划分网格的方法。

在全局变量设计上，需考虑以下变量：

1） 节点坐标变量：该变量用于记录节点的全局坐标，其行数等于节点数；由于本例为平面问题，故节点坐标变量采用两列，分别记录坐标值；

2） 单元定义变量：该变量用于指定组成各单元的节点编号，其行数等于单元数；对于三节点三角形单元，需采用三列来记录各单元的三个节点；

3） 力边界变量：该变量用于定义结构所受到外力，该变量可根据外力的不同而呈现不同结构，并结合荷载分配函数一同使用；

4） 位移约束变量：该变量用于指定特定节点的位移，考虑后面对总体刚度矩阵操作的便利，该变量可由两列构成，分别对应被约束的自由度编号以及该自由度的位移；

5） 单元面积：为了对常应变单元的结果进行优化，采用绕节点加权平均的方式，可以以面积为权重，为了避免单元面积的重复计算，可增加该变量以记录各单元面积；

6） 其他变量：如弹性模量，泊松比等。

2.2 刚度矩阵的计算与组装

总体刚度矩阵是由单元刚度矩阵依节点组装而成，在计算得到单元刚度矩阵时，可直接更新相应位置总体刚度矩阵的值。因此刚度矩阵可封装成一个函数。

2.2.1 单元刚度矩阵的计算

设三角形单元的节点编号分别为 i、j、k，全局坐标定义及单元示意如图 2-1 所示。

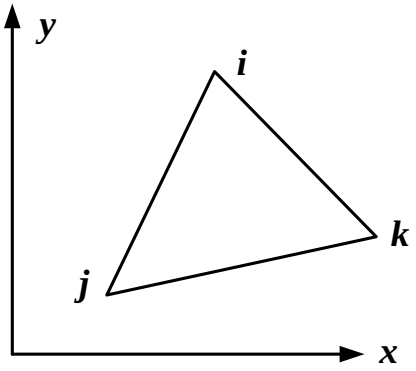


图 2-1 三节点三角形单元

该三角形的三个节点在平面内共有 6 自由度，分配到两个方向的位移，位移分量分别可以有三个待定系数，根据 Pascal 三角形选择位移函数为

$$\begin{cases} u(x,y)=\alpha_1+\alpha_2 x+\alpha_3 y \\ v(x,y)=\alpha_4+\alpha_5 x+\alpha_6 y \end{cases} \tag{2.1}$$

代入节点位移后，可得单元形函数为

$$N_k(x,y)=(a_k+b_k x+c_k)/(2A) \quad (k=i,j,m) \tag{2.2}$$

其中

$$\begin{cases} a_i=x_j y_m-x_m y_j \\ b_i=y_j-y_m \\ c_i=x_m-x_j \end{cases} \tag{2.3}$$

式(2.3)满足轮换对称性。在形函数表示下，单元位移函数可记为

$$\mathbf{f}=\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix}=\begin{bmatrix} N_i & 0 & N_j & 0 & N_m & 0 \\ 0 & N_i & 0 & N_j & 0 & N_m \end{bmatrix}\begin{bmatrix} u_i \\ v_i \\ u_j \\ v_j \\ u_m \\ v_m \end{bmatrix}=N\mathbf{d}^e \tag{2.4}$$

将位移函数代入几何方程，可得单元应变为

$$\boldsymbol{\varepsilon}=\frac{1}{2A}\begin{bmatrix} b_i & 0 & b_j & 0 & b_m & 0 \\ 0 & c_i & 0 & c_j & 0 & c_m \\ c_i & b_i & c_j & b_j & c_m & b_m \end{bmatrix}\mathbf{d}^e = B\mathbf{d}^e \tag{2.5}$$

考虑单元的弹性矩阵 D 对于平面应力问题和平面应变问题有所不同，对于平面应力问题，有

$$D_\sigma=\frac{E}{1-\mu^2}\begin{bmatrix} 1 & \mu & 0 \\ \mu & 1 & 0 \\ 0 & 0 & \dfrac{1-\mu}{2} \end{bmatrix} \tag{2.6}$$

对于平面应变问题，只需要同时将 E 换成 $E/(1-\mu^2)$，μ 换成 $\mu/(1-\mu)$，得到弹性矩阵为

$$D_\varepsilon=\frac{E(1-\mu)}{(1+\mu)(1-2\mu)}\begin{bmatrix} 1 & \dfrac{\mu}{1-\mu} & 0 \\ \dfrac{\mu}{1-\mu} & 1 & 0 \\ 0 & 0 & \dfrac{1-2\mu}{2(1-\mu)} \end{bmatrix} \tag{2.7}$$

根据不同的问题，应力可统一用应力矩阵 S 表示为

$$\boldsymbol{\sigma}=DB\mathbf{d}^e=S\mathbf{d}^e \tag{2.8}$$

对于一个平衡单元，设节点力向量为 $\mathbf{F}^e$，根据弹性体虚位移原理，得

$$\delta(\mathbf{d}^e)^T\mathbf{F}^e=\int_V \delta\boldsymbol{\varepsilon}^T\boldsymbol{\sigma}dV \tag{2.9}$$

整理上式，有

$$\begin{aligned} \delta(\mathbf{d}^e)^T\mathbf{F}^e &= \int_A \left(B\delta\mathbf{d}^e\right)^T DB\mathbf{d}^e t dA \\ \delta(\mathbf{d}^e)^T\mathbf{F}^e &= \delta(\mathbf{d}^e)^T\left(\int_A B^T DBt dA\right)\mathbf{d}^e \\ \mathbf{F}^e &= \left(\int_A B^T DBt dA\right)\mathbf{d}^e \end{aligned} \tag{2.10}$$

