

# 理解共轭梯度法

赵海亮\*

浙江大学计算机科学与技术学院

hliangzhao@zju.edu.cn

July 10, 2020

## Contents

|                           |           |
|---------------------------|-----------|
| <b>1 引言</b>               | <b>1</b>  |
| <b>2 二次型</b>              | <b>2</b>  |
| <b>3 最速下降法</b>            | <b>3</b>  |
| 3.1 方向和最优步长 . . . . .     | 3         |
| 3.2 特征值的相关特性 . . . . .    | 5         |
| 3.2.1 谱半径 . . . . .       | 5         |
| 3.2.2 雅各比迭代 . . . . .     | 6         |
| 3.3 收敛性分析 . . . . .       | 7         |
| 3.3.1 简单情形 . . . . .      | 7         |
| 3.3.2 一般情形 . . . . .      | 9         |
| <b>4 共轭方向法</b>            | <b>12</b> |
| 4.1 什么是共轭 . . . . .       | 12        |
| 4.2 共轭格莱姆-施密特过程 . . . . . | 14        |
| 4.3 最优性分析 . . . . .       | 15        |
| 4.4 共轭梯度法 . . . . .       | 17        |
| 4.5 收敛性分析 . . . . .       | 17        |
| 4.5.1 选择最佳的多项式 . . . . .  | 18        |
| 4.5.2 切比雪夫多项式 . . . . .   | 20        |
| 4.6 复杂度分析 . . . . .       | 21        |
| <b>5 共轭梯度法的实现</b>         | <b>22</b> |
| 5.1 起点与终点 . . . . .       | 22        |
| 5.2 预处理 . . . . .         | 22        |
| 5.3 扩展到一般场景 . . . . .     | 23        |
| <b>6 后记</b>               | <b>24</b> |

---

\*<http://hliangzhao.me>

# 1 引言

求解无约束非线性优化问题的算法有很多种。最常用的无约束非线性优化算法是一维搜索 (line search)，其核心是找到合适的迭代方向和每次迭代的最优步长。如果没有曲线信息可供利用 (一阶、二阶甚至更高阶的导数)，那么可以用黄金分割法、斐波那契法等方法缩小搜索区间，也可以借助多项式的强大表示能力执行插值法 (牛顿法就是二点二次插值)；如果有信息可用，那么通常借助泰勒展开式获得迭代方向。具体地，如果有一阶信息可供利用，那么可以采用梯度法 (最速下降法) 及其一系列的改进方法 (泰勒一阶展开)；如果有二阶信息可供利用，那么可以采用牛顿法及其一系列的改进方法 (泰勒二阶展开)。牛顿法的两大问题分别是 Hessian 矩阵不总正定和 Hessian 矩阵的逆太难算。对于前者，改进策略是提出各种修正方案保证迭代方向一定是函数下降的方向。共轭梯度法 (Conjugate Gradient method, CG) 就是一种常用的改进方法，其收敛速度介于梯度法和牛顿法之间；对于后者，改进策略是各种拟牛顿法 (用别的矩阵来代替 Hessian 矩阵的逆)。除了一维搜索，另一类无约束非线性优化的算法是信赖域方法。本质上，信赖域方法是在局部用二次型插值目标曲线，仍就是借助多项式的强大表示能力。

本文将重点介绍共轭梯度法 (以下简称 CG)。CG 是解决大型线性方程组问题最流行的迭代算法。它对于如下形式的问题特别有效：

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad (1.1)$$

其中  $\mathbf{A} \in \mathbb{R}^{n \times n}$  是一个对称正定矩阵<sup>1</sup>， $\mathbf{b} \in \mathbb{R}^n$  是一个已知向量， $\mathbf{x} \in \mathbb{R}^n$  是待求解的未知向量。当然，我们有很多算法可以求解线性方程组问题，例如高斯消去法 (对  $\mathbf{A}$  执行 LU 分解)、基于迭代的 Jacobi 方法、Gauss-Seidel 方法以及松弛法等。作为一种基于迭代的方法，和高斯消去法相比，CG 的优势在于当  $\mathbf{A}$  是大规模稀疏矩阵的时候，其速度和内存占用都小于高斯消去法；和 Jacobi 方法及其改进相比，当  $\mathbf{A}$  是对称正定矩阵时，CG 可以保证收敛到  $\mathbf{Ax} = \mathbf{b}$  的真值。这些结论我们都会在后文中给出证明。

本文将从二次型出发，首先给出梯度法及其性能分析，随后深入分析 CG 的理论原理、算法步骤、最优性以及收敛速率。此外，除非明确指出，否则本文提到的矩阵均为  $n$  阶方阵。

## 2 二次型

二次型 (the Quadratic Form) 是一个从向量映射到标量的函数：

$$f(\mathbf{x}) \triangleq \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x} + c, \quad (2.1)$$

其中  $\mathbf{A} \in \mathbb{R}^{n \times n}$ ,  $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$ ,  $c \in \mathbb{R}$ 。根据矩阵微积分可知  $\frac{\partial \mathbf{x}^T \mathbf{A} \mathbf{x}}{\partial \mathbf{x}} = (\mathbf{A} + \mathbf{A}^T) \mathbf{x}$ ,  $\frac{\partial \mathbf{b}^T \mathbf{x}}{\partial \mathbf{x}} = \mathbf{b}$ 。当  $\mathbf{A}$  是一个对称矩阵时， $\frac{\partial \mathbf{x}^T \mathbf{A} \mathbf{x}}{\partial \mathbf{x}} = 2\mathbf{A}\mathbf{x}$ ,  $\nabla_{\mathbf{x}} f(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}$ 。

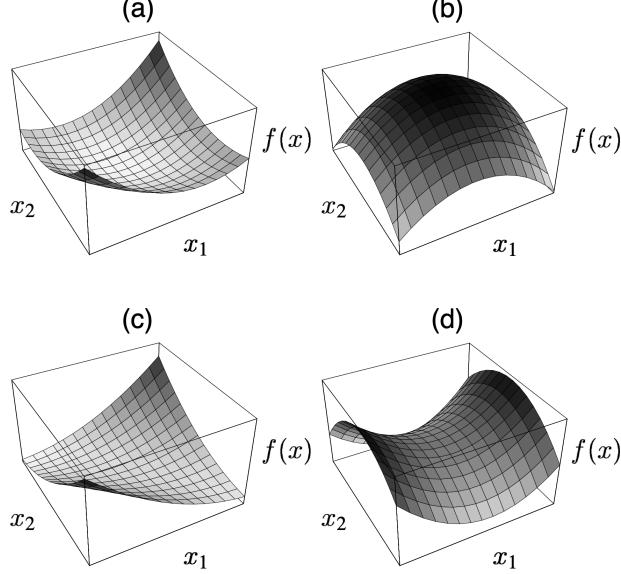
根据局部极小值存在的一阶必要条件， $f(\mathbf{x})$  的局部极小值由  $\frac{1}{2}(\mathbf{A} + \mathbf{A}^T)\mathbf{x} = \mathbf{b}$  的解给出。当  $\mathbf{A}$  是一个对称正定矩阵时， $\mathbf{A}$  必然可逆<sup>2</sup>，不妨设  $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$ ，即  $\nabla_{\mathbf{x}} f(\mathbf{x}) = 0$  的解。令  $\mathbf{x} = \mathbf{x}^* + \mathbf{e}$ ，则  $\forall \mathbf{e} \in \mathbb{R}^n$ ，

$$\begin{aligned} f(\mathbf{x}^* + \mathbf{e}) &= \frac{1}{2}(\mathbf{x}^* + \mathbf{e})^T \mathbf{A}(\mathbf{x}^* + \mathbf{e}) - \mathbf{b}^T(\mathbf{x}^* + \mathbf{e}) + c \\ &= f(\mathbf{x}^*) + \frac{1}{2} \mathbf{e}^T \mathbf{A} \mathbf{e} \\ &> f(\mathbf{x}^*). \end{aligned}$$

<sup>1</sup> 方阵  $\mathbf{A} \in \mathbb{R}^{n \times n}$  是正定的当且仅当  $\forall \mathbf{x} \in \mathbb{R}^n$ ,  $\mathbf{x}^T \mathbf{A} \mathbf{x} > 0$  恒成立。正定矩阵必然可逆，但不一定可以对角化。对称矩阵必然可以正交对角化，但不一定可逆。更多细节可以阅读<http://hliangzhao.me/math/math.pdf>。

<sup>2</sup> 这是因为正定矩阵必然可逆，与是否对称无关。简要证明：若存在  $\lambda$  和非零向量  $\mathbf{x}$  使得  $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ ，则  $\mathbf{x}^T \mathbf{A} \mathbf{x} > 0 \Leftrightarrow \lambda \|\mathbf{x}\|^2 > 0 \Leftrightarrow \lambda > 0$ 。即，只要存在特征值和对应的非零特征向量，那么特征值必然为正数。因为  $|\mathbf{A}|$  等于其所有特征值的乘积，所以行列式必然不为零，故  $\mathbf{A}$  必然可逆。

这说明当  $\mathbf{A}$  是一个对称正定矩阵时，通过求解  $\mathbf{Ax} = \mathbf{b}$  得到的就是  $f(\mathbf{x})$  的全局最小值。这相当于把一个二次型的最优化问题转变成了线性方程组的求解问题。当  $\mathbf{A}$  是一个对称正定矩阵时， $f(\mathbf{x})$  的图像是一个图形开口向上的抛物面。如果  $\mathbf{A}$  非正定，那么其全局最小值点就可能不止一个。Fig. 2.1展示了不同的  $\mathbf{A} \in \mathbb{R}^{2 \times 2}$  对  $f(\mathbf{x})$  的图像的影响。



**Figure 2.1:** 不同的  $\mathbf{A}$  对  $f(\mathbf{x})$  的图像的影响。**(a)** 正定矩阵的二次型；**(b)** 负定矩阵的二次型；**(c)** 奇异矩阵和非正定矩阵的二次型；**(d)** 不定矩阵的二次型：此时解是一个鞍点，梯度法和 CG 均无法处理该问题。

### 3 最速下降法

#### 3.1 方向和最优步长

最速下降法即梯度法。作为一种一维搜索算法，梯度法采取函数梯度的反方向作为迭代方向，并通过求导的方式确定迭代的最优步长。因为梯度是函数值增长最快的方向，所以梯度下降的方向是函数值减少最快的方向，这就是为什么梯度法又被称为最速下降法的原因。在梯度法中，我们将从任意点  $\mathbf{x}_{(0)}$  开始，经过迭代得到  $\mathbf{x}_{(1)}, \mathbf{x}_{(2)}, \dots$ ，直到达到最大迭代次数或者误差满足要求为止。在第  $i$  轮迭代中，梯度法选择的方向为  $\mathbf{b} - \mathbf{Ax}_{(i)}$ <sup>3</sup>。为了简化描述，我们定义如下变量：

- 误差向量： $\mathbf{e}_{(i)} \triangleq \mathbf{x}_{(i)} - \mathbf{x}^*$ ，表示第  $i$  轮迭代时  $\mathbf{x}_{(i)}$  和全局最小值点  $\mathbf{x}^*$  的距离；
- 残差向量： $\mathbf{r}_{(i)} \triangleq \mathbf{b} - \mathbf{Ax}_{(i)}$ ，表示第  $i$  轮迭代时  $\mathbf{Ax}_{(i)}$  和  $\mathbf{b}$  之间的距离。

残差向量其实是最速下降的方向，且  $\mathbf{r}_{(i)} = \mathbf{b} - \mathbf{Ax}_{(i)} = \mathbf{Ax}^* - \mathbf{Ax}_{(i)} = -\mathbf{A}\mathbf{e}_{(i)}$ 。这个等价变换会反复用到，务必记住。因此  $\mathbf{x}$  将按照如下公式进行更新：

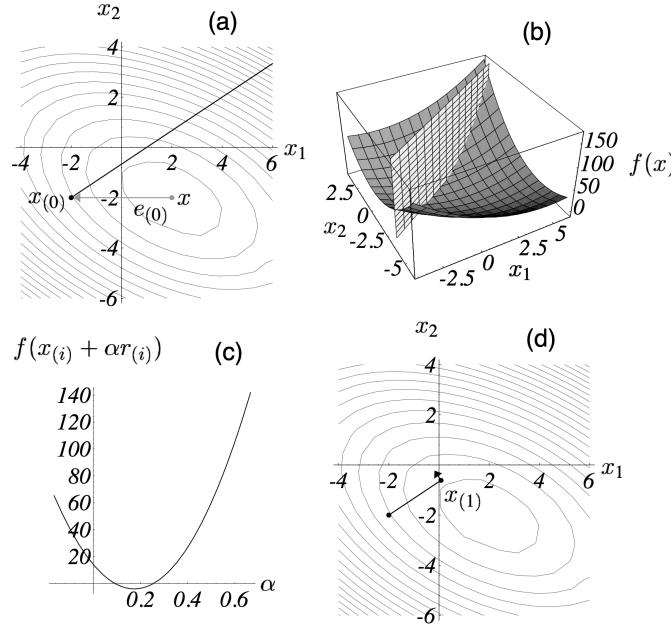
$$\mathbf{x}_{(i+1)} \leftarrow \mathbf{x}_{(i)} + \alpha \cdot \mathbf{r}_{(i)}, \quad (3.1)$$

<sup>3</sup>别忘了只有当  $\mathbf{A}$  是对称矩阵的时候才有  $\nabla_{\mathbf{x}} f = \mathbf{Ax} - \mathbf{b}$ 。不管  $\mathbf{A}$  是否为对称阵，梯度法都会选取  $\mathbf{b} - \mathbf{Ax}_{(i)}$  作为迭代方向，因此梯度法并不能够保证收敛。我们会在后文只会给出当  $\mathbf{A}$  为对称阵和对称正定阵时的收敛速度。

其中  $\alpha$  是第  $i$  轮迭代时的最优步长。如何确定  $\alpha$  呢？如果我们将  $f(\mathbf{x}_{(i+1)})$  视为  $\alpha$  的函数  $\varphi(\alpha)$ ，那么根据局部极小值存在的一阶必要条件， $\alpha$  应当使得  $\nabla_\alpha \varphi(\alpha) = 0$ ，即

$$\begin{aligned}\nabla_\alpha \varphi(\alpha) &= \nabla_\alpha f(\mathbf{x}_{(i+1)}) \\ &= (\nabla_{\mathbf{x}_{(i+1)}} f(\mathbf{x}_{(i+1)}))^T \nabla_\alpha \mathbf{x}_{(i+1)} \\ &= -\mathbf{r}_{(i+1)}^T \mathbf{r}_{(i)} \\ &= 0.\end{aligned}\tag{3.2}$$

(3.2) 表明最优步长  $\alpha$  使得前后两轮的迭代方向相互垂直（正交）。关于  $\alpha$  的确定，Fig. 3.1 给了一个形象的描述。



