第二讲 Krylov 子空间方法

目录

子空间方法

子空间方法基本思想: 降维

原问题: $Ax = b \Longleftrightarrow \min\limits_{x\in\mathbb{R}^n} \|b - Ax\|$

降维: 设 $\mathcal{K}$ 是 $\mathbb{R}^n$ 的一个子空间, $\dim(\mathcal{K}) = m \ll n$.

将原问题转化 (近似) 为: $\min\limits_{x\in\mathcal{K}} \|b - Ax\|$ 或 $\min\limits_{x\in\mathcal{K}} \|x - x_*\|$

子空间方法

子空间方法基本思想: 降维

原问题: $$Ax = b \Longleftrightarrow \min_{x\in\mathbb{R}^n} \|b - Ax\|$$

降维: 设 $\mathcal{K}$ 是 $\mathbb{R}^n$ 的一个子空间, $\dim(\mathcal{K}) = m \ll n$.

将原问题转化 (近似) 为: $$\min_{x\in\mathcal{K}} \|b - Ax\|$$ 或 $$\min_{x\in\mathcal{K}} \|x - x_*\|$$

优点: 自由度大幅减少, 问题规模从 n 降为 m

运算量主要集中在矩阵向量乘积, 非常适合稀疏矩阵

问题: (1) 怎么找合适的子空间? (2) 怎么寻找好的近似解?

2-1 | 投影方法

什么是投影方法

在 m 维子空间 $\mathcal{K}$ 中寻找 x_* 的某个 **投影**, 作为一个好的近似解.

2-1 | 投影方法

什么是投影方法

在 m 维子空间 $\mathcal{K}$ 中寻找 x_* 的某个 **投影**, 作为一个好的近似解.

定解条件: 设置 m 个约束 $\rightarrow$ 保证最佳近似存在唯一

$$r = b - A\tilde{x} \perp \mathcal{L},$$

其中 $\tilde{x}$ 是我们所要寻找的近似解, $\mathcal{L}$ 是另一个 m 维子空间.

- 正交性条件称为 **Petrov-Galerkin 条件**. 若 $\mathcal{L} = \mathcal{K}$, 则称为 **Galerkin 条件**
- 子空间 $\mathcal{K}$ 通常称为**搜索空间**, $\mathcal{L}$ 通常称为**约束空间**.

子空间投影法的一般描述

因此, 投影方法一般可以描述为

$$\text{find} \quad \tilde{x} \in \mathcal{K} \quad \text{such that} \quad b - A\tilde{x} \perp \mathcal{L}. \tag{2.1}$$

子空间投影法的一般描述

因此, 投影方法一般可以描述为

$$\text{find} \quad \tilde{x} \in \mathcal{K} \quad \text{such that} \quad b - A\tilde{x} \perp \mathcal{L}. \tag{2.1}$$

如果给定初值 $x^{(0)}$, 为充分利用初值信息, 改为在仿射空间 $x^{(0)} + \mathcal{K}$ 中寻找投影, 即

$$\text{find} \quad \tilde{x} \in x^{(0)} + \mathcal{K} \quad \text{such that} \quad b - A\tilde{x} \perp \mathcal{L}. \tag{2.2}$$

子空间投影法的一般描述

因此, 投影方法一般可以描述为

$$\text{find} \quad \tilde{x} \in \mathcal{K} \quad \text{such that} \quad b - A\tilde{x} \perp \mathcal{L}. \tag{2.1}$$

如果给定初值 $x^{(0)}$, 为充分利用初值信息, 改为在仿射空间 $x^{(0)} + \mathcal{K}$ 中寻找投影, 即

$$\text{find} \quad \tilde{x} \in x^{(0)} + \mathcal{K} \quad \text{such that} \quad b - A\tilde{x} \perp \mathcal{L}. \tag{2.2}$$

事实上, 如果将 $\tilde{x}$ 写成: $\tilde{x} = x^{(0)} + \hat{x}$, 其中 $\hat{x} \in \mathcal{K}$, 则 (2.2) 就是

$$\text{find} \quad \hat{x} \in \mathcal{K} \quad \text{such that} \quad r_0 - A\hat{x} \perp \mathcal{L},$$

其中 $r_0 = b - Ax^{(0)}$ 是初始残量. 这与 (2.1) 的形式是一样的.

子空间方法需要考虑的三个问题

- 如何选择 $\mathcal{K}$ 和 $\mathcal{L}$?
- 如何计算投影 (近似解) $\tilde{x}$?
- 若 $\tilde{x}$ 不满足精度要求，如何寻找新的 $\mathcal{K}$ 和 $\mathcal{L}$?

投影 (近似解) 的计算

$$\text{find} \quad \tilde{x} \in x^{(0)} + \mathcal{K} \quad \text{such that} \quad b - A\tilde{x} \perp \mathcal{L}.$$

设 $V = [v_1, v_2, \ldots, v_m]$ 和 $W = [w_1, w_2, \ldots, w_m]$ 分别是 $\mathcal{K}$ 和 $\mathcal{L}$ 一组基.

投影 (近似解) 的计算

$$\text{find} \quad \tilde{x} \in x^{(0)} + \mathcal{K} \quad \text{such that} \quad b - A\tilde{x} \perp \mathcal{L}.$$

设 $V = [v_1, v_2, \ldots, v_m]$ 和 $W = [w_1, w_2, \ldots, w_m]$ 分别是 $\mathcal{K}$ 和 $\mathcal{L}$ 一组基.

- 由于 $\tilde{x} \in x^{(0)} + \mathcal{K}$, 即 $\tilde{x} - x^{(0)} \in \mathcal{K}$, 因此存在向量 $y = [y_1, y_2, \ldots, y_m]^{\mathsf{T}} \in \mathbb{R}^m$ 使得

$$\tilde{x} = x^{(0)} + Vy$$

投影 (近似解) 的计算

$$\text{find} \quad \tilde{x} \in x^{(0)} + \mathcal{K} \quad \text{such that} \quad b - A\tilde{x} \perp \mathcal{L}.$$

设 $V = [v_1, v_2, \ldots, v_m]$ 和 $W = [w_1, w_2, \ldots, w_m]$ 分别是 $\mathcal{K}$ 和 $\mathcal{L}$ 一组基.

- 由于 $\tilde{x} \in x^{(0)} + \mathcal{K}$, 即 $\tilde{x} - x^{(0)} \in \mathcal{K}$, 因此存在向量 $y = [y_1, y_2, \ldots, y_m]^{\mathsf{T}} \in \mathbb{R}^m$ 使得

$$\tilde{x} = x^{(0)} + Vy$$

根据正交性条件可得

$$r_0 - AVy \perp w_i, \quad (i = 1, 2, \ldots, m) \quad \Longrightarrow \quad W^{\mathsf{T}} AVy = W^{\mathsf{T}} r_0$$

投影 (近似解) 的计算

$$\text{find} \quad \tilde{x} \in x^{(0)} + \mathcal{K} \quad \text{such that} \quad b - A\tilde{x} \perp \mathcal{L}.$$

设 $V = [v_1, v_2, \ldots, v_m]$ 和 $W = [w_1, w_2, \ldots, w_m]$ 分别是 $\mathcal{K}$ 和 $\mathcal{L}$ 一组基.

- 由于 $\tilde{x} \in x^{(0)} + \mathcal{K}$, 即 $\tilde{x} - x^{(0)} \in \mathcal{K}$, 因此存在向量 $y = [y_1, y_2, \ldots, y_m]^{\mathsf{T}} \in \mathbb{R}^m$ 使得

$$\tilde{x} = x^{(0)} + Vy$$

根据正交性条件可得

$$r_0 - AVy \perp w_i, \quad (i = 1, 2, \ldots, m) \implies W^{\mathsf{T}} AVy = W^{\mathsf{T}} r_0$$

若 $W^{\mathsf{T}} AV$ 非奇异, 则 $y = (W^{\mathsf{T}} AV)^{-1} W^{\mathsf{T}} r_0 \implies \tilde{x} = x^{(0)} + V(W^{\mathsf{T}} AV)^{-1} W^{\mathsf{T}} r_0$

定理 如果 $A, \mathcal{K}$ 和 $\mathcal{L}$ 满足下面条件之一

(1) A 正定且 $\mathcal{L} = \mathcal{K}$;

(2) A 非奇异且 $\mathcal{L} = A\mathcal{K}$,

则矩阵 $W^{\mathsf{T}} A V$ 非奇异. (留作练习)

注: 在实际计算中, 矩阵 $W^{\mathsf{T}} A V$ 通常可以直接形成, 而无需另外计算.

子空间 $\mathcal{K}$ 和 $\mathcal{L}$ 的选取

在实施投影法时, 我们需要考虑下面两个问题:

- 怎样选择搜索空间 $\mathcal{K}$ 和约束空间 $\mathcal{L}$?
- 如果找到的近似解 $\tilde{x}$ 达不到精度要求, 则需要更换搜索空间 $\mathcal{K}$, 此时如何更新 $\mathcal{K}$?

目前能够很好地解决上面两个问题的方法是采用 Krylov 子空间.

2-2 | Krylov 子空间与 Arnoldi 过程

2.2 Krylov 子空间与 Arnoldi 过程

https://math.ecnu.edu.cn/~jypan/Teaching/IMP

2-2-1 Krylov 子空间

设 $A \in \mathbb{R}^{n\times n}$, $r \in \mathbb{R}^n$, 我们称

$$\mathcal{K}_m(A, r) \triangleq \operatorname{span}\{r, Ar, \ldots, A^{m-1}r\} \subseteq \mathbb{R}^n$$

是由 A 和 r 生成的 **Krylov 子空间**, 通常简记为 $\mathcal{K}_m$, 或 $\mathcal{K}$.

2-2-1 Krylov 子空间

设 $A \in \mathbb{R}^{n\times n}$, $r \in \mathbb{R}^n$, 我们称

$$\mathcal{K}_m(A, r) \triangleq \operatorname{span}\{r, Ar, \ldots, A^{m-1}r\} \subseteq \mathbb{R}^n$$

是由 A 和 r 生成的 **Krylov 子空间**, 通常简记为 $\mathcal{K}_m$, 或 $\mathcal{K}$.

基本性质

- $\dim(\mathcal{K}_m) \leq m$
- Krylov 子空间是嵌套的, 即: $\mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq \cdots \subseteq \mathcal{K}_m \subseteq \cdots$
- $\mathcal{K}_m(A, r) = \left\{x = p(A)r : p \text{ 为次数不超过 } m-1 \text{ 的多项式}\right\}$

Krylov 子空间方法

以 Krylov 子空间为搜索空间的迭代法就是 **Krylov 子空间方法**

Krylov 子空间方法

以 Krylov 子空间为搜索空间的迭代法就是 **Krylov 子空间方法**

约束空间 $\mathcal{L}$ 的选取

目前被广泛采用的取法有:

- $\mathcal{L} = \mathcal{K}$: 如 FOM, CG, SYMMLQ $\quad \rightarrow \quad$ **正交投影法**
- $\mathcal{L} = A\mathcal{K}$: 如 MINRES, GMRES
- $\mathcal{L} = \mathcal{K}(A^{\mathsf{T}}, r)$: 如 BiCG

不同的 $\mathcal{L}$ 导出不同的 Krylov 子空间方法

2-2-2 标准正交基: Arnoldi 过程

算法 基于 MGS (Modified Gram-Schmidt) 的 Arnoldi 过程

1: 计算 $v_1 = r/\|r\|_2$
2: **for** $j = 1, 2, \ldots, m-1$ **do**
3: $\quad w_j = Av_j$
4: $\quad$ **for** $i = 1, 2, \ldots, j$ **do**
5: $\quad\quad h_{ij} = (w_j, v_i)$
6: $\quad\quad w_j = w_j - h_{ij}v_i$
7: $\quad$ **end for**
8: $\quad h_{j+1,j} = \|w_j\|_2$
9: $\quad$ **if** $h_{j+1,j} = 0$ **then** break, **end if**
10: $\quad v_{j+1} = w_j/h_{j+1,j}$
11: **end for**

注记

- 如果计算到第 k $(k < m)$ 步时有 $h_{k+1,k} = 0$, 则方法会 **提前终止**.
  此时 Av_k 必定可以由 $v_1, v_2, \ldots, v_k$ 线性表出 (不考虑舍入误差)
- 算法中的向量 v_i 称为 **Arnoldi 向量**.
  需要指出的是, 在算法中我们是用 A 乘以 v_j, 然后与之前的 Arnoldi 向量正交化, 而不是用 $A^j r$.