则单元刚度矩阵为

$$K^e=\left(\int_A B^T DBt dA\right)=B^T DBtA \tag{2.11}$$

式(2.11)的积分号可去，是因为对于三节点三角形单元，应变矩阵在单元内为定值，积分可简化为对面积单元的积分，其等于单元的面积。

2.2.2 总体刚度矩阵的组装

总体刚度矩阵是由单元刚度矩阵依据节点编号组装而成的，具体表述为：单元刚度矩阵 *i* 节点对 *j* 节点的刚度会贡献到整体刚度矩阵的第 *i* 行的第 *j* 列。这一点可借助 MATLAB 的矩阵索引实现。

刚度矩阵计算的 MATLAB 源码如下。

源码 1：刚度矩阵计算 (LearnFEM\three_node_triangle\getTriangleK.m)

```
% K = getTriangleK(node,elem,E,mu,t,type) compute stiffness for plane problem
% Three-node triangle element
%     node --- coordinate of nodes in observer's frame
%     elem --- node index of triangle elements
%     E    --- elastic modulus of every element
%     mu   --- Poisson ratio
%     t    --- thickness
%     type --- type = 1 for plane stress problem
%              type = 2 for plane strain problem

% XiaoCY 2019-11-28

%% main
function [K,A] = getTriangleK(node,elem,E,mu,t,type)
    [Nnode,~] = size(node);
    [Nelem,~] = size(elem);
    K = zeros(2*Nnode);
    A = zeros(Nelem);

    switch type
        case 1
            D = [ 1   mu    0
                mu    1    0
                0    0  (1-mu)/2];
            D = E/(1-mu^2)*D;
        case 2
            E = E/(1-mu^2);
            mu = mu/(1-mu);
            D = [ 1   mu    0
                mu    1    0
                0    0  (1-mu)/2];
            D = E/(1-mu^2)*D;
        otherwise
```

```
34            error('Wrong type flag!')
35    end
36
37    for n = 1:Nelem
38        x1 = node(elem(n,1),1);
39        y1 = node(elem(n,1),2);
40        x2 = node(elem(n,2),1);
41        y2 = node(elem(n,2),2);
42        x3 = node(elem(n,3),1);
43        y3 = node(elem(n,3),2);
44
45 %        a1 = x2*y3-x3*y2;
46 %        a2 = x3*y1-x1*y3;
47 %        a3 = x1*y2-x2*y1;
48        An = det([1 x1 y1; 1 x2 y2; 1 x3 y3])/2;
49        An = abs(An);
50        A(n) = An;
51
52        b1 = y2-y3;
53        b2 = y3-y1;
54        b3 = y1-y2;
55        c1 = x3-x2;
56        c2 = x1-x3;
57        c3 = x2-x1;
58
59        B = [ b1  0 b2  0 b3  0
60            0 c1  0 c2  0 c3
61            c1 b1 c2 b2 c3 b3]/An/2;
62        Ke = B'*D*B*t*An;
63
64        Index = zeros(1,6);
65        Index([1 3 5]) = elem(n,:)*2-1;
66        Index([2 4 6]) = elem(n,:)*2;
67        K(Index,Index) = K(Index,Index)+Ke;
68    end
69 end
```

2.3 边界条件的引入与求解

边界条件可分为力边界条件和位移边界条件，其中，力边界条件以节点编号为索引，给出节点编号和外力作用即可；位移边界条件可按自由度进行索引，一般来说，第 i 节点的 x 方向平动位移对应于第 $2i$-1 自由度，y 方向平动位移为第

2*i* 自由度。

2.3.1 集中力边界条件

对于有集中力作用的边界，可以通过在集中力处设置节点，则该集中力直接表现为节点力，可直接整合到总体力向量中。因此，集中力的边界条件不进行单独编程。集中力可以用三列表示，其中，第一列为节点编号，第二列和第三列分别稳集中力在全局坐标系两个方向的投影。但是总体刚度矩阵对应的力向量为单列向量，因此集中力的向量应当修正，修正规范为：总体力为列向量，其行号与自由度编号相同，第 *i* 行的值为第 *i* 自由度的力约束。

2.3.2 面分布力边界条件

三维结构的面分布力在二维情形下表现为线分布，如图 2-2 所示。

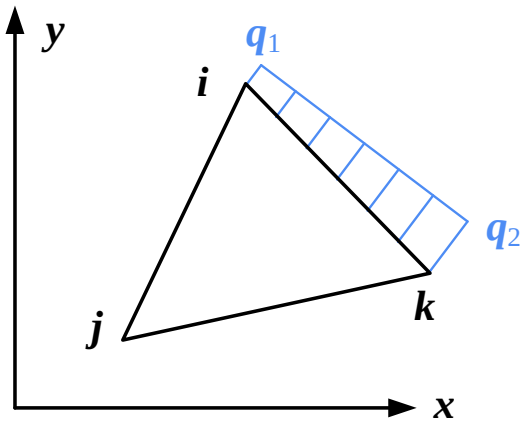


图 2-2 面分布力二维表现

对于三节点三角形，其位移为线性模型，可根据静力学等效原则求得各节点的等效载荷，即

$$\begin{cases} F_i = \dfrac{1}{3}q_1 l + \dfrac{1}{6}q_2 l \\ F_k = \dfrac{1}{6}q_1 l + \dfrac{1}{3}q_2 l \end{cases} \tag{2.12}$$

