数学应用課題选編

第三集

（参考教材）

上海师范大学数学系编

毛主席語录

列宁为什么說对资产阶级專政，這个问題要搞淸楚。這个问題不搞淸楚，就会变修正主义。要使全国知道。

教育必須为无产阶级政治服务，必須同生产劳动相结合。

农业学大寨

目　　录

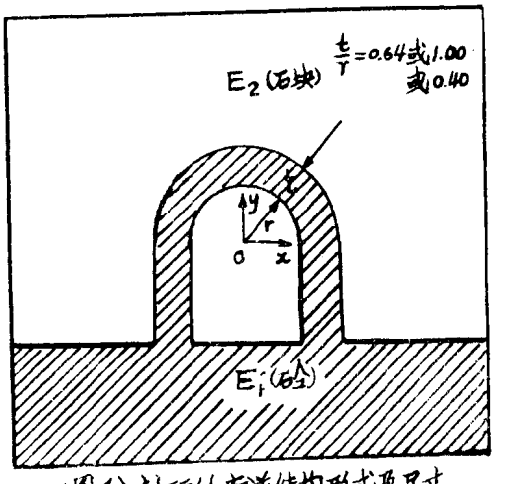
用有限元素法进行水坝廊道应力分析

1973年，我们首届工农兵学员和部分教师在开门办学中与浙江大学固体力学教研组协作为浙江云溪水库浆砌块石坝进行了某些孔口和廊道应力分析的数值计算。我们运用有限元素法对标准廊道附近的应力分布进行研究。並为此编写了计算程序，在709机上进行计算。該程序对浆砌块石坝中一种结构形式的标准廊道计算了十例。计算结果表明，浆砌块石坝确能明显地节省水泥，同时还能节省钢筋。为了說明计算的可靠性，对混凝土坝情形的廊道应力分布用同一程序进行了计算，並将计算点与試验值及解析解进行了比较和分析。我们还对同一程序进行了很少的修改而对圆孔情形又进行了计算，並将计算值与精确解进行了比较，都极为符合，表明计算有較高的精度。在计算程序中还設置了校核部分，计算结果表明符合原来設想的要求。

(一)问題的处理

坝体内廊道的应力分析，一般作为一个弹性力学的平面应变问題来处理。廊道结构形式如图1。我们分别计算三种载荷情况（图2）。由于问題的对称性，只需取右面一半进行计算，坐标轴选取如图1所示。

计算中認为坝体内混凝土部分及浆砌块石部分均为各向同性的均匀连续体，分别具有弹性模量E_1和E_2，泊松比均取$\frac{1}{6}$。



〈图1〉計算的廊道結构形式及尺寸

(二)计算步骤

1. 用三角形单元划分计算区域，得564个单元，344个结点。每个单元的三个结点i，j，m按由小至大顺序编号，x_i，

〈图2〉 三种对称载荷示意图

y_i 表示结点 i 的坐标。逐个计算各单元 e 的刚度矩阵 $[K]^e$，而 $[K]^e$ 具有如下形式：

$$[K]^e=\begin{bmatrix} K_{ii}^e & K_{ij}^e & K_{im}^e \\ K_{ji}^e & K_{jj}^e & K_{jm}^e \\ K_{mi}^e & K_{mj}^e & K_{mm}^e \end{bmatrix} \qquad (1)$$

其中

$$[K_{rs}]^e=\frac{E(1-\mu)t}{4(1+\mu)(1-2\mu)|\Delta|}\begin{bmatrix} b_rb_s+\frac{1-2\mu}{2(1-\mu)}c_rc_s & \frac{\mu}{1-\mu}b_rc_s+\frac{1-2\mu}{2(1-\mu)}c_rb_s \\ \frac{\mu}{1-\mu}c_rb_s+\frac{1-2\mu}{2(1-\mu)}b_rc_s & c_rc_s+\frac{1-2\mu}{2(1-\mu)}b_rb_s \end{bmatrix}$$

$$(r=i,j,m\ ;\ s=i,j,m)$$

$$b_i=y_j-y_m\ ,\quad c_i=x_m-x_j\quad (i,j,m)^*$$

$|\Delta|=\frac{1}{2}|b_jc_m-b_mc_j|$ 表示该三角形单元的面积。

E：弹性模量，对于浆砌块石部分的单元取 $E=5\times10^4\ kg/cm^2$，

对于混凝土部分的单元取 $E=25\times10^4\ kg/cm^2$，

μ：泊松比，对所有单元取 $\frac{1}{6}$，

t：单元厚度，本问题中取 1.