引理 如果 Arnoldi 过程不提前终止 (即 $h_{k+1,k} \neq 0$, $k = 1, 2, \ldots, m-1$), 则

$$v_j \in \mathcal{K}_j(A, r), \quad j = 1, 2, \ldots, m.$$

(板书)

$\mathcal{K}_m$ 的标准正交基

定理 如果 Arnoldi 过程不提前终止, 则向量 $v_1, v_2, \ldots, v_m$ 构成 $\mathcal{K}_m$ 的一组标准正交基, 这里

$$\mathcal{K}_m(A, r) = \mathrm{span}\{r, Ar, \ldots, A^{m-1}r\}.$$

Arnoldi 过程的矩阵表示

由 Arnoldi 过程可知 $h_{j+1,j}v_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij}v_i$, 因此

$$
\begin{aligned}
Av_j &= h_{j+1,j}v_{j+1} + \sum_{i=1}^{j} h_{ij}v_i \\
&= [v_1, v_2, \ldots, v_{j+1}] \begin{bmatrix} h_{1j} \\ h_{2j} \\ \vdots \\ h_{j+1,j} \end{bmatrix} = [v_1, \ldots, v_{j+1}, v_{j+2}, \ldots, v_{m+1}] \begin{bmatrix} h_{1j} \\ \vdots \\ h_{j+1,j} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\
&= V_{m+1}H_{m+1,m}(:, j),
\end{aligned}
$$

定理 设 $V_m = [v_1, v_2, \ldots, v_m]$, 则

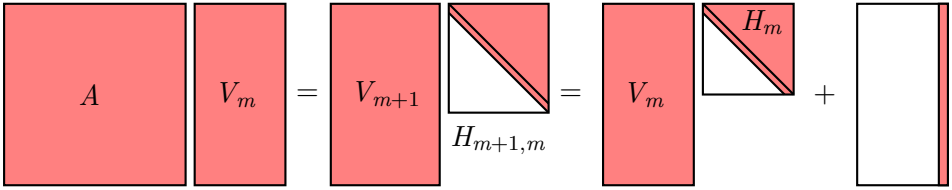
$$A V_m = V_{m+1} H_{m+1,m} = V_m H_m + h_{m+1,m} v_{m+1} e_m^{\mathsf{T}}, \tag{2.3}$$

$$V_m^{\mathsf{T}} A V_m = H_m, \tag{2.4}$$

其中 $e_m = [0, \ldots, 0, 1]^{\mathsf{T}} \in \mathbb{R}^m$, $H_m = H_{m+1,m}(1:m, 1:m) \in \mathbb{R}^{m\times m}$

$$H_{m+1,m} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1,m-1} & h_{1,m} \\ h_{21} & h_{22} & \cdots & h_{2,m-1} & h_{2,m} \\ 0 & h_{32} & \cdots & h_{3,m-1} & h_{2,m} \\ 0 & 0 & \cdots & h_{4,m-1} & h_{4,m} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & h_{m,m-1} & h_{m,m} \\ 0 & 0 & \cdots & 0 & h_{m+1,m} \end{bmatrix} \in \mathbb{R}^{(m+1)\times m}.$$

定理中的结论可以由下图表示.



Arnoldi 过程提前终止

如果 Arnoldi 过程提前终止, 则我们可以得到一个不变子空间.

定理 如果 Arnoldi 过程在第 k $(k \leq m)$ 步终止, 即 $h_{k+1,k} = 0$, 则有

$$A V_k = V_k H_k,$$

即 $\mathcal{K}_k$ 是 A 的一个不变子空间.

Krylov 子空间方法

Krylov 子空间方法的一般流程:

(1) 令 $m = 1$

(2) 定义 Krylov 子空间 $\mathcal{K}_m(A, r_0)$;

(3) 在 **仿射空间** $x^{(0)} + \mathcal{K}_m$ 中找出原问题的近似解 $(b - A\tilde{x} \perp \mathcal{L})$;

(4) 如果这个近似解满足精度要求, 则迭代结束;
否则令 $m \leftarrow m + 1$, 返回第 (2) 步.

Krylov 子空间方法的基本框架

1: 选取初始向量 $x^{(0)}$
2: 计算 $r_0 = b - Ax^{(0)}$, $v_1 = r_0/\|r_0\|_2$
3: 寻找近似解: $x^{(1)} \in x^{(0)} + \mathcal{K}_1$
4: **if** $x^{(1)}$ 满足精度要求 **then**
5: 　　终止迭代
6: **end if**
7: **for** $m = 2$ to n **do**
8: 　　调用 Arnoldi 过程计算向量 v_m
9: 　　寻找近似解: $x^{(m)} \in x^{(0)} + \mathcal{K}_m$ 　　$(b - Ax^{(m)} \perp \mathcal{L})$
10: 　　**if** $x^{(m)}$ 满足精度要求 **then**
11: 　　　　终止迭代
12: 　　**end if**
13: **end for**

2-3 | 非对称线性方程组

2.3 非对称线性方程组

2-3-1 完全正交方法 (FOM)

取 $\mathcal{L}_m = \mathcal{K}_m$, 得到的方法为**完全正交方法** (Full Orthogonalization Method)

$$\text{find} \quad \tilde{x} \in x^{(0)} + \mathcal{K}_m \quad \text{such that} \quad b - A\tilde{x} \perp \mathcal{K}_m$$

2-3-1 完全正交方法 (FOM)

取 $\mathcal{L}_m = \mathcal{K}_m$, 得到的方法为**完全正交方法** (Full Orthogonalization Method)

$$\text{find} \quad \tilde{x} \in x^{(0)} + \mathcal{K}_m \quad \text{such that} \quad b - A\tilde{x} \perp \mathcal{K}_m$$

设 $\tilde{x} = x^{(0)} + V_m \tilde{y}$, 其中 $\tilde{y} \in \mathbb{R}^m$. 由正交性条件 $b - A\tilde{x} \perp \mathcal{K}_m$ 可知

$$\begin{aligned} 0 = V_m^{\mathsf{T}}(b - A\tilde{x}) &= V_m^{\mathsf{T}}(b - Ax^{(0)} - A V_m \tilde{y}) \\ &= V_m^{\mathsf{T}} r_0 - V_m^{\mathsf{T}} A V_m \tilde{y} \\ &= V_m^{\mathsf{T}}(\beta v_1) - V_m^{\mathsf{T}} A V_m \tilde{y} = \beta e_1 - H_m \tilde{y} \end{aligned}$$

如果 H_m 非奇异, 则 $\tilde{y} = H_m^{-1} e_1$. 所以

$$\tilde{x} = x^{(0)} + \beta V_m H_m^{-1} e_1.$$

算法 完全正交方法 (FOM)

1: 给定初值 $x^{(0)}$, 计算 $r_0 = b - Ax^{(0)}$ 和 $\beta = \|r_0\|_2$
2: 计算 $v_1 = r_0/\beta$
3: **for** $j = 1, 2, \ldots, m-1$ **do** % 开始迭代
4: $w_j = Av_j$
5: **for** $i = 1, 2, \ldots, j$ **do** % Arnoldi 过程
6: $h_{ij} = (w_j, v_i), \quad w_j = w_j - h_{ij}v_i$
7: **end for**
8: $h_{j+1,j} = \|w_j\|_2$
9: **if** $h_{j+1,j} = 0$, **then** set $m = j$ and break, **end if** % Arnoldi 过程提前终止
10: $v_{j+1} = w_j/h_{j+1,j}$
11: **end for**
12: 求解线性方程组 $H_m\tilde{y} = \beta e_1$
13: 计算近似解 $\tilde{x} = x^{(0)} + V_m\tilde{y}$

注记

- 由于 m 通常远远小于 n, 因此 $H_m\tilde{y} = \beta e_1$ 可通过直接法求解, 比如列主元 LU 分解或基于 Givens 变换的 QR 分解.
- FOM 方法的一个难点是如何选取 m.
  实际计算时, 我们采用动态调整的方法, 不断增长 m 的值, 直到残量满足精度要求.
  因此, 在每一次迭代后, 需要估计残量的范数.

残量的计算

定理 设 $\tilde{x} \in x^{(0)} + \mathcal{K}_m$ 是由 FOM 算法得到的近似解, 则

$$\tilde{r} \triangleq b - A\tilde{x} = -h_{m+1,m}\left(e_m^{\mathsf{T}}\tilde{y}\right) v_{m+1},$$

其中 $\tilde{y} = \beta H_m^{-1} e_1$, $e_m = [0, \ldots, 0, 1]^{\mathsf{T}} \in \mathbb{R}^m$. 因此,

$$\|\tilde{r}\|_2 = h_{m+1,m}|e_m^{\mathsf{T}}\tilde{y}| = h_{m+1,m}|\tilde{y}(m)|.$$

(板书)

注记

- 每次迭代结束我们就可以直接计算出残量 $\tilde{r}$ 的范数, 当满足精度要求时, 我们再计算近似解 $\tilde{x}$.
- 在不考虑舍入误差的情况下, 当 $m = n$ 时, 必有 $\|\tilde{r}\|_2 = 0$.
- 如果 Arnoldi 过程提前终止, 即存在整数 k $(k < n)$ 使得 $h_{k+1,k} = 0$, 则由定理可知, 此时 $\tilde{r} = 0$, 因此近似解 $\tilde{x}$ 就是方程组的精确解.

算法 实用的 FOM 方法

1: 给定初值 $x^{(0)}$ 和 (相对) 精度要求 $\varepsilon > 0$
2: 计算 $r_0 = b - Ax^{(0)}$, $\beta = \|r_0\|_2$ 和 $v_1 = r_0/\beta$
3: **for** $j = 1, 2, \ldots$ **do**　　% 这里可以设置最大迭代步数
4:　　$w_j = Av_j$
5:　　**for** $i = 1, 2, \ldots, j$ **do**
6:　　$h_{ij} = (w_j, v_i)$,　$w_j = w_j - h_{ij}v_i$
7:　　**end for**
8:　　$h_{j+1,j} = \|w_j\|_2$
9:　　求解方程 $H_j\tilde{y} = \beta e_1$
10:　　**if** $h_{j+1,j}|\tilde{y}(j)|/\beta < \varepsilon$ **then**　% relative residual
11:　　break
12:　　**end if**
13:　　$v_{j+1} = w_j/h_{j+1,j}$
14: **end for**
15: 计算近似解 $\tilde{x} = x^{(0)} + V_j\tilde{y}$

2-3-2 广义极小残量法 (GMRES)

约束空间取为 $\mathcal{L} = A\mathcal{K}$, 则可得到 GMRES 方法:

$$\text{寻找}\quad \tilde{x} \in x^{(0)} + \mathcal{K} \quad \text{满足}\quad b - A\tilde{x} \perp A\mathcal{K}. \tag{2.5}$$

广义极小残量法 (Generalized Minimum Residual, GMRES) 是当前最受欢迎的 Krylov 子空间方法之一, 也是求解非对称线性方程组的首选方法.

GMRES 的最优性质

定理 设 $A \in \mathbb{R}^{n\times n}$, 则 $\tilde{x}$ 是 (2.5) 的解当且仅当 $\tilde{x}$ 是下面的极小化问题的解

$$\min_{x\in x^{(0)}+\mathcal{K}} \|b - Ax\|_2, \tag{2.6}$$

即在仿射空间 $x^{(0)} + \mathcal{K}$ 中, $\tilde{x}$ 所对应的残量范数最小.

(板书)

近似解的计算

对任意 $x \in x^{(0)} + \mathcal{K}_m$, 存在向量 $y \in \mathbb{R}^m$ 使得 $x = x^{(0)} + V_m y$, 于是

$$\begin{aligned} b - Ax &= b - A(x^{(0)} + V_m y) \\ &= r_0 - A V_m y = \beta v_1 - V_{m+1} H_{m+1,m} y = V_{m+1}(\beta e_1 - H_{m+1,m} y), \end{aligned}$$

其中 $e_1 = [1, 0, \ldots, 0]^{\mathsf{T}} \in \mathbb{R}^{m+1}$.

