拓扑优化

孙天阳

中国科学技术大学数学科学学院

tysun@mail.ustc.edu.cn

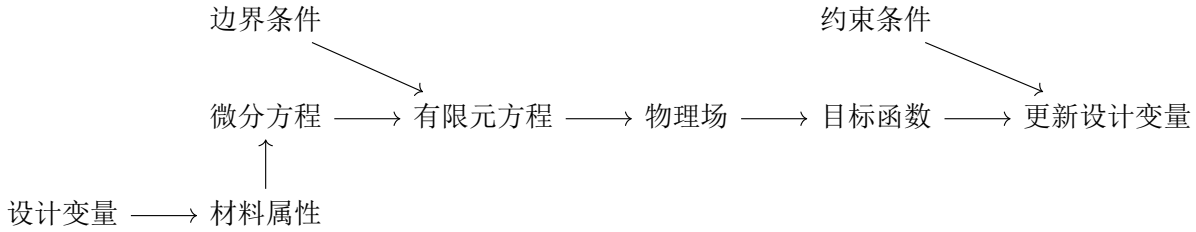
2025 年 3 月 1 日

目录

Chapter 1

宏观结构设计

拓扑优化是一种结构优化方法, 通过确定材料在空间中的分布, 使得结构在满足某些约束条件（如体积、应力等）下, 达到最优的性能指标.



1 第一个例子

给定一个结构, 给定材料的属性（如杨氏模量和泊松比）, 给定固定的点, 给定载荷（也就是什么位置受什么方向什么大小的力）, 可以通过线弹性静力学的方程组及有限元方法求解出该结构在该边界条件下的位移场. 我们希望设计一个轻量化结构在该载荷下抵抗变形的能力最强. 回忆

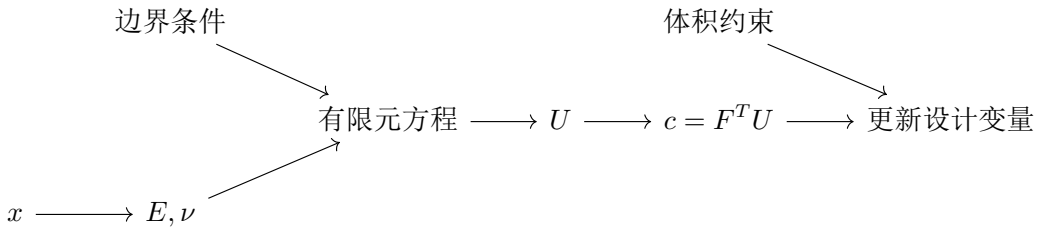
$$\int_\Omega C_{ijkl}\epsilon_{kl}(u)\epsilon_{ij}(v)\mathrm{d}x = \int_\Omega f_i v_i \mathrm{d}x + \int_{\Gamma_2} g_i v_i \mathrm{d}S \Longleftrightarrow V^T K U = V^T F.$$

我们选择结构在载荷下达到平衡时的弹性势能或者叫应变能 $U^T KU/2$ 作为抵抗变形的能力的量度, 弹性势能越小我们认为抵抗变形的能力越强. 常数不起作用所以可以略去, 再结合方程 $KU = F$, 所以最后我们的目标函数是 $c = F^T U$, 被称作柔度.

我们考虑的集合是形状不超出某个边框的所有结构, 而目标函数就是定义在这个集合上的函数, 我们要找这个函数的最小值, 这当然是一个优化问题. 现在要给所有结构的集合一个数学表达, 简单起见, 设考虑的区域 Ω 是一个矩形, 将其划分为 nely 行 nelx 列的正方形小单元, 那么所有的结构的集合就是 $\{0,1\}^{\mathrm{nelx}\times\mathrm{nely}}$, 也就是说我们只需要决定每个单元有没有材料, 0 代表没有材料 1 代表有材料, 我们就决定出了一个结构. 这显然是一个离散优化问题, 但我们并不把它当成一个离散优化问题来做（可能是因为计算效率的问题, 暂时我给不了什么 comment）, 而是使用取值在 $[0,1]$ 的连续设计变量（以便利用梯度信息）, 再通过各种技术得到离散的结果.

算法倾向于添加材料以最小化目标函数. 为了对这一点建立直观感受, 可以思考弹簧的例子. 在相同的外力下, 弹簧的弹性系数 k 越大, 平衡状态时的弹性势能就越小. 而根据我们之后要介绍的材料属性的插值模型, 设计变量的值越靠近 1, 该单元的杨氏模量就越大. 所以我们其实是在“轻量化”

和“高刚度”这两个矛盾的目标之间寻求一个平衡, 这也是拓扑优化方法的魅力所在. 为了避免收敛到每个位置都填充材料的平凡解, 为了实现“轻量化”的目标, 我们要给优化问题加上体积约束.