**Figure 3.1:** 梯度法第一轮迭代的形象描述。(a) (b) 在  $\mathbf{x}_{(0)} = [-2, -2]^T$  处经过梯度下降的方向做垂直于  $x_1x_2$  轴的平面；(c) 截面和  $f(\mathbf{x})$  的曲面的交线是一条抛物线，抛物线底部点即对应于函数值最小值点，将其视为  $\alpha$  的函数即可根据一阶导为零求出  $\alpha$  的值；(d)  $\alpha$  的取值让第一轮迭代和第二轮迭代的方向相互垂直。

根据(3.2)可得：

$$\begin{aligned}\mathbf{r}_{(i+1)}^T \mathbf{r}_{(i)} &= (\mathbf{b} - \mathbf{A}\mathbf{x}_{(i+1)})^T \mathbf{r}_{(i)} \\ &= [\mathbf{b} - \mathbf{A}(\mathbf{x}_{(i)} + \alpha \cdot \mathbf{r}_{(i)})]^T \mathbf{r}_{(i)} \\ &= (\mathbf{b} - \mathbf{A}\mathbf{x}_{(i)})^T \mathbf{r}_{(i)} - \alpha (\mathbf{A}\mathbf{r}_{(i)})^T \mathbf{r}_{(i)} \\ &= \mathbf{r}_{(i)}^T \mathbf{r}_{(i)} - \alpha \mathbf{r}_{(i)}^T \mathbf{A} \mathbf{r}_{(i)} \\ &= 0,\end{aligned}\tag{3.3}$$

因此可以确定  $\alpha$  的取值：

$$\alpha = \frac{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}}{\mathbf{r}_{(i)}^T \mathbf{A} \mathbf{r}_{(i)}}.\tag{3.4}$$

至此，梯度法已经讲解完毕。其过程为：对于任意初始点  $\mathbf{x}_{(0)}$ ，将(3.4)代入(3.1)中，根据(3.1)更新依次得到  $\mathbf{x}_{(1)}, \mathbf{x}_{(2)}, \dots$ ，直到达到最大迭代次数或者误差满足要求为止。当  $\mathbf{A}$  为对称正定阵的时候，梯度法一定可以收敛到全局最优解，因此迭代的终止条件可设为  $\mathbf{r}_k = \mathbf{0}$ 。梯度法的算法描述在 **Algorithm 1** 中。

---

**Algorithm 1:** 梯度法 (Steepest Descent) .

---

**Input:** 对称正定矩阵  $\mathbf{A} \in \mathbb{R}^{n \times n}$ , 向量  $\mathbf{b} \in \mathbb{R}^n$   
**Output:**  $\mathbf{x}^*$ , 既是线性方程组  $\mathbf{Ax} = \mathbf{b}$  的解, 也是  $\operatorname{argmin}_{\mathbf{x}} f(\mathbf{x})$

```
1  $\mathbf{r}_{(0)} = \mathbf{b} - \mathbf{Ax}_{(0)}$ 
2  $k = 0$ 
3 while  $\mathbf{r}_k \neq 0$  do
4    $k = k + 1$ 
5    $\alpha_{(k)} = (\mathbf{r}_{(k-1)}^\top \mathbf{r}_{(k-1)}) / (\mathbf{r}_{(k-1)}^\top \mathbf{Ar}_{(k-1)})$ 
6    $\mathbf{x}_{(k)} = \mathbf{x}_{(k-1)} + \alpha_{(k)} \cdot \mathbf{r}_{(k-1)}$ 
7    $\mathbf{r}_{(k)} = \mathbf{b} - \mathbf{Ax}_{(k)}$ 
8 end while
9 return  $\mathbf{x}_{(k)}$ 
```

---

**Algorithm 1** 的每一轮迭代中有两次矩阵-向量乘法运算 (step 5 和 step 7), 这是很耗时的。因此, 我们可以想办法减少一次矩阵-向量乘法来降低复杂度。怎么操作? 观察 **Algorithm 1** 第 6 行, 有

$$\begin{aligned} \mathbf{x}_{(k)} &= \mathbf{x}_{(k-1)} + \alpha_{(k)} \cdot \mathbf{r}_{(k-1)} \iff \\ -\mathbf{Ax}_{(k)} + \mathbf{b} &= -(\mathbf{Ax}_{(k-1)} - \mathbf{b}) - \alpha_{(k)} \mathbf{Ar}_{(k-1)} \iff \\ \mathbf{r}_{(k)} &= \mathbf{r}_{(k-1)} - \alpha_{(k)} \mathbf{Ar}_{(k-1)}. \end{aligned} \quad (3.5)$$

(3.5) 表明我们可以用  $\mathbf{r}_{(k-1)} - \alpha_{(k)} \mathbf{Ar}_{(k-1)}$  代替  $\mathbf{b} - \mathbf{Ax}_{(k)}$  来更新  $\mathbf{r}_{(k)}$ ! 因为  $\mathbf{Ar}_{(k-1)}$  的结果可以复用, 所以这就避免了第 7 行的矩阵-向量乘法。但是, 有得必有失。这种降低复杂度的策略是以牺牲计算  $\mathbf{r}_{(k)}$  的精度为代价的。因为  $\mathbf{r}_{(k)}$  的计算没有用到最新的  $\mathbf{x}_{(k)}$ , 所以会有舍入误差。这个问题可以通过定期重新采用  $\mathbf{b} - \mathbf{Ax}_{(k)}$  更新  $\mathbf{r}_{(k)}$  解决。改进策略的算法描述在 **Algorithm 2** 中。

---

**Algorithm 2:** 改进的梯度法.

---

**Input:** 对称正定矩阵  $\mathbf{A} \in \mathbb{R}^{n \times n}$ , 向量  $\mathbf{b} \in \mathbb{R}^n$ , 停止条件  $\varepsilon$ , 最大迭代次数  $\text{MaxIter}$

**Output:** 线性方程组  $\mathbf{Ax} = \mathbf{b}$  的一个满足精度要求的解

```
1  $\mathbf{r}_{(0)} = \mathbf{b} - \mathbf{Ax}_{(0)}, \delta_{(0)} = \mathbf{r}_{(0)}^\top \mathbf{r}_{(0)}$ 
2  $k = 0$ 
3 while  $k < \text{MaxIter}$  and  $\delta_{(k)} > \varepsilon^2 \delta_{(0)}$  do
4    $\mathbf{q} = \mathbf{Ar}_{(k)}, \alpha_{(k)} = \delta_{(k)} / \mathbf{r}_{(k)}^\top \mathbf{q}$ 
5    $\mathbf{x}_{(k+1)} = \mathbf{x}_{(k)} + \alpha_{(k)} \mathbf{r}_{(k)}$ 
6   if  $k$  is divisible by  $\lceil \sqrt{n} \rceil$  then
7     /* Periodically update by the original formula */ 
8      $\mathbf{r}_{(k+1)} = \mathbf{b} - \mathbf{Ax}_{(k+1)}$ 
9   else
10    /* Reduce complexity according to (3.5) */ 
11     $\mathbf{r}_{(k+1)} = \mathbf{r}_{(k)} - \alpha_{(k)} \mathbf{q}$ 
12   end if
13    $k = k + 1, \delta_{(k)} = \mathbf{r}_{(k)}^\top \mathbf{r}_{(k)}$ 
14 end while
15 return  $\mathbf{x}_{(k)}$ 
```

---

## 3.2 特征值的相关特性

### 3.2.1 谱半径

求解线性方程组的迭代算法通常会反复将一个矩阵作用在一个向量上。因此本章节先谈一谈特征值和特征向量的相关特性，再开展梯度法的收敛性分析。

设  $v$  是矩阵  $B$  的一个特征向量，则存在常数  $\lambda$  使得  $Bv = \lambda v$ 。也就是说，当  $B$  作为一个线性变换作用到向量  $v$  上时，只会让  $v$  在所在方向上发生长度的伸缩或变成反向向量，而不会发生旋转变换。不断地左乘  $B$ ，我们主要关心如下两种不同的情形：

- 若  $|\lambda| < 1$ ，那么  $\lim_{k \rightarrow \infty} B^k v$  会收缩至 0；
- 若  $|\lambda| > 1$ ，那么  $\lim_{k \rightarrow \infty} B^k v$  会趋于  $\infty$ 。

Fig. 3.2 和 Fig. 3.3 分别给出了一个对应的例子。

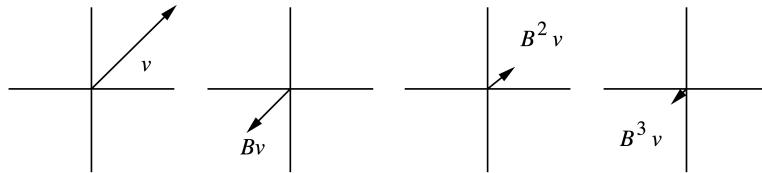


Figure 3.2:  $\lambda = -0.5$  时  $B^k v$  的变化情况。

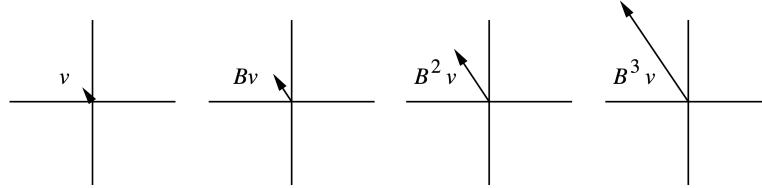


Figure 3.3:  $\lambda = 2$  时  $B^k v$  的变化情况。

如果将向量看成线性空间中的一个点，那么左乘矩阵可以看成是对点施加的一种运动（使点发生位移）。特征值反映了运动的速度，而特征向量则反映了运动的方向。线性变换是各个方向上不同速度的运动的叠加。从这个理解出发，就会想明白为什么对于任意向量  $x$ ，不断左乘  $B$ ， $x$  会不断贴合到  $B$  最大特征值对应的特征空间上。我们知道，特征值分解就是和对角阵互为相似矩阵，即对角化。任意矩阵  $B \in \mathbb{R}^{n \times n}$  可对角化当且仅当该矩阵具有  $n$  个线性无关的特征向量<sup>4</sup>。任意对称矩阵都可以进行正交对角化，即其  $n$  个线性无关的特征向量组成的矩阵是一个正交矩阵。当我们对称矩阵  $B$  作用在任意向量  $x$  上时，可以将  $x$  用  $B$  的  $n$  个线性无关的特征向量  $\{v_1, \dots, v_n\}$  的线性组合表示，即

$$x = \sum_{i=1}^n c_i v_i,$$

那么  $Bx = \sum_{i=1}^n c_i \lambda_i v_i$ ，其中  $\lambda_i$  为  $v_i$  对应的特征值。注意，我们总可以选取合适的特征向量使得  $\forall i, c_i = 1$ 。因此上式可简化为  $x = \sum_{i=1}^n v_i$ 。那么，只要存在一个  $i$  使得  $|\lambda_i| > 1$ ， $B^k x$  的长度就会发散至无穷大。只有当所有特征值的绝对值均小于 1 时， $B^k x$  才能收敛到  $0$ 。可以看出， $B^k x$  能否收敛到  $0$  是由最大的  $|\lambda_i|$  决定的，我们将这个数值定义为  $B$  的谱半径 (spectral radius)：

$$\rho(B) \triangleq \max_i |\lambda_i|. \quad (3.6)$$

<sup>4</sup> 关于线性代数的更多基础内容请参阅 <http://hliangzhao.me/math/math.pdf>，此处将不再给出相关基础理论的证明。

如果  $\mathbf{B}$  不具备  $n$  个线性无关的特征向量，那么我们称  $\mathbf{B}$  是退化矩阵 (defective matrix)。此时无法对角化 (特征值分解)。本文只关心对称正定阵，因此关于退化矩阵的内容不再展开描述。

### 3.2.2 雅各比迭代

我们先前提到过，对于线性方程组问题  $\mathbf{A}\mathbf{x} = \mathbf{b}$ ，一类典型的方法是 Jacobi 算法。Gauss-Seidel 方法和松弛法均是在 Jacobi 算法的基础上的改进版本。本章节我们将深入理解 Jacobi 算法，其中涉及的一些观念将会用于梯度法的收敛性分析。

将  $\mathbf{A}\mathbf{x} = \mathbf{b}$  展开可得

$$x_i = \frac{b_i - \sum_{j \neq i} a_{ij}x_j}{a_{ii}}, \quad (3.7)$$

由此可得 Jacobi 的迭代公式：

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)}}{a_{ii}} \quad (3.8)$$

显然该方法要求  $\forall i, a_{ii} \neq 0$ 。如果在更新  $x_i^{(k+1)}$  的时候对于所有排在其前面的  $x_j$  采用最新的数据  $x_j^{(k+1)}$ ，那么就得到了

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)}}{a_{ii}}, \quad (3.9)$$

这就是 Gauss-Seidel 方法的迭代公式。不出意外地，Gauss-Seidel 方法的收敛速度应当快于 Jacobi 方法。

我们也可以采用矩阵分解的方式表达上述迭代方程。具体地，令  $\mathbf{D} = \text{diag}(\mathbf{A})$ ， $\mathbf{E}$  是  $\mathbf{A}$  严格下三角部分的负组成的下三角矩阵， $\mathbf{F}$  是  $\mathbf{A}$  严格上三角部分的负组成的上三角矩阵，则  $\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F}$ 。所以  $\mathbf{A}\mathbf{x} = \mathbf{b}$  等价于

$$\begin{aligned} (\mathbf{D} - \mathbf{E} - \mathbf{F})\mathbf{x} &= \mathbf{b} \\ \mathbf{D}^{-1}(\mathbf{D} - \mathbf{E} - \mathbf{F})\mathbf{x} &= \mathbf{D}^{-1}\mathbf{b} \\ \mathbf{x} &= \mathbf{D}^{-1}(\mathbf{E} + \mathbf{F})\mathbf{x} + \mathbf{D}^{-1}\mathbf{b}. \end{aligned}$$

定义  $\mathbf{B} \triangleq \mathbf{D}^{-1}(\mathbf{E} + \mathbf{F})$ ， $\mathbf{z} \triangleq \mathbf{D}^{-1}\mathbf{b}$ ，则 Jacobi 迭代公式(3.8)可写作

$$\mathbf{x}_{(k+1)} = \mathbf{B}\mathbf{x}_{(k)} + \mathbf{z}. \quad (3.10)$$

我们将  $\mathbf{A}\mathbf{x} = \mathbf{b}$  的真值记为  $\mathbf{x}^*$ ，并称之为驻点 (stationary point)。第  $i$  轮迭代中的  $\mathbf{x}_{(i)}$  可以用真值  $\mathbf{x}^*$  和误差向量  $\mathbf{e}_{(i)}$  之和来表示，那么根据(3.10)可得

$$\begin{aligned} \mathbf{x}_{(i+1)} &= \mathbf{B}\mathbf{x}_{(i)} + \mathbf{z} \\ &= \mathbf{B}(\mathbf{x}^* + \mathbf{e}_{(i)}) + \mathbf{z} \\ &= \mathbf{B}\mathbf{x}^* + \mathbf{z} + \mathbf{B}\mathbf{e}_{(i)} \quad \triangleright \mathbf{x}^* = \mathbf{B}\mathbf{x}^* + \mathbf{z} \text{ 恒成立} \\ &= \mathbf{x}^* + \mathbf{B}\mathbf{e}_{(i)} \end{aligned}$$