近似解的计算

对任意 $x \in x^{(0)} + \mathcal{K}_m$, 存在向量 $y \in \mathbb{R}^m$ 使得 $x = x^{(0)} + V_m y$, 于是

$$\begin{aligned} b - Ax &= b - A(x^{(0)} + V_m y) \\ &= r_0 - A V_m y = \beta v_1 - V_{m+1} H_{m+1,m} y = V_{m+1}(\beta e_1 - H_{m+1,m} y), \end{aligned}$$

其中 $e_1 = [1, 0, \ldots, 0]^{\mathsf{T}} \in \mathbb{R}^{m+1}$. 又 V_{m+1} 的列向量是标准正交的, 故

$$\|b - Ax\|_2 = \| V_{m+1}(\beta e_1 - H_{m+1,m} y)\|_2 = \|\beta e_1 - H_{m+1,m} y\|_2.$$

因此, 最小化问题 (2.6) 的解为 $\tilde{x} = x^{(0)} + V_m \tilde{y}$, 其中 $\tilde{y}$ 是下面最小二乘问题的解

$$\min_{y \in \mathbb{R}^m} \|\beta e_1 - H_{m+1,m} y\|_2$$

最小二乘问题求解

通常 m 不是很大, 因此可以使用 QR 方法来求解.

最小二乘问题求解

通常 m 不是很大, 因此可以使用 QR 方法来求解.

设 $H_{m+1,m} = Q_{m+1}^{\mathsf{T}} R_{m+1,m}$, 于是

$$\begin{aligned}\|\beta e_1 - H_{m+1,m} y\|_2 &= \|\beta e_1 - Q_{m+1}^{\mathsf{T}} R_{m+1,m} y\|_2 \\ &= \|\beta Q_{m+1} e_1 - R_{m+1,m} y\|_2 = \left\| \beta q_1 - \begin{bmatrix} R_m \\ 0 \end{bmatrix} y \right\|_2,\end{aligned}$$

其中 q_1 是 Q_{m+1} 的第一列, $R_m \in \mathbb{R}^{m\times m}$ 表示 $R_{m+1,m}$ 的前 m 行.

最小二乘问题求解

通常 m 不是很大, 因此可以使用 QR 方法来求解.

设 $H_{m+1,m} = Q_{m+1}^{\mathsf{T}} R_{m+1,m}$, 于是

$$\begin{aligned}\|\beta e_1 - H_{m+1,m} y\|_2 &= \|\beta e_1 - Q_{m+1}^{\mathsf{T}} R_{m+1,m} y\|_2 \\ &= \|\beta Q_{m+1} e_1 - R_{m+1,m} y\|_2 = \left\| \beta q_1 - \begin{bmatrix} R_m \\ 0 \end{bmatrix} y \right\|_2,\end{aligned}$$

其中 q_1 是 Q_{m+1} 的第一列, $R_m \in \mathbb{R}^{m\times m}$ 表示 $R_{m+1,m}$ 的前 m 行.

所以 y 可以通过求解下面的上三角方程组获得

$$R_m \tilde{y} = \beta q_1(1:m) \tag{2.7}$$

其中 $q_1(1:m)$ 表示 q_1 的前 m 个元素组成的向量.

残量的计算

定理 设 $\tilde{x} \in x^{(0)} + \mathcal{K}_m$ 是由 GMRES 方法所得到的近似解, 则

$$\|\tilde{r}\|_2 = \|b - A\tilde{x}\|_2 = \beta\,|q_1(m+1)|,$$

其中 $q_1(m+1)$ 表示 $q_1 \in \mathbb{R}^{m+1}$ 的最后一个分量. (板书)

算法 广义极小残量法 (GMRES)

1: 给定初值 $x^{(0)}$ 和 (相对) 精度要求 $\varepsilon > 0$
2: 计算 $r_0 = b - Ax^{(0)}$, $\beta = \|r_0\|_2$ 和 $v_1 = r_0/\beta$
3: **for** $j = 1, 2, \ldots$ **do**　　% 开始迭代, 可以设置最大迭代步数
4:　　$w_j = Av_j$
5:　　**for** $i = 1, 2, \ldots, j$ **do**　　% Arnoldi process
6:　　$h_{ij} = (w_j, v_i)$,　　$w_j = w_j - h_{ij}v_i$
7:　　**end for**
8:　　$h_{j+1,j} = \|w_j\|_2$
9:　　**if** $h_{j+1,j} = 0$, **then** set $m = j$ and break, **end if**　　% Arnoldi 过程提前终止
10:　　$v_{j+1} = w_j/h_{j+1,j}$
11:　　**if** $\|\tilde{r}\|_2/\beta < \varepsilon$, **then** set $m = j$ and break, **end if**　　% check convergence
12: **end for**
13: 求解上三角线性方程组 $R_m\tilde{y} = \beta q_1(1:m)$　　% 回代求解
14: 计算近似解 $\tilde{x} = x^{(0)} + V_m\tilde{y}$

GMRES 的提前终止

如果在第 k $(k < n)$ 步时有 $h_{k+1,k} = 0$, 则 GMRES 会提前终止, 此时

$$A V_k = V_k H_k, \quad \tilde{y} = H_k^{-1}(\beta e_1).$$

因此

$$\begin{aligned} \|\tilde{r}\|_2 = \|b - A\tilde{x}\|_2 &= \|b - Ax^{(0)} - A V_k \tilde{y}\|_2 \\ &= \|r_0 - V_k H_k \tilde{y}\|_2 = \|\beta v_1 - V_k(\beta e_1)\|_2 = 0. \end{aligned}$$

这意味着 $\tilde{x}$ 就是原方程组的精确解. 因此, **这种提前终止是好事**.

反之, 如果在第 k 步时有 $\tilde{r} = b - A\tilde{x} = 0$, 则必有 $h_{k+1,k} = 0$.

定理 设 $A \in \mathbb{R}^{n\times n}$ 非奇异, 则 GMRES 方法在第 k 步提前终止的充要条件是近似解即为精确解. (留作练习)

2-3-3 GMRES 方法的实施细节

需要解决的问题

(1) 残量范数计算: $\|\tilde{r}\|_2 = \beta|q_1(m+1)|$ $\rightarrow$ 计算 $H_{j+1,j}$ 的 QR 分解.

(2) 最后的求解: $R_m\tilde{y} = \beta q_1(1:m)$ $\rightarrow$ 计算 $H_{j+1,j}$ 的 QR 分解.

设计高效的 $H_{j+1,j}$ 的 QR 分解算法 ⬅ ❶ Givens 变换; ❷ 递推方法.

$H_{j+1,j}$ 的 QR 分解的递推算法

1. 当 $j=1$ 时, 对 $H_{2,1}\in\mathbb{R}^{2\times 1}$ 直接做一次 Givens 变换即可得到其 QR 分解.

$H_{j+1,j}$ 的 QR 分解的递推算法

1. 当 $j=1$ 时, 对 $H_{2,1}\in\mathbb{R}^{2\times 1}$ 直接做一次 Givens 变换即可得到其 QR 分解.
2. 假定 $H_{j,j-1}\in\mathbb{R}^{j\times(j-1)}$ 的 QR 分解 (基于 Givens 变换) 为

$$H_{j,j-1}=(G_{j-1}G_{j-2}\cdots G_1)^{\mathsf{T}}R_{j,j-1}=Q_j^{\mathsf{T}}\begin{bmatrix}R_{j-1}\\0\end{bmatrix}_{j\times(j-1)}$$

其中 $R_{j-1}\in\mathbb{R}^{(j-1)\times(j-1)}$ 是上三角矩阵.

$H_{j+1,j}$ 的 QR 分解的递推算法

❶ 当 $j=1$ 时, 对 $H_{2,1}\in\mathbb{R}^{2\times 1}$ 直接做一次 Givens 变换即可得到其 QR 分解.

❷ 假定 $H_{j,j-1}\in\mathbb{R}^{j\times(j-1)}$ 的 QR 分解 (基于 Givens 变换) 为

$$H_{j,j-1}=(G_{j-1}G_{j-2}\cdots G_1)^{\mathsf{T}}R_{j,j-1}=Q_j^{\mathsf{T}}\begin{bmatrix}R_{j-1}\\0\end{bmatrix}_{j\times(j-1)}$$

其中 $R_{j-1}\in\mathbb{R}^{(j-1)\times(j-1)}$ 是上三角矩阵.

❸ 考虑 $H_{j+1,j}=\begin{bmatrix}H_{j,j-1}&h_j\\0&h_{j+1,j}\end{bmatrix}$ 的 QR 分解, 其中 h_j 是 $H_{j+1,j}$ 最后一列的前 j 行.

$$\begin{bmatrix}Q_j&0\\0&1\end{bmatrix}H_{j+1,j}=\begin{bmatrix}Q_jH_{j,j-1}&h_j\\0&h_{j+1,j}\end{bmatrix}=\begin{bmatrix}\begin{bmatrix}R_{j-1}\\0\end{bmatrix}&Q_jh_j\\0&h_{j+1,j}\end{bmatrix}=\begin{bmatrix}R_{j-1}&\tilde{h}_{j-1}\\0&\hat{h}_{j,j}\\0&h_{j+1,j}\end{bmatrix},$$

其中 $\tilde{h}_{j-1}$ 是 Q_jh_j 的前 $j-1$ 行, $\hat{h}_{j,j}$ 是 Q_jh_j 的最后一个元素.

下面利用 Givens 变换, 将 $h_{j+1,j}$ 化为 0. 计算

$$\tilde{h}_{j,j} = \sqrt{\hat{h}_{j,j}^2 + h_{j+1,j}^2}, \quad c_j = \frac{\hat{h}_{j,j}}{\tilde{h}_{j,j}}, \quad s_j = \frac{h_{j+1,j}}{\tilde{h}_{j,j}}.$$

构造 Givens 变换 $G_j = \begin{bmatrix} I_{j-1} & 0 & 0 \\ 0 & c_j & s_j \\ 0 & -s_j & c_j \end{bmatrix}$, **令** $Q_{j+1} = G_j \begin{bmatrix} Q_j & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{(j+1)\times(j+1)}$, **则**

$$Q_{j+1}H_{j+1,j} = G_j \begin{bmatrix} R_{j-1} & \tilde{h}_{j-1} \\ 0 & \hat{h}_{j,j} \\ 0 & h_{j+1,j} \end{bmatrix} = \begin{bmatrix} R_{j-1} & \tilde{h}_{j-1} \\ 0 & \tilde{h}_{j,j} \\ 0 & 0 \end{bmatrix} \triangleq R_{j+1,j}.$$

所以 $H_{j+1,j} = Q_{j+1}^{\mathsf{T}} R_{j+1,j}$ **就是** $H_{j+1,j}$ **的 QR 分解.**

主要工作: ❶ 计算 $Q_j h_j$ **❷ 计算 Givens 变换** $G(c_j, s_j)$ **并更新** $\tilde{h}_{j,j}$

需要解决的问题

(1) **残量范数计算**: $\|\tilde{r}\|_2 = \beta|q_1(j+1)|$

☞ $q_1 = Qe_1 = G_j(G_{j-1}\cdots G_1 e_1)$, **逐次计算, 每次迭代只需一次** Givens **变换**

(2) **最后的求解**: $R_m\tilde{y} = \beta q_1(1:m)$

☞ **逐次计算** R **的第** j **列**: $G_j\cdots G_1 H(1:j+1,j)$, **每次迭代需** j **次** Givens **变换**

需要解决的问题

(1) 残量范数计算: $\|\tilde{r}\|_2 = \beta|q_1(j+1)|$

☞ $q_1 = Qe_1 = G_j(G_{j-1}\cdots G_1 e_1)$, 逐次计算, 每次迭代只需一次 Givens 变换

(2) 最后的求解: $R_m\tilde{y} = \beta q_1(1:m)$

☞ 逐次计算 R 的第 j 列: $G_j\cdots G_1 H(1:j+1,j)$, 每次迭代需 j 次 Givens 变换

实施细节

- H, Q 和 R 的维数都是随着迭代步数的增加而不断变大
- 实际计算时:
  1. 不用计算 Q, 保存所有 Givens 变换即可
  2. R 可以直接存放在 H 中.