同一节点获得的来自不同边界的等效载荷只需要简单相加就可以得到节点的总等效载荷。这里，我们假设单元的面力分布为线性，是考虑到大多数工程载

荷为线性载荷，对于非线性载荷，我们可以增加单元数量，使载荷在单元边界上近似为线性。这样可以减轻编程的负担。

面分布力分配的函数需要给定面力在各节点处的值，即其包含三列，分别为节点编号和该节点分布力对应的数值在全局坐标下两个分量的投影。据此，面分布力分配函数如下：

源码 2：面分布力的分配 (LearnFEM\three_node_triangle\distriQs.m)

```
% Fn = distriQs(node,inode,q) distribute concentrated force to nodes
%     node  --- coordinate of nodes in observer's frame
%     q     --- distributed force, contains [nodeIdx qx qy]

% XiaoCY 2019-11-28

%% main
function Fn = distriQs(node,q)
    [Nnode,~] = size(node);
    [Nq,~] = size(q);
    Fn = zeros(Nnode*2,1);

    for n = 1:Nq-1
        n1 = q(n,1);
        n2 = q(n+1,1);
        q1 = q(n,[2 3])';
        q2 = q(n+1,[2 3])';
        L = sqrt((node(n1,1)-node(n2,1))^2+(node(n1,2)-node(n2,2))^2);

        idx1 = [2*n1-1; 2*n1];
        idx2 = [2*n2-1; 2*n2];
        Fn(idx1) = Fn(idx1)+q1*L/3+q2*L/6;
        Fn(idx2) = Fn(idx2)+q1*L/6+q2*L/3;
    end
end
```

2.3.3 体分布力边界条件

体分布力相对面分布力计算更为简单，在线性位移模式下，只需要将体分布力平均分配到各节点，然后将各节点在不同单元内分配到的节点力相加即可。源码如下：

源码 3：体分布力的分配 (LearnFEM\three_node_triangle\distriQv.m)

```
% Fn = distriFc(node,elem,ielem,Fe) distribute distributed force
to nodes
%    node  --- coordinate of nodes in observer's frame
%    elem  --- node index of triangle elements
%    Q     --- distributed force, contains [elemIdx Qx Qy]

% XiaoCY 2019-11-28

%% main
function Fn = distriQv(node,elem,Q)
    [Nnode,~] = size(node);
    Fn = zeros(Nnode*2,1);

    for n = Q(:,1)'
        x1 = node(elem(n,1),1);
        y1 = node(elem(n,1),2);
        x2 = node(elem(n,2),1);
        y2 = node(elem(n,2),2);
        x3 = node(elem(n,3),1);
        y3 = node(elem(n,3),2);

        A = det([1 x1 y1; 1 x2 y2; 1 x3 y3])/2;
        A = abs(A);
        Fe = Q(n,[2 3])*A;

        nFx = elem(n,:)*2-1;
        nFy = elem(n,:)*2;
        Fn(nFx) = Fn(nFx)+Fe(1)/3;
        Fn(nFy) = Fn(nFy)+Fe(2)/3;
    end
end
```

2.3.4 位移边界条件与求解

位移边界条件通常与总体刚度矩阵的处理放在一起，一般有划行列法、乘大数法、对角元素置一法等。出于 MATLAB 对矩阵操作的便捷性，本程序在引入位移约束时采用对角元素置一法。

对角元素置一法的基本思路是将已知自由度对应的行列中除了对角位置，其他位置元素全部置零，对角元素置一，然后将各节点力减去相应刚度与已知位移

的乘积。例如，设第 r 自由度的位移为 $u_r = c_r$，则刚度矩阵可化为

$$\begin{bmatrix} k_{11} & k_{12} & 0 & \cdots & k_{1N} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ k_{N1} & k_{N2} & 0 & \cdots & k_{NN} \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_r \\ \vdots \\ v_N \end{bmatrix} = \begin{bmatrix} u_1 - k_{1r}c_r \\ \vdots \\ c_r \\ \vdots \\ v_N - k_{Nr}c_r \end{bmatrix} \tag{2.13}$$

边界条件引入后，可直接利用 MATLAB 矩阵“左除”的方法解得结点位移。源码如下：

源码 4：约束引入与求解 (LearnFEM\three_node_triangle\solveTriangle.m)

```
% d = solveTriangle(K,F,cons) introduces constraint and solves FEM
%     K   --- total stiffness matrix
%     F   --- force vector on nodes
%    cons --- displacement constraint, constains [dof,u]

% XiaoCY 2019-11-28

%% main
function d = solveTriangle(K,F,cons)
    for n = 1:length(cons(:,1))
        idx = cons(n,1);
        K(idx,:) = 0;
        K(:,idx) = 0;
        K(idx,idx) = 1;
        F = F-K(:,idx)*cons(2);
        F(idx) = cons(n,2);
    end
    d = K\F;
end
```

2.4 计算结果整理

上述程序已完成各节点位移的计算，进一步，重新将节点位移回代到各个单元，即可得到各单元的应力应变情况。除此之外，计算所获得的数值结果应当以一定的方式展现出来，以便在整体进行判断和分析。本小节将讨论计算结果的整理。