2. 单元刚度矩阵 $[K_{rs}]^e$ 的下标 r, s 分别指出了它在总刚度矩阵中所占的行、列位置。按照这个原则由564个单元刚度矩阵，得出整体刚度矩阵 $[K]$.

$$[K]=\begin{bmatrix} K_{11},\cdots,K_{1j},\cdots,K_{1,344} \\ \cdots\cdots, \\ K_{i1},\cdots,K_{ij},\cdots,K_{i,344} \\ \cdots\cdots, \\ K_{344,1},\cdots,K_{344,j},\cdots,K_{344,344} \end{bmatrix}_{(688\times688)} \qquad (2)$$

其中 $[K_{ij}]=\sum_e[K_{ij}]^e$.

3. 分别按三种情形进行计算：承受铅直载荷；承受水平载荷；承受剪切载荷。

---

*表示该式按 i,j,m 顺序改变 x,y 的下标，便得到其余的 b_j,c_j，b_m,c_m. 下同.

将载荷等效移置到各结点上得结点载荷向量$\{F\}$。

4. 由于问题的对称性，在对称面上的结点要进行一定的约束处理。在承受铅直水平载荷的情况下，对称面上的结点没有水平方向的位移，即对于这些结点位移向量$u_i=0$。在承受剪切载荷的情况下，对称面上的结点没有垂直方向的位移，即对于这些结点位移分量$v_i=0$。

此外为防止发生刚体平移和刚体旋转，选取一个结点为固结点，即不发生水平方向和垂直方向的位移，它的位移分量$u_i=v_i=0$。同时选取与该点在同一水平位置上的某个结点没有垂直方向的位移，即$v_i=0$。

由于以上结点的某些位移分量为0，必须对方程组作相应的处理，也就是对刚度矩阵相对的对角元素为1，而相应行列的其它元素为0。

5. 列出线性代数方程组

$$[K]\{u\}=\{F\} \tag{3}$$

从中解出结点位移向量$\{u\}=\begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \vdots \\ u_{344} \\ v_{344} \end{Bmatrix}$，其中$u_i, v_i$为未知的位移分量。

6. 由此求得的结点位移分量$\{u\}$求出各单元e的应力分量。

$$\{\sigma\}=\begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix}=[S_i, S_j, S_m]\begin{Bmatrix} \delta_i \\ \delta_j \\ \delta_m \end{Bmatrix} \tag{4}$$

其中

$$[S_i]=\frac{E(1-\mu)}{2(1+\mu)(1-2\mu)\Delta}\begin{bmatrix} b_i & \frac{\mu}{1-\mu}c_i \\ \frac{\mu}{1-\mu}b_i & c_i \\ \frac{1-2\mu}{2(1-\mu)}c_i & \frac{1-2\mu}{2(1-\mu)}b_i \end{bmatrix},\quad (i,j,m)$$

$$\{\delta_i\}=\begin{Bmatrix} u_i \\ v_i \end{Bmatrix}\quad (i,j,m)$$

结点应力则用此结点周围的单元应力的算术平均值计算。

7. 由所求得的结点位移向量 u 代入对称面上结点所对应的方程中求出结点反力与外力及外力矩进行总体平衡校核。

(三)计算程序

用有限元素法归结为线性代数方程组的求解问题，一般可以采取直接法和迭代法两种方法求解。直接法中也有消去法和平方根法等多种。由于刚度矩阵 $[K]$ 具有对称正定性，我们采取了改进平方根法求解。该方法简述如下：

