Chapter 2

Iterative Methods and High Perofrmance Computing

2.1 稀疏矩阵技术

在科学计算中人们往往需要解决具有大量自由度的计算问题，如一个在三维立方体上采用100x100x100的网格的有限差分方法离椭圆方程，将得到一个大约1,000,000阶的线性方程组问题。 因此出现了“高性能科学计算”这一概念。高性能计算涉及到硬件架构、编程框架和编程语言、软件设计以及算法的并行化设计等诸多方面，在这里我们关注的是如何尽可能地减少算法在存储空间和运算时间等方面的开销。我们将针对典型的稀疏线性方程组求解问题

$$Ax = b \tag{2.1}$$

以及稀疏矩阵特征值问题

$$Bx = \lambda x \tag{2.2}$$

where A and B are sparse matrix, which is defined as the following

定义2.1.1 (Sparse Matrix)**.** It is the definition of Sparse matrix!

直接法在矩阵规模比较小(在偏微分方程数值解中小于 1 百万)的情形是高效的,但是实际问题往往要求解很大的n的矩阵,而且往往含有大量的0元素,在用直接法时就会耗费大量的时间和存储单元。稀疏矩阵的直接求逆是不可取的，因为在消元过程中会将大量原本为0的元素被消元运算转换为非0。尽管有一些自由度重排序算法减少稀疏矩阵的”带宽“以尽可能减少这种非零元填充事件的发生，随着矩阵规模增长(如至千万维)，直接求逆算法将很快耗尽内存和计算力。

2.1.1 稀疏矩阵存储技术

2.1.2 稀疏矩阵乘法运算

另一方面注意到稀疏矩阵的乘法具有较高的效率，迭代法

2.2 变分迭代法

基 记秩为N的矩阵,可以由一组线性无关的向量$\{\psi_k\}_{k=1}^N$。

大规模线性方程组的求解是有困难的. 带来困难的原因不仅仅在于其规模之大, 另一个重要的原因是线性方程组的不适定性(即矩阵奇异). 利用线性空间的理论, 基于变分(即投影)理论的迭代解法是计算近似解的有效方法.

Let A be an $n \times n$ real matrix and $\mathcal{K}$ and $\mathcal{L}$ be two m -dimensional subspaces of R_n. A projection technique onto the subspace $\mathcal{K}$ and orthogonal to $\mathcal{L}$ is a process described as

$$\text{Find } \tilde{x} \in x_0 + \mathcal{K}, \text{ such that } b - A\tilde{x} \perp \mathcal{L}$$

等价于

$$\text{Find } \tilde{x} = x_0 + \delta, \delta \in \mathcal{K}, \text{ such that } (r_0 - A\delta, \omega) = 0, \forall \omega \in \mathcal{L}$$

最速下降法和极小残量法分别是由两类不同的投影方法导致的:

最速下降(Steep Descent)法

```
for j = 0, 1, ..., until convergence do
    r_j = b − A x_j;
    α_j = (r_j, r_j)/(A r_j, r_j);
    x_{j+1} = x_j + α_j r_j;
end
```

极小残量(Minimal Residual)法

```
for j = 0, 1, ..., until convergence do
    r_j = b − A x_j;
    α_j = (A r_j, r_j)/(A r_j, A r_j);
    x_{j+1} = x_j + α_j r_j;
end
```

1950年,美国国家标准局数值分析研究所 Hestenes，Stiefel 和 Lanczos，发明了Krylov子空间迭代法求解$A\mathbf{x} = \mathbf{b}$。构造迭代

$$K\mathbf{x}_{i+1} = K\mathbf{x}_i + (\mathbf{b} - A\mathbf{x}_i)$$

其中，K(来源于作者俄国人Nikolai Krylov姓氏的首字母)是一个用投影法构造得到的接近于A的矩阵，根据K所属空间$\mathcal{K}$的不同取法得到求解不同类型的迭代格式。

该迭代形式的算法的妙处在于，它将复杂问题化简为阶段性的易于计算的子步骤。特别地在第m步，构造子空间

$$\mathcal{K}_m(A, \mathbf{r}_0) = span\{\mathbf{r}_0, A\mathbf{r}_0, \cdots, A^{m-1}\mathbf{r}_0\}.$$

只要分别采用$\mathbf{L}_m = \mathbf{K}_m$ 或$\mathbf{L}_m = A\mathbf{K}_m$ 即可得到不同类型的变分迭代法. 让我们来了解一下如何构造子空间的数值方法.