所以  $\mathbf{e}_{(i+1)} = \mathbf{x}_{(i+1)} - \mathbf{x}^* = \mathbf{B}\mathbf{e}_{(i)}$ 。这表明 Jacobi 方法是通过不断减小误差项来向真值逼近的。若  $\rho(\mathbf{B}) < 1$ ，则误差项可以收敛到  $\mathbf{0}$ ，Jacobi 方法可以解出  $\mathbf{x}^*$ 。同时我们也可以发现，初始向量  $\mathbf{x}_{(0)}$  虽然不影响最终的结果，但却会影响到收敛到给定误差范围内需要迭代的次数。Jacobi 的方法的劣势在于： $\mathbf{B}$  并不总是对称阵，甚至可能是退化矩阵，因此 Jacobi 方法并非对于所有的  $\mathbf{A}$  都能收敛，即使  $\mathbf{A}$  是正定矩阵也无法保证收敛。

### 3.3 收敛性分析

#### 3.3.1 简单情形

为了分析梯度法的收敛速度，我们首先考虑一种简单的情形，即误差向量  $e_{(i)}$  是  $\mathbf{A}$  的特征值  $\lambda_e$  所对应的特征向量。那么残差向量  $r_{(i)} = -\mathbf{A}e_{(i)} = -\lambda_e e_{(i)}$ 。显然

$$\mathbf{A}r_{(i)} = -\lambda_e \mathbf{A}e_{(i)} = \lambda_e r_{(i)}, \quad (3.11)$$

所以  $r_{(i)}$  也是  $\mathbf{A}$  的特征值  $\lambda_e$  所对应的特征向量。

根据(3.4)可得最优步长  $\alpha$  满足

$$\begin{aligned} \alpha &= \frac{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}}{\mathbf{r}_{(i)}^T \mathbf{A} \mathbf{r}_{(i)}} = \frac{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}}{\mathbf{r}_{(i)}^T \lambda_e \mathbf{r}_{(i)}} \quad \triangleright (3.11) \\ &= \frac{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}}{\lambda_e \mathbf{r}_{(i)}^T \mathbf{r}_{(i)}} = \frac{1}{\lambda_e}. \end{aligned} \quad (3.12)$$

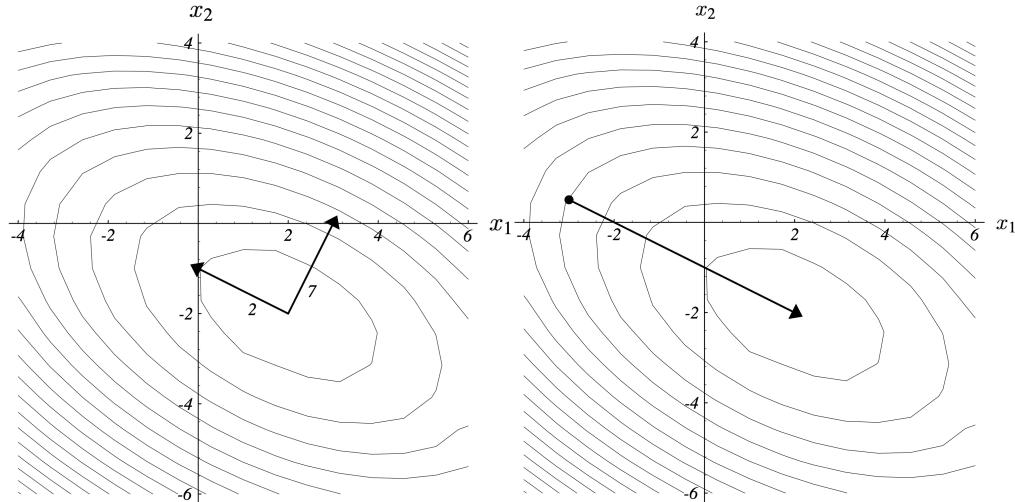
又由(3.1)可得  $e_{(i+1)} + x^* = e_{(i)} + x^* + \alpha \cdot r_{(i)}$ , 因此

$$e_{(i+1)} = e_{(i)} + \alpha \cdot r_{(i)}. \quad (3.13)$$

从而

$$\begin{aligned} e_{(i+1)} &= e_{(i)} + \frac{1}{\lambda_e} r_{(i)} \quad \triangleright (3.12) \\ &= e_{(i)} - \frac{1}{\lambda_e} \lambda_e e_{(i)} \\ &= \mathbf{0}. \end{aligned}$$

这说明在该条件下梯度法只需要一次迭代即可抵达  $x^*$ ! Fig. 3.4对该结论给出了一个形象的解释。



**Figure 3.4:**  $\mathbf{A} = [[3, 2], [2, 6]]$ 。左图:  $\mathbf{A}$  的两个特征向量分别为  $v_1 = [1, 2]^T$ ,  $v_2 = [-2, 1]^T$ 。对应地,  $\lambda_1 = 7$ ,  $\lambda_2 = 2$ ; 右图: 误差向量 (迭代方向) 为特征向量, 一次迭代即可收敛。

当误差向量  $e_{(i)}$  不是  $\mathbf{A}$  的特征值  $\lambda_e$  所对应的特征向量时, 梯度法的收敛性又如何呢? 假设  $\mathbf{A}$  是对称阵, 则  $\mathbf{A}$  可进行正交对角化:

$$\mathbf{A} = \mathbf{P} \mathbf{D} \mathbf{P}^{-1} = \mathbf{P} \mathbf{D} \mathbf{P}^T, \quad (3.14)$$

其中  $\mathbf{P} = \{v_1, \dots, v_n\}$  的每一列是相互单位正交的特征向量 (显然  $\{v_1, \dots, v_n\}$  线性无关),  $\mathbf{D} =$

$\text{diag}(\lambda_1, \dots, \lambda_n)$  是一个由  $n$  个实特征值（包含重根）组成的对角阵。不妨将误差向量  $e_{(i)}$  用特征向量的线性组合表示：

$$e_{(i)} = \sum_{i=1}^n \xi_i v_i, \quad (3.15)$$

则  $\|e_{(i)}\|^2 = \sum_{i=1}^n \xi_i^2$ , 且  $r_{(i)} = -\mathbf{A}e_{(i)} = -\sum_{i=1}^n \xi_i \lambda_i v_i$ , 这表明残差向量也可以表达为特征向量的线性组合。所以

$$e_{(i)}^T \mathbf{A} e_{(i)} = \left( \sum_{i=1}^n \xi_i v_i^T \right) \left( \sum_{i=1}^n \xi_i \lambda_i v_i \right) = \sum_{i=1}^n \xi_i^2 \lambda_i, \quad (3.16)$$

$$\begin{aligned} r_{(i)}^T \mathbf{A} r_{(i)} &= \left( \sum_{i=1}^n \xi_i \lambda_i v_i^T \right) \mathbf{A} \left( \sum_{i=1}^n \xi_i \lambda_i v_i \right) = \left( \sum_{i=1}^n \xi_i \lambda_i v_i \right) \left( \sum_{i=1}^n \xi_i \lambda_i^2 v_i \right) \\ &= \sum_{i=1}^n \xi_i^2 \lambda_i^3. \end{aligned} \quad (3.17)$$

由  $e_{(i+1)} = e_{(i)} + \alpha \cdot r_{(i)}$  可得

$$\begin{aligned} e_{(i+1)} &= e_{(i)} + \frac{r_{(i)}^T r_{(i)}}{r_{(i)}^T \mathbf{A} r_{(i)}} r_{(i)} \\ &= e_{(i)} + \frac{\sum_{i=1}^n \xi_i^2 \lambda_i^2}{\sum_{i=1}^n \xi_i^2 \lambda_i^3} r_{(i)} \end{aligned} \quad (3.18)$$

请注意，到目前为止所推导出的结论均是基于  $\mathbf{A}$  是对称阵这一假设的。若所有特征向量的特征值均相同且为  $\lambda$ , 则(3.18)变成了

$$\begin{aligned} e_{(i+1)} &= e_{(i)} + \frac{\lambda^2 \sum_{i=1}^n \xi_i^2}{\lambda^3 \sum_{i=1}^n \xi_i^2} (-\mathbf{A}e_{(i)}) \\ &= e_{(i)} + \frac{\lambda^2 \sum_{i=1}^n \xi_i^2}{\lambda^3 \sum_{i=1}^n \xi_i^2} (-\lambda e_{(i)}) \\ &= \mathbf{0}. \end{aligned} \quad (3.19)$$

上述推导用到了  $\mathbf{A}e_{(i)} = \mathbf{A} \sum_{i=1}^n \xi_i v_i = \sum_{i=1}^n \xi_i \mathbf{A} v_i = \sum_{i=1}^n \xi_i \lambda v_i = \lambda e_{(i)}$ 。这意味着当  $\mathbf{A}$  为对称阵且所有特征值相同时, 梯度法依然可以做到一步迭代即收敛。本质上, 这是因为  $f(\mathbf{x})$  从椭球体的表面变成了球体的表面, 任意点的梯度都经过球心。

让我们来总结一下本章节得出的结论:

- 对于任意  $n$  阶方阵  $\mathbf{A}$ , 若误差向量  $e_{(i)}$  是  $\mathbf{A}$  的特征值  $\lambda_e$  所对应的特征向量, 那么梯度法一步迭代即收敛;
- 若  $\mathbf{A}$  是对称阵且所有特征值相同, 则梯度法依然可以做到一步迭代即收敛。

### 3.3.2 一般情形

当误差向量为任意向量, 而此时又存在多个不相等的非零特征值时, 梯度法的收敛性如何分析呢? 为了解决这个问题, 我们需要定义误差向量的能量范数 (energy norm):

$$\|e\|_{\mathbf{A}} \triangleq (e^T \mathbf{A} e)^{\frac{1}{2}} \quad (3.20)$$

从 Fig. 3.5可以看出, 从点  $\mathbf{x}^*$  出发的两个向量如果能量范数相同, 那么它们的终点在  $f(\mathbf{x})$  的同一条等势线上。

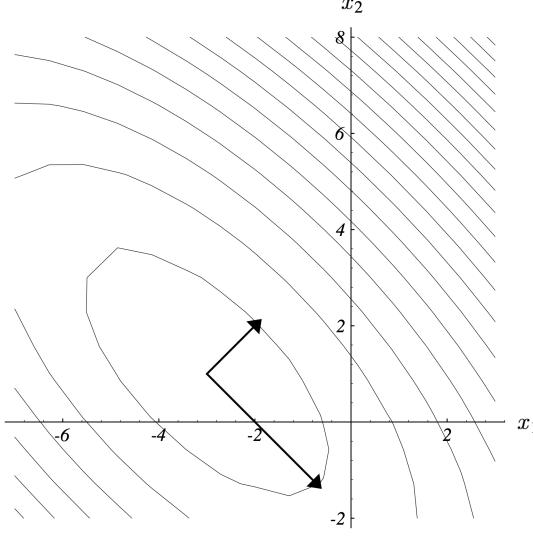


Figure 3.5: 能量范数相同的两个向量。

根据能量范数的定义可得

$$\begin{aligned}
 \|e_{(i+1)}\|_A^2 &= e_{(i+1)}^T A e_{(i+1)} \quad \triangleright (3.13) \\
 &= (e_{(i)}^T + \alpha \cdot r_{(i)}^T) A (e_{(i)} + \alpha \cdot r_{(i)}) \\
 &= e_{(i)}^T A e_{(i)} + 2\alpha r_{(i)}^T A e_{(i)} + \alpha^2 r_{(i)}^T A r_{(i)} \quad \triangleright (3.4) \\
 &= \|e_{(i)}\|_A^2 + 2 \frac{r_{(i)}^T r_{(i)}}{r_{(i)}^T A r_{(i)}} (-r_{(i)}^T r_{(i)}) + \left( \frac{r_{(i)}^T r_{(i)}}{r_{(i)}^T A r_{(i)}} \right)^2 r_{(i)}^T A r_{(i)} \\
 &= \|e_{(i)}\|_A^2 - \frac{(r_{(i)}^T r_{(i)})^2}{r_{(i)}^T A r_{(i)}} \\
 &= \|e_{(i)}\|_A^2 \left( 1 - \frac{(r_{(i)}^T r_{(i)})^2}{(r_{(i)}^T A r_{(i)})(e_{(i)}^T A e_{(i)})} \right) \\
 &= \|e_{(i)}\|_A^2 \left( 1 - \frac{(\sum_{i=1}^n \xi_i^2 \lambda_i^2)^2}{(\sum_{i=1}^n \xi_i^2 \lambda_i)(\sum_{i=1}^n \xi_i^2 \lambda_i^3)} \right) \quad \triangleright (3.16) \& (3.17) \\
 &= \|e_{(i)}\|_A^2 \omega^2 \quad \triangleright \omega^2 \triangleq 1 - \frac{(\sum_{i=1}^n \xi_i^2 \lambda_i^2)^2}{(\sum_{i=1}^n \xi_i^2 \lambda_i)(\sum_{i=1}^n \xi_i^2 \lambda_i^3)} \quad (3.21)
 \end{aligned}$$