已知方程组 $KU=P$，如果 K 为对称正定方阵，则可作如下分解：

$$K=LD^{-1}L^T$$

其中

$$L=\begin{pmatrix} L_{11} & & & 0 \\ L_{21} & L_{22} & & \\ \vdots & \vdots & \ddots & \\ L_{n1} & L_{n2} & \cdots & L_{nn} \end{pmatrix},\quad D=\begin{pmatrix} L_{11} & & & \\ & L_{22} & & 0 \\ & & \ddots & \\ 0 & & & L_{nn} \end{pmatrix}$$

L 矩阵元素 l_{ij} 由下面递推公式给出：

$$l_{ij}=\begin{cases} K_{ij} & j=1 \\ K_{ij}-\sum_{t=1}^{j-1} L_{it}l_{jt}/l_{tt} & 1<j\le i \end{cases}$$

于是求解过程可归结为下列步骤：

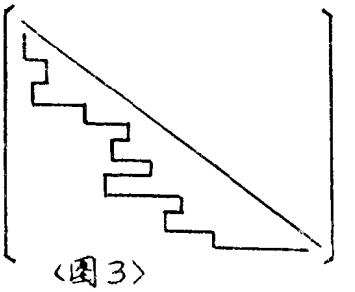
令 $D^{-1}L^TU=Y$，

由 $LY=P$ 计算 Y

由 $L^TU=DY$ 即可解出 U.

由于通常 $[K]$ 的阶数很高，给机器存储带来一定的困难，然而利用 $[K]$ 的稀疏性的特点，可以采取不等带宽的紧缩一维数组存放（如图3）。便存放下三角非零元素色络线以内的部分。不难看出，在平方根法整个过程中，$[K]$ 矩阵保持了下三角部分的带宽和形状。因此，紧缩的一维数组存放是可行的。

计算中对于已知位移为0的点在线性方程中所对应的行列上除对角元素1外其余的全部系数元0，而对应的载荷分量也元0．这样处理比较方便。



〈图3〉

解方程组求得的结点位移向量仍存在载荷列阵中。

1. 程序说明

符号意义.

NE 单元总数，约600左右.

NP 结点总数，约400左右.

NC 有约束的节点总数.

NM 刚度矩阵存放元素总数.

NF_1 有垂直载荷的结点数.

NF_2 有水平载荷的结点数.

II 单元结点紧缩编号数组.

如单元E的三个结点分别为 i、j、m，则II[E]的输入方式为 0000 $\underbrace{xxx}_{i}$ $\underbrace{xxx}_{j}$ $\underbrace{xxx}_{m}$ 0.

X,Y 分别表示节点的x,y坐标数组.

N_1 刚度矩阵Kii 在一维存放中的编号.

G_1,H_1 有垂直载荷的结点编号及相应的载荷值.

G_2,H_2 有水平载荷的结点编号及相应的载荷值.

CH 有约束的结点编号.

NN 刚度矩阵最宽行元素数－1.

KB 一维存放的刚度矩阵.

ME_1,ME_2 分别表示两种材料的弹性模量.

MN 弹性模量为ME_1的单元个数.

MU 泊松比.

F_1,F_2 外力载荷数组，解方程后是位移数组.

SI_1,SI_2,SI_3 单元应力数组.