边界条件

接下来思考边界条件的施加. 我想说边界条件的施加有两类方式, 第一类是在微分方程的阶段提出明确的边界条件, 再看根据有限元的操作能对这些节点向量如 U, F 说些什么. 第二类是不管具体的细节, 直接对节点向量提要求.

$$\int_\Omega C_{ijkl}\epsilon_{kl}(u)\epsilon_{ij}(v)\mathrm{d}x = \int_\Omega f_i v_i \mathrm{d}x + \int_{\Gamma_2} g_i v_i \mathrm{d}S$$

这个式子已经用上了 Γ_1 的 Dirichlet 边界条件和 Γ_2 的 Neumann 边界条件. 通过有限元得到向量的等式 $V^T K U = V^T F$, 对任意满足 Dirichelt 边界条件的 V（也就是有些位置的值需要是 0）成立. 这个等式是左侧很多项的求和最后等于右侧很多项的求和, 但因为 V, U 要满足 Dirichlet 边界条件所以其中很多项都为 0, 丢掉这些为零的项等式依然成立, 并且剩下的项依然能写成矩阵相乘的形式, 这时我们得到 $\tilde{V}^T \tilde{K} \tilde{U} = \tilde{V}^T \tilde{F}$, 此时的 $\tilde{V}$ 是一个任意的向量, 我们可以根据它的任意性消去它得到 $\tilde{F} = \tilde{K}\tilde{U}$. 注意是不能够根据 $V^T K U = V^T F$ 得到 $F = KU$ 的, 因为这里的 V 不具有任意性. 综上所述我们在有限元编程中施加 Dirichlet 边界条件的方式就是指定 U 的哪些位置为 0 然后解方程的时候丢掉相应的位置. 而我们施加 Neumann 边界条件的方式也并不是指定面力 g 然后仔细去算会得到什么样的 F, 而是直接对 F 赋予想要的值. 这个操作也是解释的通的, 因为如果物理上只在某个节点施加某个方向的力, 去算相应的面力 g 应该会算出来狄拉克测度 δ, 再按部就班根据有限元去算应该正好能算出来相应的节点向量 F.

材料插值：SIMP

优化方法：OC

2 编程实现

```
nodenrs = reshape(1:(1+nelx)*(1+nely),1+nely,1+nelx);
```

这行代码的效果是为所有的节点编号, 以 $nelx = nely = 4$ 为例, 编号出来的效果就是

$$\begin{bmatrix} 1 & 6 & 11 & 16 & 21 \\ 2 & 7 & 12 & 17 & 22 \\ 3 & 8 & 13 & 18 & 23 \\ 4 & 9 & 14 & 19 & 24 \\ 5 & 10 & 15 & 20 & 25 \end{bmatrix}$$

在研究二维问题的时候, 每个节点有两个位移自由度, 所以按顺序对自由度编号得到

$$\begin{bmatrix} 1,2 & 11,12 & 21,22 & 31,32 & 41,42 \\ 3,4 & 13,14 & 23,24 & 33,34 & 43,44 \\ 5,6 & 15,16 & 25,26 & 35,36 & 45,46 \\ 7,8 & 17,18 & 27,28 & 37,38 & 47,48 \\ 9,10 & 19,20 & 29,30 & 39,40 & 49,50 \end{bmatrix}$$

为了有限元的矩阵组装, 我们需要记录每个单元的所有 8 个自由度的编号. 首先要决定一个开始的顺序, 为了尊重 Sigmund 的代码, 我们从左下角的 x 分量开始计数, 比如, 第一个单元的 8 个自由度

$$\begin{bmatrix} 3 & 4 & 13 & 14 & 11 & 12 & 1 & 2 \end{bmatrix}$$

我们可以发现, 以第一个自由度为基准, 所有 8 个自由度实际上可以表达为

$$\begin{bmatrix} 0 & 1 & 2(nely+1) & 2(nely+1)+1 & 2(nely+1)-2 & 2(nely+1)-1 & -2 & -1 \end{bmatrix}$$

所以在代码中, 我们首先记录每个单元左下角的 x 分量在总体自由度中的编号

```
edofVec = reshape(2*nodenrs(1:end-1,1:end-1)+1,nelx*nely,1);
```

$$edofVec = \begin{bmatrix} 3 \\ 5 \\ 7 \\ 9 \\ 13 \\ \vdots \\ 39 \end{bmatrix}$$