首先,可以利用Arnoldi算法构造正交子空间$\mathbf{K}_m$,如下述算法:

Choose a unit vector $\mathbf{v}_1$;
for $j = 1, 2, \ldots, m$ **do**
  Compute $h_{ij} = (A\mathbf{v}_j, \mathbf{v}_i), \quad \forall i = 1, 2, \cdots, j$;
  Compute $\mathbf{u}_j = A\mathbf{v}_j - \sum_{i=1}^{j} h_{ij}\mathbf{v}_i$;
  $h_{j+1,j} = \|\mathbf{u}_j\|_2$;
  if $h_{j+1,j} == 0$ **then**
    Stop
  else
    $\mathbf{v}_{j+1} = \mathbf{u}_j / h_{j+1,j}$
  end
end

接着实施Arnoldi-Modified Gram-Schmidt正交化过程:

Choose a unit vector $\mathbf{v}_1$;
for $j = 1, 2, \ldots, m$ **do**
  Compute $\mathbf{u}_j = A\mathbf{v}_j$;
  for $i = 1, \ldots, j$ **do**
    $h_{i,j} = (\mathbf{u}_j, \mathbf{v}_i)$;
    $\mathbf{u}_j = \mathbf{u}_j - h_{i,j}\mathbf{v}_i$;
  end
  $h_{j+1,j} = \|\mathbf{u}_j\|_2$;
  if $h_{j+1,j} == 0$ **then**
    Stop
  else
    $\mathbf{v}_{j+1} = \mathbf{u}_j / h_{j+1,j}$
  end
end

Full Orthogonalization Method(FOM)结合了上述两个过程:

Compute $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\beta = \|\mathbf{r}_0\|_2$, and $\mathbf{v}_1 = \mathbf{r}_0/\beta$;
Define the $m \times m$ matrix $H_m = \{h_{i,j}\}_{i,j=1,2,\ldots,m}$, set $H_m = 0$;
for $j = 1, 2, \ldots, m$ **do**
  Compute $\mathbf{u}_j = A\mathbf{v}_j$;
  for $i = 1, \ldots, j$ **do**
    $h_{i,j} = (\mathbf{u}_j, \mathbf{v}_i)$;
    $\mathbf{u}_j = \mathbf{u}_j - h_{i,j}\mathbf{v}_i$;
  end
  $h_{j+1,j} = \|\mathbf{u}_j\|_2$;
  if $h_{j+1,j} == 0$ **then**
    set $m = j$, and goto 16
  else
    Comput $\mathbf{v}_{j+1} = \mathbf{u}_j / h_{j+1,j}$
  end
end
Compute $\mathbf{y}_m = H_m^{-1}(\beta\mathbf{e}_1)$ and $\mathbf{x}_m = \mathbf{x}_0 + V_m\mathbf{y}_m$;

