基于 99 行程序的拓扑优化学习报告

（一）背景和前言

随着汽车工业的飞速发展以及日益突出的能源问题，汽车工业面临的挑战以及竞争环境也越来越激烈，对汽车产品提出了降低其制造成本及燃油经济性的新要求。在提高汽车安全性、减少汽车排放和解决能源消耗的背景下，提出了汽车轻量化技术。实现汽车轻量化的途径包括三个方面：结构优化技术、新型材料和先进性制造工艺。其中，我们所讨论的是结构优化技术，其中结构优化设计分为三个层次：尺寸优化（Size Optimization）、形状优化（Shape Optimization）和拓扑优化（Topology Optimization）。本文我们基于 99 行 matlab 程序初步学习拓扑优化技术中的理论和优化方法。

拓扑优化技术指的是在给定的设计空间内寻求最佳的材料布局，同时在满足平衡方程、物理关系、几何关系和边界约束条件下使得结构达到某种性能最优的应用技术。

拓扑优化的理论研究最早可以追溯到 Michel 提出的桁架理论，连续体结构的拓扑优化由于描述和数值计算得困难，发展一直相对缓慢，直到 Bendsoe 和 Kikuchi 在 1988 年提出的均匀化方法之后才得到迅速的发展，其基本思想是在组成拓扑结构中引入微结构，通过微结构的几何参数作为设计变量，通过微结构的增加和删减实现结构的拓扑形状的改变，实现拓扑优化和尺寸优化的统一。

在微结构的基础上，我们介绍变密度法的应用，变密度法是在均匀化方法的基础上产生的，把材料引入微结构代之以密度在 0~1 之间变化的假想材料，把密度作为设计变量，从而实现材料的删减，因其模型简单、计算变量相对较少成为目前广泛采用的方法。根据不同的插值模式，变密度法又有不同的插值模型：SIMP 法（Solid Isotropic Material with Penalization）、Hashin-Shtrikman 法，以及 RAMP 法（Rational Approximation of Material Properties）。

（二）拓扑优化问题的描述

（1）材料插值模型

变密度理论的材料插值模型通过引入中间密度单元，将离散型问题转化成连续型优化问题，而实际上，中间密度单元是无法存在和制造，因此要尽量避免中间密度单元的产生，减少中间密度单元的数目，此时就需要对设计变量中出现的中间密度值进行惩罚。

目前材料插值模型主要分为两类，SIMP 法和 RAMP 法，基于 SIMP 格式的材料插值模型为：

$$E(x_i)=E_{min}+(x_i)^p(E_0-E_{min}),x_i\in[0,1] \tag{1}$$

其中：

$E(x_i)$——插值以后的弹性模量

E_0——实体部分材料的弹性模量

E_{min}——孔洞部分材料的弹性模量

x_i——单元相对密度，取值为 1 时表示有材料，为 0 时表示无材料即孔洞

p——惩罚因子

在本文中，在本篇文章中，讨论的 SIMP 的表达为：

$$E(x_{i,j})=(x_{i,j})^pE_0 \tag{2}$$

其中：$x_{i,j}$ 为第 i 个子域内第 j 个单元的相对密度。

（2）数学模型

根据在体积或质量约束下求最小柔度，即最大刚度来建立优化模型，基于变密度理论的 SIMP 法的周期性拓扑优化问题的数学模型可以表达为：

$$find \quad X=(x_{1,1},x_{1,2},x_{1,3},\cdots,x_{i,j})^T\in R$$

$$i=1,2,\cdots,m \qquad j=1,2,\cdots,n$$

$$\min \quad C(X)=FU=U^TKU=\sum_{i=1}^{m}\sum_{j=1}^{n}u_{i,j}^Tk_{i,j}u_{i,j}=\sum_{i=1}^{m}\sum_{j=1}^{n}(x_{i,j})^pu_{i,j}^Tk_0u_{i,j}$$

$$s.t \quad KU=F$$

$$V=fV_0=\sum_{i=1}^{m}\sum_{j=1}^{n}x_{i,j}\cdot v_{i,j}$$

$$0<x_{\min}\le x_{i,j}\le x_{\max}\le 1$$

其中：

X——单元相对密度矢量

C——结构的柔度

F——载荷矢量

U——位移矢量

K ——结构刚度矩阵

$u_{i,j}$ ——单元位移矢量

$k_{i,j}$ ——单元刚度矩阵

k_0 ——初始段元刚度矩阵

$v_{i,j}$ ——单元体积

V ——优化后的体积

f ——保留的体积分数

V_0 ——初始体积

$x_{\min}$ ——设计变量的取值下限

$x_{\max}$ ——设计变量的取值上限

n ——子域内单元的个数

（三）程序解读

99 行程序主要包括主函数、OC 优化准则、网格过滤、有限元分析、灵敏度分析 5 个部分，其流程图为：

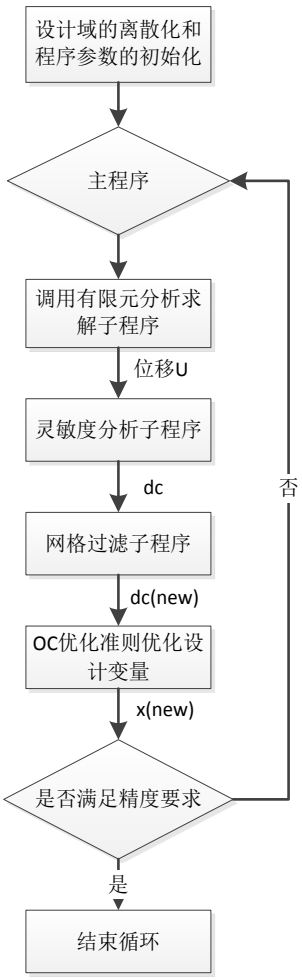


图 1. 99 行程序的流程图

具体程序分析：

总体程序为：

%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 %%%

%%%% CODE MODIFIED FOR INCREASED SPEED, September 2002, BY OLE SIGMUND %%%

function top(nelx,nely,volfrac,penal,rmin);

nelx=80;

nely=20;

volfrac=0.4;

penal=3;

rmin=2;