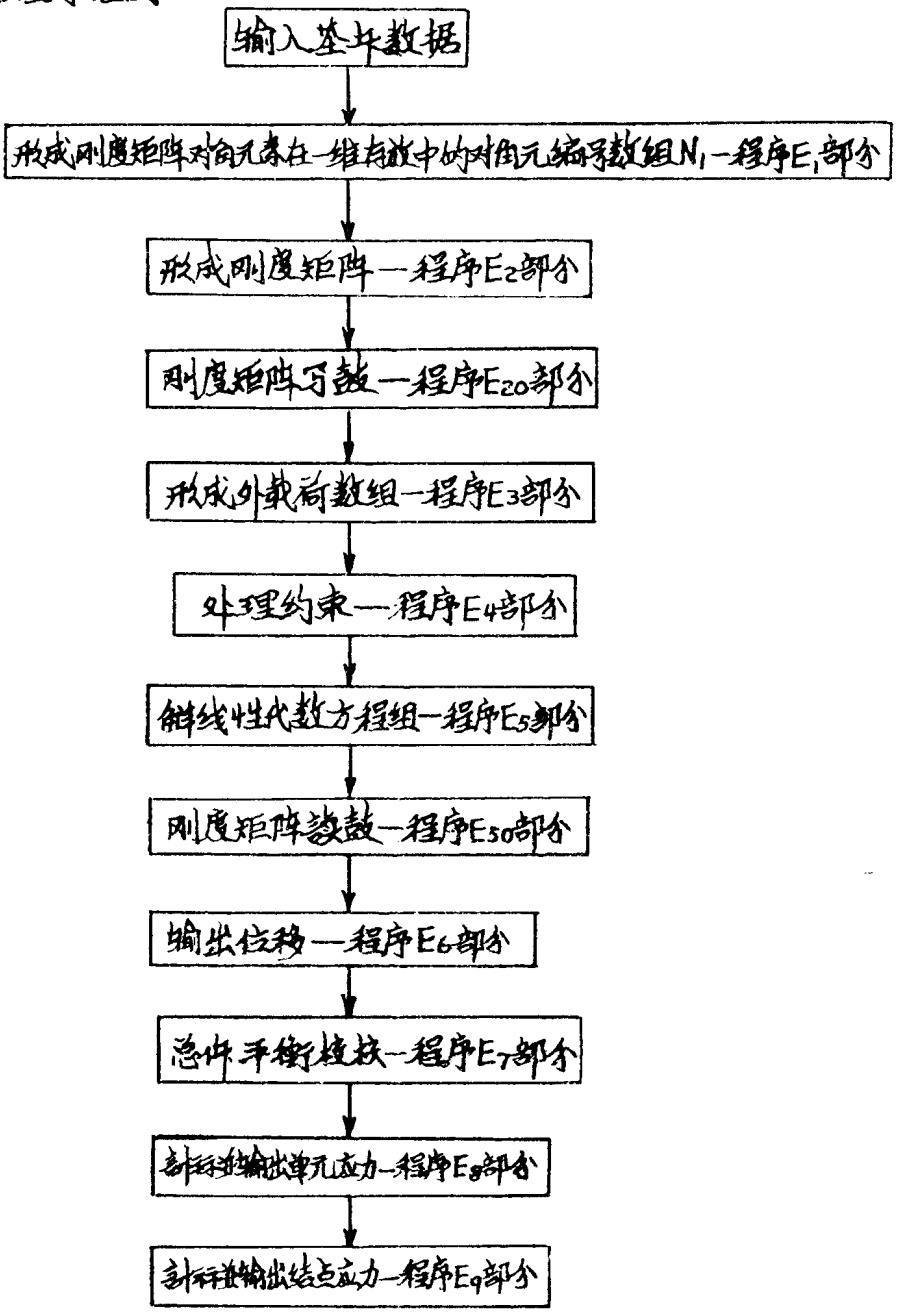
布尔量K,K_1以控制不同载荷情况的计算，K为false计算剪切载荷，K为true根据K_1的真假分别计算垂直水平载荷情况.

过程IJM：将输入的单元E的结点紧缩编号II[E]分解为I,J,M三个数.

过程BCS：形成单元E的单体刚度矩阵的基本数据.

过程PQ：进行节点P的约束处理。当Q=1时表示节点P的位移x为0，当Q=2时表示节点P的y位移为0.

2.程序框图



3.输入输出数据

输入数据

NE, NP, NC, NF_1, NF_2,

II,
X, Y,
G_1, H_1, G_2, H_2, CH,
MN, ME_1, ME_2.