将该向量复制成相同的 8 列, 再每行加上偏差, 就得到了每个单元的 8 个自由度的索引

```
edofMat = repmat(edofVec,1,8)+repmat([0 1 2*nely+[2 3 0 1] -2 -1],nelx*nely,1);
```

$$\begin{bmatrix} 3 & 4 & 13 & 14 & 11 & 12 & 1 & 2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 39 & 40 & 49 & 50 & 47 & 48 & 37 & 38 \end{bmatrix}$$

下面我们来看如何应用这些东西来组装矩阵

$$\int_{\Omega} \text{被积项} = \sum_{e} \int_{\Omega_e} \text{被积项} = \sum_{e} \begin{pmatrix} v_1 & \cdots & v_8 \end{pmatrix} K_e \begin{pmatrix} u_1 \\ \vdots \\ u_8 \end{pmatrix} = VKU$$

我们需要将 K_e 中的值放到 K 的合适的位置. K_e 的列数等于一个单元的右侧向量自由度的个数. 考虑 K_e 中的一个元素, 该元素的列指标也就是 j 决定了它与第 j 个右侧向量自由度相乘, 所以它在 K 中的列指标就是该单元的第 j 个右侧向量自由度在总体右侧向量自由度中的位置. 我们要记录每个单元的 K_e 的每个元素最终的列指标. 这个向量的长度应该是

$$\text{单元的总数} \times K_e \text{ 的元素个数}$$

取来每个单元的右侧自由度在总体右侧自由度中位置的矩阵, 他的行数是单元的总数, 列数与 K_e 的列数相同, 不妨就考虑 1 个单元, 列向量指标应该是 edofMat 中的每个元素重复 K_e 的行数次,

```
jK = reshape(kron(列edofMat,ones(1,行_单元自由度个数))',单个矩阵元素数*nelx*nely,
sK = reshape(KE(:)*(Emin+xPhys(:)'.^penal*(E0-Emin)),64*nelx*nely,1);
```

第 i 行第 j 列中的元素, 它与 v_i 和 u_j 相乘, 所以看这个元素在 K 中该放到第几行和第几列实际上是看第 1 个单元的第 i 个自由度在总自由度中的编号和第 1 个单元的第 j 个自由度在总自由度中的编号. 假设我们按 Matlab 的规则将 Ke 展成一维向量, 然后用两个同样大小的一维向量记录对应元素应该被放置在 K 中的行指标和列指标, 那么行指标就是

$$iK = \begin{bmatrix} 3 & 4 & 13 & 14 & 11 & 12 & 1 & 2 & \cdots & 2 \end{bmatrix}$$

列指标就是

$$jK = \begin{bmatrix} 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & \cdots & 2 \end{bmatrix}$$

按这种方式去生成所有的 iK, jK, 再把所有的 Ke 按顺序展开并拼接好

```
iK = reshape(kron(edofMat,ones(8,1))',64*nelx*nely,1);
jK = reshape(kron(edofMat,ones(1,8))',64*nelx*nely,1);
sK = reshape(KE(:)*(Emin+xPhys(:)'.^penal*(E0-Emin)),64*nelx*nely,1);
```

3 20241212

这里我们暂时不追究刚度矩阵是怎么来的, 也不打算完全解释清楚刚度矩阵的意义, 仅仅解释如何从局部刚度矩阵拼凑整体刚度矩阵. 单元与节点的概念我们已经在有限元的课程与弹性力学的例子中熟悉, 我们可以大胆的假定这两个概念是基本的. 下一个概念是解空间的维数, 或者说解的自由度, 我相信它与节点是有关系的, 但这个关系依赖于具体的问题, 比如在一维的两点边值问题且我们使用分片线性函数进行逼近的时候, 解的自由度等于自由节点（没有被指定边值条件的节点）的个数; 在弹性力学的问题中, 解的自由度等于自由节点的个数的二倍; 在一维的两点边值问题且我们使用分片二次函数进行逼近的时候, 解除了在自由节点处有自由度, 在额外的位置还有自由度. 但总之, 每个问题, 我们可以确定解空间的维数, 解向量就是在选定一组基后解的系数. 在有限元分析中, 刚度矩阵是在将微分方程乘上测试函数分部积分后得到变分方程

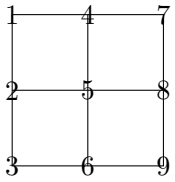
$$a(u,v) = (f,v), \quad \forall v \in V_h$$