2.4.1 节点应力应变的计算

在第 2.2.1 小节讨论了单元应力、应变与节点位移之间的关系，故只需要将节点位移重新代入式(2.5)和式(2.8)即可。对于三节点三角形单元，应力和应变在单元内为定值，故可假设节点应力与单元应力相等、节点应变与单元应变相等。考虑到节点可能为多个单元所共有，则节点应力应变会出现多个不同的数值，我们可以用平均的方法取舍。本程序采用绕节点面积加权的方法求平均值，即对于节点 i，其应力（或应变）等于绕该节点所有单元应力（或应变）对面积的加权平均，用公式表示为：

$$\sigma_i = \frac{\sum_e \sigma_i^e A^e}{\sum_e A^e} \tag{2.14}$$

求解应力应变的源代码如下：

源码 5：节点应力应变求解 (LearnFEM\three_node_triangle\solveStrAll.m)

```
% This function is to solve strain and stress
% [epsi,sigm] = solveStrAll(node,elem,A,d,E,mu,type)
%     epsi --- strain
%     sigm --- stress
%     node --- coordinate of nodes in observer's frame
%     elem --- node index of triangle elements
%     A    --- elements' area
%     d    --- nodes' displacement
%     E    --- elastic modulus of every element
%     mu   --- Poisson ratio
%     type --- type = 1 for plane stress problem
%              type = 2 for plane strain problem

% XiaoCY 2019-11-28

%% main
function [epsi,sigm] = solveStrAll(node,elem,A,d,E,mu,type)
    [Nnode,~] = size(node);
    [Nelem,~] = size(elem);
    strA = zeros(Nnode,7);

    switch type
        case 1
            D = [ 1   mu    0
```

```
25                mu    1    0
26                0    0  (1-mu)/2];
27            D = E/(1-mu^2)*D;
28        case 2
29            E = E/(1-mu^2);
30            mu = mu/(1-mu);
31            D = [ 1   mu    0
32                mu    1    0
33                0    0  (1-mu)/2];
34            D = E/(1-mu^2)*D;
35        otherwise
36            error('Wrong type flag!')
37    end
38
39    for n = 1:Nelem
40        x1 = node(elem(n,1),1);
41        y1 = node(elem(n,1),2);
42        x2 = node(elem(n,2),1);
43        y2 = node(elem(n,2),2);
44        x3 = node(elem(n,3),1);
45        y3 = node(elem(n,3),2);
46
47        b1 = y2-y3;
48        b2 = y3-y1;
49        b3 = y1-y2;
50        c1 = x3-x2;
51        c2 = x1-x3;
52        c3 = x2-x1;
53
54        BA = [ b1  0 b2  0 b3  0
55            0 c1  0 c2  0 c3
56            c1 b1 c2 b2 c3 b3]/2;
57
58        idx = [ elem(n,1)*2-1
59                elem(n,1)*2
60                elem(n,2)*2-1
61                elem(n,2)*2
62                elem(n,3)*2-1
63                elem(n,3)*2 ];
64        dn = d(idx);
65
66        epsiAn = BA*dn;
67        sigmAn = D*epsiAn;
```

```
68          strA(elem(n,:),1:3) =
    strA(elem(n,:),1:3)+repmat(epsiAn',3,1);
69          strA(elem(n,:),4:6) =
    strA(elem(n,:),4:6)+repmat(sigmAn',3,1);
70          strA(elem(n,:),7) = strA(elem(n,:),7)+A(n);
71      end
72
73      epsi = strA(:,1:3)./strA(:,7);
74      sigm = strA(:,4:6)./strA(:,7);
75  end
```

2.4.2 结果的图形化输出

上述计算已经完全求解了有限元问题，但结果以数值的方式储存。最直观展现计算结果的方式是用图形来展现。对于二维问题，有两种绘图方式：

1. 采用三维坐标：两位绘制结构平面，第三维表示结果；
2. 采用颜色表示：仅绘制结构平面，用颜色表示结果。

三维坐标绘图精度高，能够很容易寻找极值出现的位置，但不容易讨论计算结果在整体的分布特性。考虑到自编有限元采用的三节点三角形单元并不具有特别高的精度，且一般分析中更注重物理量的分布状况，因此本程序采用第二中绘图方法，用颜色表示结果的大小。

更进一步，由于计算结果都转移到了节点上，因此可以对节点进行插值来近似单元内应力应变分布，这一点利用 MATLAB 的命令可以很容易实现。结果的图形化输出函数源码如下：

源码 6：结果的图形化输出 (LearnFEM\three_node_triangle\showResult.m)

```
1   % showResult(node,elem,rst) plots result in color map
2   %     node --- coordinate of nodes in observer's frame
3   %     elem --- node index of triangle elements
4   %     rst  --- result to be showed
5
6   % XiaoCY 2019-11-28
7
8   %% main
9   function showResult(node,elem,rst)
10      [Nelem,~] = size(elem);
11      x = zeros(3,Nelem);
12      y = zeros(3,Nelem);
```

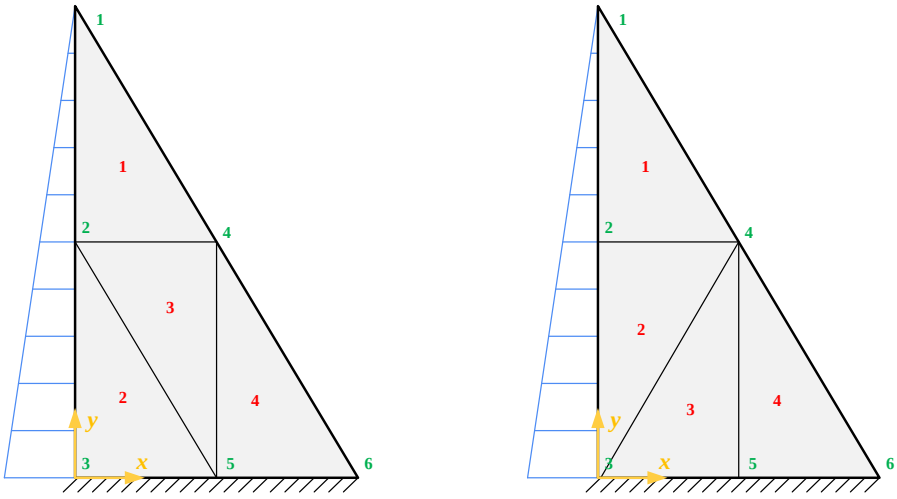
```
    c = zeros(3,Nelem);

    figure
    for n = 1:Nelem
        xn = node(elem(n,:),1);
        yn = node(elem(n,:),2);
        cn = rst(elem(n,:));

        x(:,n) = xn;
        y(:,n) = yn;
        c(:,n) = cn;
    end
    patch(x,y,c,'Facecolor','interp','EdgeColor','interp')
    grid on
    colorbar
    axis equal
end
```