% INITIALIZE

```
x(1:nely,1:nelx) = volfrac;
loop = 0;
change = 1.;
% START ITERATION
while change > 0.01
  loop = loop + 1;
  xold = x;
% FE-ANALYSIS
  [U]=FE(nelx,nely,x,penal);
% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
  [KE] = lk;
  c = 0.;
  for ely = 1:nely
     for elx = 1:nelx
       n1 = (nely+1)*(elx-1)+ely;
       n2 = (nely+1)* elx    +ely;
       Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2; 2*n1+1;2*n1+2],1);
       c = c + x(ely,elx)^penal*Ue'*KE*Ue;
       dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
     end
  end
% FILTERING OF SENSITIVITIES
  [dc]    = check(nelx,nely,rmin,x,dc);
% DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
  [x]     = OC(nelx,nely,x,volfrac,dc);
% PRINT RESULTS
  change = max(max(abs(x-xold)));
  disp([' It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',c) ...
         ' Vol.: ' sprintf('%6.3f',sum(sum(x))/(nelx*nely)) ...
         ' ch.: ' sprintf('%6.3f',change )])
% PLOT DENSITIES
```

```
  colormap(gray); imagesc(-x); axis equal; axis tight; axis off;pause(1e-6);
end
%%%%%%%%%% OPTIMALITY CRITERIA UPDATE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [xnew]=OC(nelx,nely,x,volfrac,dc)
l1 = 0; l2 = 100000; move = 0.2;
while (l2-l1 > 1e-4)
  lmid = 0.5*(l2+l1);
  xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
  if sum(sum(xnew)) - volfrac*nelx*nely > 0;
     l1 = lmid;
  else
     l2 = lmid;
  end
end
%%%%%%%%%% MESH-INDEPENDENCY FILTER %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [dcn]=check(nelx,nely,rmin,x,dc)
dcn=zeros(nely,nelx);
for i = 1:nelx
  for j = 1:nely
     sum=0.0;
     for k = max(i-floor(rmin),1):min(i+floor(rmin),nelx)
       for l = max(j-floor(rmin),1):min(j+floor(rmin),nely)
         fac = rmin-sqrt((i-k)^2+(j-l)^2);
         sum = sum+max(0,fac);
         dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);
       end
     end
```

```
    dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
  end
end
%%%%%%%%%%                                                          FE-
ANALYSIS   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
function [U]=FE(nelx,nely,x,penal)
[KE] = lk;
K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
F = sparse(2*(nely+1)*(nelx+1),1); U = zeros(2*(nely+1)*(nelx+1),1);
for elx = 1:nelx
  for ely = 1:nely
    n1 = (nely+1)*(elx-1)+ely;
    n2 = (nely+1)* elx    +ely;
    edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
    K(edof,edof) = K(edof,edof) + x(ely,elx)^penal*KE;
  end
end
% DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
F(2*(nelx/2+1)*(nely+1),1) = 1;
fixeddofs   = [2*(nely/2+1),2*nelx*(nely+1)+2*(nely/2+1)];
alldofs     = [1:2*(nely+1)*(nelx+1)];
freedofs    = setdiff(alldofs,fixeddofs);
% SOLVING
U(freedofs, ：  ）= K(freedofs,freedofs) \ F(freedofs,：  ）；
U(fixeddofs,：  ）= 0;
%%%%%%%%%%                   ELEMENT                   STIFFNESS
MATRIX %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function [KE]=lk
E = 1.;
```

```
nu = 0.3;
k=[ 1/2-nu/6    1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
   -1/4+nu/12 -1/8-nu/8  nu/6         1/8-3*nu/8];
KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
                  k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
                  k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
                  k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
                  k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
                  k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
                  k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
                  k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
%
```

根据程序的运行的先后顺序，依次详细分析各行程序：

```
function top(nelx,nely,volfrac,penal,rmin);
```

```
nelx=80;          x 轴方向的单元数目
nely=20;          y 轴方向的单元数目
volfrac=0.4;      体积比，这个参数是用来确定保留的材料量
penal=3;          材料插值的惩罚因子，由于变密度理论的材料插值模型引入中间密度单元，所以将离散型问题转化成连续型优化问题，而在实际过程中，是无法存在和制造的，所以要尽量避免中间密度单元的产生，减少中间密度单元的数目，所以需要对中间密度值进行惩罚，penal 是惩罚因子。
rmin=2;       灵敏度过滤的过滤半径
```

关于体积比、惩罚因子和过滤半径不同引起结果的不同，我们将在之后的讨论中进行。

以上是初始化参数的程序段

```
x(1:nely,1:nelx) = volfrac;          x 是设计变量，为某一个单元的相对密度，在此
```

给出整体网格编排的顺序表：

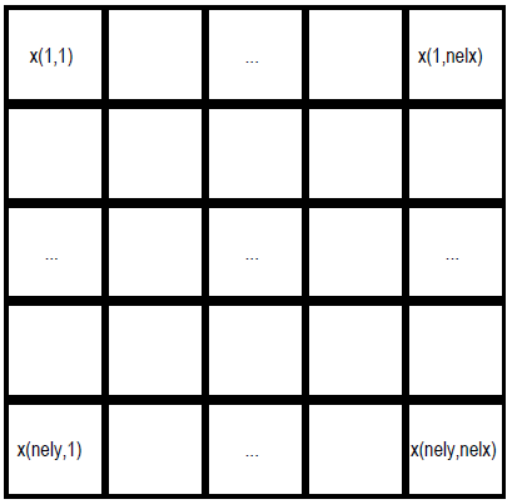


图 2. 整体网格编排顺序表

故对于整个网格，在垂直方向上，有 nely 个单元，在水平方向上，有 nelx 个单元。

loop = 0;　　存放迭代次数的变量

change = 1.;　　每次迭代之后，目标函数的改变值，可以用来判断何时收敛

while change > 0.01　　当两次连续目标函数迭代的差小于 0.01 时，结束迭代

loop = loop + 1;迭代次数加 1

xold = x;　　将前一次的设计变量赋值给 xold

[U]=FE(nelx,nely,x,penal);　　进行有限元分析，需要输入的参数为 nelx、nely、x、penal

进入有限元分析子程序

function [U]=FE(nelx,nely,x,penal)

[KE] = lk;　　单元刚度矩阵分析

进入单元刚度矩阵子程序

```
function [KE]=lk
E = 1.;
k=[ 1/2-nu/6    1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
   -1/4+nu/12 -1/8-nu/8  nu/6         1/8-3*nu/8];
KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
                  k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
                  k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
                  k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
                  k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
                  k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
                  k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
                  k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
```

此处单独解释有限元理论，通过平面问题的 4 节点矩形单元描述

（1） 单元的几何和节点描述

平面 4 节点矩形单元如图 3 所示，单元的节点位移共有 8 个自由度（DOF），节点的编号为 1、2、3、4，各自的位置坐标为 $(x_i, y_i)\ i=1,2,3,4$，各个节点的位移（分别沿 x 和 y 方向），为 $(u_i, v_i), i=1,2,3,4$。