输出数据

Q*

DISPIACE (位移)

P	UX	UY
(节点编号)	(x方向位移)	(y方向位移)

*Q=1,2,3 分别表示垂直、水平、剪切三种载荷的计标数据.

P=(节点编号)　　F=(反力)

**F=(反力合力)

*RF=(反力合力矩)

STRESS

E	CX	CY	TXY
(单元编号)	(σx)	(σy)	(τxy)
(节点编号)	(σx)	(σy)	(τxy)

4.程序 (编入开文时, 需作部分修改)

```
begin
    real NE, NP, NC, NS, NM, NF1, NF2, NN;
    *read(0, '10', NE, NP, NC, NF1, NF2);
        NS:=NP*2;
  begin
    real E, I, J, M, P, Q, S, T;
    array II(800)[1:NE],
        XY(500)[1:NP],
```

```
        N1(1000)[1:NS],
        G1, H1, (50)[1:NF1],
        G2, H2, (50)[1:NF2],
        CH(100)[1:NC];
    procedure IJM;
      begin
        real P;
            I:=*entire(II[E]*1000);
            P:=(II[E]*1000-I)*1000;
            J:=*entire(P);
            M:=*entire((P-J)*1000+0.5)
      end;
E1: N1[0]:=0; NN:=0;
    for P:=1 step 1 until NP do
    begin
            Q:=P;
            for E:=1 step 1 until NE do
      begin IJM;
            if(J=P∨M=P)∧I<Q then Q:=I
      end;
            Q:=P-Q;
            if Q>NN then NN:=Q;
            N1[2*P-1]:=N1[2*P-2]+2*Q+1;
            N1[2*P]:=N1[2*P-1]+2*Q+2
    end;
            NN:=2*NN+1;
```

```
            NM:=N1[NS];
            *print(0,'15X,3HNM=,I6',NM);
begin
  real SS, ME, ME1, ME2, MN, MU, A1, A2, A3,
       I1, I2, J1, J2, G, KI, KJ, KK;
  boolean K, K1;
  array KB(24000)[1:NM], B,C,II1[1:3], F1,F2(1000)[1:NS];
  procedure BCS;
  begin IJM;
        II1[1]:=I; II1[2]:=J; II1[3]:=M;
        B[1]:=Y[J]-Y[M];   C[1]:=X[M]-X[J];
        B[2]:=Y[M]-Y[I];   C[2]:=X[I]-X[M];
        B[3]:=Y[I]-Y[J];   C[3]:=X[J]-X[I];
        SS:=(B[2]*C[3]-B[3]*C[2])/2;
        ME:=if E>MN then ME2 else ME1.
  end;
  procedure PQ(P,Q);
  value P,Q;
  real  P,Q;
    begin
      real M,I,J,J1;
          M:=2*(P-1)+Q;
          J:=N1[M];
          J1:=N1[M-1];
      for I:=J1+1 step 1 until J-1 do
          KB[I]:=0;
          KB[J]:=1;
      for I:=M+1 step 1 until NS do
       if I-M<N1[i]-N1[i-1] then KB[N1[I]-I+M]:=0
    end;
```

```
            MU:=1.
            A1:=(1-2*MU)/(1-MU)/2;
            A2:=MU/(1-MU);
            A3:=(1-MU)/(4*(1+MU)*(1-2*MU));
E2:     *read(0,'10', MN, ME1, ME2);
        for P:=1 step 1 until NM do
            KB[P]:=0;
        for E:=1 step 1 until NE do
        begin BCS;
            SS:=A3*ME/*abs(SS);
        for S:=1, 2, 3 do
        begin I2:=II1[S];
        for T:=1 step 1 until S do
          begin J2:=II1[T];
                I1:=N1[2*I2-1]-2*(I2-J2);
                J1:=N1[2*I2]-2*(I2-J2)-1;
                KB[I1]:=KB[I1]+(B[S]*B[T]+A1*C[S]*
                                        C[T])*SS;
                KB[J1]:=KB[J1]+(A2*B[T]*C[S]+A1*
                                        B[S]*C[T])*SS;
                KB[J1+1]:=KB[J1+1]+(C[S]*C[T]+A1*
                                        B[S]*B[T])*SS;
          if I2>J2 then KB[I1+1]:=KB[I1+1]+(A2*B[S]*C[T]+A1
                                        *B[T]*C[S])*SS
          end
        end