3 计算结果对比与分析

为探究不同网格情况下有限元程序计算结果的异同，利用自编函数对比了不同网格划分、不同网格数量和不同边界条件的差异；为判断自编函数的正确性，将自编函数计算结果与商业化程序的计算结果进行了粗略对比。

3.1 不同网格划分对比

针对第 1.1 小节描述的结构，采用两种方法划分网格，如图 3-1 所示。



（1）网格 A　　（2）网格 B

图 3-1 不同网格划分

图中，绿色数字表示节点编号，红色数字为单元编号。全局坐标系如黄色箭头所示。以网格 A 为例，调用自编函数进行计算的代码如下：

源码 7：计算示例 (LearnFEM\three_node_triangle\example1.m)

```
clear;clc
close all

E = 3e10;
mu = 0.3;
t = 1;
type = 2;

node = [ 0 10; 0 5; 0 0; 3 5; 3 0; 6 0];
elem = [1 2 4
        2 3 5
        2 4 5
        4 5 6];
q = [1 0 0
    2 5e4*t 0
    3 10e4*t 0];
cons = [5 0; 6 0; 9 0; 10 0; 11 0; 12 0];

[K,A] = getTriangleK(node,elem,E,mu,t,type);
F = distriQs(node,q);
d = solveTriangle(K,F,cons);
[epsi,sigm] = solveStrAll(node,elem,A,d,E,mu,type);

showResult(node,elem,epsi(:,1))
```

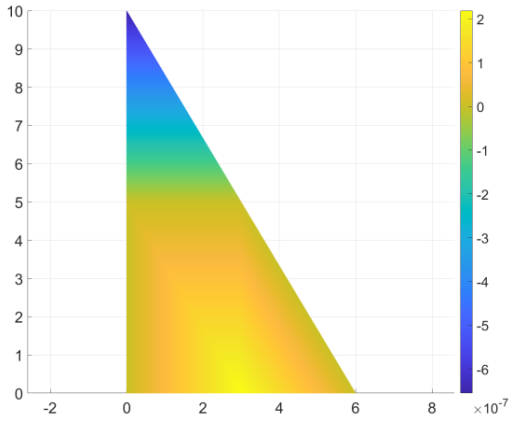
该程序中，前 2 行是 MATLAB 环境的初始化，包括清除变量和关闭其他图窗；4~7 行是导入结构的力学参数；9~17 行定义了节点坐标和单元组成，并引入边界条件。21~22 行是计算的核心部分，其相关函数见第 2 节。最后一行是计算结果的展示，这里仅展示了应变在 x 方向的分量。

两种不同网格划分下，结构在 x 方向的应变分量如图 3-2 所示。两种网格划分情况下差异很大，导致这种差异的主要原因有：

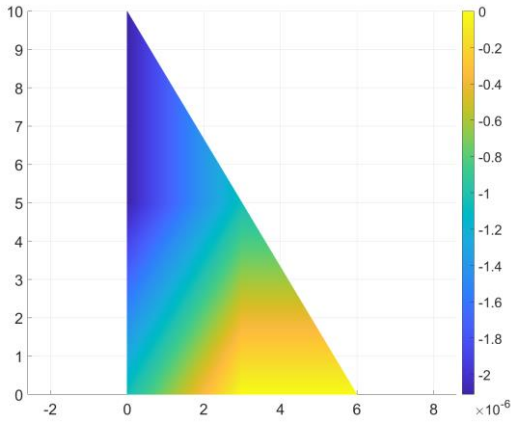
1）网格数量少，计算精度低；

2）位移边界条件约束太强，整体结果受限于边界条件的设置；

3）应变是位移的导数，数值解的导数会导致降解，精度变差。



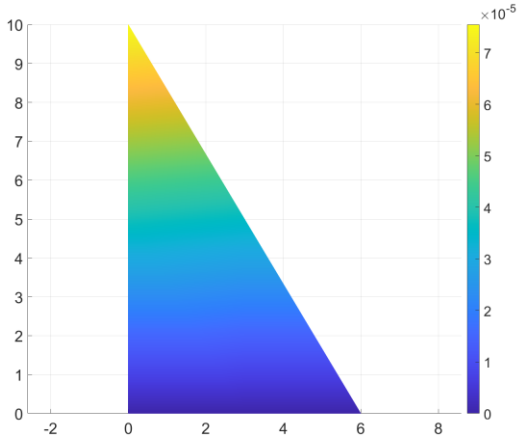
（1）网格 A 计算结果



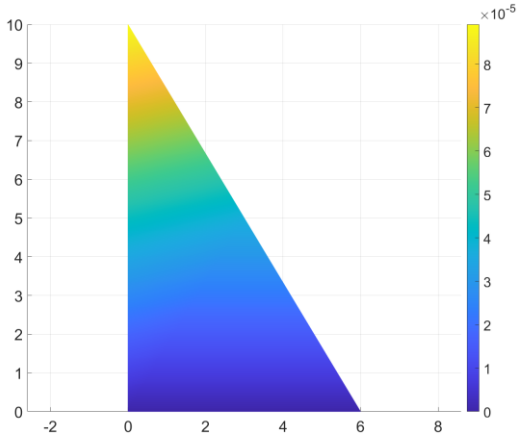
（2）网格 B 计算结果

图 3-2 不同网格划分 *x* 方向应变分布

考虑到上述第三点原因，不妨比较结构的位移，如图 3-3 所示。



（1）网格 A 计算结果



（2）网格 B 计算结果

图 3-3 不同网格划分 *x* 方向位移分布

可见，由于直接采用位移法，两种网格划分下位移的计算几乎一致，但数量上略有差别，这受限于网格的划分数量。并由此对比可以确定，两种网格划分下应变的差异确实是来自导数使方程降阶引入的误差。为使应变精度提高，可增加网格的数量，或使用更高阶单元，甚至可直接采用力法进行有限元计算。