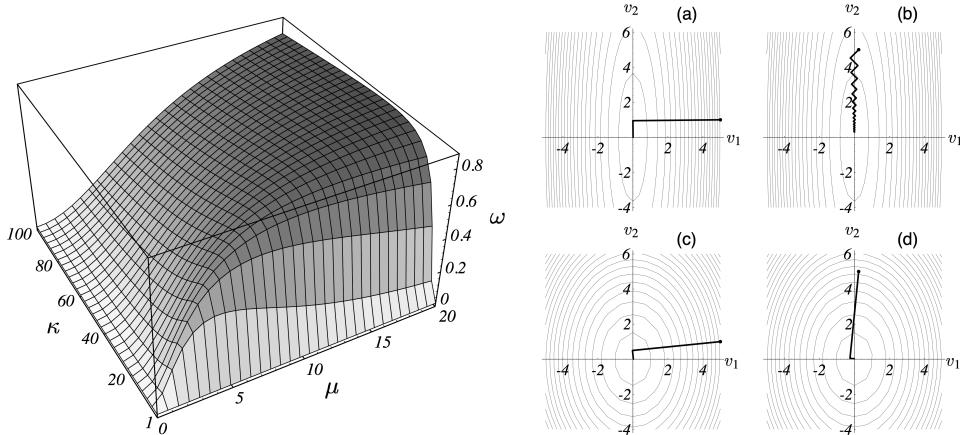
我们只要分析  $\omega^2$  的上界，就可以知道梯度法在最坏情况下的表现了。 $\omega$  越小，收敛速度越快。至于原因，此处暂时按下不表，后文会给出解释。我们先针对一个简单的情况： $A \in \mathbb{R}^{2 \times 2}$  是个正定对称阵进行分析。前文已经说明说明过，当  $A$  是正定矩阵时，其特征值  $\lambda_1, \lambda_2$  均为正数，且特征向量  $v_1, v_2$  线性无关。因此不妨设  $\lambda_1 > \lambda_2$ ，此时  $A$  的谱条件数（spectral condition number） $\kappa = \lambda_1/\lambda_2 \geq 1$ 。请回顾(3.15)<sup>5</sup>，我们将误差向量  $e_{(i)}$  相对于特征向量张成的空间/坐标系的斜率定义为  $\mu \triangleq \xi_2/\xi_1$ ，

<sup>5</sup> 在(3.15)中，我们将  $v_i$  视为基，相应地， $\xi_i$  是对应基上的坐标。将  $v_1$  视为  $x$  轴， $v_2$  视为  $y$  轴，即可类比得到斜率  $\mu$ 。

由此可对  $\omega^2$  作如下化简：

$$\begin{aligned}\omega^2 &= 1 - \frac{\left(\frac{\xi_1^2 \lambda_1^2 + \xi_2^2 \lambda_2^2}{\xi_1^2 \lambda_2^2}\right)^2}{\left(\frac{\sum_{i=1}^2 \xi_i^2 \lambda_i}{\xi_1^2 \lambda_2}\right) \left(\frac{\sum_{i=1}^2 \xi_i^2 \lambda_i^3}{\xi_1^2 \lambda_2^3}\right)} \\ &= 1 - \frac{(\kappa^2 + \mu^2)^2}{(\kappa + \mu^2)(\kappa^3 + \mu^2)}. \end{aligned} \quad (3.22)$$

上式表明我们可以将  $\omega$  看成  $\kappa$  和  $\mu$  的函数（只考虑正数部分）。观察 Fig. 3.6 左图可发现  $\kappa$  和  $\mu$  越小， $\omega$  越小。此外， $\mu = \pm \kappa$  时  $\omega$  最大（这个结论从图像上看很直观，也可以通过对(3.22)求导得到）。分析这个图像也可以理解梯度法是如何做到一步迭代即收敛的。当  $e_{(i)}$  是一个特征向量时，此时误差向量在特征向量组成的坐标系的坐标轴上，因此斜率  $\mu$  为 0 或  $\infty$ ，此时均有  $\omega = 0$ ；当  $\mathbf{A}$  的所有特征值均相等时， $\kappa = 1$ ，同样有  $\omega = 0$ 。此外，我们可以观察一下  $\omega$  随  $\kappa$  和  $\mu$  的变化情况。当  $\kappa$  很小的时候，不论  $\mu$  如何，不论初始点如何选取，收敛速度都是比较快的，这是因为  $\mathbf{A}$  各个方向的特征值相似，所以  $f(\mathbf{x})$  接近球形，在各个点处均有合适的迭代方向；当  $\kappa$  很大的时候，不同的特征向量的能量范数差距很大，此时  $f(\mathbf{x})$  的一极相对另一极长很多（想象一个  $a$  和  $b$  差异很大的椭圆  $x^2/a^2 + y^2/b^2 = 1$ ）。如果初始点在山脊处，那么迭代方向还是很好找的，收敛速度不算慢；如果初始点在山谷处，那么梯度法只能朝着波谷方向曲折而缓慢的前进，每一步迭代作出的有效改进很小，收敛速度很慢。Fig. 3.6 右图给四种情形下梯度法的迭代情况做了一个形象化的展示。



**Figure 3.6:** 左图： $\omega$  关于  $\kappa$  和  $\mu$  的函数图像；右图：四种情形下梯度法的迭代情况：(a)  $\kappa$  大， $\mu$  小；(b)  $\kappa$  大， $\mu$  大；(c)  $\kappa$  小， $\mu$  小；(d)  $\kappa$  小， $\mu$  大。

我们再来算一算最差的情况下，即  $\mu = \pm \kappa$  时  $\omega$  的上界是多少。对于对称正定阵  $\mathbf{A}$ ，我们将条件数（condition number）定义为特征值之比的最大值：

$$\kappa \triangleq \frac{\lambda_{max}}{\lambda_{min}}. \quad (3.23)$$

当  $\mu = \pm \kappa$  时有  $\mu^2 = \kappa^2$ ，因此(3.22)满足

$$\omega^2 \leq 1 - \frac{4\kappa^4}{\kappa^5 + 2\kappa^4 + \kappa^3} \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^2,$$

进而得到

$$\omega \leq \frac{\kappa - 1}{\kappa + 1}. \quad (3.24)$$

该结论在  $n \geq 2$  时依然成立。将(3.24)绘制出来可得

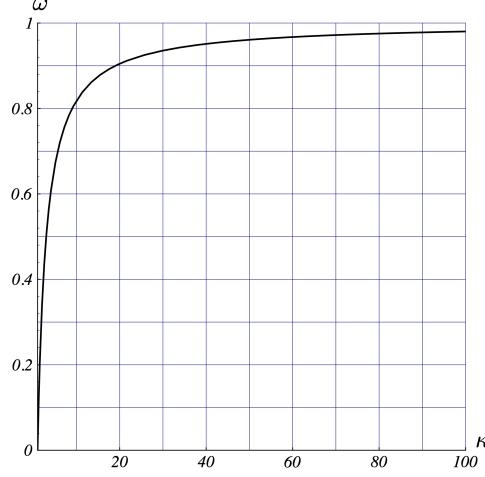


Figure 3.7: 梯度法中, 当  $\mathbf{A}$  为对称正定阵时, 收敛系数  $\omega$  关于条件数  $\kappa$  的函数图像。

现在我来回答一下前文埋下的一个问题。即, 梯度法的收敛速度和  $\omega$  之间的关系: 因为

$$\frac{f(\mathbf{x}_{(i)}) - f(\mathbf{x}^*)}{f(\mathbf{x}_{(0)}) - f(\mathbf{x}^*)} = \frac{\frac{1}{2}\mathbf{e}_{(i)}^T \mathbf{A} \mathbf{e}_{(i)}}{\frac{1}{2}\mathbf{e}_{(0)}^T \mathbf{A} \mathbf{e}_{(0)}} = \frac{\|\mathbf{e}_{(i)}\|_{\mathbf{A}}^2}{\|\mathbf{e}_{(0)}\|_{\mathbf{A}}^2} = \omega^2, \quad (3.25)$$

所以  $\omega$  越大, 收敛速度就越慢。换句话说,  $\|\mathbf{e}_{(i)}\|_{\mathbf{A}}^2$  反应的是第  $i$  轮迭代时当前解和最优解之间在优化目标上的差距。结合上式和(3.24)可得

$$\|\mathbf{e}_{(i)}\|_{\mathbf{A}} \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^i \|\mathbf{e}_{(0)}\|_{\mathbf{A}}, \quad (3.26)$$

所以

$$\frac{f(\mathbf{x}_{(i)}) - f(\mathbf{x}^*)}{f(\mathbf{x}_{(0)}) - f(\mathbf{x}^*)} \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^{2i}. \quad (3.27)$$

这就是当  $\mathbf{A}$  为对称正定阵时梯度法的最坏收敛速率。

## 4 共轭方向法

### 4.1 什么是共轭

观察 Fig. 3.6右图 (b) 可发现在梯度法中, 同一个迭代方向可能会反复出现。这意味着, 对于重复出现的方向, 对应的迭代没有完全消除误差向量在这个方向上的分量。如果我们能找到  $n$  个相互正交的搜索方向  $\mathbf{d}_{(0)}, \mathbf{d}_{(1)}, \dots, \mathbf{d}_{(n-1)}$ , 在每个方向上只移动一步即可完美消除误差向量在这个方向上的分量, 那么只需要  $n$  轮迭代即可收敛到全局最优, 复杂度将会大大降低。为什么是  $n$  而不是别的数字呢? 请回顾(3.15), 稍加思索你就会明白。

和梯度法一样, 在共轭方向法中,  $\mathbf{x}_{(i)}$  将会按照如下的迭代式进行更新:

$$\mathbf{x}_{(i+1)} \leftarrow \mathbf{x}_{(i)} + \alpha_{(i)} \mathbf{d}_{(i)}. \quad (4.1)$$

那么问题来了, 如何按照上述思想求解出  $\alpha_{(i)}$  和  $\mathbf{d}_{(i)}$  呢? 本质上, 完全消除误差向量在各个搜索方向的分量, 等价于

$$-\mathbf{e}_{(0)} = \mathbf{x}^* - \mathbf{x}_{(0)} = \alpha_{(0)} \cdot \mathbf{d}_{(0)} + \dots + \alpha_{(n-1)} \cdot \mathbf{d}_{(n-1)}, \quad (4.2)$$

即“反向的初始误差向量  $-\mathbf{e}_{(0)}$ ”可由各个搜索方向的线性组合表示, 各搜索方向上的系数就是该方

向上的搜索步长。由此可知第  $i+1$  轮时的误差向量可表示为

$$\begin{aligned}\mathbf{e}_{(i+1)} &= \mathbf{e}_{(0)} + \sum_{k=0}^i \alpha_{(k)} \mathbf{d}_{(k)} \\ &= - \sum_{k=i+1}^{n-1} \alpha_{(k)} \mathbf{d}_{(k)},\end{aligned}\quad (4.3)$$

所以

$$\begin{aligned}\mathbf{d}_{(i)} \cdot \mathbf{e}_{(i+1)} &= - \sum_{k=i+1}^n \alpha_{(k)} \mathbf{d}_{(i)}^\top \mathbf{d}_{(k)} \quad \triangleright \mathbf{d}_{(i)}^\top \mathbf{d}_{(j)} = 0 \text{ if } i \neq j \\ &= 0.\end{aligned}\quad (4.4)$$

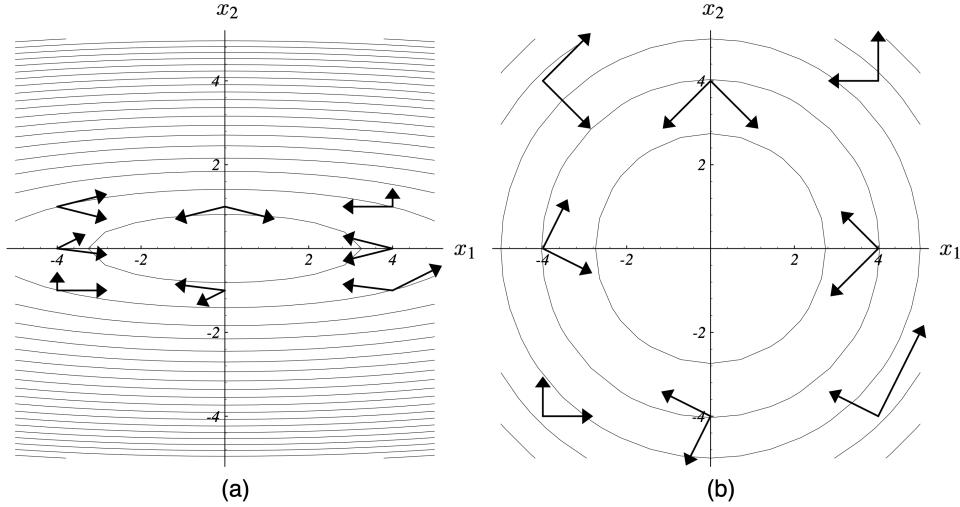
这表明本轮的误差向量和上一轮的搜索方向必然正交。从而

$$\begin{aligned}\mathbf{d}_{(i)}^\top (\mathbf{e}_{(i)} + \alpha_{(i)} \mathbf{d}_{(i)}) &= 0 \implies \\ \alpha_{(i)} &= - \frac{\mathbf{d}_{(i)}^\top \mathbf{e}_{(i)}}{\mathbf{d}_{(i)}^\top \mathbf{d}_{(i)}}.\end{aligned}\quad (4.5)$$

(4.5)给出了每一轮最优的迭代步长，这其实就是“反向误差向量  $-\mathbf{e}_{(i)}$ ”在第  $i$  个搜索方向上投影的坐标值。但是， $\mathbf{e}_{(i)}$  是未知的，即使给定了  $\mathbf{d}_{(i)}$ ，我们仍不知道  $\alpha_{(i)}$  该如何计算。解决办法是把找到  $n$  个相互正交的搜索方向  $\mathbf{d}_{(0)}, \mathbf{d}_{(1)}, \dots, \mathbf{d}_{(n-1)}$  转变为找到  $n$  个相互  $\mathbf{A}$ -正交的搜索方向  $\mathbf{d}_{(0)}, \mathbf{d}_{(1)}, \dots, \mathbf{d}_{(n-1)}$ 。其中， $\mathbf{A}$ -正交 ( $\mathbf{A}$ -orthogonal) 的定义为：若两个向量  $\mathbf{p}$  和  $\mathbf{q}$  满足

$$\mathbf{p}^\top \mathbf{A} \mathbf{q} = 0, \quad (4.6)$$

则称  $\mathbf{p}$  和  $\mathbf{q}$  是  $\mathbf{A}$ -正交的。在这个新的思路下，我们可以真正地计算出  $\alpha_{(i)}$ 。不过，在此之前，我们先深入理解一下  $\mathbf{A}$ -正交这个概念。如 Fig. 4.1 所示，(a) 是一般的  $\mathbf{A}$ -正交的向量对的样子，显然这些向量对并不正交。当  $\mathbf{A}$  是对称阵且所有特征值都相同时，通过特征值分解可发现  $\mathbf{p}$  和  $\mathbf{q}$  正交，如 (b) 所示。从 (a) 到 (b) 可以理解为把能量范数小的特征向量拉伸到和能量范数大的特征向量一样长。