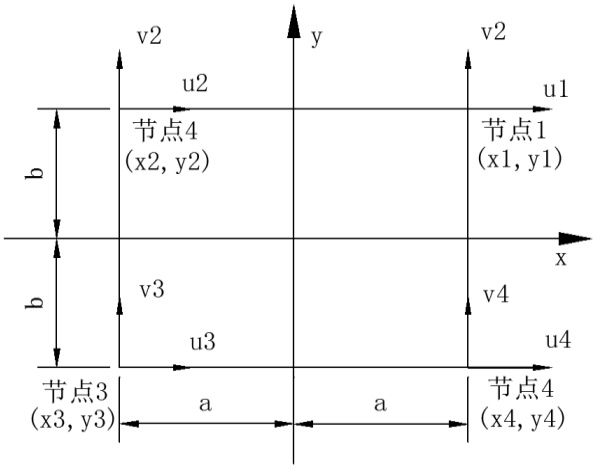


图 3.平面 4 节点矩形单元

将所有节点上的位移组成一个列矩阵，记作 $U^{(e)}$，将所有节点上的力组成一个列矩阵，记作 $F^{(e)}$，则：

$$U^{(e)}=\begin{bmatrix}u_1 & v_1 & u_2 & v_2 & u_3 & v_3 & u_4 & v_4\end{bmatrix}^T \tag{3}$$

$$F^{(e)}=\begin{bmatrix}F_{x1} & F_{y1} & F_{x2} & F_{y2} & F_{x3} & F_{y3} & F_{x4} & F_{y4}\end{bmatrix}^T \tag{4}$$

（2） 单元场的位移描述

从图 3 可以看出，节点条件共有 8 个，x 方向上 4 个 (u_1,u_2,u_3,u_4)，y 方向上 4 个 (v_1,v_2,v_3,v_4)，故在 x 和 y 方向的位移场可以各有 4 个待定系数，取以下多项式作为位移场模式：

$$\begin{aligned}u(x,y)&=a_0+a_1x+a_2y+a_3xy\\ v(x,y)&=b_0+b_1x+b_2y+b_3xy\end{aligned} \tag{5}$$

要考虑到的特点是当 x 或 y 不变时，沿 y 或 x 方向位移函数是线性变化的，因此不选 x^2 和 y^2 项。由于节点条件，在 $x=x_i, y=y_i$ 处，有：

$$\left.\begin{aligned}u(x_i,y_i)&=u_i\\ v(x_i,y_i)&=v_i\end{aligned}\right\} i=1,2,3,4 \tag{6}$$

将四个节点的坐标 $(x_1,y_1),(x_2,y_2),(x_3,y_3),(x_4,y_4)$ 代入（5）方程中，综合（6）

方程得：

$$\begin{cases} a_0 = \dfrac{u_1+u_2+u_3+u_4}{4} \\ a_1 = \dfrac{u_1+u_2-u_3-u_4}{4a} \\ a_2 = \dfrac{u_1-u_2-u_3+u_4}{4b} \\ a_3 = \dfrac{u_1-u_2+u_3-u_4}{4ab} \end{cases} \qquad \begin{cases} b_0 = \dfrac{v_1+v_2+v_3+v_4}{4} \\ b_1 = \dfrac{v_1+v_2-v_3-v_4}{4a} \\ b_2 = \dfrac{v_1-v_2-v_3+v_4}{4b} \\ b_3 = \dfrac{v_1-v_2+v_3-v_4}{4ab} \end{cases}$$

经过整理，令（5）式的格式为：

$$\left.\begin{aligned} u(x,y) &= N_1(x,y)u_1 + N_2(x,y)u_2 + N_3(x,y)u_3 + N_4(x,y)u_4 \\ v(x,y) &= N_1(x,y)v_1 + N_2(x,y)v_2 + N_3(x,y)v_3 + N_4(x,y)v_4 \end{aligned}\right\} \tag{7}$$

可以整理得：

$$\left.\begin{aligned} N_1(x,y) &= \frac{1}{4}(1+\frac{x}{a})(1+\frac{y}{b}) \\ N_2(x,y) &= \frac{1}{4}(1-\frac{x}{a})(1+\frac{y}{b}) \\ N_3(x,y) &= \frac{1}{4}(1-\frac{x}{a})(1-\frac{y}{b}) \\ N_4(x,y) &= \frac{1}{4}(1+\frac{x}{a})(1-\frac{y}{b}) \end{aligned}\right\} \tag{8}$$

将（7）式写成矩阵形式：

$$\underset{(2\times1)}{u}(x,y) = \begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix}$$

$$= \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{bmatrix} = \underset{(2\times8)}{N} \cdot \underset{(8\times1)}{U^{(e)}} \tag{9}$$

其中 $N(x,y)$ 为该单元的形状函数矩阵。

（3） 单元应变场的描述

由弹性力学中节点的几何方程：

$$\varepsilon_{xx} = \frac{\partial u}{\partial x}, \quad \varepsilon_{yy} = \frac{\partial v}{\partial y}, \quad \gamma_{xy} = \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}$$

得到单元应变的表达：

$$\underset{(3\times1)}{\varepsilon}(x,y)=\begin{bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ \gamma_{xy}\end{bmatrix}=\underset{(3\times2)}{[\partial]}\underset{(2\times1)}{u}=\underset{(3\times2)}{[\partial]}\underset{(2\times8)}{N}\underset{(8\times1)}{U^{(e)}}=\underset{(3\times8)}{B}\underset{(8\times1)}{U^{(e)}} \tag{10}$$

其中：

$$\underset{(3\times2)}{[\partial]}=\begin{bmatrix}\dfrac{\partial}{\partial x} & 0\\ 0 & \dfrac{\partial}{\partial y}\\ \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial x}\end{bmatrix}$$

单元的形状函数矩阵：

$$\underset{(2\times8)}{N}=\begin{bmatrix}N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0\\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4\end{bmatrix}$$

定义单元的几何矩阵为：

$$\begin{aligned}\underset{(3\times8)}{B}(x,y)&=\underset{(3\times2)}{[\partial]}\underset{(2\times8)}{N}=\begin{bmatrix}\dfrac{\partial}{\partial x} & 0\\ 0 & \dfrac{\partial}{\partial y}\\ \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial x}\end{bmatrix}\begin{bmatrix}N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0\\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4\end{bmatrix}\\ &=\begin{bmatrix}\underset{(3\times2)}{B_1} & \underset{(3\times2)}{B_2} & \underset{(3\times2)}{B_3} & \underset{(3\times2)}{B_4}\end{bmatrix}\end{aligned} \tag{11}$$

式（11）中的子矩阵 $\boldsymbol{B}_i$ 为：

$$\underset{(3\times2)}{B_i}=\begin{bmatrix}\dfrac{\partial N_i}{\partial x} & 0\\ 0 & \dfrac{\partial N_i}{\partial y}\\ \dfrac{\partial N_i}{\partial y} & \dfrac{\partial N_i}{\partial x}\end{bmatrix},i=1,2,3,4 \tag{12}$$