算法 实用的 GMRES 方法

给定初值 $x^{(0)}$ 和 (相对) 精度要求 $\varepsilon > 0$

计算 $r_0 = b - Ax^{(0)}$, $\beta = \|r_0\|_2$ 和 $v_1 = r_0/\beta$

令 $\xi = \beta e_1$ % 用于计算和存放 q_1

for $j = 1, 2, \ldots$ **do** % 开始迭代, 可以设置最大迭代步数

  $w_j = Av_j$

  for $i = 1, 2, \ldots, j$ **do** % Arnoldi process

    $h_{ij} = (w_j, v_i), \quad w_j = w_j - h_{ij}v_i$

  end for

  $h_{j+1,j} = \|w_j\|_2$

  for $i = 1, 2, \ldots, j-1$ **do** % Apply $G_{j-1}, \ldots, G_1$ to the last column of $H_{j+1,j}$

$$\begin{bmatrix} h_{i,j} \\ h_{i+1,j} \end{bmatrix} = \begin{bmatrix} c_i & s_i \\ -s_i & c_i \end{bmatrix} \begin{bmatrix} h_{i,j} \\ h_{i+1,j} \end{bmatrix}$$

  end for

  if $h_{j+1,j} = 0$, **then** set $m = j$ and break, **end if** % Arnoldi 过程提前终止

  $v_{j+1} = w_j/h_{j+1,j}$

15: **if** $|h_{j,j}| > |h_{j+1,j}|$ **then** % Form the Givens rotation G_j

16: $c_j = \frac{1}{\sqrt{1+\tau^2}}$, $s_j = c_j\tau$ where $\tau = \frac{h_{j+1,j}}{h_{j,j}}$

17: **else**

18: $s_j = \frac{1}{\sqrt{1+\tau^2}}$, $c_j = s_j\tau$ where $\tau = \frac{h_{j,j}}{h_{j+1,j}}$

19: **end if**

20: $h_{j,j} = c_j h_{j,j} + s_j h_{j+1,j}$, $h_{j+1,j} = 0$ % Apply G_j to last column of $H_{j+1,j}$

21: $\begin{bmatrix} \xi_j \\ \xi_{j+1} \end{bmatrix} = \begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix} \begin{bmatrix} \xi_j \\ 0 \end{bmatrix}$ % Apply G_j to the right-hand side

22: **if** $|\xi_{j+1}|/\beta < \varepsilon$, **then** set $m = j$ and break, **end if** % Check convergence

23: **end for**

24: 计算 $\tilde{y} = R_m^{-1}\xi^{(m)}$, 其中 $R_m = H(1:m, 1:m)$, $\xi^{(m)} = \xi(1:m)$

25: 计算近似解 $\tilde{x} = x^{(0)} + V_m\tilde{y}$

- ☞ 注 1: 运算量主要集中在矩阵向量乘积, 即 Av_j.
- ☞ 注 2: 在某些特定场合, Arnoldi 过程中可使用 Householder 变换以增加稳定性.

2-3-4 带重启的 GMRES

为什么要重启

- 如果不考虑舍入误差, 则 GMRES 迭代 n 步后残量为 0, 因此肯定收敛.
- 但对于大规模的线性方程组, 我们不可能迭代 n 步, 因为 GMRES 每一步迭代的运算量和存储量都会随着迭代步数的增加而增加, 所以当迭代步数很大时, 运算时间和存储量会大大增加, 这使得方法失去竞争性.
- 因此我们希望将迭代步数控制在一个能接受的范围内. 目前实现这一目标的通用做法是采用**重启**策略, 即给定一个固定迭代步数 m (通常远远小于 n, 如 20, 50 等) 如果 GMRES 迭代到 m 步后仍不收敛, 则计算出近似解 $x^{(m)} \in x^{(0)} + \mathcal{K}_m$. 然后将其作为新的迭代初值, 即令 $x^{(0)} = x^{(m)}$, 重新启动 GMRES 方法. 该过程可以重复执行, 直到找到满意的近似解为止.

算法 GMRES(m): **带重启的** GMRES **方法**

1: **给定正整数** $m \ll n$, **初值** $x^{(0)}$ **和 (相对) 精度要求** $\varepsilon > 0$
2: **计算** $r_0 = b - Ax^{(0)}$ **和** $\beta = \|r_0\|_2$
3: $v_1 = r_0/\beta$
4: **for** $j = 1, 2, \ldots, m$ **do**
5: 　　**执行** GMRES **算法的第** 5 **步至第** 22 **步**
6: **end for**
7: **计算** $\tilde{y} = R_m^{-1}\xi^{(m)}$, **其中** $R_m = H(1:m, 1:m)$, $\xi^{(m)} = \xi(1:m)$
8: **计算近似解** $\tilde{x} = x^{(0)} + V_m\tilde{y}$
9: **if** $|\xi_{m+1}|/\beta < \varepsilon$ **then**
10: 　　stop and output the results
11: **end if**
12: **令** $x^{(0)} = x^{(m)}$, **返回第** 2 **步**

重启带来的不利因素

- 加入重启技术可以节省运算量和存储量, 但随之也会带来一些负面效应. 如收敛速度会减慢, 这是因为在重启方法时, 一些有用信息会被丢弃.
- 如果 A 不是正定矩阵, 则可能还会出现方法停滞不前的情形, 即残量范数很难减小. 此时即使总迭代步数已经达到 n 步, 或者甚至超过 n 步, 但方法仍然不收敛.
- 另外, 怎样选取合适的重启步数 m, 目前仍然没有很好的办法.

举例 (一)

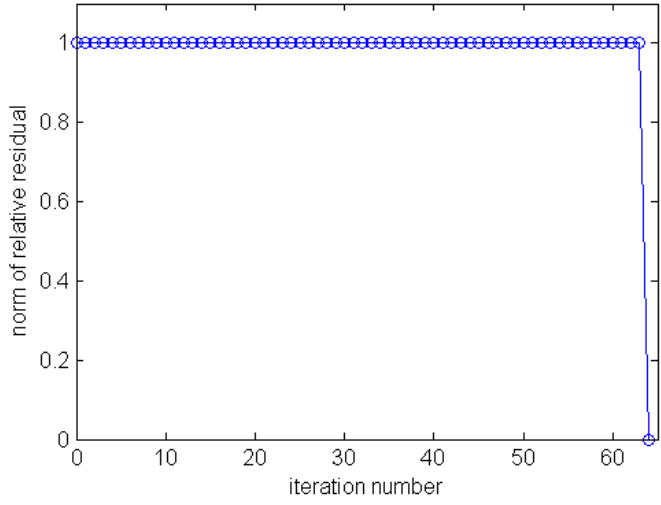
由 GMRES 的最优性质可知, 残量范数 $\|r_k\|_2$ 递减, 但不一定严格递减. 在最坏的情况下, 可能会出现 $\|r_k\|_2$ 保持不变, 直到最后一步才降为 0.

例 考虑线性方程组 $Ax = b$, 其中

$$A = \begin{bmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ a_1 & a_2 & \cdots & a_{n-1} & a_n \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}.$$

求解过程参见 MATLAB 程序 GMRES_Example01.m.

$n = 64$ 时的 (相对) 残量范数下降曲线



对于该问题, 重启 GMRES 将导致算法失败.

举例 (二)

例 考虑下面的带齐次 Dirichlet 边界条件的偏微分方程:

$$
\begin{aligned}
&-\Delta u(x,y)+u_x(x,y)+u_y(x,y)+u(x,y)=f(x,y),\\
&\quad (x,y)\in\Omega\triangleq[0,1]\times[0,1]\\
&\quad u(x,0)=u(x,1)=u(0,y)=u(1,y)=0,\quad 0<x,y<1.
\end{aligned}
$$

用五点差分离散, 步长 $h_x=h_y=1/(N+1)$, 可得

$$
\frac{4u_{i,j}-u_{i+1,j}-u_{i-1,j}-u_{i,j+1}-u_{i,j-1}}{h^2}+\frac{u_{i+1,j}-u_{i-1,j}}{2h}+\frac{u_{i,j+1}-u_{i,j-1}}{2h}+u_{i,j}=f_{i,j}.
$$

对应的离散线性方程组为

$$
[I\otimes T+T\otimes I+h(I\otimes D)+h(D\otimes I)+h^2I]u=h^2f.
$$

其中 $T=\mathrm{tridiag}(-1,2,-1)$, $D=\mathrm{tridiag}(-1/2,0,1/2)$.

- 在测试中, 我们假设解析解为

$$u(x,y) = xy(1-x)(1-y).$$

由此可以求出右端项

$$f(x,y) = (3-2x)(1-y)y + (3-2y)(1-x)x + x(1-x)y(1-y).$$

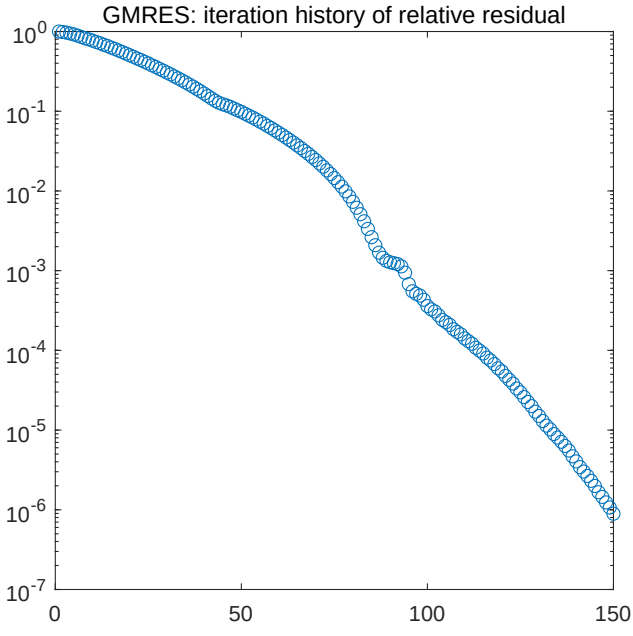
由于系数矩阵是非对称的, 因此用 GMRES 方法求解.

- 最大迭代步数设为 $\mathrm{IterMax} = N^2$, 迭代终止条件

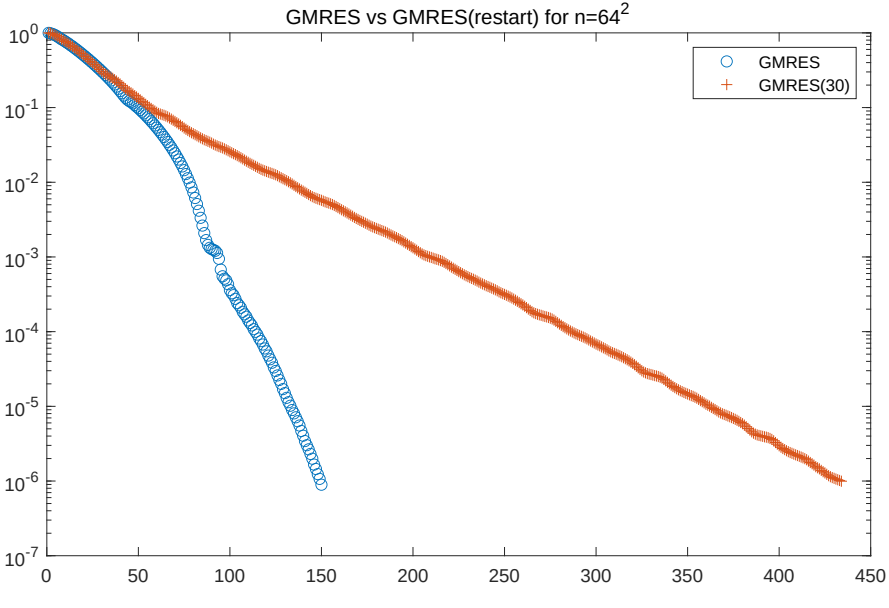
$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq \mathrm{tol} \triangleq 10^{-6}.$$

- MATLAB 程序见 GMRES_Example02_PDE.m

$n = 32^2$ 时的相对残量下降曲线.



GMRES vs GMRES(restart) for $n = 64^2$.