3.2 不同网格数量对比

为比较不同网格数量下有限元计算的结果，补充如图 3-4 所示网格划分。计算程序除节点坐标和单元定义不同外，其他与源码 7 类似，此处不再赘述。

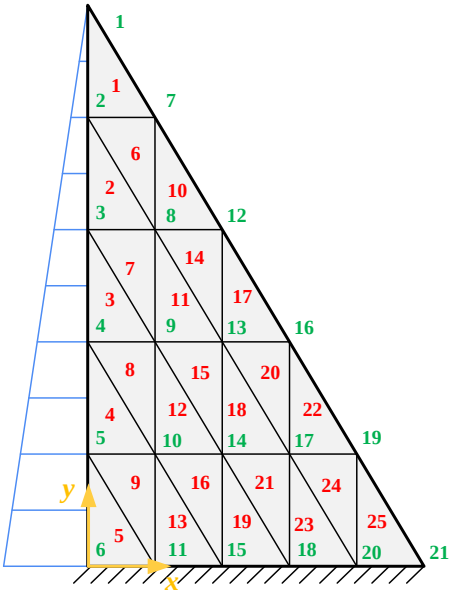
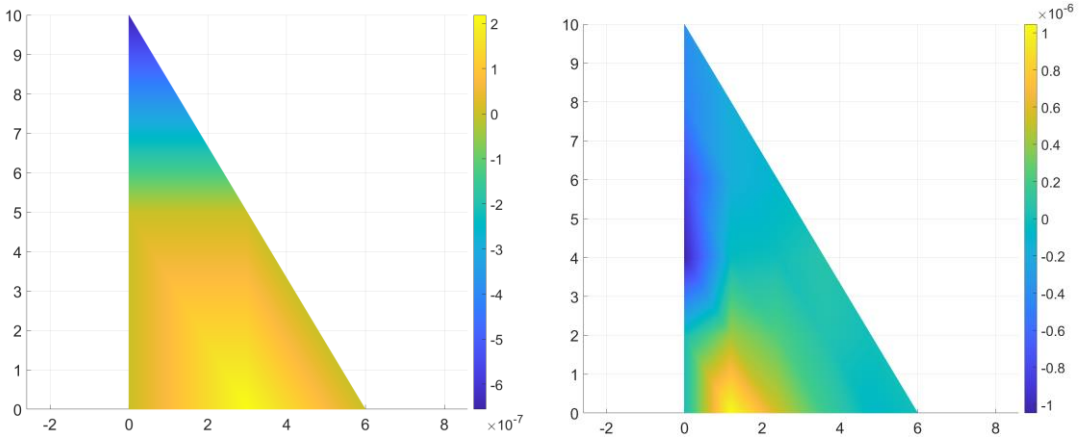


图 3-4 网格 C 示意图

对比网格 A 和网格 C 在 x 方向应变的计算结果，如图 3-5 所示。

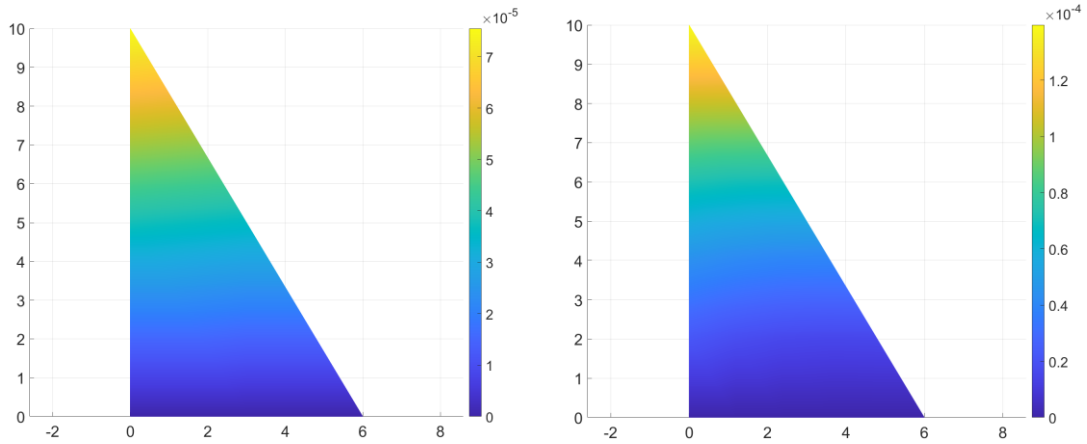


（1）网格 A 计算结果　　（2）网格 C 计算结果

图 3-5 不同网格数量 x 方向应变分布

容易看出，网格 C 由于数量更多，能够看到更加精细的应变分布。与经验判断相同，坝体左侧受压面应当以压应力为主，因而产生的应变在 x 方向主要为压应变（表现为负值）。而为了支撑该压力，坝体由此应当以一定的压应变抵消左侧的水压力，根据圣维南原理，由于应力传递将使应力变小且趋于均匀，右侧压应变较左侧更小。同时，类比梁的下侧受拉，坝体左下角将会由于水压产生一定的拉力，故应变为正。