（4） 单元应力场的表达

由弹性力学中平面问题的物理方程：

$$\left.\begin{aligned}\varepsilon_{xx}&=\frac{1}{E}(\sigma_x-\mu\sigma_y)\\ \varepsilon_{yy}&=\frac{1}{E}(\sigma_y-\mu\sigma_x)\\ \gamma_{xy}&=\frac{2(1+\mu)}{E}\tau_{xy}\end{aligned}\right\} \tag{13}$$

将（13）式换成应力和应变的关系表达为：

$$\left.\begin{aligned}\sigma_x&=\frac{E}{1-\mu^2}(\varepsilon_{xx}+\mu\varepsilon_{yy})\\ \sigma_y&=\frac{E}{1-\mu^2}(\varepsilon_{yy}+\mu\varepsilon_{xx})\\ \tau_{xy}&=\frac{E}{2(1+\mu)}\gamma_{xy}\end{aligned}\right\} \tag{14}$$

故有：

$$\underset{(3\times1)}{\sigma}=\begin{bmatrix}\sigma_x\\ \sigma_y\\ \tau_{xy}\end{bmatrix}=\frac{E}{1-\mu^2}\begin{bmatrix}1&\mu&0\\ \mu&1&0\\ 0&0&\dfrac{1-\mu}{2}\end{bmatrix}\begin{bmatrix}\varepsilon_{xx}\\ \varepsilon_{yy}\\ \gamma_{xy}\end{bmatrix}=\underset{(3\times3)}{D}\underset{(3\times1)}{\varepsilon} \tag{15}$$

其中定义 $\underset{(3\times3)}{D}$ 为弹性矩阵：

$$\underset{(3\times3)}{D}=\frac{E}{1-\mu^2}\begin{bmatrix}1&\mu&0\\ \mu&1&0\\ 0&0&\dfrac{1-\mu}{2}\end{bmatrix} \tag{16}$$

把（10）式代入到（15）式中得：

$$\underset{(3\times1)}{\sigma}=\underset{(3\times3)}{D}\underset{(3\times1)}{\varepsilon}=\underset{(3\times3)}{D}\underset{(3\times8)}{B}\underset{(8\times1)}{U^{(e)}}=\underset{(3\times8)}{S}\underset{8\times1}{U^{(e)}} \tag{17}$$

其中应力函数矩阵为：

$$\underset{(3\times8)}{S}=\underset{(3\times3)}{D}\underset{(3\times8)}{B} \tag{18}$$

（5） 单元势能的表达

至此，已经通过（9）（10）（17）式将单元的三个基本变量 σ,ε,u 用节点位移矩阵 $U^{(e)}$ 来进行表达，将这三个式代入到单元的势能表达式中有：

$$\prod{}^{e}=\frac{1}{2}U^{(e)T}K^{(e)}U^{(e)}-F^{(e)T}U^{(e)} \tag{19}$$

其中 $K^{(e)}$ 是四节点矩形单元的刚度矩阵：

$$\underset{(8\times8)}{K^{(e)}}=\int_{A^{(e)}}\underset{(8\times3)}{B^{T}}\underset{(3\times3)}{D}\underset{(3\times8)}{B}dA\cdot t=\begin{bmatrix}k_{11}&&&\\k_{21}&k_{22}&sym&\\k_{31}&k_{32}&k_{33}&\\k_{41}&k_{42}&k_{43}&k_{44}\end{bmatrix}\tag{20}$$

其中t为平面单元的厚度，按照（11）式，可以对（20）式中的各个部分进行分块，各个子块矩阵为：

$$\underset{(2\times2)}{k_{rs}}=\int_{A^{(e)}}\underset{(2\times3)}{B_r^{T}}\underset{(3\times3)}{D}\underset{(3\times2)}{B}\cdot t\cdot dxdy\ ,\quad r,s=1,2,3,4\tag{21}$$

已经求得了几何矩阵 $\underset{(3\times2)}{B}$ 和弹性矩阵 $\underset{(3\times3)}{D}$，故可以对每一个刚度分量进行积分求解，其中的积分过程省略，最终可以得到刚度矩阵：

$$\underset{(8\times8)}{K^{(e)}}=\frac{Et}{ab(1-\mu^2)}\cdot\begin{bmatrix}k_{11}&k_{12}&k_{13}&k_{14}&k_{15}&k_{16}&k_{17}&k_{18}\\&k_{22}&k_{23}&k_{24}&k_{25}&k_{26}&k_{27}&k_{28}\\&&k_{33}&k_{34}&k_{35}&k_{36}&k_{37}&k_{38}\\&&&k_{44}&k_{45}&k_{46}&k_{47}&k_{48}\\&&&&k_{55}&k_{56}&k_{57}&k_{58}\\&&&&&k_{66}&k_{67}&k_{68}\\&&&&&&k_{77}&k_{78}\\&&&&&&&k_{88}\end{bmatrix}\tag{22}$$

其中：

$$k_{11}=k_{33}=k_{55}=k_{77}=\frac{1}{3}(b^2+\frac{1-\mu}{2}a^2)$$

$$k_{22}=k_{44}=k_{66}=k_{88}=\frac{1}{3}(a^2+\frac{1-\mu}{2}b^2)$$

$$k_{23}=k_{45}=k_{18}=k_{67}=\frac{ab}{8}(1-3\mu)$$

$$k_{14}=k_{27}=k_{36}=k_{58}=-\frac{ab}{8}(1-3\mu)$$

$$k_{12}=k_{56}=k_{47}=k_{38}=\frac{ab}{8}(1+\mu)$$

$$k_{16}=k_{25}=k_{34}=k_{78}=-\frac{ab}{8}(1+\mu)$$

$$k_{15}=k_{37}=-\frac{1}{6}(b^2+\frac{1-\mu}{2}a^2)$$

$$k_{26}=k_{48}=-\frac{1}{6}(a^2+\frac{1-\mu}{2}b^2)$$

$$k_{28}=k_{46}=\frac{1}{3}(-a^2+\frac{1-\mu}{4}b^2)$$

$$k_{13}=k_{57}=\frac{1}{3}(-b^2+\frac{1-\mu}{4}a^2)$$

$$k_{42}=k_{68}=\frac{1}{6}\left[a^2-(1-\mu)b^2\right]$$

$$k_{53}=k_{17}=\frac{1}{6}\left[b^2-(1-\mu)a^2\right]$$

对于程序中的问题，给定 $a=b=0.5$，则：

$$k_{11}=k_{33}=k_{55}=k_{77}=k_{22}=k_{44}=k_{66}=k_{88}=\frac{1}{2}-\frac{\mu}{6}=k_1$$