**Figure 4.1:** 左图： $\mathbf{A}$ -正交的向量对；右图：变换过后的、 $\mathbf{A}$ -正交的向量对。

这类通过相互  $\mathbf{A}$ -正交的搜索方向来找到最优解的方法就被统称为共轭方向法 (conjugate direc-

tions methods)。把上面的数学分析搬过来，可以得到和(4.4)类似的结论：

$$\mathbf{d}_{(i)}^T \mathbf{A} \mathbf{e}_{(i+1)} = 0. \quad (4.7)$$

同样地，最优步长的结果也和(4.5)类似：

$$\begin{aligned} \alpha_{(i)} &= -\frac{\mathbf{d}_{(i)}^T \mathbf{A} \mathbf{e}_{(i)}}{\mathbf{d}_{(i)}^T \mathbf{A} \mathbf{d}_{(i)}} \quad \triangleright \mathbf{r}_{(i)} = -\mathbf{A} \mathbf{e}_{(i)} \\ &= \frac{\mathbf{d}_{(i)}^T \mathbf{r}_{(i)}}{\mathbf{d}_{(i)}^T \mathbf{A} \mathbf{d}_{(i)}} \end{aligned} \quad (4.8)$$

和(4.5)不一样的是，只要搜索方向  $\mathbf{d}_{(i)}$  定了，(4.8)是可以直接计算的。在继续分析共轭方向法之前，我们先看看(4.7)意味着什么。在梯度法中，最优步长的选取是根据(3.2)来执行的。类似地，当我们采用(4.1)来更新  $\mathbf{x}$  的时候，就可以根据

$$\begin{aligned} \nabla_\alpha f(\mathbf{x}_{(i+1)}) &= (\nabla_{\mathbf{x}_{(i+1)}} f(\mathbf{x}_{(i+1)}))^T \nabla_\alpha \mathbf{x}_{(i+1)} \\ &= -\mathbf{r}_{(i+1)}^T \mathbf{d}_{(i)} \\ &= -(\mathbf{d}_{(i)}^T \mathbf{r}_{(i+1)})^T \\ &= (\mathbf{d}_{(i)}^T \mathbf{A} \mathbf{e}_{(i+1)})^T \\ &= 0 \end{aligned}$$

得到(4.7)! 这意味着梯度法中沿着搜索方向前进最优步长的过程就是对  $\mathbf{A}$ -正交性的运用。如果在每轮迭代中我们就将  $\mathbf{d}_{(i)}$  设置为  $\mathbf{r}_{(i)}$ ，那么共轭方向法就退化成了梯度法。

和(3.15)类似，我们采用搜索方向的线性组合来表示初始误差向量  $\mathbf{e}_{(0)}$ ：

$$\mathbf{e}_{(0)} = \sum_{i=0}^{n-1} \delta_i \mathbf{d}_{(i)}. \quad (4.9)$$

因为

$$\begin{aligned} \mathbf{d}_{(j)}^T \mathbf{A} \mathbf{e}_{(0)} &= \sum_{i=0}^{n-1} \delta_i \mathbf{d}_{(j)}^T \mathbf{A} \mathbf{d}_{(i)} \quad \triangleright \mathbf{d}_{(i)}^T \mathbf{A} \mathbf{d}_{(j)} = 0 \text{ if } i \neq j \\ &= \delta_j \mathbf{d}_{(j)}^T \mathbf{A} \mathbf{d}_{(j)}, \end{aligned}$$

所以可得

$$\begin{aligned} \delta_j &= \frac{\mathbf{d}_{(j)}^T \mathbf{A} \mathbf{e}_{(0)}}{\mathbf{d}_{(j)}^T \mathbf{A} \mathbf{d}_{(j)}} \\ &= \frac{\mathbf{d}_{(j)}^T \mathbf{A} \mathbf{e}_{(0)} + \sum_{i=0}^{j-1} \alpha_{(i)} \mathbf{d}_{(j)}^T \mathbf{A} \mathbf{d}_{(i)}}{\mathbf{d}_{(j)}^T \mathbf{A} \mathbf{d}_{(j)}} \quad \triangleright \mathbf{d}_{(i)}^T \mathbf{A} \mathbf{d}_{(j)} = 0 \text{ if } i \neq j \\ &= \frac{\mathbf{d}_{(j)}^T \mathbf{A} (\mathbf{e}_{(0)} + \sum_{i=0}^{j-1} \alpha_{(i)} \mathbf{d}_{(i)})}{\mathbf{d}_{(j)}^T \mathbf{A} \mathbf{d}_{(j)}} \quad \triangleright \mathbf{e}_{(j)} = \mathbf{e}_{(0)} + \sum_{i=0}^{j-1} \alpha_{(i)} \mathbf{d}_{(i)} \\ &= \frac{\mathbf{d}_{(j)}^T \mathbf{A} \mathbf{e}_{(j)}}{\mathbf{d}_{(j)}^T \mathbf{A} \mathbf{d}_{(j)}} = -\frac{\mathbf{d}_{(j)}^T \mathbf{r}_{(j)}}{\mathbf{d}_{(j)}^T \mathbf{A} \mathbf{d}_{(j)}} \\ &= -\alpha_{(j)}. \quad \triangleright (4.8) \end{aligned} \quad (4.10)$$

因此可以得到最优步长的等价计算公式： $\alpha_{(i)} = -\delta_i$ 。据此，我们可以对每一轮的误差向量  $\mathbf{e}_{(i)}$  作如

下等价变换：

$$\begin{aligned}
\mathbf{e}_{(i)} &= \mathbf{e}_{(0)} + \sum_{j=0}^{i-1} \alpha_{(j)} \mathbf{d}_{(j)} \quad \triangleright (4.3) \\
&= \sum_{j=0}^{n-1} \delta_j \mathbf{d}_{(j)} - \sum_{j=0}^{i-1} \delta_j \mathbf{d}_{(j)} \quad \triangleright (4.9) \& (4.10) \\
&= \sum_{j=i}^{n-1} \delta_j \mathbf{d}_{(j)}. \tag{4.11}
\end{aligned}$$

这个公式表明共轭方向法的每轮迭代是在逐项消除误差向量的分量。这和本章节开头部分的分析完全一致。

## 4.2 共轭格莱姆-施密特过程

现在，只要每轮的迭代方向  $\mathbf{d}_{(i)}$  确定，我们就能根据(4.1)和(4.8)来执行共轭方向法了。本章节将会采用共轭格莱姆-施密特过程（conjugate Gram-Schmidt process）来确定这些相互  $\mathbf{A}$ -正交的迭代方向。

在谈及共轭格莱姆-施密特过程之前，我们先来谈谈什么是格莱姆-施密特过程（Gram-Schmidt process）。格莱姆-施密特过程是一个从任意一组线性无关的基  $\{\mathbf{u}_{(0)}, \dots, \mathbf{u}_{(n-1)}\}$  中构造出一组对应的（单位）正交基  $\{\mathbf{q}_{(0)}, \dots, \mathbf{q}_{(n-1)}\}$  的过程。具体地，

$$\left\{
\begin{array}{l}
\mathbf{q}_{(0)} = \mathbf{u}_{(0)}, \\
\mathbf{q}_{(1)} = \mathbf{u}_{(1)} - \frac{\mathbf{u}_{(1)} \cdot \mathbf{q}_{(0)}}{\mathbf{q}_{(0)} \cdot \mathbf{q}_{(0)}} \mathbf{q}_{(0)}, \\
\vdots \\
\mathbf{q}_{(n-1)} = \mathbf{u}_{(n-1)} - \sum_{i=0}^{n-2} \left( \frac{\mathbf{u}_{(n-1)} \cdot \mathbf{q}_{(i)}}{\mathbf{q}_{(i)} \cdot \mathbf{q}_{(i)}} \mathbf{q}_{(i)} \right).
\end{array}
\right. \tag{4.12}$$

以  $\mathbf{q}_{(1)}$  的计算为例，以上过程就是将  $\mathbf{u}_{(1)}$  进行正交分解，其中一个分量在  $\mathbf{q}_{(0)}$  上，且系数为  $\frac{\mathbf{u}_{(1)} \cdot \mathbf{q}_{(0)}}{\mathbf{q}_{(0)} \cdot \mathbf{q}_{(0)}}$ （可通过  $\mathbf{u}_{(1)}$  在  $\mathbf{q}_{(0)}$  上的投影得到），另一个分量就是  $\mathbf{q}_{(1)}$ ，这样就能保证  $\mathbf{q}_{(1)}$  和  $\mathbf{q}_{(0)}$  是正交的。对于后续的  $\mathbf{u}_{(i)}$  而言，在执行分解的时候，让  $i$  个分量分别在  $\mathbf{q}_{(0)}, \dots, \mathbf{q}_{(i-1)}$  上，最后剩下的就是  $\mathbf{q}_{(i)}$ 。如果要构造的是单位正交的一组基，那么  $\forall i, \mathbf{q}_{(i)}$  按照如下方式计算：

$$\mathbf{q}_{(i)} = \mathbf{p}_{(i)} - \sum_{k=0}^{i-1} (\mathbf{p}_{(i)} \cdot \mathbf{q}_{(k)}) \mathbf{q}_{(k)}, \quad \mathbf{q}_{(i)} \leftarrow \frac{\mathbf{q}_{(i)}}{\|\mathbf{q}_{(i)}\|}. \tag{4.13}$$

把上述过程中的正交替换为  $\mathbf{A}$ -正交，即可得到共轭格莱姆-施密特过程。将  $\mathbf{q}_{(i)}$  替换为  $\mathbf{d}_{(i)}$ ，即可得迭代方向的计算公式：

$$\forall i : \mathbf{d}_{(i)} = \mathbf{p}_{(i)} - \sum_{k=0}^{i-1} \left( \frac{\mathbf{p}_{(i)}^T \mathbf{A} \mathbf{d}_{(k)}}{\mathbf{d}_{(k)}^T \mathbf{A} \mathbf{d}_{(k)}} \mathbf{d}_{(k)} \right). \tag{4.14}$$

我们将系数部分  $-\frac{\mathbf{p}_{(i)}^T \mathbf{A} \mathbf{d}_{(k)}}{\mathbf{d}_{(k)}^T \mathbf{A} \mathbf{d}_{(k)}}$  定义为  $\beta_{ik}$ 。以上结论是直接套用格莱姆-施密特过程得到的，我们当然也可以手动推导出来这个结果。下面将给出推导过程。当  $i = 0$  时，直接取  $\mathbf{d}_{(i)} = \mathbf{p}_{(i)}$  即可。对于之后的每一步 ( $0 < i < n$ )，取

$$\mathbf{d}_{(i)} = \mathbf{p}_{(i)} + \sum_{k=0}^{i-1} \beta_{ik} \mathbf{d}_{(k)}, \tag{4.15}$$

则

$$\begin{aligned}
\mathbf{d}_{(i)}^T \mathbf{A} \mathbf{d}_{(j)} &= \mathbf{p}_{(i)}^T \mathbf{A} \mathbf{d}_{(j)} + \sum_{k=0}^{i-1} \left( \beta_{ik} \mathbf{d}_{(k)}^T \mathbf{A} \mathbf{d}_{(j)} \right) \\
&= \mathbf{p}_{(i)}^T \mathbf{A} \mathbf{d}_{(j)} + \beta_{ij} \mathbf{d}_{(j)}^T \mathbf{A} \mathbf{d}_{(j)}, \tag{4.16}
\end{aligned}$$

所以

$$\beta_{ij} = -\frac{\mathbf{p}_{(i)}^T \mathbf{A} \mathbf{d}_{(j)}}{\mathbf{d}_{(j)}^T \mathbf{A} \mathbf{d}_{(j)}}. \quad (4.17)$$

结合(4.15)和(4.17)就得到了(4.14)。

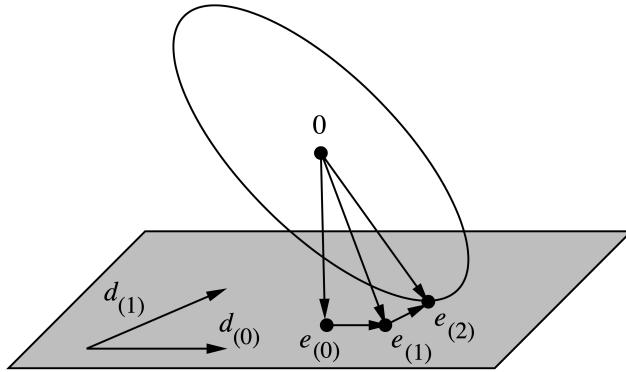
那么  $\mathbf{p}_{(i)}$  如何设置呢？其实有很多选择。在 CG 中， $\mathbf{p}_{(i)} \leftarrow \mathbf{r}_{(i)}$ ，即迭代方向是由残差向量构造出来的！我在后面的章节会解释这样做的优越性。至此，共轭方向法所有理论细节已经准备完毕。不过，在给出算法描述之前，我们先来看看共轭方向法的最优化分析。

### 4.3 最优化分析

在允许的搜索范围内，每一轮迭代中，共轭方向法总可以让误差向量  $\mathbf{e}_{(i)}$  的能量范数  $\|\mathbf{e}_{(i)}\|_{\mathbf{A}}$  最小<sup>6</sup>。那么，什么是允许的搜索范围呢？不妨定义  $\mathcal{D}_i$  为前  $i$  个搜索方向所张成的空间，即  $\mathcal{D}_i \triangleq \text{span}\{\mathbf{d}_{(0)}, \dots, \mathbf{d}_{(i-1)}\}$ ，则根据(4.3)有

$$\mathbf{e}_{(i)} = \mathbf{e}_{(0)} + \sum_{i=0}^{j-1} \alpha_{(i)} \mathbf{d}_{(i)} \in \mathbf{e}_{(0)} + \mathcal{D}_i. \quad (4.18)$$

Fig. 4.2 给了一个切实的案例来帮助理解  $\|\mathbf{e}_{(i)}\|_{\mathbf{A}} (i = 2)$  是如何被最小化的。



**Figure 4.2:** 阴影区域为  $\mathbf{e}_{(0)} + \mathcal{D}_2$ ，椭圆是和该区域相切的  $\|\mathbf{e}_{(i)}\|_{\mathbf{A}}$  的能量范数等高线。共轭方向法可以理解为通过在  $\mathbf{e}_{(0)} + \mathcal{D}_i$  上找到合适的  $\mathbf{e}_{(i)}$  来最小化当前解和最优解之间的差距，即  $\mathbf{e}_{(i)} = \arg\min_{\mathbf{e} \in \mathbf{e}_{(0)} + \mathcal{D}_i} \|\mathbf{e}\|_{\mathbf{A}}$ 。

接下来我们从理论上证明为什么在每一轮迭代中，共轭方向法都能做到最小化  $\|\mathbf{e}_{(i)}\|_{\mathbf{A}}$ 。我们将  $\|\mathbf{e}_{(i)}\|_{\mathbf{A}}^2$  表示成累加的形式：

$$\begin{aligned} \|\mathbf{e}_{(i)}\|_{\mathbf{A}}^2 &= \mathbf{e}_{(i)}^T \mathbf{A} \mathbf{e}_{(i)} \quad \triangleright (4.11) \\ &= \left( \sum_{j=i}^{n-1} \delta_j \mathbf{d}_{(j)} \right)^T \mathbf{A} \left( \sum_{k=i}^{n-1} \delta_k \mathbf{d}_{(k)} \right) \quad \triangleright \mathbf{d}_{(i)}^T \mathbf{A} \mathbf{d}_{(j)} = 0 \text{ if } i \neq j \\ &= \sum_{j=i}^{n-1} (\delta_j \mathbf{d}_{(j)})^T \mathbf{A} (\delta_j \mathbf{d}_{(j)}). \end{aligned} \quad (4.19)$$