2-4 | 对称线性方程

2.4 对称线性方程

2-4-1 Lanczos 过程

如果 A 是对称的, 则由 Arnoldi 过程所生成的上 Hessenberg 矩阵 $H_m = V_m^{\mathsf{T}} A V_m$ 也对称, 所以是**对称三对角矩阵**.

- 为了书写方便, 我们记 $\alpha_j = h_{j,j}$, $\beta_j = h_{j,j+1}$, 并用 T 来表示, 即

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix}.$$

三项递推公式

与 Arnoldi 过程一样, 我们有下面的性质

$$A V_k = V_k T_k + \beta_k v_{k+1} e_k^{\mathsf{T}}, \tag{2.8}$$

$$V_k^{\mathsf{T}} A V_k = T_k. \tag{2.9}$$

考察关系式 (2.8) 两边的第 j 列可知

$$\beta_j v_{j+1} = A v_j - \alpha_j v_j - \beta_{j-1} v_{j-1}, \quad j = 1, 2, \ldots,$$

这里我们令 $v_0 = 0$ 和 $\beta_0 = 0$.

Lanczos 过程

算法 基于三项递推公式的 Lanczos 过程

1: 给定非零向量 r, 令 $v_0 = 0$, $\beta_0 = 0$
2: 计算 $v_1 = r/\|r\|_2$
3: **for** $j = 1, 2, \ldots$ **do**
4: $\quad w_j = Av_j - \beta_{j-1} v_{j-1}$ % 三项递推公式
5: $\quad \alpha_j = (w_j, v_j)$
6: $\quad \tilde{w}_j = w_j - \alpha_j v_j$
7: $\quad \beta_j = \|\tilde{w}_j\|_2$
8: $\quad$ **if** $\beta_j = 0$ **then**
9: $\quad\quad$ break
10: $\quad$ **end if**
11: $\quad v_{j+1} = \tilde{w}_j/\beta_j$
12: **end for**

Lanczos 向量的正交性

如果不考虑舍入误差, 则 Lanczos 过程生成的向量是相互正交的.

定理 设 $v_1, v_2, \ldots, v_k, \ldots$ 由 Lanczos 算法生成, 则

$$(v_i, v_j) = \delta_{ij} \triangleq \begin{cases} 1, & i = j, \\ 0, & i \neq j, \end{cases} \qquad \text{for } i, j = 1, 2, \ldots.$$

(留作课外自习)

注记

- 在 Lanczos 过程中, 我们只需正交化相邻的三个向量即可. 因此在计算过程中只需存储三个向量, 而且每个迭代步的运算量也是固定不变的.
- 在实际计算中, 由于舍入误差的原因, Lanczos 算法生成的向量可能会逐渐失去正交性. 这时需要采取一些补救措施, 如重新全部正交化或部分正交化.

2-4-2 共轭梯度法 (CG)

共轭梯度法 (Conjugate Gradient, CG) 是当前求解大规模**对称正定**线性方程组的首选方法. 事实上, CG 方法与 FOM 方法是等价的, 即选取的约束空间为 $\mathcal{L} = \mathcal{K}$. 但由于 A 是对称正定的, 因此我们可以借助三项递推公式来简化运算.

2-4-2 共轭梯度法 (CG)

共轭梯度法 (Conjugate Gradient, CG) 是当前求解大规模**对称正定**线性方程组的首选方法. 事实上, CG 方法与 FOM 方法是等价的, 即选取的约束空间为 $\mathcal{L} = \mathcal{K}$. 但由于 A 是对称正定的, 因此我们可以借助三项递推公式来简化运算.

共轭梯度法

设 $x^{(0)}$ 是迭代初值, 我们需要在仿射空间 $x^{(0)} + \mathcal{K}_k$ 中寻找近似解 $x^{(k)}$, 满足

$$x^{(k)} \in x^{(0)} + \mathcal{K}_k \quad \text{且} \quad b - Ax^{(k)} \perp \mathcal{K}_k.$$

计算公式推导

与 FOM 方法类似, 我们可得

$$x^{(k)} = x^{(0)} + V_k y_k, \qquad y_k = T_k^{-1}(\beta e_1^{(k)}),$$

其中 $e_1^{(k)} = [1, 0, \ldots, 0]^{\mathsf{T}} \in \mathbb{R}^k$.

- 由于 A 对称正定, 所以 T_k 也对称正定, 因此可以使用 $\mathrm{LDL}^{\mathsf{T}}$ 分解进行求解.
- 设 T_k 的 $\mathrm{LDL}^{\mathsf{T}}$ 分解为 $T_k = L_k D_k L_k^{\mathsf{T}}$, 于是可得

$$x^{(k)} = x^{(0)} + V_k y_k = x^{(0)} + V_k T_k^{-1}(\beta e_1^{(k)}) = x^{(0)} + (V_k L_k^{-\mathsf{T}})(\beta D_k^{-1} L_k^{-1} e_1^{(k)}).$$

- 如果 $x^{(k)}$ 满足精度要求, 则计算结束, 否则需要计算

$$x^{(k+1)} = x^{(0)} + V_{k+1} T_{k+1}^{-1}(\beta e_1^{(k+1)}) = x^{(0)} + (V_{k+1} L_{k+1}^{-\mathsf{T}})(\beta D_{k+1}^{-1} L_{k+1}^{-1} e_1^{(k+1)}),$$

这里假定 T_{k+1} 的 $\mathrm{LDL}^{\mathsf{T}}$ 分解为 $T_{k+1} = L_{k+1} D_{k+1} L_{k+1}^{\mathsf{T}}$.

计算 $x^{(k+1)}$ 的递推方法

记

$$\tilde{P}_k \triangleq V_k L_k^{-\mathsf{T}} = [\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_k] \in \mathbb{R}^{n\times k},$$
$$\tilde{y}_k \triangleq \beta D_k^{-1} L_k^{-1} e_1^{(k)} = [\eta_1, \ldots, \eta_k]^{\mathsf{T}} \in \mathbb{R}^k. \quad k = 1, 2, \ldots.$$

计算 $x^{(k+1)}$ 的递推方法

记

$$\tilde{P}_k \triangleq V_k L_k^{-\mathsf{T}} = [\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_k] \in \mathbb{R}^{n\times k},$$
$$\tilde{y}_k \triangleq \beta D_k^{-1} L_k^{-1} e_1^{(k)} = [\eta_1, \ldots, \eta_k]^{\mathsf{T}} \in \mathbb{R}^k. \quad k = 1, 2, \ldots.$$

引理 下面的递推公式成立:

$$\tilde{P}_{k+1} \triangleq V_{k+1} L_{k+1}^{-\mathsf{T}} = [\tilde{P}_k, \tilde{p}_{k+1}],$$
$$\tilde{y}_{k+1} \triangleq \beta D_{k+1}^{-1} L_{k+1}^{-1} e_1^{(k+1)} = [\tilde{y}_k^{\mathsf{T}}, \eta_{k+1}]^{\mathsf{T}}, \quad k = 1, 2, \ldots.$$

(板书)

证明.

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix}, \quad T_{k+1} = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \ddots & \ddots & & \\ & \ddots & \ddots & \beta_{k-1} & \\ & & \beta_{k-1} & \alpha_k & \beta_k \\ & & & \beta_k & \alpha_{k+1} \end{bmatrix}.$$

设 T_k 的 $\mathrm{LDL}^{\mathsf{T}}$ 分解为

$$T_k = L_k D_k L_k^{\mathsf{T}} = \begin{bmatrix} 1 & & & \\ l_1 & 1 & & \\ & \ddots & \ddots & \\ & & l_{k-1} & 1 \end{bmatrix} \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_k \end{bmatrix} \begin{bmatrix} 1 & & & \\ l_1 & 1 & & \\ & \ddots & \ddots & \\ & & l_{k-1} & 1 \end{bmatrix}^{\mathsf{T}}.$$

由待定系数法可知 $d_1 = \alpha_1$, $l_i = \beta_i / d_i$, $\quad d_{i+1} = \alpha_{i+1} - l_i \beta_i$, $\quad i = 1, 2, \ldots, k-1$.

记 $\gamma_k = [0, \ldots, 0, \beta_k]^{\mathsf{T}} \in \mathbb{R}^k$ **则**

$$T_{k+1} = \begin{bmatrix} T_k & \gamma_k \\ \gamma_k^{\mathsf{T}} & \alpha_{k+1} \end{bmatrix} = \begin{bmatrix} L_k & 0 \\ * & 1 \end{bmatrix} \begin{bmatrix} D_k & \\ & * \end{bmatrix} \begin{bmatrix} L_k & 0 \\ * & 1 \end{bmatrix}^{\mathsf{T}} = L_{k+1} D_{k+1} L_{k+1}^{\mathsf{T}}$$

$$= \begin{bmatrix} 1 & & & & \\ l_1 & 1 & & & \\ & \ddots & \ddots & & \\ & & l_{k-1} & 1 & \\ & & & \boldsymbol{l_k} & 1 \end{bmatrix} \begin{bmatrix} d_1 & & & & \\ & d_2 & & & \\ & & \ddots & & \\ & & & d_k & \\ & & & & \boldsymbol{d_{k+1}} \end{bmatrix} \begin{bmatrix} 1 & & & & \\ l_1 & 1 & & & \\ & \ddots & \ddots & & \\ & & l_{k-1} & 1 & \\ & & & \boldsymbol{l_k} & 1 \end{bmatrix}^{\mathsf{T}},$$

其中 $\boxed{l_k = \beta_k / d_k,\ d_{k+1} = \alpha_{k+1} - l_k \beta_k}$. **记** $\tilde{\gamma}_k = [0, \ldots, 0, l_k]^{\mathsf{T}} \in \mathbb{R}^k$, **则**

$$L_{k+1} = \begin{bmatrix} L_k & 0 \\ \tilde{\gamma}_k^{\mathsf{T}} & 1 \end{bmatrix}, \qquad L_{k+1}^{-1} = \begin{bmatrix} L_k^{-1} & 0 \\ -\tilde{\gamma}_k^{\mathsf{T}} L_k^{-1} & 1 \end{bmatrix}.$$

所以有

$$\tilde{P}_{k+1} = V_{k+1}L_{k+1}^{-\mathsf{T}} = [V_k, v_{k+1}]\begin{bmatrix} L_k^{-\mathsf{T}} & -L_k^{-\mathsf{T}}\tilde{\gamma}_k \\ 0 & 1 \end{bmatrix} = [V_k L_k^{-\mathsf{T}}, -V_k L_k^{-\mathsf{T}}\tilde{\gamma}_k + v_{k+1}].$$

又 $V_k L_k^{-\mathsf{T}}\tilde{\gamma}_k = \tilde{P}_k[0,\ldots,0,l_k]^{\mathsf{T}} = l_k\tilde{p}_k$, 所以 $\tilde{P}_{k+1} = [\tilde{P}_k, \tilde{p}_{k+1}]$, 其中

$$\tilde{p}_{k+1} = -l_k\tilde{p}_k + v_{k+1} \tag{2.10}$$

另外,

$$\begin{aligned}\tilde{y}_{k+1} &= \beta D_{k+1}^{-1}L_{k+1}^{-1}e_1^{(k+1)} \\ &= \beta\begin{bmatrix} D_k^{-1} & 0 \\ 0 & d_{k+1}^{-1}\end{bmatrix}\begin{bmatrix} L_k^{-1} & 0 \\ -\tilde{\gamma}_k^{\mathsf{T}}L_k^{-1} & 1\end{bmatrix}\begin{bmatrix} e_1^{(k)} \\ 0\end{bmatrix} = \begin{bmatrix} \beta D_k^{-1}L_k^{-1}e_1^{(k)} \\ -\beta d_{k+1}^{-1}\tilde{\gamma}_k^{\mathsf{T}}L_k^{-1}e_1^{(k)}\end{bmatrix} = \begin{bmatrix}\tilde{y}_k \\ \eta_{k+1}\end{bmatrix},\end{aligned}$$