$$k_{12}=k_{56}=k_{47}=k_{38}=\frac{1}{8}(1+\mu)=k_2$$

$$k_{13}=k_{57}=k_{28}=k_{46}=-\frac{1}{4}-\frac{\mu}{12}=k_3$$

$$k_{14}=k_{27}=k_{36}=k_{58}=-\frac{1}{8}+\frac{3\mu}{8}=k_4$$

$$k_{15}=k_{37}=k_{26}=k_{48}=-\frac{1}{4}+\frac{\mu}{12}=k_5$$

$$k_{16}=k_{25}=k_{34}=k_{78}=-\frac{1}{8}-\frac{\mu}{8}=k_6$$

$$k_{17}=k_{53}=k_{42}=k_{68}=\frac{\mu}{6}=k_7$$

$$k_{23}=k_{45}=k_{18}=k_{67}=\frac{1}{8}-\frac{3\mu}{8}=k_8$$

以上定义的 $k_1,k_2,k_3,k_4,k_5,k_6,k_7,k_8$ 即为程序中定义的刚度矩阵。

回到程序中，走完刚度矩阵的定义，回到有限元分析子程序中：

```
K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
```

此处开始涉及到单元内所有节点在 x 和 y 两个方向上的计算分析，综合图 2 和图 3，观察所有节点的节点定义分析：

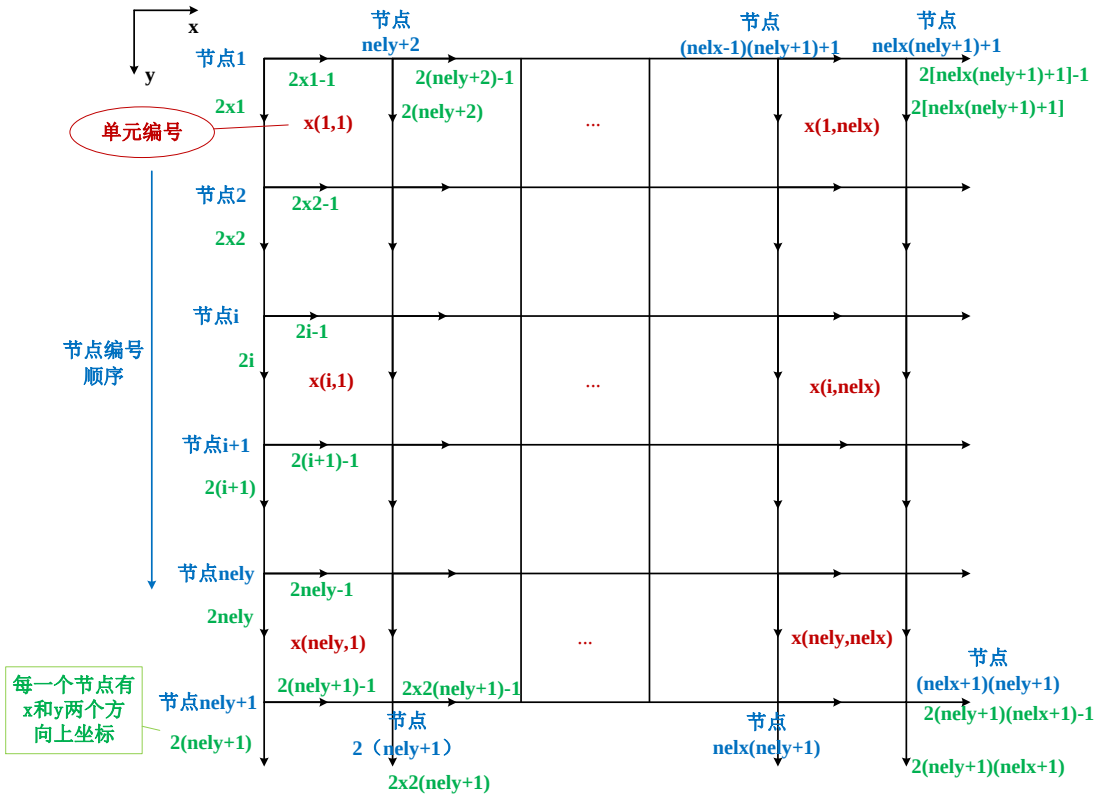


图 4.节点定义分析

在所绘制的图中：

红色区域是单元编号，从 $x(1,1), x(1,2)...x(nely,1)...x(nely,nelx)$；

蓝色的是节点的定义和顺序，从节点 1，节点 2…节点 $(nely+1)(nelx+1)$；

绿色的是每一个节点上 x 和 y 方向上的坐标，从 $2\times1-1$， 2×1，到 $2\times(nely+1)(nelx+1)-1$， $2\times(nely+1)(nelx+1)$。

所以定义刚度的时候，考虑到每一个坐标上的应力会对其他任何一个位置上的坐标产生相应应变，所以这一句定义刚度矩阵的行列数都为 $2(nely+1)(nelx+1)$ 的稀疏矩阵。

F = sparse(2*(nely+1)*(nelx+1),1); U = zeros(2*(nely+1)*(nelx+1),1);

对于每一个节点上的位移只有一个，所以定义位移矩阵为行为 $2(nely+1)(nelx+1)$，列为 1 的零矩阵。

由于有：$F=KU$ 所以分析行列数，可知力矩阵的行为 $2(nely+1)(nelx+1)$，列为 1，同样定义一个稀疏矩阵。

```
for elx = 1:nelx
  for ely = 1:nely
     n1 = (nely+1)*(elx-1)+ely;
     n2 = (nely+1)* elx    +ely;
     edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
```

```
    K(edof,edof) = K(edof,edof) + x(ely,elx)^penal*KE;
  end
end
```