极小残差(Generalized Minimized Residual)法

简称GMRes. 通过上述算法的铺垫,可以归结为如下算法

1 Compute $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\beta = \|\mathbf{r}_0\|_2$, and $\mathbf{v}_1 = \mathbf{r}_0/\beta$;
2 Define the $(m+1) \times m$ matrix $H_m = \{h_{i,j}\}_{1\le i\le(m+1)}^{1\le j\le m}$, set $H_m = 0$;
3 **for** $j = 1, 2, \ldots, m$ **do**
4 Compute $\mathbf{u}_j = A\mathbf{v}_j$;
5 **for** $i = 1, \ldots, j$ **do**
6 $h_{i,j} = (\mathbf{u}_j, \mathbf{v}_i)$;
7 $\mathbf{u}_j = \mathbf{u}_j - h_{i,j}\mathbf{v}_i$;
8 **end**
9 $h_{j+1,j} = \|\mathbf{u}_j\|_2$;
10 **if** $h_{j+1,j} == 0$ **then**
11 set $m = j$, and goto 16
12 **else**
13 Comput $\mathbf{v}_{j+1} = \mathbf{u}_j/h_{j+1,j}$
14 **end**
15 **end**
16 Compute $\mathbf{y}_m$ to **minimize** $\|\beta\mathbf{e}_1 - H_m\mathbf{y}\|_2$ and $\mathbf{x}_m = \mathbf{x}_0 + V_m\mathbf{y}_m$;

共轭梯度法(Conjugate Gradient)及其变形

GMRes迭代算法对于非对称矩阵A尤其适用. 在A对称时的特殊情形,利用Lanczos三项递推关系可将FOM简化成如下简洁的算法:

1 Compute $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\mathbf{p}_0 = \mathbf{r}_0$;
2 **for** $j = 0, 1, \ldots,$*until convergence* **do**
3 $\alpha_j = (\mathbf{r}_j, \mathbf{r}_j)/(A\mathbf{p}_j, \mathbf{p}_j)$;
4 $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j\mathbf{p}_j$;
5 $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j A\mathbf{p}_j$;
6 $\beta_j = (\mathbf{r}_{j+1}, \mathbf{r}_{j+1})/(\mathbf{r}_j, \mathbf{r}_j)$;
7 $\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j\mathbf{p}_j$;
8 **end**

同理得到的迭代方法即为共轭梯度法(CG), 该迭代法对于对称矩阵A具有收敛性. 令$\mathbf{x}_m$ 是执行第$m-$步Conjugate Gradient algorithm得到的近似解, 并且$\mathbf{x}^*$是精确解，那么

$$||\mathbf{x}^* - \mathbf{x}_m||_A \le \left[\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right]^m ||\mathbf{x}^* - \mathbf{x}_0||_A$$

其中κ是矩阵A最大与最小特征值的比值，即所谓条件数。可见κ越接近1，则共轭梯度法收敛越快!

CG算法的其他主要常见变形有

- ICCG：Incomplete Cholesky预处理的CG迭代
- BiCG：双正交共轭梯度法
- BiCGstab：稳定化的BiCG

2.2.1 预条件处理技术

预处理技术可以减少迭代法运行时的迭代次数(注意不一定能减少计算量). 预处理的优点在于能结合不同的迭代法优势,在很多情形可以减少总体的计算量. 以CG迭代法为例，该方法的主要思想是: 设M是non-singular矩阵，并且$M^{-1}A$的条件数相对教小，则求解

$$(M^{-1}A)\mathbf{x} = M^{-1}\mathbf{b}$$

相对容易，或者

$$(AM^{-1})\mathbf{y} = \mathbf{b},$$

再求$M\mathbf{x} = \mathbf{y}$得到原方程的解。

显然地,$M\mathbf{x} = \mathbf{y}$相比于原问题更容易求解. 另一方面,CG算法采用的M也需要是对称正定的, 这样不会破坏CG算法的收敛性. 举例来说, 假设$A = L + D + L^T$,那么可以构造Jacobi预处理子

$$M := M_{Jacobi} = D^{-1}$$

或SOR预处理子

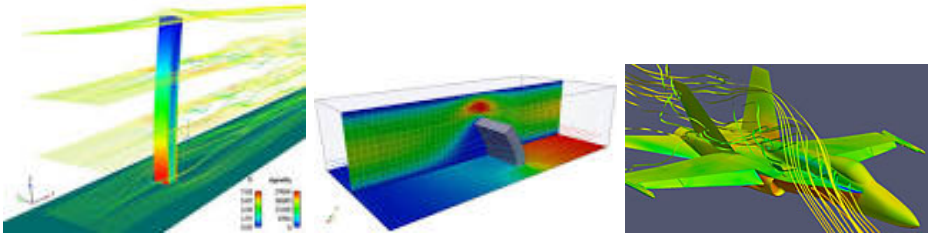
$$M := M_{SOR} = \frac{1}{\omega(2-\omega)}(D+\omega L)D^{-1}(D+\omega L^T), 0 < \omega < 2.$$