因此结论成立. □

递推公式

$$x^{(k)} = x^{(0)} + (V_k L_k^{-\mathsf{T}})(\beta D_k^{-1} L_k^{-1} e_1^{(k)})$$

$$x^{(k+1)} = x^{(0)} + (V_{k+1} L_{k+1}^{-\mathsf{T}})(\beta D_{k+1}^{-1} L_{k+1}^{-1} e_1^{(k+1)})$$

$$\tilde{P}_k \triangleq V_k L_k^{-\mathsf{T}}, \quad \tilde{P}_{k+1} \triangleq V_{k+1} L_{k+1}^{-\mathsf{T}} = [\tilde{P}_k, \tilde{p}_{k+1}]$$

$$\tilde{y}_k \triangleq \beta D_k^{-1} L_k^{-1} e_1^{(k)}, \quad \tilde{y}_{k+1} \triangleq \beta D_{k+1}^{-1} L_{k+1}^{-1} e_1^{(k+1)} = [\tilde{y}_k^{\mathsf{T}}, \eta_{k+1}]^{\mathsf{T}}$$

从 $x^{(k)}$ 到 $x^{(k+1)}$ 的递推公式

$$x^{(k+1)} = x^{(0)} + \tilde{P}_{k+1}\tilde{y}_{k+1} = x^{(0)} + [\tilde{P}_k, \tilde{p}_{k+1}] \begin{bmatrix} \tilde{y}_k \\ \eta_{k+1} \end{bmatrix} = x^{(k)} + \eta_{k+1}\tilde{p}_{k+1}.$$

残量计算

通过直接计算可知

$$r_{k+1} = b - Ax^{(k+1)} = b - A(x^{(k)} + \eta_{k+1}\tilde{p}_{k+1}) = r_k - \eta_{k+1}A\tilde{p}_{k+1}.$$

另一方面, 我们有

$$\begin{aligned} r_k = b - Ax^{(k)} &= b - A(x^{(0)} + V_k y_k) \\ &= r_0 - AV_k y_k = \beta v_1 - V_k T_k y_k - \beta_k v_{k+1} e_k^{\mathsf{T}} y_k = -\beta_k (e_k^{\mathsf{T}} y_k) v_{k+1}, \end{aligned}$$

即 r_k 与 v_{k+1} 平行. 记

$$r_k = \tau_k v_{k+1}, \quad k = 0, 1, 2, \ldots, \tag{2.11}$$

其中

$$\tau_0 = \beta = \|r_0\|_2, \quad \tau_k = -\beta_k (e_k^{\mathsf{T}} y_k), \quad k = 1, 2, \ldots.$$

$\tilde{p}_{k+1}$ 的计算

记 $p_k = \tau_{k-1}\tilde{p}_k,\ k = 1, 2, \ldots$, 由 (2.10) 和 (2.11) 可知

$$p_{k+1} = \tau_k \tilde{p}_{k+1} = \tau_k(v_{k+1} - l_k \tilde{p}_k) = r_k + \mu_k p_k, \tag{2.12}$$

其中 $\mu_k = -\dfrac{l_k \tau_k}{\tau_{k-1}},\ k = 1, 2, \ldots$. 于是就得到下面的递推公式

$$r_{k+1} = r_k - \eta_{k+1} A\tilde{p}_{k+1} = r_k - \xi_{k+1} A p_{k+1} \tag{2.13}$$

$$x^{(k+1)} = x^{(k)} + \eta_{k+1}\tilde{p}_{k+1} = x^{(k)} + \xi_{k+1} p_{k+1}, \tag{2.14}$$

其中 $\xi_{k+1} = \dfrac{\eta_{k+1}}{\tau_k},\ k = 0, 1, 2, \ldots$.

ξ_{k+1} 和 μ_k 的计算

引理 下面的结论成立:

(1) $r_1, r_2, \ldots, r_k, \ldots$ 相互正交;

(2) $p_1, p_2, \ldots, p_k, \ldots$ 相互 A 共轭 (或 A 正交), 即当 $i \neq j$ 时有 $p_i^{\mathsf{T}} A p_j = 0$.

证明. (1) 由于 r_k 与 v_{k+1} 平行, 所以结论成立.

(2) 只需证明 $P_k^{\mathsf{T}} A P_k$ 是对角矩阵即可, 即证 $\tilde{P}_k^{\mathsf{T}} A \tilde{P}_k$ 是对角矩阵. 通过直接计算可得

$$\begin{aligned} \tilde{P}_k^{\mathsf{T}} A \tilde{P}_k = (V_k L_k^{-\mathsf{T}})^{\mathsf{T}} A V_k L_k^{-\mathsf{T}} &= L_k^{-1} V_k^{\mathsf{T}} A V_k L_k^{-\mathsf{T}} \\ &= L_k^{-1} T_k L_k^{-\mathsf{T}} \\ &= L_k^{-1} (L_k D_k L_k^{\mathsf{T}}) L_k^{-\mathsf{T}} = D_k. \end{aligned}$$

□

下面给出 ξ_{k+1} 和 μ_k 的计算公式.

- 等式 (2.12) 两边左乘 $p_{k+1}^{\mathsf{T}}A$ 得 $p_{k+1}^{\mathsf{T}}Ap_{k+1} = p_{k+1}^{\mathsf{T}}Ar_k + \mu_k p_{k+1}^{\mathsf{T}}Ap_k = r_k^{\mathsf{T}}Ap_{k+1}$
- 等式 (2.13) 两边左乘 r_k^{T} 得 $0 = r_k^{\mathsf{T}}r_{k+1} = r_k^{\mathsf{T}}r_k - \xi_{k+1}r_k^{\mathsf{T}}Ap_{k+1}$

$$\text{故}\quad \xi_{k+1} = \frac{r_k^{\mathsf{T}}r_k}{r_k^{\mathsf{T}}Ap_{k+1}} = \frac{r_k^{\mathsf{T}}r_k}{p_{k+1}^{\mathsf{T}}Ap_{k+1}} \tag{2.15}$$

- 等式 (2.12) 两边左乘 $p_k^{\mathsf{T}}A$ 得 $0 = p_k^{\mathsf{T}}Ap_{k+1} = p_k^{\mathsf{T}}Ar_k + \mu_k p_k^{\mathsf{T}}Ap_k$

$$\text{故}\quad \mu_k = -\frac{p_k^{\mathsf{T}}Ar_k}{p_k^{\mathsf{T}}Ap_k} = -\frac{r_k^{\mathsf{T}}Ap_k}{p_k^{\mathsf{T}}Ap_k}. \tag{2.16}$$

为了进一步减少运算量, 我们将上式进行改写: 等式 (2.13) 两边左乘 r_{k+1}^{T} 得

$$r_{k+1}^{\mathsf{T}} r_{k+1} = r_{k+1}^{\mathsf{T}} r_k - \xi_{k+1} r_{k+1} A p_{k+1} = -\xi_{k+1} r_{k+1}^{\mathsf{T}} A p_{k+1}$$

故 $\xi_{k+1} = -\dfrac{r_{k+1}^{\mathsf{T}} r_{k+1}}{r_{k+1}^{\mathsf{T}} A p_{k+1}}$. 所以 $\xi_k = -\dfrac{r_k^{\mathsf{T}} r_k}{r_k^{\mathsf{T}} A p_k}$, 即 $r_k^{\mathsf{T}} A p_k = -r_k^{\mathsf{T}} r_k / \xi_k$. 代入 (2.16) 得

$$\mu_k = -\frac{r_k^{\mathsf{T}} A p_k}{p_k^{\mathsf{T}} A p_k} = \frac{r_k^{\mathsf{T}} r_k}{p_k^{\mathsf{T}} A p_k} \cdot \frac{1}{\xi_k} = \frac{r_k^{\mathsf{T}} r_k}{p_k^{\mathsf{T}} A p_k} \cdot \frac{p_k^{\mathsf{T}} A p_k}{r_{k-1}^{\mathsf{T}} r_{k-1}} = \frac{r_k^{\mathsf{T}} r_k}{r_{k-1}^{\mathsf{T}} r_{k-1}}. \tag{2.17}$$

综合公式 (2.12), (2.13), (2.14) 和 (2.15), (2.17) 即可得 CG 方法的迭代格式:

$$\begin{aligned}
&p_{k+1} = r_k + \mu_k p_k, \quad \text{其中} \quad \mu_k = r_k^{\mathsf{T}} r_k / r_{k-1}^{\mathsf{T}} r_{k-1}, \\
&x^{(k+1)} = x^{(k)} + \xi_{k+1} p_{k+1}, \\
&r_{k+1} = r_k - \xi_{k+1} A p_{k+1}, \quad \text{其中} \quad \xi_{k+1} = r_k^{\mathsf{T}} r_k / p_{k+1}^{\mathsf{T}} A p_{k+1}.
\end{aligned}$$

需要指出的是, 以上递推公式是从 $k=1$ 开始的.

因此 $k=0$ 时的计算公式需要另外推导.

需要指出的是, 以上递推公式是从 $k=1$ 开始的.

因此 $k=0$ 时的计算公式需要另外推导.

- p_1 的计算: 由 $\tilde{p}_1$ 的定义可知 $\tilde{p}_1 = \tilde{P}_1 = V_1 L_1^{-\mathsf{T}} = v_1$. 因此

$$p_1 = \tau_0 \tilde{p}_1 = \beta v_1 = r_0.$$

需要指出的是, 以上递推公式是从 $k=1$ 开始的.

因此 $k=0$ 时的计算公式需要另外推导.

- p_1 的计算: 由 $\tilde{p}_1$ 的定义可知 $\tilde{p}_1 = \tilde{P}_1 = V_1 L_1^{-\mathsf{T}} = v_1$. 因此

$$p_1 = \tau_0 \tilde{p}_1 = \beta v_1 = r_0.$$

- $x^{(1)}$ 的计算: 由 Lanczos 过程可知 $T_1 = \alpha_1 = v_1^{\mathsf{T}} A v_1$, 注意到 $\beta^2 = r_0^{\mathsf{T}} r_0$, 所以

$$x^{(1)} = x^{(0)} + V_1 T_1^{-1}(\beta e_1) = x^{(0)} + \frac{\beta}{v_1^{\mathsf{T}} A v_1} v_1 = x^{(0)} + \frac{r_0^{\mathsf{T}} r_0}{p_1^{\mathsf{T}} A p_1} p_1.$$

令 $\xi_1 = \dfrac{r_0^{\mathsf{T}} r_0}{p_1^{\mathsf{T}} A p_1}$ (之前的公式只对 $k \geq 1$ 有定义), 则递推公式对 $k=0$ 仍然成立.

- r_1 的计算:

$$r_1 = b - Ax^{(1)} = b - Ax^{(0)} - \frac{r_0^{\mathsf{T}} r_0}{p_1^{\mathsf{T}} A p_1} A p_1 = r_0 - \xi_1 A p_1,$$

即当 $k = 0$ 时关于 r_{k+1} 的递推公式也成立.

综上所述, 就可以得到 CG 算法.

算法 共轭梯度法 (CG)

1: 给定初值 $x^{(0)}$, (相对) 精度要求 $\varepsilon > 0$ 和最大迭代步数 IterMax
2: 计算 $r_0 = b - Ax^{(0)}$ 和 $\beta = \|r_0\|_2$
3: **for** $k = 1$ to IterMax **do**
4: $\quad \rho = r_{k-1}^{\mathsf{T}} r_{k-1}$
5: $\quad$ **if** $k > 1$ **then**
6: $\quad\quad \mu_{k-1} = \rho/\rho_0, \quad p_k = r_{k-1} + \mu_{k-1} p_{k-1}$
7: $\quad$ **else**
8: $\quad\quad p_k = r_0$
9: $\quad$ **end if**
10: $\quad q_k = Ap_k, \quad \xi_k = \rho/(p_k^{\mathsf{T}} q_k)$
11: $\quad x^{(k)} = x^{(k-1)} + \xi_k p_k, \quad r_k = r_{k-1} - \xi_k q_k$
12: $\quad$ relres $= \|r_k\|_2/\beta$
13: $\quad$ **if** relres $< \varepsilon$, **then** break, **end if**
14: $\quad \rho_0 = \rho$
15: **end for**
16: **if** relres $< \varepsilon$, 输出近似解 $x^{(k)}$ 及相关信息
17: $\quad$ **else** 输出算法失败信息
18: **end if**

注记

(1) 在优化领域, CG 方法中的向量 p_k 称为**搜索方向**, 而 ξ_k 称为**步长**.
因此, 搜索方法是 A 共轭的, 这就是方法名称的由来.