解释这一段循环程序，定义*ely*和*elx*为单元编号的*x*和*y*坐标，即图 4 中的红色坐标。

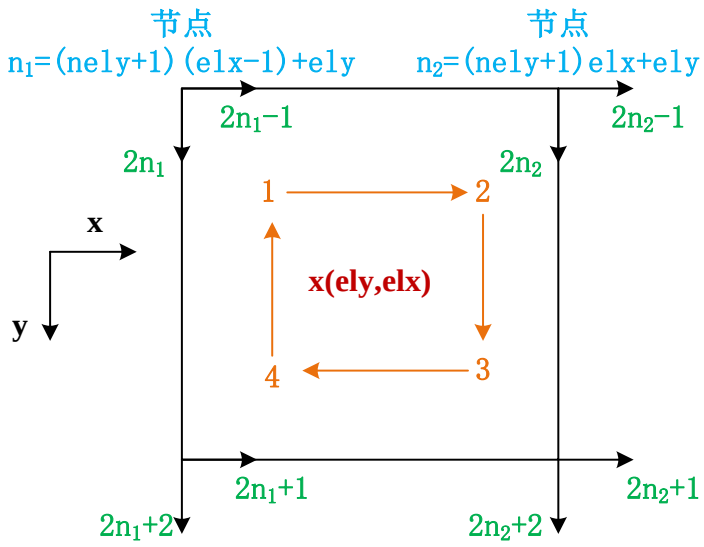


图 5.任意单元附近的节点信息

蓝色坐标即为定义的节点编号，在程序中为了避免繁琐，在一个单元上只定义两个节点：

$$n_1 = (nely+1)(elx-1)+ely$$
$$n_2 = (nely+1)elx+ely$$

然后定义一个单元上四个节点 8 个坐标，其中的坐标顺序如图 5 中橙色的 1,2,3,4 和箭头所示。

故定义一个单元上的坐标为：

```
edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
```

接下来，将单元刚度矩阵组装成总体刚度矩阵：

```
K(edof,edof) = K(edof,edof) + x(ely,elx)^penal*KE;
```

其中是根据材料密度模型中的：

$$k_{i,j} = (x_{i,j})^p k_0 \qquad (23)$$

通过循环程序组装成总体刚度矩阵。程序循环结束之后：

```
F(2*(nelx/2+1)*(nely+1),1) = 1;
```

此处我们通过这个例子来讲解程序中给定坐标的位置确定和给定位置坐

标的写法：

力矩阵 F(2*(nelx/2+1)*(nely+1),1)，是一个列矩阵，很明显是确定y方向上的加载力，则可知其节点数为(nelx/2+1)*(nely+1)，对于节点数，每一列有(nely+1)个，可以参考图 4 的表示，则 nelx/2+1 表示列数，显然对于一个有 nelx 列的单元，nelx/2+1 表示最中间的那一列节点（在此处，我们一般给定 nelx 为偶数）。

fixeddofs = [2*(nely/2+1),2*nelx*(nely+1)+2*(nely/2+1)];

此处为定义固定节点的坐标数，很容易知道固定的都是y轴上的坐标，nely/2+1 为第一列节点中间位置的节点数，2*(nely/2+1)为该节点y方向上的坐标；对于坐标为 2*nelx*(nely+1)+2*(nely/2+1)，我们可以知道，它是节点为 nelx*(nely+1)+ (nely/2+1)在y方向上的约束。

此处要注意的是，在 matlab 中括号的定义是不同的，中括号用于形成一个向量或矩阵，小括号用于表达一般的算术预算、还用于表达下标等。
定义载荷是指下标为$(2(nelx/2+1)(nely+1),1)$的力，是用小括号定义的，它是指下标，而在固定节点中，是用中括号给定的，该程序中定义固定坐标矩阵是一个1×2的矩阵，其中包含固定坐标的的两个坐标数。

在此，我们给出固定节点和施加载荷的位置示意图：

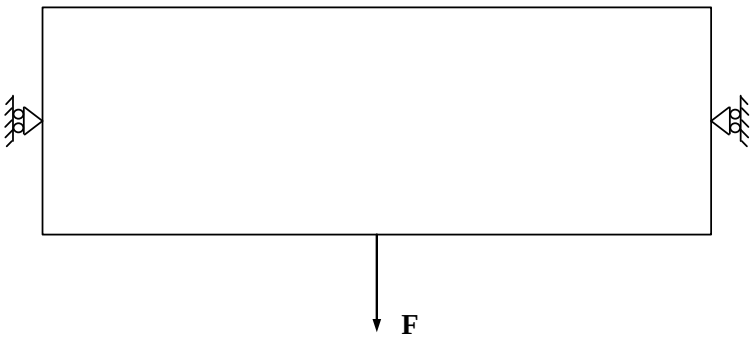


图 6.边界约束情况

alldofs=[1:2*(nely+1)*(nelx+1)];

定义所有节点的坐标值是从 1 到 2*(nely+1)*(nelx+1)。用根据之前的理解，中括号定义的是一个$1\times 2(nely+1)(nelx+1)$的矩阵。

freedofs=setdiff(alldofs,fixeddofs);

其中利用了 matlab 中的 setdiff 函数，它的作用是返回在 alldofs 中有，但是在 fixeddofs 中没有的量，相当于集合论中的$C=A-B$，即C ~~setdiff~~ (A,B)。此处得到除固定节点坐标外的其他坐标，把它们设为自由坐标。

U(freedofs,:） = K(freedofs,freedofs) \ F(freedofs,:）

之前已经组装了总体刚度矩阵 K，并且在稀疏力矩阵中确定了施加载荷的大小和方向，此处通过公式$U=\frac{F^{(e)}}{K}$计算自由坐标上的位移值，从而得到位移场。

U(fixeddofs,:）= 0;

同时定义固定节点坐标上的位移为 0，固定的坐标就是在侧边中点y方向上的位移。

至此，已经解释完有限元分析的程序段。

回到主程序中：

[KE] = lk;

定义单元刚度矩阵，不再重复。

c = 0.;

此处的 c 是存放目标函数为柔度的变量，我们的优化目标是整个结构的刚度最大，也就是柔度最小。理解为在一定载荷下，优化结构，使结构的变形最小。

接下来：

```
for ely = 1:nely
   for elx = 1:nelx
    n1 = (nely+1)*(elx-1)+ely;
    n2 = (nely+1)* elx    +ely;
    Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2; 2*n1+1;2*n1+2],1);
    c = c + x(ely,elx)^penal*Ue'*KE*Ue;
    dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
   end
end
```

n_1和n_2已经解释过了，作为节点的定义，其中$U^{(e)}$为某一个单元上 4 个节点 8 个方向上的位移，其中这个循环中节点坐标的位移矩阵已经通过有限元分析子程序求解出来了，此处计算每一个单元的柔度，此处利用目标函数的表达式：

c = c + x(ely,elx)^penal*Ue'*KE*Ue;

$$\min \quad C(X)=FU=U^TKU=\sum_{i=1}^{m}\sum_{j=1}^{n}u_{i,j}^Tk_{i,j}u_{i,j}=\sum_{i=1}^{m}\sum_{j=1}^{n}(x_{i,j})^pu_{i,j}^Tk_0u_{i,j}$$

k_0为单元刚度矩阵。

dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;

这句是计算每一个单元的灵敏度的，目标函数的灵敏度表达：

在每一个单元上目标函数C对设计变量X的求导：

$$\frac{\partial c}{\partial x_e} = F^T \frac{\partial U}{\partial x_e} \tag{24}$$

根据方程$F = KU$，同时两边对设计变量进行求导：

$$0 = \frac{\partial K}{\partial x_e} U + K \frac{\partial U}{\partial x_e}$$

$$\frac{\partial U}{\partial x_e} = -K^{-1} \frac{\partial K}{\partial x_e} U \tag{25}$$