2.3 多物理耦合

基本物理过程的有限元建模和计算已相对成熟,然而多物理、多尺度现象的建模和计算仍是难点，有限元方法仍面临的挑战和机遇。”未来的几年内，**多物理场(Multiphysics)**分析工具(基于有限元/有限体积法)将会给学术界和工程界带来震惊。单调的“设计－校验”的设计方法将会慢慢被淘汰，虚拟造型技术将让你的思想走得更远，通过模拟仿真将会点燃创新的火花”，Dr. Louis Komzsik说道，他是Chief Numerical Analyst at Siemens PLM Software，被世人誉为’the father of algorithms for Nastran’。

2.3.1 Fluid Structure Interaction

考察可变形/移动的物体与流体环境的相互影响 主要有 stable 和 oscillatory 两种情形：固体结构



的应变产生应力，以抵消环境流体的作用力；或由固体结构在多个空间位置往复运动。让我们看一个有趣的FSI数学模型 **Reference**: Andrew T. Barker and Xiao-Chuan Cai, SIAM J. Sci. Comput., 32(4), 2395-2417. 流动问题de数值方法，参考lecturenoteforNS.pdf

求解耦合问题FSI 常用数值方案：couple 与 decouple。利用牛顿（Newton-Raphson）迭代法，同时在流体和固体区域求解耦合方程组（One Shot方法）或者利用不动点(Fixed-point)迭代: 分别求解流动问题（the flow problem）和结构问题（structural problem），直至变化量趋向于零。适用情形：Interaction between the fluid and the structure is weak, only one fixed-point iteration is required within each time step.

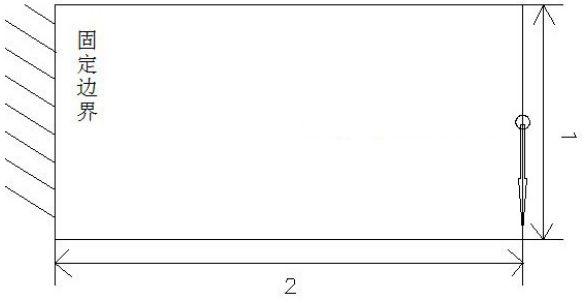
$$(\tilde{M}-(\Delta t/2)\tilde{K})y^{n+1}=(\tilde{M}+(\Delta t/2)\tilde{K})y^{n},$$

$$y^{n}=\begin{pmatrix}u_f\\ p_f\\ x_s\\ \dot{x}_s\\ p_s\\ x_f\end{pmatrix}^{n},\quad \tilde{M}=\begin{pmatrix}M_f&0&0&0&0&0\\0&0&0&0&0&0\\0&0&I&0&0&0\\0&0&0&\rho_s M_s&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\end{pmatrix},$$

$$\rho_s\frac{\partial^2}{\partial t^2}\mathbf{x}_s=\nabla\cdot\sigma_s+\beta\frac{\partial}{\partial t}(\Delta\mathbf{x}_s)-\gamma\mathbf{x}_s,\ \Rightarrow\ \begin{array}{l}\dfrac{\partial\mathbf{u}_f}{\partial t}+(\mathbf{u}_f\cdot\nabla)\mathbf{u}_f+\dfrac{1}{\rho_f}\nabla p_f=\nu_f\Delta\mathbf{u}_f,\\ \nabla\cdot\mathbf{u}_f=0.\end{array}\quad \tilde{K}=\begin{pmatrix}N(u_f)-\nu_f K_f&-Q_f^T&0&0&0&0\\Q_f&0&0&0&0&0\\0&0&0&I&0&0\\A_u&A_p&K_s+\gamma M_s&\beta K_s&-Q_s^T&0\\0&0&Q_s&0&0&0\\0&0&0&0&0&K_m\end{pmatrix}$$