(2) 在 CG 方法中, 只需要额外存储四个向量 x, p, Ap 和 r,
每步迭代的主要运算为一个矩阵向量乘积和两个内积.

CG 方法的最优性质

定理 设 A 对称正定, 则

$$x^{(k)} = \arg \min_{x \in x^{(0)} + \mathcal{K}_k} \|x - x_*\|_A \tag{2.18}$$

当且仅当

$$x^{(k)} \in x^{(0)} + \mathcal{K}_k \quad \text{且} \quad b - Ax^{(k)} \perp \mathcal{K}_k. \tag{2.19}$$

(板书)

补充: 最佳逼近与正交投影

设 $A \in \mathbb{R}^{n \times n}$ 对称正定, $\mathbb{S}$ 是 $\mathbb{R}^n$ 的子空间, $z \in \mathbb{R}^n$,

则 x_* 是 $\min_{x \in \mathbb{S}} \|x - z\|_A$ 的解的充要条件是 $x_* \in \mathbb{S}$ 且 $A(x_* - z) \perp \mathbb{S}$

证明. 首先证明**充分性**. 设 $x^{(k)}$ 满足 (2.19). 记 $\tilde{x} = x^{(k)} - x^{(0)}$, 则 $\tilde{x} \in \mathcal{K}_k$ 且

$$r_0 - A\tilde{x} \perp \mathcal{K}_k.$$

由正交投影的性质可知, $\tilde{x}$ 是最佳逼近问题 $\min\limits_{x\in\mathcal{K}_k} \|x - A^{-1}r_0\|_A$ 的解, 即

$$\begin{aligned}\tilde{x} = \arg\min_{x\in\mathcal{K}_k} \|x - A^{-1}r^{(0)}\|_A &= \arg\min_{x\in\mathcal{K}_k} \|x - A^{-1}(b - Ax^{(0)})\|_A \\ &= \arg\min_{x\in\mathcal{K}_k} \|(x^{(0)} + x) - x_*\|_A.\end{aligned}$$

所以, $x^{(k)} = x^{(0)} + \tilde{x}$ 是最佳逼近问题

$$\min_{x\in x^{(0)}+\mathcal{K}_k} \|x - x_*\|_A$$

的解, 即结论 (2.18) 成立.

必要性: 仍然利用正交投影的性质, 留作练习. □

注记

- GMRES 极小化残量的 2-范数, CG 方法极小化绝对误差的 A 范数或残量的 A^{-1} 范数.
- 如果 A 对称, 但不定, 此时 A 范数就没有定义, 因此最优性结论不再成立.

  从方法推导过程可知, 此时仍可以使用 CG 方法, 但由于 $\mathrm{LDL}^{\mathsf{T}}$ 分解不一定存在, 因此方法可能会中断.

例 考虑带齐次 Dirichlet 边界条件的 Poisson 方程: (CG_Poisson2D.m)

$$
\begin{aligned}
-\Delta u(x,y) &= f(x,y), \qquad (2.20)\\
(x,y) &\in \Omega \triangleq [0,1]\times[0,1]\\
u(x,y) &= 0, \quad (x,y)\in\partial\Omega.
\end{aligned}
$$

用五点差分离散, 步长为 $h_x = h_y = 1/(N+1)$, 得代数方程组

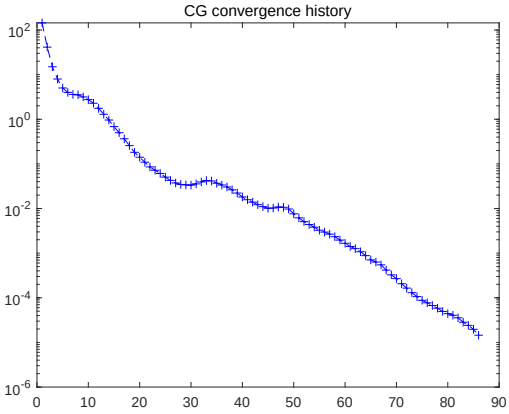
$$(I\otimes T + T\otimes I)u = h^2 f.$$

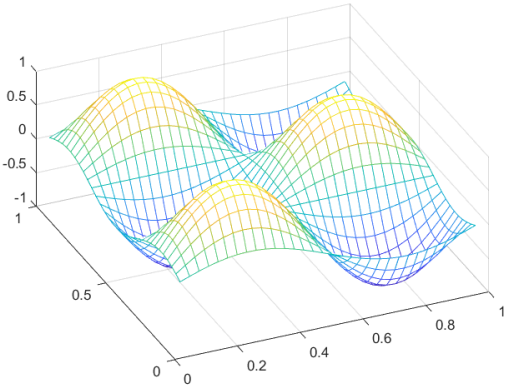
其中 $T = \text{tridiag}(-1, 2, -1)$.

最大迭代步数设为 $\text{IterMax} = N^2$, 迭代终止条件

$$\frac{\|r_k\|_2}{\|r_0\|_2} \le \text{tol} \triangleq 10^{-6}.$$

设解析解为 $u(x,y)=\sin(2\pi x)\sin(3\pi y)$, 则 $f(x,y)=13\pi^2\sin(2\pi x)\sin(3\pi y)$.





$n=32$ 时的相对残量下降曲线和数值解

✍ **注:** 程序里的初始向量是一个随机向量, 如果取零向量, 大家试试结果会怎样.

2-4-3 极小残量法 (MINRES)

假定 A 对称, 但不定, 此时 CG 方法不再适用.

与 GMRES 方法类似, 我们可以取约束空间为 $\mathcal{L}_k = A\mathcal{K}_k$, 即

$$\text{find} \quad x^{(k)} \in x^{(0)} + \mathcal{K}_k \quad \text{such that} \quad b - Ax^{(k)} \perp A\mathcal{K}_k$$

这就是 **极小残量法** (MINRES).

MINRES 可以看作是 GMRES 的对称情形, 但由于 A 对称, 因此在计算 $\mathcal{K}_k$ 的标准正交基时, 可以采用 Lanczos 算法, 即三项递推方法, 从而节省运算量.

MINRES 方法的最优性质

定理 设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵, $x^{(k)} \in x^{(0)} + \mathcal{K}_k$ 是 MINRES 方法迭代 k 步后得到的近似解, 则

$$x^{(k)} = \arg \min_{x\in x^{(0)}+\mathcal{K}_k} \|b - Ax\|_2,$$

即 $x^{(k)}$ 是残量在仿射空间 $x^{(0)} + \mathcal{K}_k$ 中的唯一最小值点.

算法推导

由 GMRES 的推导过程可知, 当 MINRES 方法迭代 k 步之后, 近似解可表示为

$$x^{(k)} = x^{(0)} + V_k y_k, \tag{2.21}$$

其中

$$y_k = \arg\min_{y\in\mathbb{R}^k} \|\beta e_1 - T_{k+1,k} y\|_2. \tag{2.22}$$

这里 $T_{k+1,k}$ 是由 Lanczos 算法生成的 $(k+1)\times k$ 的上 Hessenberg 矩阵

$$T_{k+1,k} = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \\ & & & \beta_k \end{bmatrix}.$$

最小二乘问题的求解

我们仍然用 QR 分解来求解最小二乘问题 (2.22). 设

$$T_{k+1,k} = Q_{k+1}^{\mathsf{T}} R_{k+1,k}$$

由于 $T_{k+1,k}$ 是三对角的, 所以 $R_{k+1,k}$ 也只有三条对角线非零, 即

$$R_{k+1,k} = \begin{bmatrix} \tau_1^{(1)} & \tau_1^{(2)} & \tau_1^{(3)} & & & \\ & \tau_2^{(1)} & \tau_2^{(2)} & \tau_2^{(3)} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \tau_{k-2}^{(1)} & \tau_{k-2}^{(2)} & \tau_{k-2}^{(3)} \\ & & & & \tau_{k-1}^{(1)} & \tau_{k-1}^{(2)} \\ & & & & & \tau_k^{(1)} \\ 0 & & \cdots & & \cdots & 0 \end{bmatrix}_{(k+1)\times k} \triangleq \begin{bmatrix} R_k \\ 0 \end{bmatrix},$$

其中 R_k 是由 $R_{k+1,k}$ 的前 k 行组成的 k 阶矩阵.

于是, 我们有

$$\|\beta e_1 - T_{k+1,k} y\|_2 = \|\beta e_1 - Q_{k+1}^{\mathsf{T}} R_{k+1,k}\, y)\|_2$$
$$= \left\| Q_{k+1}(\beta e_1) - \begin{bmatrix} R_k \\ 0 \end{bmatrix} y \right\|_2 = \left\| \beta q_1 - \begin{bmatrix} R_k y \\ 0 \end{bmatrix} \right\|_2,$$

其中 $q_1^{(k+1)}$ 是由 Q_{k+1} 的第 1 列. 如果 R_k 非奇异, 则最小二乘问题 (2.22) 的解为

$$y_k = \beta R_k^{-1} q_1^{(k+1)}(1\!:\!k)$$

其中 $q_1^{(k+1)}(1\!:\!k)$ 表示 $q_1^{(k+1)}$ 的前 k 个分量组成的向量. 因此,

$$x^{(k)} = x^{(0)} + \beta\, V_k R_k^{-1} q_1^{(k+1)}(1\!:\!k) \tag{2.23}$$

残量范数为

$$\|r_k\|_2 = \|\beta e_1 - T_{k+1,k} y_k\|_2 = \beta |q_1^{(k+1)}(k+1)|.$$

QR 分解的递推算法

与 GMRES 类似, 可通过递推方法利用一次 Givens 变换得到 $T_{k+1,k}$ 的 QR 分解.

假定 $T_{k,k-1}$ 的 QR 分解为

$$T_{k,k-1} = (G_{k-1}G_{k-2}\cdots G_1)^{\mathsf{T}} R_{k,k-1} = Q_k^{\mathsf{T}} R_{k,k-1},$$

$$R_{k,k-1} = \begin{bmatrix} \tau_1^{(1)} & \tau_1^{(2)} & \tau_1^{(3)} & & & \\ & \tau_2^{(1)} & \tau_2^{(2)} & \tau_2^{(3)} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \tau_{k-3}^{(3)} \\ & & & & \ddots & \tau_{k-2}^{(2)} \\ & & & & & \tau_{k-1}^{(1)} \\ 0 & & \cdots & & \cdots & 0 \end{bmatrix}.$$

因此,

$$T_{k+1,k} = \left[\begin{array}{c|c} T_{k,k-1} & \begin{matrix} 0 \\ \vdots \\ 0 \\ \beta_{k-1} \\ \alpha_k \end{matrix} \\ \hline 0 & \beta_k \end{array}\right] = \begin{bmatrix} Q_k^{\mathsf{T}} & 0 \\ 0 & 1 \end{bmatrix} \left[\begin{array}{c|c} R_{k,k-1} & Q_k \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \beta_{k-1} \\ \alpha_k \end{bmatrix} \\ \hline 0 & \beta_k \end{array}\right] \triangleq \begin{bmatrix} Q_k^{\mathsf{T}} & 0 \\ 0 & 1 \end{bmatrix} \tilde{T}_{k+1,k},$$

其中 $Q_k \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \beta_{k-1} \\ \alpha_k \end{bmatrix} = G_{k-1} G_{k-2} \cdots G_1 \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \beta_{k-1} \\ \alpha_k \end{bmatrix} = G_{k-1} G_{k-2} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \beta_{k-1} \\ \alpha_k \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ \tau_{k-2}^{(3)} \\ \tau_{k-1}^{(2)} \\ \tilde{\alpha}_k \end{bmatrix}.$

构造 Givens 变换 $G_k = \begin{bmatrix} I_{k-1} & & \\ & c_k & s_k \\ & -s_k & c_k \end{bmatrix} \in \mathbb{R}^{(k+1)\times(k+1)}$, $\left(\text{针对 } \begin{bmatrix} \tilde{\alpha} \\ \beta_k \end{bmatrix}\right)$

消去 $\tilde{T}_{k+1,k}$ **最后一列的最后一个元素** β_k, **即选取**

$$c_k = \frac{\tilde{\alpha}_k}{\tau_k^{(1)}}, \quad s_k = \frac{\beta_k}{\tau_k^{(1)}} \quad \text{其中} \quad \tau_k^{(1)} = \sqrt{\tilde{\alpha}_k^2 + \beta_k^2}.$$

由于 $R_{k,k-1}$ **的最后一行全为** 0, **且左乘** G_k **只会影响** $\tilde{T}_{k+1,k}$ **的最后两行, 因此**

$$G_k \tilde{T}_{k+1,k} = \left[\begin{array}{c|c} & 0 \\ & \vdots \\ & 0 \\ R_{k,k-1} & \tau_{k-2}^{(3)} \\ & \tau_{k-1}^{(2)} \\ & \tau_k^{(1)} \\ \hline 0 & 0 \end{array}\right] = \begin{bmatrix} \tau_1^{(1)} & \tau_1^{(2)} & \tau_1^{(3)} & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \tau_{k-3}^{(3)} & & \\ & & & \ddots & \tau_{k-2}^{(2)} & \tau_{k-2}^{(3)} & \\ & & & & \tau_{k-1}^{(1)} & \tau_{k-1}^{(2)} & \\ & & & & & \tau_k^{(1)} & \\ 0 & & \cdots & & \cdots & 0 & \end{bmatrix} \triangleq R_{k+1,k}.$$

于是我们就得到 $T_{k+1,k}$ 的 QR 分解

$$T_{k+1,k} = Q_{k+1}^{\mathsf{T}} R_{k+1,k},$$

其中 (这里我们假定 Givens 变换 G_i 的维数是自动增长的)

$$Q_{k+1} = G_k \begin{bmatrix} Q_k & 0 \\ 0 & 1 \end{bmatrix} = G_k G_{k-1} \dots G_1.$$

需要指出的是, $R_{k,k-1}$ 是 $R_{k+1,k}$ 的前 k 行和前 $k-1$ 列.