将式（25）代入式（24）中：

$$\frac{\partial c}{\partial x_e} = -U^T \frac{\partial K}{\partial x_e} U = -p(x_e)^{p-1} u_e^T k_0 u_e \tag{26}$$

对每一个节点进行计算之后，完成循环，程序来到网格过滤子程序：

```
[dc]=check(nelx,nely,rmin,x,dc);
```

进入子程序：

```
dcn=zeros(nely,nelx);
```

首先定义所有节点的灵敏度矩阵为零矩阵。

```
for i = 1:nelx
  for j = 1:nely
     sum=0.0;
     for k = max(i-floor(rmin),1):min(i+floor(rmin),nelx)
       for l = max(j-floor(rmin),1):min(j+floor(rmin),nely)
         fac = rmin-sqrt((i-k)^2+(j-l)^2);
         sum = sum+max(0,fac);
         dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);
       end
     end
     dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
  end
end
```

首先解释此处的数学和物理模型基础：

考虑整个网格的编排，包括边界附近和中间部分，我们定义有过滤半径，在图中我们就以第$x(i, j)$作为圆心，以$r_{\min} = 1$举例作为过滤半径作图，在不同区域附近有不同的情况。如图 7：

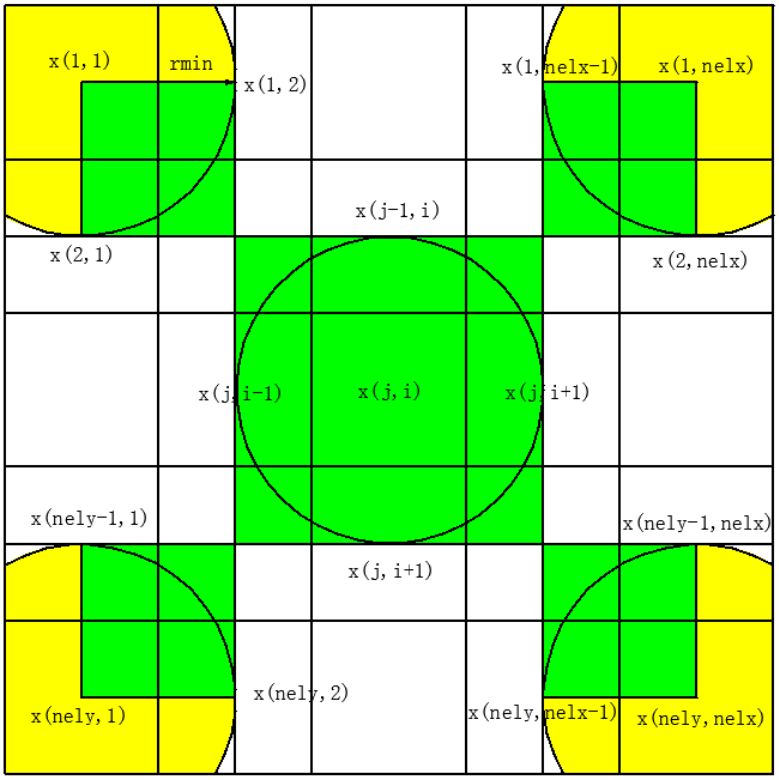


图 7.敏度过滤图解分析

```
for i = 1:nelx
  for j = 1:nely
```

总循环，以每一个单元的坐标进行循环，其中 j 是列坐标，从 1 到 nely，i 是横坐标，从 1 到 nelx。所以每一次分析都是以每一个单元作为分析，其顺序和图 4 中蓝色箭头方向一致。

```
sum=0.0;
```

这个变量是用来计算算子 fac 的变量。

```
for k = max(i-floor(rmin),1): min(i+floor(rmin),nelx)
 for l = max(j-floor(rmin),1):min(j+floor(rmin),nely)
```

主循环已经确定了某一个单元坐标 $x(j,i)$，这段程序是遍历附近的单元，如果满足以单元 $x(j,i)$ 中心为圆心以 $r_{\min}$ 作为半径的圆，如图 7 中的黄色区域所示，在圆内（包括边线）上如果包含其他单元的中心坐标的话，就要添加进入加权叠加灵敏度计算。

对于语句：

```
for k = max(i-floor(rmin),1): min(i+floor(rmin),nelx)
```

floor 是 matlab 中的向下取整函数。

程序语句代表如果在 x 坐标上，初始状态：max(i-floor(rmin),1)，代表圆的

范围超过了边界，如图 7 中单元 $x(1,1)$ 的横坐标（左侧圆黄色部分）所示，则在边界上 k 的过滤边界（绿色部分）选择较大值 1，此时它的最大值为 min(i+floor(rmin),nelx)= i+floor(rmin)=1+ floor(rmin)，故灵敏度过滤的选择单元在 x 方向上的范围为 $x(1,1)$ 为 1: 1+floor(rmin)，可以通过图 4 中 $x(1,1)$ 附近绿色区域理解。

最终状态：min(i+floor(rmin),nelx)，同理，如果圆所包含的边界超过了设计范围，如单元 $x(1,nelx)$，右侧圆（黄色部分）超过了设计范围，则过滤边界（绿色区域）选择较小值 nelx，此时灵敏度过滤在 x 方向上的范围为 nelx-floor(rmin): nelx。可以通过图 4 中单元 $x(1,nelx)$ 附近的绿色过滤区域理解。

同理，如果在 y 坐标上，如果是类似于 $x(1,1)$ 和 $x(nely,1)$ 单元一样，圆部分（黄色区域）超过了 y 方向的设计范围，则同理按照以上解释去理解。

```
fac = rmin-sqrt((i-k)^2+(j-l)^2);
sum = sum+max(0,fac);
```

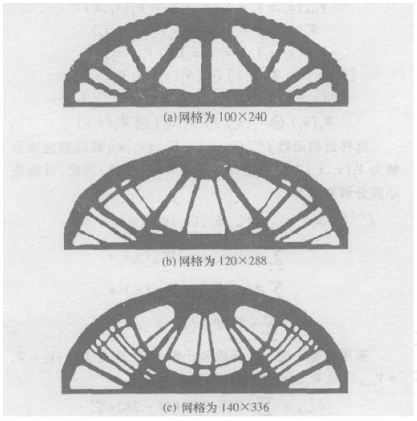
此处解释数学模型：

Sigmund 提出的敏度过滤技术，是一种局部意义上的约束方法，这种敏度过滤技术通过引入卷积算子（或权重因子）对过滤半径以内的其他单元的敏度进行加权平均，修正单元的灵敏度，从而避免棋盘格现象，通常能够减少棋盘格现象的方法都能减少网格依赖性。