2.3.2 Shape Optimization with PDE Constrains

形状优化 目标函数为柔度(极小化)



$$J(\Omega)=\int_{\Omega}f*udx+\int_{\Gamma_N}g*uds=\int_{\Omega}Ae(u)*e(u)dx \tag{2.3}$$

约束条件:弹性平衡条件。 应用实例：桥梁设计、发动机支架设计等

例2.3.1 (线弹性模型的有限元方法). 弹性平衡方程

$$\begin{cases}-div(Ae(u)=f) & in\quad \Omega,\\ u=0 & on\quad \Gamma_D,\\ (Ae(u))n=g & on\quad \Gamma_N,\end{cases} \tag{2.4}$$

有限元方法（参考前一讲）离散得到线性代数方程组

$$F=\begin{pmatrix}F_1\\F_2\\\vdots\\F_n\end{pmatrix}=\begin{pmatrix}k_{11}&k_{12}&\cdots&k_{1n}\\k_{21}&k_{22}&\cdots&k_{2n}\\\vdots&\ddots&&\vdots\\\vdots&&\ddots&\vdots\\k_{n1}&k_{n2}&\cdots&k_{nn}\end{pmatrix}\begin{pmatrix}u_1\\u_2\\\vdots\\u_n\end{pmatrix}=K\cdot u$$

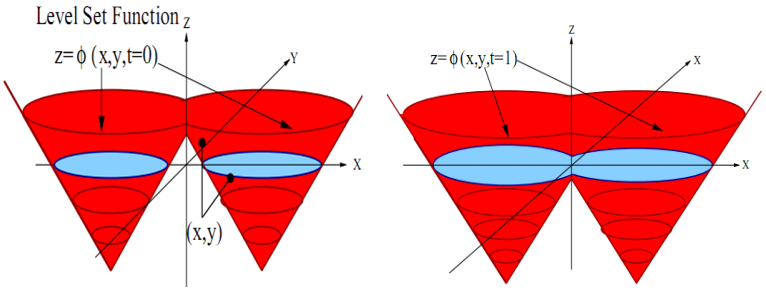
2.3.3 水平集方法

Motivation： 平面封闭曲线可隐式表示为二维函数的水平集线（一般使用零水平集，即c=0）

$$C = \{(x, y), u(x, y) = c\}$$

若平面封闭曲线随时间t变化则（如下图所示）

$$C(t) = \{(x, y), u(x, y, t) = c\}$$



水平集方法的控制方程

$$\begin{cases} \psi(x) = 0 & \Leftrightarrow x \in \partial\Omega, \\ \psi(x) < 0 & \Leftrightarrow x \in \Omega \\ \psi(x) > 0 & \Leftrightarrow x \in (D \backslash \bar{\Omega}) \end{cases}$$

零水平集满足：$\psi(t, x(t)) = 0, \forall x(t) \in \partial\Omega(t)$,则关于时间求导

$$\frac{\partial \psi}{\partial t} + \nabla\psi * \frac{\partial \vec{X}}{\partial t} = \frac{\partial \psi}{\partial t} + Vn * \nabla\psi = 0,$$

其中，外法向量$n = \nabla\psi/|\nabla\psi|$。可得Hamilton-Jacobi方程

$$\frac{\partial \psi}{\partial t} + V|\nabla\psi| = 0, \tag{2.5}$$

H-J方程的数值解法

$$\frac{\psi_i^{n+1} - \psi_i^n}{\Delta t} + min(V_i^n, 0)g^-(D_x^+\psi_i^n, D_x^-\psi_i^n) + max(V_i^n, 0)g^+(D_x^+\psi_i^n, D_x^-\psi_i^n) = 0,$$