仔细观察上式，可发现  $\|\mathbf{e}_{(i)}\|_{\mathbf{A}}^2$  全部由尚未搜索过的方向的能量范数组成。这意味着但凡是搜索过的方向，误差在该方向上的分量都已经被消除。因此这样得到的  $\|\mathbf{e}_{(i)}\|_{\mathbf{A}}$  必然是最小的。至此，最优化已经分析完毕。在给出 CG 之前，我们还有三个推论需要说明。

<sup>6</sup>在 §3.3.2 中，我们说过能量范数反映了第  $i$  轮迭代时当前解和最优解之间在优化目标上的差距。

将(4.11)中的  $i$  替换为  $j$  (取  $j > i$ ), 并在其左右两端分别左乘  $-\mathbf{d}_{(i)}^T \mathbf{A}$  可得

$$\begin{aligned}\mathbf{d}_{(i)}^T \mathbf{r}_{(j)} &= -\mathbf{d}_{(i)}^T \mathbf{A} \mathbf{e}_{(j)} = -\mathbf{d}_{(i)}^T \mathbf{A} \sum_{k=j}^{n-1} \delta_k \mathbf{d}_{(k)} \\ &= -\sum_{k=j}^{n-1} \delta_k \mathbf{d}_{(i)}^T \mathbf{A} \mathbf{d}_{(k)} \quad \triangleright \mathbf{d}_{(i)} \text{ 与 } \mathbf{d}_{(k)} \text{ 相互 } \mathbf{A}-\text{正交} \\ &= 0.\end{aligned}\tag{4.20}$$

这说明在共轭方向法中, 当  $j \in \{i+1, \dots, n-1\}$  时, 残差向量  $\mathbf{r}_{(j)}$  与迭代方向  $\mathbf{d}_{(0)}, \dots, \mathbf{d}_{(i)}$  都正交 【推论 1】。这是因为每当我们沿着某个方向迈出一步之后, 后续过程中就不再会往这个方向走了。因此误差向量与之前的所有搜索方向永远是  $\mathbf{A}$ -正交的。又因为  $\mathbf{r}_{(j)} = -\mathbf{A} \mathbf{e}_{(j)}$ , 故残差向量也必定与已经搜索过的方向正交。

另外, 在(4.15)左右两端点乘  $\mathbf{r}_{(j)}$  (取  $j > i$ ) 可得

$$\begin{aligned}\mathbf{d}_{(i)}^T \mathbf{r}_{(j)} &= \mathbf{p}_{(i)}^T \mathbf{r}_{(j)} + \sum_{k=0}^{i-1} \beta_{ik} \mathbf{d}_{(k)}^T \mathbf{r}_{(j)} \iff \\ 0 &= \mathbf{p}_{(i)}^T \mathbf{r}_{(j)},\end{aligned}\tag{4.21}$$

即在共轭方向法中, 当  $j \in \{i+1, \dots, n-1\}$  时, 残差向量  $\mathbf{r}_{(j)}$  与  $\mathbf{p}_{(0)}, \dots, \mathbf{p}_{(i)}$  都正交 【推论 2】。在(4.15)左右两端点乘  $\mathbf{r}_{(j)}$  (取  $j = i$ ) 可得

$$\mathbf{d}_{(i)}^T \mathbf{r}_{(i)} = \mathbf{p}_{(i)}^T \mathbf{r}_{(i)}. \tag{4.22}$$

这就是 【推论 3】。

最后, 有一个很重要的等价变换会在后文用到, 这里先给出。在 **Algorithm 2** 中, 我们通过(3.5)来减少了一次矩阵-向量乘法运算。这个降低复杂度的操作在共轭方向法中依然适用。具体地, 我们将公式(4.1)按照如下方式变换:

$$\begin{aligned}\mathbf{x}_{(k)} &= \mathbf{x}_{(k-1)} + \alpha_{(k-1)} \cdot \mathbf{d}_{(k-1)} \iff \\ -\mathbf{A} \mathbf{x}_{(k)} + \mathbf{b} &= -(\mathbf{A} \mathbf{x}_{(k-1)} - \mathbf{b}) - \alpha_{(k-1)} \mathbf{A} \mathbf{d}_{(k-1)} \iff \\ \mathbf{r}_{(k)} &= \mathbf{r}_{(k-1)} - \alpha_{(k-1)} \mathbf{A} \mathbf{d}_{(k-1)}.\end{aligned}\tag{4.23}$$

上式表明我们可以用  $\mathbf{r}_{(k-1)} - \alpha_{(k-1)} \mathbf{A} \mathbf{d}_{(k-1)}$  代替  $\mathbf{b} - \mathbf{A} \mathbf{x}_{(k)}$  来更新  $\mathbf{r}_{(k)}$ 。因为  $\mathbf{A} \mathbf{d}_{(k-1)}$  是之前就已经计算过并保存下来的结果, 所以直接复用就可以减少一次矩阵-向量乘法运算。当然, 也会出现舍入误差, 因此需要定期用  $\mathbf{b} - \mathbf{A} \mathbf{x}_{(k)}$  来更新。

## 4.4 共轭梯度法

在 §4.2 中, 我们已经说过, CG 中, 搜索方向  $\mathbf{d}_{(0)}, \dots, \mathbf{d}_{(n-1)}$  是由残差向量  $\mathbf{r}_{(0)}, \dots, \mathbf{r}_{(n-1)}$  根据共轭格莱姆-施密特过程构造出来的。此时, (4.21)变成了

$$\forall i \neq j : \mathbf{r}_{(i)}^T \mathbf{r}_{(j)} = 0. \tag{4.24}$$

在此基础上, 采用(4.8)计算迭代步长并根据(4.1)更新  $\mathbf{x}$  即可。

现在, 我来解释一下选择用残差向量来构造迭代方向的优势在哪里。回顾(4.15), 仔细观察这个式子, 它意味着构造  $\mathbf{d}_{(i)}$  的时候需要用到前面所有的迭代方向  $\mathbf{d}_{(0)}, \dots, \mathbf{d}_{(i-1)}$ , 所以每一个迭代方向都需要被存储下来。当  $n$  很大的时候, 时间和空间的开销都很大。当  $\mathbf{p}_{(i)}$  被设置为  $\mathbf{r}_{(i)}$  的时候, 奇

迹发生了：首先

$$\mathbf{r}_{(i)}^T \mathbf{r}_{(j+1)} = \mathbf{r}_{(i)}^T \mathbf{r}_{(j)} - \alpha_{(j)} \mathbf{r}_{(i)}^T \mathbf{A} \mathbf{d}_{(j)} \quad \triangleright (4.23)$$

$$\alpha_{(j)} \mathbf{r}_{(i)}^T \mathbf{A} \mathbf{d}_{(j)} = \mathbf{r}_{(i)}^T \mathbf{r}_{(j)} - \mathbf{r}_{(i)}^T \mathbf{r}_{(j+1)} \quad \triangleright (4.24)$$

$$\mathbf{r}_{(i)}^T \mathbf{A} \mathbf{d}_{(j)} = \begin{cases} \frac{1}{\alpha_{(i)}} \mathbf{r}_{(i)}^T \mathbf{r}_{(i)}, & i = j \\ -\frac{1}{\alpha_{(i-1)}} \mathbf{r}_{(i)}^T \mathbf{r}_{(i)}, & i = j + 1 \\ 0, & \text{otherwise} \end{cases} \quad (4.25)$$

进而可得

$$\beta_{ij} = \begin{cases} \frac{1}{\alpha_{(i-1)}} \frac{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}}{\mathbf{d}_{(i-1)}^T \mathbf{A} \mathbf{d}_{(i-1)}}, & j = i - 1 \\ 0, & j < i - 1 \end{cases} \quad \triangleright (4.17)$$

这意味着除了最近一次的迭代方向的常系数不为零，其他都是零！因此，我们不再需要存储之前的搜索方向了，这使得时空开销显著降低。这一优势让 CG 在每一轮迭代中的时间和空间复杂度均从  $O(n^2)$  降低到了  $O(m)$ ，其中  $m$  是  $\mathbf{A}$  中非零元素的数量。

当  $j = i - 1$  时，将  $\alpha_{(i-1)}$  的取值代入， $\beta_{ij}$  还可以进一步化简：

$$\begin{aligned} \beta_{ij} &= \frac{\mathbf{d}_{(i-1)}^T \mathbf{A} \mathbf{d}_{(i-1)}}{\mathbf{d}_{(i-1)}^T \mathbf{r}_{(i-1)}} \cdot \frac{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}}{\mathbf{d}_{(i-1)}^T \mathbf{A} \mathbf{d}_{(i-1)}} \quad \triangleright (4.8) \\ &= \frac{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}}{\mathbf{d}_{(i-1)}^T \mathbf{r}_{(i-1)}} \quad \triangleright (4.22) \& \mathbf{p}_{(i-1)} \leftarrow \mathbf{r}_{(i-1)} \\ &= \frac{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}}{\mathbf{r}_{(i-1)}^T \mathbf{r}_{(i-1)}} \end{aligned} \quad (4.27)$$

为了方便描述，我们直接将  $\beta_{ij}$  ( $j = i - 1$ ) 改写为  $\beta_{(i-1)}$ ，则(4.15)更新为

$$\mathbf{d}_{(i+1)} = \mathbf{r}_{(i+1)} + \beta_{(i)} \mathbf{d}_{(i)}. \quad (4.28)$$

结合以上分析，我们终于得到了 CG 的算法描述（见 **Algorithm 3**）。

## 4.5 收敛性分析

虽然 CG 在理论上只需要  $n$  轮迭代即可结束，但是收敛性分析依然是有意义的。一方面是因为我们偷懒用(4.23)来代替矩阵-向量运算是会导致舍入误差越来越大，从而导致残差向量越来越失准；另一方面是因为删除误差（cancellation error）会导致搜索向量逐渐丧失  $\mathbf{A}$ -正交性。

### 4.5.1 选择最佳的多项式

因为（共轭）格莱姆-施密特变换是不改变基所张成的空间的，即

$$\text{span}\{\mathbf{r}_{(0)}, \dots, \mathbf{r}_{(i)}\} = \text{span}\{\mathbf{d}_{(0)}, \dots, \mathbf{d}_{(i)}\} = \mathcal{D}_{i+1}, \quad (4.29)$$

所以显然有  $\mathbf{r}_{(i)} \in \mathcal{D}_{(i+1)}$ 。此外，仔细观察(4.23)可知，第  $i + 1$  轮迭代后的残差向量  $\mathbf{r}_{(i+1)}$  可以表示成上一轮迭代的残差向量  $\mathbf{r}_{(i)}$  和  $\mathbf{A} \mathbf{d}_{(i)}$  的线性组合。因此， $\mathbf{r}_{(i+1)}$  所在的张成子空间等于  $\mathbf{r}_{(i)}$  所张成的子空间和子空间  $\mathbf{A} \mathcal{D}_i$  的并集，即

$$\mathcal{D}_{i+2} = \mathcal{D}_{i+1} \cup \mathbf{A} \mathcal{D}_{i+1}. \quad (4.30)$$

---

**Algorithm 3:** 共轭梯度法 (CG) .

---

**Input:** 对称正定矩阵  $\mathbf{A} \in \mathbb{R}^{n \times n}$ , 向量  $\mathbf{b} \in \mathbb{R}^n$ , 停止条件  $\varepsilon$ , 最大迭代次数 MaxIter

**Output:** 线性方程组  $\mathbf{Ax} = \mathbf{b}$  的一个满足精度要求的解

```

1   $k = 0, \mathbf{r}_{(0)} = \mathbf{b} - \mathbf{Ax}_{(0)}$ 
2   $\mathbf{d}_{(0)} = \mathbf{r}_{(0)}$                                 // Set  $\mathbf{d}_{(0)}$  as  $\mathbf{r}_{(0)}$ 
3   $\delta_{(0)} = \mathbf{r}_{(0)}^\top \mathbf{r}_{(0)}$ 
4  while  $k < \text{MaxIter}$  and  $\delta_{(k)} > \varepsilon^2 \delta_{(0)}$  do
5     $\mathbf{q} = \mathbf{Ad}_{(k)}, \alpha_{(k)} = \delta_{(k)} / \mathbf{d}_{(k)}^\top \mathbf{q}$ 
6     $\mathbf{x}_{(k+1)} = \mathbf{x}_{(k)} + \alpha_{(k)} \mathbf{d}_{(k)}$ 
7    if  $k$  is divisible by  $\lceil \sqrt{n} \rceil$  then
8      /* Periodically update by the original formula           */
9       $\mathbf{r}_{(k+1)} = \mathbf{b} - \mathbf{Ax}_{(k+1)}$ 
10   else
11     /* Reduce complexity according to (4.23)                 */
12      $\mathbf{r}_{(k+1)} = \mathbf{r}_{(k)} - \alpha_{(k)} \mathbf{q}$ 
13   end if
14    $\delta_{(k+1)} = \mathbf{r}_{(k+1)}^\top \mathbf{r}_{(k+1)}, \beta = \delta_{(k+1)} / \delta_{(k)}$ 
15    $\mathbf{d}_{(k+1)} = \mathbf{r}_{(k+1)} + \beta \mathbf{d}_{(k)}$           // Set  $\mathbf{d}_{(k+1)}$  by (4.28)
16    $k = k + 1$ 
17 end while
18 return  $\mathbf{x}_{(k)}$ 

```

---

利用递归性可得

$$\begin{aligned}
\mathcal{D}_i &= \mathcal{D}_{i-1} \cup \mathbf{A}\mathcal{D}_{i-1} \\
&= \mathcal{D}_1 \cup \mathbf{A}\mathcal{D}_1 \cup \mathbf{A}^2\mathcal{D}_1 \cup \dots \cup \mathbf{A}^{i-1}\mathcal{D}_1 \\
&= \text{span}\{\mathbf{d}_{(0)}, \mathbf{Ad}_{(0)}, \dots, \mathbf{A}^{i-1}\mathbf{d}_{(0)}\} \\
&= \text{span}\{\mathbf{r}_{(0)}, \mathbf{Ar}_{(0)}, \dots, \mathbf{A}^{i-1}\mathbf{r}_{(0)}\} \quad \triangleright \mathbf{r}_{(i)} = -\mathbf{Ae}_{(i)} \\
&= \text{span}\{\mathbf{Ae}_{(0)}, \mathbf{A}^2\mathbf{e}_{(0)}, \dots, \mathbf{A}^i\mathbf{e}_{(0)}\}.
\end{aligned} \tag{4.31}$$