再对比不同网格数量下位移计算的结果，如图 3-5 所示。两种方法获得的位移分布相似，但由于网格数量导致的整体精度不同，位移计算的数值略有差异。



（1）网格 A 计算结果　　（2）网格 C 计算结果

图 3-6 不同网格数量 *x* 方向位移分布

3.3 不同边界条件对比

在上述计算中，位移边界条件的设置依据是坝体的底端不发生位移，即下边界所有节点的 x、y 方向位移分量均为零。这种约束实际上是过强的约束：实际的边界如果位移严格为零，局部刚度表现为无限大，任意微小应变将导致无限大应力，这与现实不符；且实际应力过大时材料进入非线性状态，局部产生微小裂纹，最终也表现为位移约束的放松。因此，在计算中可尝试使用不同的位移约束来对比计算结果。

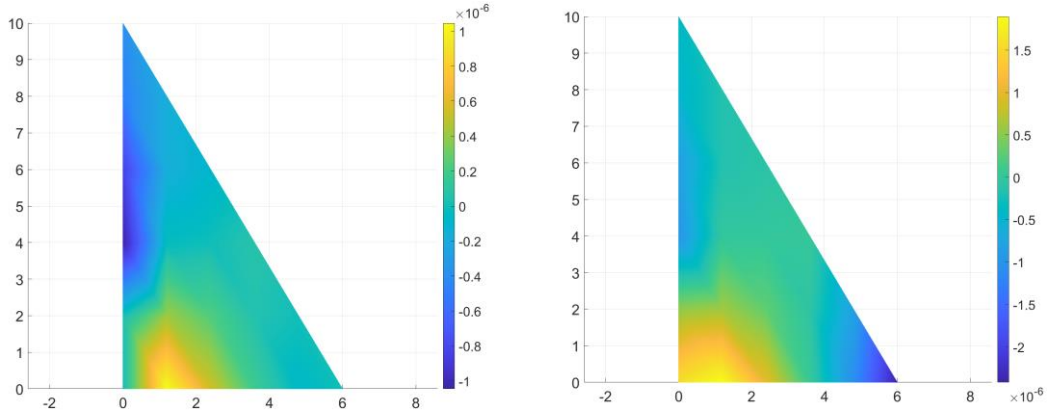
应当注意的是，约束的放松并不是任意的，必须消除结构的刚体位移，否则总体刚度矩阵仍为奇异矩阵，节点位移无法求解。

本例基于网格 C，对比以下两种位移边界条件：

边界 A：下边界所有节点所有方向位移为零；

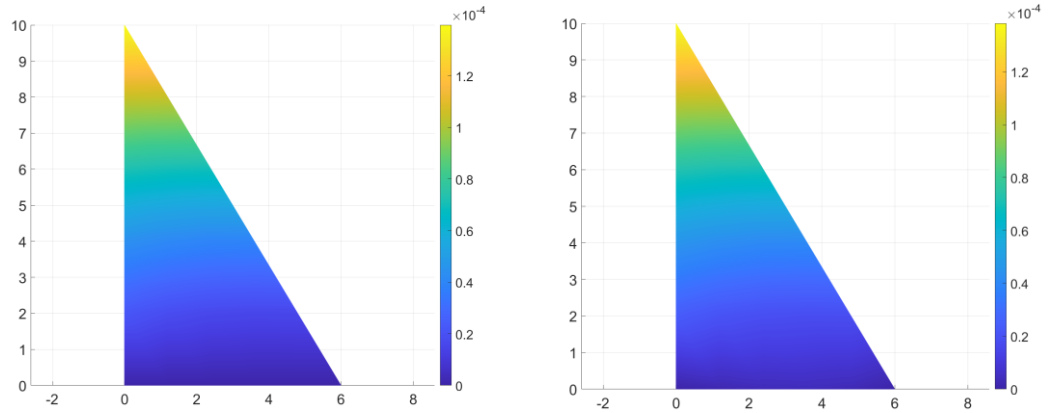
边界 B：下边界所有节点 y 方向位移为零，仅 6、21 号节点 x 方向位移为零。

两种边界条件下有限元计算结果分别如图 3-7 和图 3-8 所示。可以看出，边界条件改变时，边界附近的应变分布差异较大，而远端的分布差异较小，但数值结果存在差异。同样，由于采用的是位移法，两种边界条件下位移计算一致。这里再一次体现出基于位移法的有限元对结构位移的计算精度和一致性都很高。



（1）边界 A 计算结果　　（2）边界 B 计算结果

图 3-7 不同边界条件 *x* 方向应变分布



（1）边界 A 计算结果　　（2）边界 B 计算结果

图 3-8 不同边界条件 *x* 方向位移分布

3.4 自编函数与商业工具包对比

为了与自编函数的结果进行对比，采用 MATLAB 集成的偏微分工具箱（PDE toolbox）进行有限元分析。源代码如下：

源码 8：PDE 求解有限元(LearnFEM\three_node_triangle\solvePDE.m)

```
% Solve FEM using MATLAB PDE toolbox

% XiaoCY 2019-11-30

%% main
clear;clc
close all

% create model
model = createpde('structural','static-planestrain');
```

```
gd = [2 3 ...
    0 0 6 ...
    10 0 0]';
g = decsg(gd);
geometryFromEdges(model,g);

figure
pdegplot(model,'EdgeLabels','on','VertexLabels','on')
grid on

% add properties
structuralProperties(model,'YoungsModulus',3e10,'PoissonsRatio',0.
3);

% boundary condition
% structuralBC(model,'Edge',2,'XDisplacement',0);
structuralBC(model,'Edge',2,'YDisplacement',0);
structuralBC(model,'Vertex',2,'XDisplacement',0);
q = @(loc,states) 1e4*(10-loc.y);
structuralBoundaryLoad(model,'Edge',1,'Pressure',q);

% solve FEM
generateMesh(model,'GeometricOrder','linear','Hmin',1);
R = solve(model);

% results
figure
pdemesh(model)
grid on

figure
pdeplot(model,'XYData',R.Strain.exx,'ColorMap','parula')
axis equal
grid on

figure
pdeplot(model,'XYData',R.Displacement.ux,'ColorMap','parula')
axis equal
grid on
```