其中

$$D_x^+\psi_i^n = \frac{\psi_{i+1}^n - \psi_i^n}{\Delta x}, D_x^-\psi_i^n = \frac{\psi_i^n - \psi_{i-1}^n}{\Delta x}$$

$$g^+(d^+, d^-) = \sqrt{min(d^+, 0)^2 + max(d^-, 0)^2},$$

$$g^-(d^+, d^-) = \sqrt{max(d^+, 0)^2 + min(d^-, 0)^2}.$$

重新初始化

求解水平集函数，更新设计域后要重新初始化,使函数为符号距离函数，满足$|\nabla\phi| = 1$，方法求解下述方程直至收敛

$$\frac{\partial\phi}{\partial t} + s(\phi_0)(|\nabla\phi| - 1) = 0, \tag{2.6}$$

其中，

$$\begin{cases} s(\phi_0) = & 1 & \phi > 0 \\ s(\phi_0) = & 0 & \phi = 0 \\ s(\phi_0) = & -1 & \phi < 0 \end{cases}$$

关于形状导数

若定义$\Omega_\theta = (Id + \theta)(\Omega)$为区域$\Omega$的一个扰动，那么

$$J((Id + \theta)(\Omega)) = J(\Omega) + J'(\Omega)(\theta) + o(\theta)$$

一些事实: 若考虑$J(\Omega) = \int_\Omega \phi(x)dx$，则

$$J'(\Omega)(\theta) = \int_\Omega div(\theta\phi(x))dx = \int_{\partial\Omega} \theta(x)\mathbf{n}(x)\phi(x)ds \tag{2.7}$$

若考虑$J(\Omega) = \int_{\partial\Omega} \phi(x)ds$，则

$$J'(\Omega)(\theta) = \int_{\partial\Omega} \theta(x)\mathbf{n}(x)\big(\frac{\partial\phi}{\partial n} + H\phi\big)ds \tag{2.8}$$

证明参考文献[?]或[?]

优化模型（2.3）的形状导数

对于目标函数

$$J(\Omega) = \int_\Omega fudx + \int_{\partial\Omega} guds$$

$\int_\Omega fudx$借助公式（2.7）可得形状导数

$$\int_\Omega \theta(x) \cdot n(x)fuds$$

另一方面， $\int_{\partial\Omega} guds$借助公式（2.8）可得形状导数

$$\int_{\partial\Omega} \theta(x) \cdot n(x)\big(\frac{\partial(gu)}{\partial n} + Hgu\big)ds$$

终于

$$J'(\Omega) = \int_{\partial\Omega} \theta(x) \cdot n(x)\big(\frac{\partial(gu)}{\partial n} + Hgu + fu\big)ds$$

水平集方程的速度场- 形状导数

形状优化：取方程(2.5)中的速度场为

$$V = \frac{\partial(gu)}{\partial n} + Hgu + fu \tag{2.9}$$

分析：依上述V的取法

$$J'(\Omega) = \int_{\partial\Omega} V\theta \cdot \mathbf{n} ds$$

定义向量场$\theta = -V\mathbf{n}$

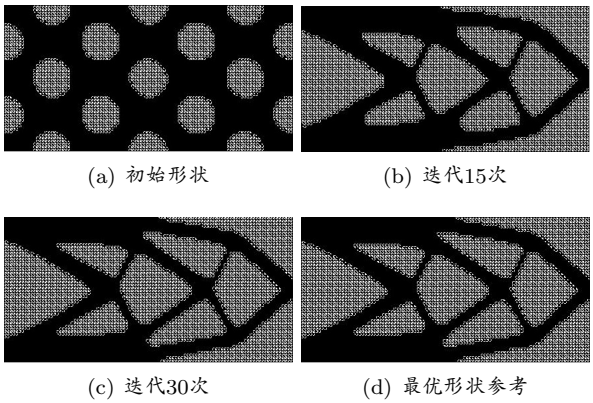
$$J((Id+\theta)(\Omega)) = J(\Omega) - \int_{\partial\Omega} V^2 ds + o(\theta),$$