上述子空间称为 Krylov 子空间, 它是一个通过将一个矩阵反复左乘一个向量构建起来的子空间。对于这个空间, 在第  $i$  轮迭代中, 误差项可以表示成如下形式 (用上述张成子空间来描述):

$$\mathbf{e}_{(i)} = \left( \mathbf{I} + \sum_{k=1}^i \psi_k \mathbf{A}^k \right) \mathbf{e}_{(0)}, \tag{4.32}$$

其中  $\psi_k$  是一个关于  $\alpha_{(k)}$  和  $\beta_{(k)}$  的函数。我们可以将圆括号内的表达式视为一个关于矩阵  $\mathbf{A}$  的多项式  $P_i(\mathbf{A})$ , 其参数既可以是矩阵, 也可以是标量。若参数是一个标量  $\lambda$ , 则  $P_i(\lambda) = 1 + \sum_{k=1}^i \psi_k \lambda^k$  (因此  $P_i(0) = 1$ )。所以(4.32)可以写作  $\mathbf{e}_{(i)} = P_i(\mathbf{A})\mathbf{e}_{(0)}$ 。和分析梯度法时所采取的策略一样, 令  $\mathbf{v}_1, \dots, \mathbf{v}_n$  是  $\mathbf{A}$  的单位正交的特征向量,  $\lambda_1, \dots, \lambda_n$  是对应的特征值, 并将误差向量  $\mathbf{e}_{(0)}$  用特征向量

的线性组合表示:  $\mathbf{e}_{(0)} = \sum_{i=1}^n \xi_i \mathbf{v}_i$ , 那么有

$$\begin{aligned}\mathbf{e}_{(i)} &= \sum_{j=1}^n \xi_j P_i(\lambda_j) \mathbf{v}_j \quad \triangleright P_i(\mathbf{A}) \mathbf{v}_j = P_i(\lambda_j) \mathbf{v}_j \\ \mathbf{A} \mathbf{e}_{(i)} &= \sum_{j=1}^n \xi_j P_i(\lambda_j) \lambda_j \mathbf{v}_j \\ \|\mathbf{e}_{(i)}\|_{\mathbf{A}}^2 &= \sum_{j=1}^n \xi_j^2 (P_i(\lambda_j))^2 \lambda_j, \quad \triangleright \|\mathbf{v}_j\| = 1\end{aligned}\tag{4.33}$$

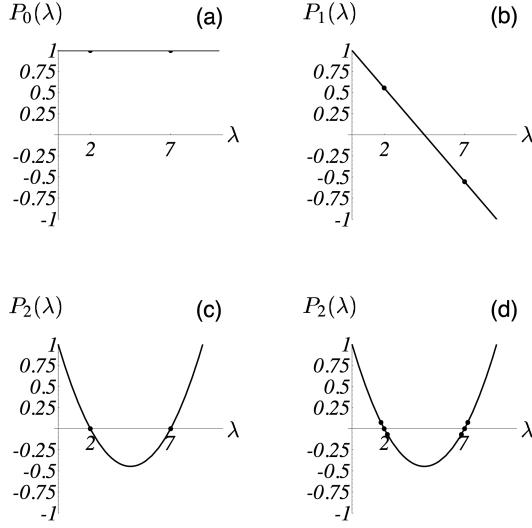
且  $\|\mathbf{e}_{(0)}\|_{\mathbf{A}}^2 = \sum_{j=1}^n \xi_j^2 \lambda_j$  (因为  $P_0(\cdot) \equiv 1$ )。我们在 §4.3 中分析过, 在每一轮迭代中, CG 都能做到最小化  $\|\mathbf{e}_{(i)}\|_{\mathbf{A}}$ 。对应到(4.33), 即 CG 每轮迭代都能找到使得  $\|\mathbf{e}_{(i)}\|_{\mathbf{A}}$  最小的多项式  $P_i$ , 其收敛性和最坏的特征向量的收敛性一致。因此,

$$\begin{aligned}\|\mathbf{e}_{(i)}\|_{\mathbf{A}}^2 &\leq \min_{P_i} \max_{\lambda} (P_i(\lambda))^2 \sum_{j=1}^n \xi_j^2 \lambda_j \\ &= \min_{P_i} \max_{\lambda} (P_i(\lambda))^2 \|\mathbf{e}_{(0)}\|_{\mathbf{A}}^2.\end{aligned}\tag{4.34}$$

同样地, 我们还是拿  $n = 2$  的情况分析看看。对于 Fig. 3.3 中的例子,  $\lambda_1 = 7, \lambda_2 = 2$ , Fig. 4.3 解释了 CG 是如何最小化  $\max_{\lambda} (P_i(\lambda))^2$  的。在 (a) 中,  $i = 0$ , 尚未开始迭代, 因此和  $\|\mathbf{e}_{(0)}\|_{\mathbf{A}}^2$  的比值恒为 1; 在 (b) 中,  $i = 1$ , 在  $P_i(0) = 1$  的约束下, 一阶多项式必然长成这个样子:  $P_1(\lambda) = a^* \lambda + 1$ , 其中  $a^*$  必然满足

$$a^* = \operatorname{argmin}_a \max \{(2a+1)^2, (7a+1)^2\},$$

求解上式可得  $a^* = -2/9$ 。所以最优的一阶多项式为  $P_1(\lambda) = -2/9\lambda + 1$ , 此时误差项的能量范数小于其初始值的  $5/9$ ; 在 (c) 中,  $i = 2$ , 我们完全有把握找到经过  $(0, 1), (2, 0), (7, 0)$  三个点的二阶多项式, 因而第二轮一结束误差项的能量范数就是 0 了, 表明迭代结束, 最优解已找到。



**Figure 4.3:** 经过第  $i$  轮之后, CG 的收敛情况取决于在  $P_i(0) = 1$  的约束下,  $P_i$  在各个特征值上距离 0 有多近。

设想一下, 若  $\lambda_1 = \lambda_2$ , 那么只需要一阶多项式即可拟合  $(0, 1), (\lambda_1, 0), (\lambda_2, 0)$  三个点, 这意味着一轮迭代就可以找到最优解了! 这充分说明 CG 算法在特征值存在重复的情况下收敛是非常快的, 重复的特征值越多, 收敛速度越快。这一点和梯度法不谋而合。此外, 如 Fig. 4.3 (d) 所示, 当特

征值不完全相同时，若这些特征值在  $\lambda_{min}$  和  $\lambda_{max}$  这两端分布较为集中，那么 CG 的收敛速度也不慢，因为此时也很容易找到一个多项式来拟合出一个不错的结果。

显然，特征值并不总是相同。因此，接下来我们将会分析最一般的情形，即特征值在  $[\lambda_{min}, \lambda_{max}]$  之间均匀分布时，CG 的收敛速度如何。

#### 4.5.2 切比雪夫多项式

我们首先引入一个著名的多项式，切比雪夫多项式 (Chebyshev polynomials)。 $i$  阶的切比雪夫多项式的表达式为

$$T_i(\omega) = \frac{1}{2} \left( (\omega + \sqrt{\omega^2 - 1})^i + (\omega - \sqrt{\omega^2 - 1})^i \right). \quad (4.35)$$

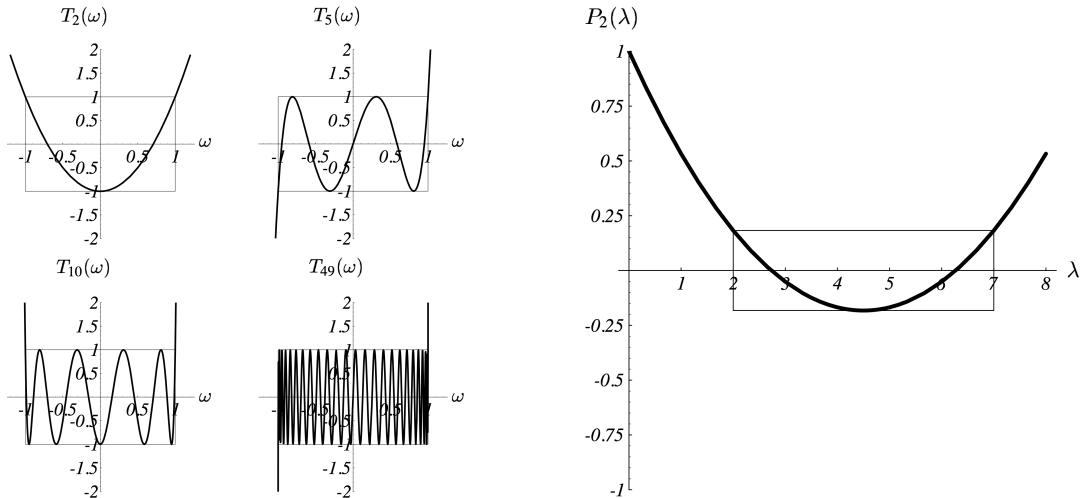
Fig. 4.4 左图给出了切比雪夫不等式在不同的阶的情况下图像。该多项式具有如下特性：

- $\omega \in [-1, 1]$  时， $T_i(\omega) \leq 1$ ，其函数值在 -1 到 1 之间反复震荡；
- $\omega \notin [-1, 1]$  时，其函数值尽其可能的迅速增长。

可以发现，当

$$P_i(\lambda) = \frac{T_i\left(\frac{\lambda_{max} + \lambda_{min} - 2\lambda}{\lambda_{max} - \lambda_{min}}\right)}{T_i\left(\frac{\lambda_{max} + \lambda_{min}}{\lambda_{max} - \lambda_{min}}\right)} \quad (4.36)$$

时<sup>7</sup>，能够最小化  $\max_{\lambda} (P_i(\lambda))^2$ 。Fig. 4.4 右图给出了  $\lambda_{min} = 2, \lambda_{max} = 7$  时上述多项式的图像。和 4.3 (c) 相比，虽然两轮迭代后还不能收敛，但是已经是一个不错的结果：误差项的能量范式将不大于其初始值的 0.183 倍。



**Figure 4.4:** 左图：切比雪夫不等式在不同的阶的情况下图像；右图：一般情况下最小化能量范式二阶多项式示意图。

<sup>7</sup>这个证明并不简单。但是，考虑到探究其证明过程与 CG 本身无关，我就没有在正文中给予描述。

根据切比雪夫多项式的特性可知  $T_i\left(\frac{\lambda_{max}+\lambda_{min}-2\lambda}{\lambda_{max}-\lambda_{min}}\right) \leq 1$ , 所以

$$\begin{aligned}\|e_{(i)}\|_A &\leq \frac{T_i\left(\frac{\lambda_{max}+\lambda_{min}-2\lambda}{\lambda_{max}-\lambda_{min}}\right)}{T_i\left(\frac{\lambda_{max}+\lambda_{min}}{\lambda_{max}-\lambda_{min}}\right)} \|e_{(0)}\|_A \\ &\leq T_i\left(\frac{\lambda_{max}+\lambda_{min}}{\lambda_{max}-\lambda_{min}}\right)^{-1} \|e_{(0)}\|_A \\ &= 2\left(\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^i + \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i\right)^{-1} \|e_{(0)}\|_A.\end{aligned}\quad (4.37)$$

随着  $i$  的增加,  $\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{-i}$  会逐渐收敛到 0, 因此更多的时候我们用下面的弱不等式来表述 CG 的收敛速度:

$$\|e_{(i)}\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i \|e_{(0)}\|_A. \quad (4.38)$$

我们可以对比一下(4.37)和(3.27), 可发现  $i = 1$  时二者相同。这是显然的, 因为 CG 的第一步和梯度法一样。 $i > 1$  时, 对比 Fig. (3.7) 可发现 CG 通常要比梯度法快。Fig. (4.5) 给出了 CG 算法 (每轮迭代) 收敛性关于条件数  $\kappa$  的函数图像, 图上略掉了系数 2。实际应用中, 由于较好的特征值分布或者比较理想的起始点的存在, CG 算法通常收敛的比(4.38)所描述的还要快。

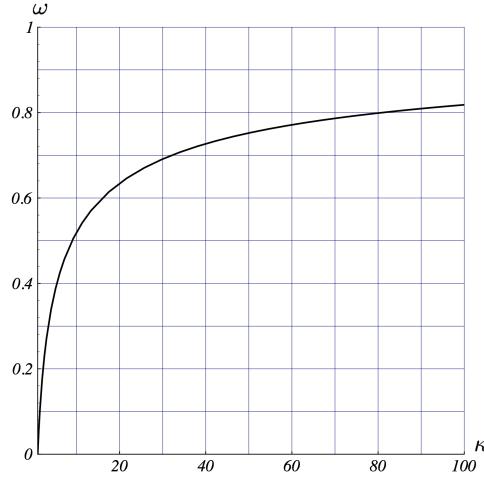


Figure 4.5: CG 算法 (每轮迭代) 收敛性关于条件数  $\kappa$  的函数图像。

## 4.6 复杂度分析

上一章节分析了 CG 的收敛性, 其收敛速度和特征值的取值是否相同以及条件数的大小有关。本章节将分析 CG 的复杂度。

就计算复杂度而言, 不论是 CG 还是梯度法, 其迭代过程中最耗时的操作都是矩阵-向量乘积。一般而言, 矩阵-向量乘积需要  $O(m)$  次操作, 其中  $m$  为矩阵非零元的个数。显然, 矩阵越稀疏, 这一操作的算力需求就越小。复杂度可以根据收敛速度得到。对于梯度法, 假设我们希望执行足够多的迭代次数来把误差的能量范式减少到其初始值的  $\epsilon$  倍, 即  $\|e_{(i)}\| = \epsilon \|e_{(0)}\|$ , 则根据(3.27)可算出梯度法实现这个边界条件的最大迭代次数需求为

$$i \leq \left\lceil \frac{1}{2} \kappa \ln\left(\frac{1}{\epsilon}\right) \right\rceil. \quad (4.39)$$

类似地，根据(4.38)可算出 CG 实现这个边界条件的最大迭代次数需求为

$$i \leq \left\lceil \frac{1}{2} \sqrt{\kappa} \ln \left( \frac{2}{\epsilon} \right) \right\rceil. \quad (4.40)$$

因此，

- 时间复杂度：梯度法为  $O(m\kappa)$ , CG 为  $O(m\sqrt{\kappa})$ ;
- 空间复杂度，二者均为  $O(m)$ 。

二阶椭圆边值问题在  $d$  维域上的有限差分和有限元逼近通常有： $\kappa \in O(n^{2/d})$ 。所以在二维问题上，梯度法的时间复杂度为  $O(n^2)$ , CG 则为  $O(n^{3/2})$ ；在三维问题上，梯度法的时间复杂度为  $O(n^{5/3})$ , CG 则为  $O(n^{4/3})$ 。