然后在 v 取遍基的时候, 得到矩阵方程

$$KU = F$$

这里 K 就是刚度矩阵. 局部刚度矩阵, 从两点边值问题来看, 是对单元进行遍历, 计算这个单元对整体刚度矩阵的贡献（如果将 $a(\phi_i, \phi_j)$ 看成积分, 我觉得计算单元的贡献就相当于在单元上积分, 也就是考虑基函数们在这个单元上的部分）; 从弹性力学的问题来看, 也是在遍历单元, 问题在于, 我不知道原来的微分方程是什么（大概是弹性力学中的方程）, 不知道变分问题是什么, 不知道基函数是什么, 所以也不知道局部刚度矩阵该怎么求. 但是有人直接告诉了我局部刚度矩阵是什么, 也告诉了我整体刚度矩阵就是把局部刚度矩阵按位置加上去, 我也知道解的自由度出现在哪. 我知道解向量的几何含义是节点的水平与竖直的自由度, 知道解向量是选定基后的系数, 知道解向量的几何含义已经确定了说明基函数其实已经选好了, 只不过我不知道选的是什么.

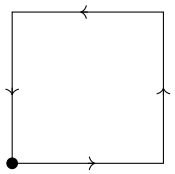
所以我们现在来看怎么拼凑. 整体刚度矩阵的维数等于解的自由度的个数等于节点的个数的二倍. 选定某个单元后, 涉及到四个节点, 也就是八个自由度. 所以, 想知道怎么把局部刚度矩阵放到整体刚度矩阵, 就需要知道所有的自由度是如何排序的, 以及对于单个刚度矩阵, 它涉及到的自由度是如何排序的. 在 88 行代码中, 所有节点的自由度按照从上到下, 从左到右的顺序排列, 比如我们以一个 2*2 的单元为例, 那么它的节点编号就为



在 Matlab 中, 我们使用如下代码实现全部节点的编号

```
nodenrs = reshape(1:(1+nelx)*(1+nely), 1+nely, 1+nelx);
```

对于某个单元, 我想要索引它的所有的节点自由度, 每个节点有水平、竖直两个自由度, 也就是每个单元涉及到 8 个自由度, 我希望给定第 i 个单元, 我能立马索引到它的 8 个节点自由度在全部节点自由度中的位置. 一个简单的观察是, 我只要给定了该单元某一特定自由度的索引, 我就能算出该单元其余自由度的索引. 比如我从左下角的水平分量出发, 按照如图的顺序,



则 8 个节点自由度的编号相对于左下角的水平自由度的编号的位移为

$$[0 \quad 1 \quad 2*\text{nely}+[2 \quad 3 \quad 0 \quad 1] \quad -2 \quad -1].$$

所以如果我想要索引每个单元的 8 个节点的编号, 我只需要先索引出每个单元的同一位置的节点的编号, 再加上位移的向量就好了. 在 88 行代码中, 我们选择先索引每个单元的左下角的水平分量

```
edofVec = reshape(2*nodenrs(1:end-1,1:end-1)+1,nelx*nely,1);
```

在 2*2 的例子中这样就得到

$$\begin{pmatrix} 3 \\ 5 \\ 9 \\ 11 \end{pmatrix}$$

现在第 i 行就是第 i 个单元的左下角水平分量在全部自由度分量中的编号. 为了得到第 i 个单元所有自由度的编号, 我们以目前第 i 行的这个数作为起点, 加在偏移向量

$$\begin{pmatrix} 0 & 1 & 6 & 7 & 4 & 5 & -2 & -1 \end{pmatrix}$$

上, 这样就得到

$$\begin{pmatrix} 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 \\ 11 & 11 & 11 & 11 & 11 & 11 & 11 & 11 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 6 & 7 & 4 & 5 & -2 & -1 \\ 0 & 1 & 6 & 7 & 4 & 5 & -2 & -1 \\ 0 & 1 & 6 & 7 & 4 & 5 & -2 & -1 \\ 0 & 1 & 6 & 7 & 4 & 5 & -2 & -1 \end{pmatrix}$$

等于

$$\begin{pmatrix} 3 & 4 & 9 & 10 & 7 & 8 & 1 & 2 \\ 5 & 6 & 11 & 12 & 9 & 10 & 3 & 4 \\ 9 & 10 & 15 & 16 & 13 & 14 & 7 & 8 \\ 11 & 12 & 17 & 18 & 15 & 16 & 9 & 10 \end{pmatrix}$$

代码实现如下

```
edofMat = repmat(edofVec,1,8)+repmat([0 1 2*nely+[2 3 0 1] -2 -1],nelx*nely,1);
```

矩阵的第 i 行就存储了第 i 个单元的所有节点自由度的信息.

4 移动渐近线法

5 柔性逆变机构

6 Min-Max 叙述

7 鲁棒拓扑优化

8 应力约束

9 局部体积约束

10 3 维拓扑优化

Chapter 2

微结构设计