设置时间步长t，将形状Ω更新到$\Omega_t = (Id + t\theta)\Omega$,

$$J(\Omega_t) = J(\Omega) - t\int_{\partial\Omega} V^2 ds + 0(t^2),$$

则$V = \frac{\partial(gu)}{\partial n} + Hgu + fu$即为各点速度标量。

数值结果



(a) 初始形状 (b) 迭代15次

(c) 迭代30次 (d) 最优形状参考

Figure 2.1: 一阶精度水平集方法

初始条件取杨氏模量E=1.0，泊松比$\mu = 0.3$,网格数量为100×50,采用一阶精度的迎风格式，时间步长dt=1.0。

初始形状的依赖性:多孔更优

作为本章的总结。 多物理/多尺度现象是自然科学和工程领域问题的理论研究所关心的广泛性问题。科学计算工作者则更关心相应问题的数学模型与数值算法的研究：

- Assume scientific computing be a secondary tool beside the practical experiments/第三种科学方法
- Interdisciplinary(交叉) topics, arising from material/nuclear science and bio-mechanics.
- Modeling meso-scale(介观尺度) problems.

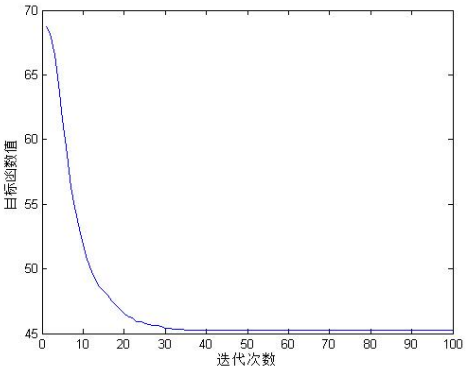
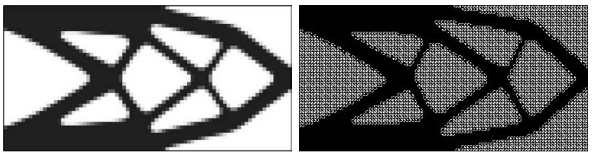
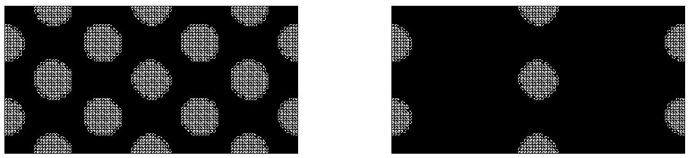
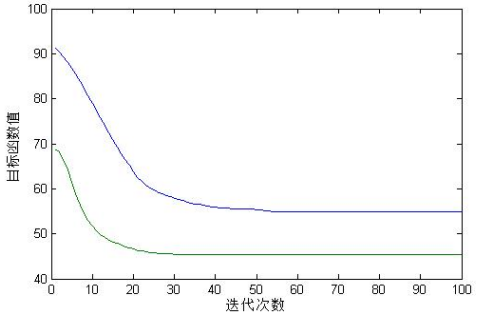


Figure 2.2: 目标函数变化



(a) 来自文献[?] (b) 最优形状





软件包与参考教材

关于特征值问题及其求解方法有不少专著

- Yousef Saad: Iterative methods for sparse systems(2nd edition)
- Yousef Saad: Numerical Methods for Large Eigenvalue Problems(2nd Edition)
- James Demmel, et. al. : Templates for the solution of algebraic eigenvalue problems
- R. S. Varga: Matrix Iterative Analysis(2nd Edition)
- H. Wilkinson: The Algebraic Eigenvalue Problem(Wilkinson Prize for Numerical Software)

2.3.4 软件包

petsc

偏微分方程数值解研究常用软件包,支持大规模并行计算.

slepc

特征值问题求解工具包

hypre

支持多重网格快速算法开发和大规模并行计算.

trillinos

偏微分方程数值解研究常用软件包,支持大规模并行计算.

Exercises

1. exe1
2. exe1
3. exe1
4. exe1
5. exe1
6. exe1
7. exe1
8. exe1
9. exe1
10. exe1
11. exe1
12. exe1

13. exe1

14. exe1

15. exe1 5