## 5 共轭梯度法的实现

§4讲解了 CG 的理论原理和算法描述，本章节将谈一谈在实现 CG 时应当注意的一些细节问题，包括起始点和算法停止迭代的条件应如何选取、需要怎样的预处理等。

### 5.1 起点与终点

起始点  $\mathbf{x}_{(0)}$  的选取对 CG 的影响其实并不大，一般取  $\mathbf{x}_{(0)} = \mathbf{0}$  即可。对于给定的二次型  $f(\mathbf{x})$  且  $\mathbf{A}$  为对称正定阵时，任意初始点最终都必将收敛。但是对于非凸优化问题，因为存在多个局部极小值点，初始点的选取将极大影响最终结果。

观察(4.27)和 Algorithm 3 的 Step 14 可知，当残差向量为零的时候， $\beta_{ij}$  的计算必然发生除零错误。因此算法需要在残差向量等于  $\mathbf{0}$  之前停止迭代。但是，由于舍入误差的存在，可能会得到一个假的零残差。因此绝大多数时候，更稳妥的做法是当残差向量（的范数）小于某个给定阈值的时候就停下来。Algorithm 3 的 Step 4 正是这样做的。

### 5.2 预处理

观察和 Fig. 4.5 以及(4.38)，可发现随着条件数  $\kappa$  的增加，CG 收敛速度越来越慢。因此，如果可以降低给定矩阵的条件数，就可以让 CG 更快地收敛。设  $\mathbf{M}$  是一个对称正定且方便求逆的矩阵，则求解  $\mathbf{Ax} = \mathbf{b}$  等价于求解

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}, \quad (5.1)$$

若  $\kappa(\mathbf{M}^{-1}\mathbf{A}) \ll \kappa(\mathbf{A})$  ( $\mathbf{M}^{-1}\mathbf{A}$  的特征值分布比  $\mathbf{A}$  更集中) 且  $\mathbf{M}^{-1}\mathbf{A}$  对称正定，则用 CG 求解上式比直接求解  $\mathbf{Ax} = \mathbf{b}$  更快。这就是“预处理”的思路， $\mathbf{M}$  被称为预处理器。但是，现实是骨感的，即使  $\mathbf{M}$  和  $\mathbf{A}$  都对称正定也无法保证  $\mathbf{M}^{-1}\mathbf{A}$  对称正定。

实际上可以绕过这个问题。对于任意对称正定矩阵  $\mathbf{M}$ ，必然存在矩阵  $\mathbf{E}$  满足  $\mathbf{EE}^T = \mathbf{M}$  (可通过 Cholesky 分解获得)。设  $\lambda$  是  $\mathbf{M}^{-1}\mathbf{A}$  的特征向量， $\mathbf{v}$  是对应的特征向量，则

$$\begin{aligned} (\mathbf{E}^{-1}\mathbf{A}\mathbf{E}^{-T})(\mathbf{E}^T\mathbf{v}) &= \mathbf{E}^{-1}\mathbf{A}(\mathbf{E}^{-T}\mathbf{E}^T)\mathbf{v} \quad \triangleright \mathbf{E}^{-T}\mathbf{E}^T = \mathbf{E}^T\mathbf{E}^{-T} = \mathbf{I} \\ &= (\mathbf{E}^T\mathbf{E}^{-T})\mathbf{E}^{-1}\mathbf{A}\mathbf{v} = \mathbf{E}^T(\mathbf{E}^{-T}\mathbf{E}^{-1})\mathbf{A}\mathbf{v} \quad \triangleright \mathbf{E}^{-T}\mathbf{E}^{-1} = \mathbf{M}^{-1} \\ &= \mathbf{E}^T\mathbf{M}^{-1}\mathbf{A}\mathbf{v} \quad \triangleright \mathbf{M}^{-1}\mathbf{A}\mathbf{v} = \lambda\mathbf{v} \\ &= \lambda(\mathbf{E}^T\mathbf{v}), \end{aligned} \quad (5.2)$$

即  $\lambda$  也是  $\mathbf{E}^{-1}\mathbf{A}\mathbf{E}^{-T}$  特征值，且相应的特征向量为  $\mathbf{E}^T\mathbf{v}$ ！因此，求解  $\mathbf{Ax} = \mathbf{b}$  等价于求解

$$\begin{cases} \mathbf{E}^{-1}\mathbf{A}\mathbf{E}^{-T}\hat{\mathbf{x}} = \mathbf{M}^{-1}\mathbf{b}, \\ \hat{\mathbf{x}} = \mathbf{E}^T\mathbf{x}. \end{cases} \quad (5.3)$$

我们可以先求解  $\hat{\mathbf{x}}$ , 再求解  $\mathbf{x}$ 。当  $\mathbf{A}$  为对称正定阵时  $\mathbf{E}^{-1}\mathbf{A}\mathbf{E}^{-\top}$  也是对称正定阵, 因此可通过 CG 求解  $\hat{\mathbf{x}}$ 。这就是变换预处理共轭梯度法 (Transformed Preconditioned Conjugate Gradient Method, TPCG), 在框架上和 CG 一致。(5.4)总结了相关的要点。

$$\left\{ \begin{array}{l} \hat{\mathbf{d}}_{(0)} = \hat{\mathbf{r}}_{(0)} = \mathbf{E}^{-1}\mathbf{b} - \mathbf{E}^{-1}\mathbf{A}\mathbf{E}^{-\top}\hat{\mathbf{x}}_{(0)} \\ \alpha_{(i)} = \frac{\hat{\mathbf{r}}_{(i)}^T \hat{\mathbf{r}}_{(i)}}{\hat{\mathbf{d}}_{(i)}^T \mathbf{E}^{-1}\mathbf{A}\mathbf{E}^{-\top}\hat{\mathbf{d}}_{(i)}} \\ \hat{\mathbf{x}}_{(i+1)} = \hat{\mathbf{x}}_{(i)} + \alpha_{(i)}\hat{\mathbf{d}}_{(i)} \\ \hat{\mathbf{r}}_{(i+1)} = \hat{\mathbf{r}}_{(i)} - \alpha_{(i)}\mathbf{E}^{-1}\mathbf{A}\mathbf{E}^{-\top}\hat{\mathbf{d}}_{(i)} \\ \beta_{(i+1)} = \frac{\hat{\mathbf{r}}_{(i+1)}^T \hat{\mathbf{r}}_{(i+1)}}{\hat{\mathbf{r}}_{(i)}^T \hat{\mathbf{r}}_{(i)}} \\ \hat{\mathbf{d}}_{(i+1)} = \hat{\mathbf{r}}_{(i+1)} + \beta_{(i+1)}\hat{\mathbf{d}}_{(i)}. \end{array} \right. \quad (5.4)$$

然而, TPCG 需要我们通过矩阵分解的方法计算出  $\mathbf{E}$ 。有没有什么办法可以避免  $\mathbf{E}$  的计算并且直接计算  $\mathbf{x}$  而非  $\hat{\mathbf{x}}$  呢? 有! 令  $\hat{\mathbf{r}}_{(i)} = \mathbf{E}^{-1}\mathbf{r}_{(i)}$  且  $\hat{\mathbf{d}}_{(i)} = \mathbf{E}^T\mathbf{d}_{(i)}$ , 再利用等价变换  $\hat{\mathbf{x}} = \mathbf{E}^T\mathbf{x}$  和  $\mathbf{E}^{-\top}\mathbf{E}^{-1} = \mathbf{M}^{-1}$  即可从(5.4)得到

$$\left\{ \begin{array}{l} \mathbf{r}_{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}_{(0)}, \mathbf{d}_{(0)} = \mathbf{M}^{-1}\mathbf{r}_{(0)} \\ \alpha_{(i)} = \frac{\mathbf{r}_{(i)}^T \mathbf{M}^{-\top} \mathbf{r}_{(i)}}{\mathbf{d}_{(i)}^T \mathbf{A} \mathbf{d}_{(i)}} \\ \mathbf{x}_{(i+1)} = \mathbf{x}_{(i)} + \alpha_{(i)}\mathbf{d}_{(i)} \\ \mathbf{r}_{(i+1)} = \mathbf{r}_{(i)} - \alpha_{(i)}\mathbf{A}\mathbf{d}_{(i)} \\ \beta_{(i+1)} = \frac{\mathbf{r}_{(i+1)}^T \mathbf{M}^{-1} \mathbf{r}_{(i+1)}}{\mathbf{r}_{(i)}^T \mathbf{M}^{-1} \mathbf{r}_{(i)}} \\ \mathbf{d}_{(i+1)} = \mathbf{M}^{-1}\mathbf{r}_{(i+1)} + \beta_{(i+1)}\mathbf{d}_{(i)}. \end{array} \right. \quad (5.5)$$

这就是未变换预处理共轭梯度法 (Untransformed Preconditioned CG Method, UPCG)。之所以叫做“未变换的”是因为和 TPCG 相比, UPCG 不需要变换为  $\hat{\mathbf{x}}$  来求解。此外, UPCG 也不再需要计算  $\mathbf{E}$  了, 因为根本没用到这个变量。

本质上, 预处理是尝试对二次型  $f(\mathbf{x})$  的图形进行拉伸, 使其更接近于球形, 以便它的特征值彼此之间更加靠近。最完美的预处理器当然是  $\mathbf{M} = \mathbf{A}$ , 因为此时  $\kappa(\mathbf{M}^{-1}\mathbf{A}) = 1$ , 相当于把  $f(\mathbf{x})$  沿着特征向量进行伸缩,  $f(\mathbf{x})$  被拉伸为一个完美的球形。但是, 这样做毫无意义, 因为经过预处理之后, 我们还是要求解  $\mathbf{M}\mathbf{x} = \mathbf{b}$  (与原始问题等价)。一种常用的预处理器  $\mathbf{M}$  是  $\mathbf{A}$  对角元素组成的对角阵。这相当于把  $f(\mathbf{x})$  沿着坐标轴进行伸缩。此外, 之所以这样选择的另一个重要原因是它的逆很好求解。人们其实发明了很多预处理器, 一种更为精细复杂的预处理器是不完备 Cholesky 预处理器 (incomplete Cholesky preconditioning), 本文不再详细介绍。

本章节所有的分析虽然是针对 CG 的, 但是同样适用于梯度法。对应地, 有变换/未变换预处理最速下降法 (Transformed/Untransformed Preconditioned Steepest Descent Method)。

### 5.3 扩展到一般场景

到目前为止, 我们只讨论当  $\mathbf{A}$  为对称正定阵的情形。对于一般的矩阵, 可否将梯度法和 CG 用上呢? 以最小二乘问题为例:

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2, \quad (5.6)$$

其解可通过  $\nabla_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 = 0$  得到, 即

$$\mathbf{A}^T \mathbf{A} \mathbf{x} = \mathbf{A}^T \mathbf{b}. \quad (5.7)$$

若  $\mathbf{A}$  为非奇异的方阵, 则上式等价于  $\mathbf{A}\mathbf{x} = \mathbf{b}$ 。若  $\mathbf{A} \in \mathbb{R}^{m \times n}$  且  $m > n$  (overconstrained), 那么(5.7)可能无解。但不论如何, (5.6)总是有解的。怎么求呢? 我们知道  $\mathbf{A}^T \mathbf{A}$  总是对称正定的, 若  $\mathbf{A}\mathbf{x} = \mathbf{b}$  不是欠约束的 (underconstrained), 即  $\mathbf{A}^T \mathbf{A}$  非奇异, 那么就可以用梯度法或者 CG 求解(5.7)。也就是说, 当  $\mathbf{A}$  不是对称正定阵的时候, 若满足特定的条件, 那么也是可以采用梯度法或

者 CG 求解的。这就将二者扩展到了更为广泛的应用场景中去了。

CG 也可以拓展到任意连续函数上，而不仅仅局限于二次型。但是要面临许多新的问题：

- 残差向量  $r_{(i)}$  不再等于  $\mathbf{b} - \mathbf{Ax}_{(i)}$ 。常用的做法是  $\mathbf{r}_{(i)} \leftarrow -f'(\mathbf{x}_{(i)})$ ；
- 最优步长  $\alpha_{(i)}$  依然通过求导获得，但是更加复杂。一阶导为零对应的方程需要用到一些数值计算的方法来求解，如牛顿法和割线法等；
- $\beta_{(i)}$  有多重可选的方案。目前常用的有 Fletcher-Reeves 方程 (FR) 和 Polak-Ribière 方程 (PR)，其表达式分别如下：

$$\begin{cases} \beta_{(i+1)}^{FR} = \frac{\mathbf{r}_{(i+1)}^T \mathbf{r}_{(i+1)}}{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}} \\ \beta_{(i+1)}^{PR} = \frac{\mathbf{r}_{(i+1)}^T (\mathbf{r}_{(i+1)} - \mathbf{r}_{(i)})}{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}} \end{cases} \quad (5.8)$$

FR 就是  $f(\mathbf{x})$  为二次型时我们所采用的那个。在起始点足够接近我们想要求解的最小值点的情况下，前者能够收敛，后者在少数情况下却会无限循环而不收敛，但是速度更快。一种解决 PR 不收敛的方案是

$$\beta_{(i+1)} \leftarrow \max \left\{ \frac{\mathbf{r}_{(i+1)}^T (\mathbf{r}_{(i+1)} - \mathbf{r}_{(i)})}{\mathbf{r}_{(i)}^T \mathbf{r}_{(i)}}, 0 \right\}, \quad (5.9)$$

本质上就是当  $\beta_{(i+1)}^{PR} < 0$  时开启一轮新的 CG。

非线性 CG 算法几乎没有像线性 CG 算法的收敛性保证。函数  $f$  和二次型函数越不像，搜索方向失去共轭性的速度就越快。此外， $f$  可能会存在多个局部最小值点，CG 算法并不保证能收敛到全局最小值点。若  $f$  没有下界的话，CG 甚至都找不到局部最小值点。

## 6 后记

撰写这份文档的初衷是完成《计算机应用数学》课程的最后一次作业。总体上，本文是对 Jonathan Richard Shewchuk 的文档 *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain* 的重新演绎，也参考了这个页面的一些信息，但加入了很多我个人的理解和感悟。此外，我删去了很多原文中在我看来有些多余的叙述。本文的核心内容是讲解共轭梯度法，但也捎带阐述了梯度法和一些线性代数的理论。本文对共轭梯度的解释是极为充实的，设计原理、算法实现以及收敛性分析均被涵盖在内，希望可以对读者有切实的帮助。最新版本可以在<http://hliangzhao.me/math/CG.pdf> 下载。

本文的图表均来自上述文档。

## 致谢

感谢 Licai Chu 教授认真细致地指出本文档的各类错误。本文档已于 2021 年 1 月修改。