该工具包具有自动划分网格的功能，计算中采用的网格如图 3-9 所示。为了与自编代码相对比，自动划分网格时限制了最小边长为 1 m。

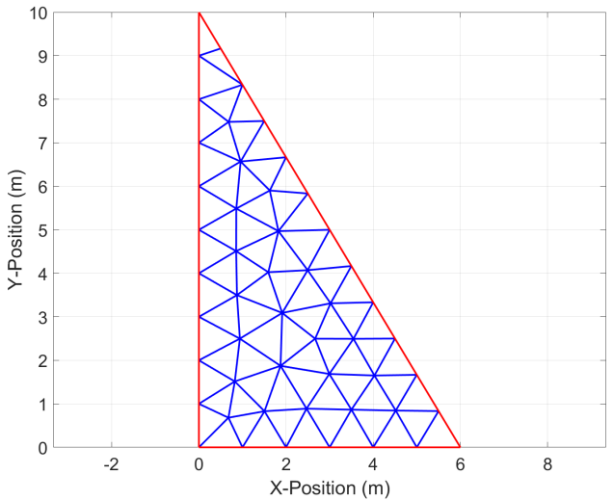
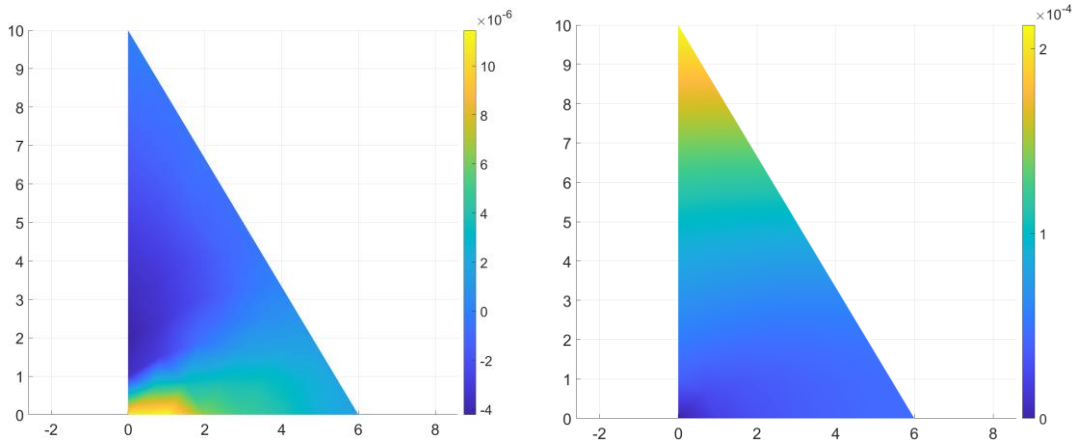


图 3-9 PDE 工具箱自动网格划分

利用该工具包进一步求解 x 方向的应变分布和位移分布，如图 3-10 所示。



（1）x 方向的应变分布　　（2）x 方向的位移分布

图 3-10 PDE 工具箱计算结果

对比图 3-5、图 3-6 和图 3-10 的结果，自编程序与商业化集成工具包计算的结果在分布上并无大异，但在数值上略有区别。区别主要来自于网格划分的不同以及边界条件设置时存在的些许差异。此外，自编程序中节点应力应变来自绕节点单元的面积加权平均，而商业化工具包 PDE 并未说明如何获得，或许这里的数据处理存在差异，这也将导致计算结果的不同。

3.5 计算结果的分析

综合上述计算结果，可以判断出坝体受齐顶水压力作用时，坝体与水接触面大部分处于受压状态，且顶部横向位移最大。由于坝体底部存在位移约束，使得

底部应变分布对底部边界条件较为敏感。本程序基于位移法计算，因此对结构位移的计算较为准确，且在远离边界的位移趋于一致。

除坝体大部分受压外，底部近水侧存在局部拉应变。考虑混凝土结构的抗拉能力小，根据该有限元仿真结果，在坝体底部受拉区域应当适当加入钢筋，增强混凝土结构的局部抗拉能力。

4 总结与展望

本次大作业以水坝平面应变问题为例，通过自己编写的有限元计算程序对各种设置下有限元计算结果进行了对比，其中包括不同划分网格方式对计算结果的影响、不同单元数量对计算结果的影响以及不同边界条件对计算结果的影响。最后利用商业化工具包的计算进行对比，结果表明自编函数基本正确，但结果的精度受限于单元数量。

通过此次大作业，本人基本掌握了有限元的一般方法，认识了网格划分和边界条件对计算结果的影响，基本满足了本人后期研究所涉及的关于有限元分析的需求。此次大作业仅仅以平面问题最基本的三节点三角形单元为基础，尚未引入六节点三角形、各种四边形单元等高阶单元，也尚未对三维结构进行建模。除此之外，本次作业也只以应力场为研究对象，未尝试磁场、温度场甚至多场耦合的情况进行尝试。这些都可在后期使用商业有限元软件时继续学习和强化认识。