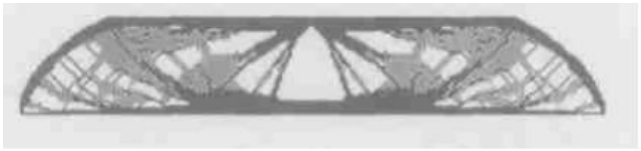
所谓棋盘格现象，就是在计算区域内，材料密度为 1 和 0 的单元呈现周期性的分布状态，如下图所示，棋盘格的优化结果在工程上的可制造性很差，没有实际意义。

所谓的网格依赖性就是对于拓扑优化的计算结果，经常与区域的网格划分密度有关，选择不同的网格划分密度会产生不同的优化计算结果，通常情况下，当网格划分密度变大时，会出现很多的细小分支结构，网格依赖性也使得计算结果的可制造性下降。

Sigmund 提出的敏度过滤技术的本质是利用过滤范围内所有单元的敏度信息修正中心单元的敏度信息，即采用过滤半径范围内各单元敏度的加权平均值代替中心单元的敏度值。



网格依赖性



棋盘格现象

目标函数对设计变量修正后的单元敏度值为：

$$\frac{\partial \hat{c}}{\partial x_e}=\frac{\sum_{i=1}^{N}\hat{H}_i x_i\frac{\partial c}{\partial x_i}}{x_e\sum_{i=1}^{N}\hat{H}_i} \tag{27}$$

式中：

卷积算子（权重算子）$\hat{H}_i=r_{\min}-dist(e,i),\{i\in N,|\ dist(e,i)\le r_{\min}\}$；

$dist(e,i)$ 为单元 e 和 i 中心的距离；

$r_{\min}$ 为过滤半径；

N 为单元数目。

故程序中的 fac= rmin-sqrt((i-k)^2+(j-l)^2);即计算卷积算子 $\hat{H}_i$。

sum = sum+max(0,fac);即计算 $\sum_{i=1}^{N}\hat{H}_i$。

dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);即计算 $\sum_{i=1}^{N}\hat{H}_i x_i\frac{\partial c}{\partial x_i}$。

在循环如图 7 中绿色矩形框范围内，在黄色圆内的其他节点的敏度加权叠加之后：

dcn(j,i) = dcn(j,i)/(x(j,i)*sum);

进行平均计算，进行敏度过滤：

$$\frac{\partial \hat{c}}{\partial x_e}=\frac{\sum_{i=1}^{N}\hat{H}_i x_i \frac{\partial c}{\partial x_i}}{x_e\sum_{i=1}^{N}\hat{H}_i}$$

至此，敏度过滤段程序已经解释完毕。

主程序继续运行：

[x]　= OC(nelx,nely,x,volfrac,dc);

这段是利用优化准则法进行优化设计变量。

```
function [xnew]=OC(nelx,nely,x,volfrac,dc)
l1 = 0; l2 = 100000; move = 0.2;
while (l2-l1 > 1e-4)
   lmid = 0.5*(l2+l1);
   xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
   if sum(sum(xnew)) - volfrac*nelx*nely > 0;
      l1 = lmid;
   else
      l2 = lmid;
   end
end
```

解释其中的程序：

l1 = 0; l2 = 100000;

这是利用二分法来进行逼近和最终终止循环的边界。

move = 0.2;定义 m 。

while (l2-l1 > 1e-4)，当二分法的边界小于这个条件时，终止循环。

lmid = 0.5*(l2+l1);二分法中，边界的中间值，之后用来缩小逼近范围。

xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));

首先介绍这句程序的数学表达：

$$x_{new}=\begin{cases}\max(x_{\min},x_e-m) & x_eB_e^{\eta}\le\max(x_{\min},x_e-m)\\ x_eB_e^{\eta} & \max(x_{\min},x_e-m)<x_eB_e^{\eta}<\min(1,x_e+m)\\ \min(1,x_e+m) & \min(1,x_e+m)\le x_eB_e^{\eta}\end{cases}\qquad(28)$$

式中，η为$\frac{1}{2}$。

分别解释三段取值的情况，我们利用坐标轴解释

第一种情况：

当$x_eB_e^{\eta}$小于$x_{\min}$或x_e-m时，取$x_{\min}$和x_e-m的较大值。

第二种情况：

当$x_e B_e^{\eta}$位于$x_{\min}$、1和$x_e - m$、$x_e + m$之间时，选择更新变量$x_e B_e^{\eta}$

第三种情况：

当$x_e B_e^{\eta}$大于1或$x_e + m$时，取1和$x_e + m$的较小值

建议对所有的逻辑情况画数轴来理解，这样会更容易体会其中的关系。此处不给出所有的逻辑关系。

对于程序中的语句，我们从最内层括号进行：

x.*sqrt(-dc./lmid)

要注意的是，此处是整个结构的单元密度矩阵，是整体进行计算。此处是利用优化条件：

-dc./lmid：$B_e = \dfrac{-\dfrac{\partial c}{\partial x_e}}{\lambda \dfrac{\partial V}{\partial x_e}}$

其中λ取为二分法中的中间值。由于每一个单元的体积为$V = x_e v_0$，其中$v_0 = 1$，故：

$$\frac{\partial V}{\partial x_e} = 1 \text{。}$$

x.*sqrt(-dc./lmid): B_e^{η}，η取为$\dfrac{1}{2}$。

xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));

此处的逻辑关系读者可以按照括号从内层往外层的观察，其逻辑关系和（28）式的表述是相同的。

if sum(sum(xnew)) - volfrac*nelx*nely > 0;

这句是判断整体体积与规定的体积约束fV_0是否相同：

如果$V \le fV_0$，则将二分法的上边界定为之前计算的中间值，从而更新设计变量的时候能够朝着增大单元密度的方向进行。

如果$V \ge fV_0$，则将二分法的下边界定为之前计算的中间值，从而更新设计变量的时候能够朝着减小单元密度的方向进行。

至此，已经解释完毕更新设计变量的程序段。

回到主程序：

change = max(max(abs(x-xold)));

这句是计算设计变量的改变量的绝对值最大值。

disp([' It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',c) ...

' Vol.: ' sprintf('%6.3f',sum(sum(x))/(nelx*nely)) ... ' ch.: ' sprintf('%6.3f',change)]);

显示在这个循环之后的信息，包括：

迭代次数、柔度值、总体密度、设计变量的改变值。

colormap(gray); imagesc(-x); axis equal; axis tight; axis off;pause(1e-6);

将这个循环中的结构显示出来。

回到主程序的循环判断：

while change > 0.01

判断设计变量的变化值是否达到要求，否则继续进行迭代计算。

程序运行结果：