        end;
E20:
  begin
      real C1, C2, I;
```

```
        array C[1:1000];
              C1:=C2:=0;
              LW1: for I:=C1+1 step 1 until C1+1000 do
              C[I-C1]:=KB[I];
              if C1<11001 then *write drum(2,C1,C)
                else begin *write drum(0,C2,C);
              C2:=C2+1000 end;
              if C1<22001 then
            begin C1:=C1+1000;
            goto LW1 end
      end
              K:=true;
E3:         for I:=1 step 1 until NS do
              F1[1]:=F2[1]:=0;
            for I:=1 step 1 until NF1 do
    begin   P:=G1[I];
              if K then F1[2*P]:=H1[I]
                else F1[2*P-1]:=-H[I]
    end;
            for I:=1 step 1 until NF2 do
      begin  P:=G2[I];
            if K then F2[2*P-1]:=H2[I]
                else F1[2*P]:=-H2[I]
      end;
E4:   for I:=1 step 1 until NC-1 do
    begin P:=CH[I];
            Q:=if K then 1 else 2;
            PQ(P,Q);
            F1[2*(P-1)+Q]:=0;
            if K then F2[2*(P-1)+Q]:=0
```

```
        end ;
            P:=CH[NC];
            PQ(P,1); PQ(P,2);
            F1[2*P-1]:=F1[2*P]:=0 ;
            if K then F2[2*P-1]:=F2[2*P]:=0;
            else begin P:=CH[NC-1];
                       PQ(P,1);
                       F1[2*P-1]:=0
                 end;
E5:         for I:=2 step 1 until NS do
        begin S:=N1[I];
            K1:=I-S+N1[I-1]+1;
            for J:=K1 step 1 until I do
          begin I1:=S-I+J;  T:=N1[J];
              KJ:=J-T+N1[J-1]+1;
              KK:= if KJ>KI then KJ else KI;
            for J1:=KK step 1 until J-1 do
            KB[I1]:=KB[I1]-KB[S-I+J1]*KB[T-J+J1]/KB[N1[J1]]
          end
        end;
          for I:=1 step 1 until NS do
          begin S:=N1[I];
              for J:=I-S+N1[I-1]+1 step 1 until I-1 do
            begin F1[I]:=F1[I]-KB[S-I+J]*F1[J];
              if K then F2[I]:=F2[I]-KB[S-I+J]*F2[J]
            end;
                  F1[I]:=F1[I]/KB[S];
              if K then F2[I]:=F2[I]/KB[S]
          end ;
```

```
            for I:=NS step -1 until 1 do
        begin   S:=N1[I];
                I1:=0;  J1:=0;
            for J:=I+1 step 1 until NS do
          begin T:=N1[J];
                if I>J-T+N1[J-1] then
            begin I1:=I1-KB[T-J+I]*F1[J];
                if K then J1:=J1-KB[T-J+I]*F2[J]
            end
          end;
                F1[I]:=F1[I]+I1/KB[S];
                if K then F2[I]:=F2[I]+J1/KB[S]
      end ;
Eso:
  begin
      real    C1, C2, I;
      array   C[1:1000];
              C1:=C2:=0;
    LW2:      if C1<11001 then *read drum(2, C1, C)
              else begin *read drum(0, C2, C);
                         C2:=C2+1000
                   end;
          for I:=C1+1 step 1 until C1+1000 do
              KB[I]:=C[I-C1];
              if C1<22001 then
                   begin C1:=C1+1000;
                         goto LW2
                   end
  end ;
```