因此, R_{k-1} 是 R_k 的 $k-1$ 阶顺序主子矩阵.

$x^{(k)}$ 的递推计算公式

$$x^{(k)} = x^{(0)} + \beta\, V_k R_k^{-1} q_1^{(k+1)}(1:k)$$

- 设 $P_k = [p_1, p_2, \ldots, p_k] \triangleq V_k R_k^{-1}$, 通过直接计算, 由 $P_k R_k = V_k$ 可知

$$\begin{aligned} p_1 &= v_1/\tau_1^{(1)}, \quad p_2 = \left(v_2 - \tau_1^{(2)} p_1\right)/\tau_2^{(1)}, \\ p_i &= \left(v_i - \tau_{i-1}^{(2)} p_{i-1} - \tau_{i-2}^{(3)} p_{i-2}\right)/\tau_i^{(1)}, \quad i = 3, 4, \ldots, k. \end{aligned} \tag{2.24}$$

又 $V_k = [V_{k-1}, v_k]$, 且 R_{k-1} 是 R_k 的 $k-1$ 阶顺序主子矩阵, 所以

$$P_k = [P_{k-1}, p_k].$$

- 用 $\xi^{(k)}$ 表示 $\beta q_1^{(k+1)}(1:k)$, 即

$$\beta q_1^{(k+1)} = \begin{bmatrix} \xi^{(k)} \\ \tilde{\xi}_{k+1} \end{bmatrix},$$

又

$$Q_{k+1} = G_k \begin{bmatrix} Q_k & 0 \\ 0 & 1 \end{bmatrix},$$

所以 $q_1^{(k+1)}(1:k-1) = q_1^{(k)}(1:k-1)$, 即 $q_1^{(k+1)}$ 和 $q_1^{(k)}$ 的前 $k-1$ 个分量相同, 其中 $q_1^{(k)}$ 表示 Q_k 的第 1 列.

于是我们可以得到 $\xi^{(k)}$ 和 $\xi^{(k-1)}$ 之间的关系式:

$$\xi^{(k)} = \begin{bmatrix} \xi^{(k-1)} \\ \xi_k \end{bmatrix}, \quad k = 2, 3, \ldots,$$

其中 ξ_k 表示 $\xi^{(k)}$ 的最后一个分量.

因此 $\xi^{(k)}$ 可以表示为

$$\xi^{(k)} = [\xi_1, \xi_2, \ldots, \xi_k]^{\mathsf{T}}, \quad k = 1, 2, \ldots.$$

于是

$$\begin{aligned} x^{(k)} &= x^{(0)} + V_k R_k^{-1}(\beta q_1^{(k+1)}(1:k)) \\ &= x^{(0)} + P_k \xi^{(k)} = x^{(0)} + [P_{k-1}, p_k] \begin{bmatrix} \xi^{(k-1)} \\ \xi_k \end{bmatrix} = \boxed{x^{(k-1)} + \xi_k p_k}, \end{aligned} \tag{2.25}$$

其中 p_k 可通过递推公式 (2.24) 来计算, 而 ξ_k 是 $\beta q_1^{(k+1)}$ 的第 k 个分量, 且

$$\beta q_1^{(k+1)} = \beta Q_{k+1} e_1 = G_k G_{k-1} \cdots G_1(\beta e_1),$$

即向量 $\beta q_1^{(k+1)}$ 可以通过将 Givens 变换依次作用在 βe_1 上得到.

残量范数的计算

$$\|r_k\|_2 = \|b - Ax^{(k)}\|_2 = |\beta q_1^{(k+1)}(k+1)| = |\tilde{\xi}_{k+1}|,$$

其中 $\tilde{\xi}_{k+1}$ 是 $\beta q_1^{(k+1)}$ 的最后一个分量.

MINRES 迭代格式

$$x^{(k)} = x^{(k-1)} + \xi_k p_k$$

$$p_k = \left(v_k - \tau_{k-1}^{(2)} p_{k-1} - \tau_{k-2}^{(3)} p_{k-2}\right) / \tau_k^{(1)}$$

$$\beta q_1^{(k+1)} = G_k G_{k-1} \cdots G_1(\beta e_1), \qquad \begin{bmatrix} \tau_{k-2}^{(3)} \\ \tau_{k-1}^{(2)} \\ \tau_k^{(1)} \\ 0 \end{bmatrix} = G_k G_{k-1} G_{k-2} \begin{bmatrix} 0 \\ \beta_{k-1} \\ \alpha_k \\ \beta_k \end{bmatrix}$$

算法 Minimal Residual (MINRES) Method

给定初值 $x^{(0)}$, **(相对) 精度要求** $\varepsilon > 0$ **和最大迭代步数** IterMax
计算 $r_0 = b - Ax^{(0)}$ **和** $\beta = \|r_0\|_2$
令 $\beta_0 = 0$, $v_0 = 0$, $p_{-1} = 0$, $p_0 = 0$
$v_1 = r_0/\beta, \quad \xi = \beta e_1$
for $k = 1$ to IterMax **do**
    $w_k = Av_k - \beta_{k-1}v_{k-1}, \quad \alpha_k = (w_k, v_k), \quad w_k = w_k - \alpha_k v_k, \quad \beta_k = \|w_k\|_2$
    if $k > 2$ **then** % apply $G_{k-1}G_{k-2}$ to the last column of $T_{k+1,k}$
        $$\begin{bmatrix} \tau_{k-2}^{(3)} \\ \tilde{\beta}_{k-1} \end{bmatrix} = \begin{bmatrix} c_{k-2} & s_{k-2} \\ -s_{k-2} & c_{k-2} \end{bmatrix} \begin{bmatrix} 0 \\ \beta_{k-1} \end{bmatrix}$$
    end if
    if $k > 1$ **then**
        $$\begin{bmatrix} \tau_{k-1}^{(2)} \\ \tilde{\alpha}_k \end{bmatrix} = \begin{bmatrix} c_{k-1} & s_{k-1} \\ -s_{k-1} & c_{k-1} \end{bmatrix} \begin{bmatrix} \tilde{\beta}_{k-1} \\ \alpha_k \end{bmatrix}$$
    end if
    if $|\tilde{\alpha}_k| > |\beta_k|$ **then** % form the Givens rotation G_k
        set $c_k = \frac{1}{\sqrt{1+\gamma^2}}$, $s_k = c_k\gamma$ where $\gamma = \beta_k/\tilde{\alpha}_k$

15: **else**

16: set $s_k = \frac{1}{\sqrt{1+\gamma^2}}$, $c_k = s_k\gamma$ where $\gamma = \alpha_k/\tilde{\beta}_k$

17: **end if**

18: $\tau_k^{(1)} = c_k\tilde{\alpha}_k + s_j\beta_k$ % apply G_k to last column of $\tilde{T}_{k+1,k}$

19: $\begin{bmatrix} \xi_k \\ \xi_{k+1} \end{bmatrix} = \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix} \begin{bmatrix} \xi_k \\ 0 \end{bmatrix}$ % apply G_k to ξ

20: $p_k = \left(v_k - \tau_{k-1}^{(2)}p_{k-1} - \tau_{k-2}^{(3)}p_{k-2}\right)/\tau_k^{(1)}$ where $p_0 = p_{-1} = 0$

21: $x^{(k)} = x^{(k-1)} + \xi_k p_k$

22: relres $= |\xi_{k+1}|/\beta$

23: **if** relres$< \varepsilon$, **then** break, **end if** % check convergence

24: $v_{k+1} = w_k/\beta_k$

25: **end for**

26: **if** relres$< \varepsilon$, 输出近似解 $x^{(k)}$ 及相关信息

27: **else** 输出算法失败信息

28: **end if**

MINRES vs GMRES

- MINRES 充分利用了系数矩阵的对称性, 这使得每步迭代的运算量不会随着迭代步数的增加而增加;
- 在迭代过程中, 我们不需要存储所有的 v_i 和 p_i, 因此存储量也与迭代步数无关.

注记

CG 方法也可以用来求解对称不定线性方程组, 但可能会产生中断. 关于对称不定线性方程组的 CG 方法和 MINRES 方法的收敛性态比较, 可以参见相关文献.

MINRES 的缺点

- 在 MINRES 中, 我们用三项递推进行正交化, 随着迭代步数的增加, 舍入误差可能会影响正交性.
- 有学者指出, 在 MINRES 中, 舍入误差是按 $\kappa(A)^2$ 的速度增长的. 而在 GMRES 中, 舍入误差只按 $\kappa(A)$ 的速度增长. 因此, 对于坏条件问题, 在使用 MINRES 需要加倍小心, 特别是迭代步数比较大的情况下.
- 另一种替代方法 — SYMMLQ 方法.

2-4-4 SYMMLQ 方法

SYMMLQ (Symmetric LQ) 方法是求解对称线性方程组的另外一种方法, 其收敛速度可能不如 MINRES, 但对于坏条件问题, SYMMLQ 通常比 MINRES 更稳定.

在 SYMMLQ 方法中, 我们取 $\mathcal{L} = \mathcal{K}$, 即

$$\text{find} \quad x^{(k)} \in x^{(0)} + \mathcal{K}_k \quad \text{such that} \quad b - Ax^{(k)} \perp \mathcal{K}_k$$

✍ 当 A 对称正定时, SYMMLQ 方法与 CG 方法是等价的.

近似解的表示

- 设 $x^{(k)}$ 是 SYMMLQ 在 $x^{(0)}+\mathcal{K}_k$ 中找到的近似解, 则 $x^{(k)}$ 可表示为

$$x^{(k)}=x^{(0)}+V_k y_k, \quad \text{其中} \quad y_k=T_k^{-1}(\beta e_1).$$

如果 A 不定, 则 T_k 也不定, 因此无法使用 $\mathrm{LDL}^{\mathsf{T}}$ 分解来求解 y_k.

- 此时, 我们改为对 T_k 进行 LQ 分解 (类似于 QR 分解), 即

$$T_k=\tilde{L}_k Q_k, \tag{2.26}$$

其中 $\tilde{L}_k\in\mathbb{R}^{k\times k}$ 是下三角矩阵, $Q_k\in\mathbb{R}^{k\times k}$ 是正交矩阵. 于是

$$x^{(k)}=x^{(0)}+V_k T_k^{-1}(\beta e_1)=x^{(0)}+V_k Q_k^{\mathsf{T}}\tilde{L}_k^{-1}(\beta e_1)=\boxed{x^{(0)}+\tilde{P}_k\tilde{z}^{(k)}},$$

其中

$$\tilde{P}_k\triangleq V_k Q_k^{\mathsf{T}}\in\mathbb{R}^{n\times k}, \quad \tilde{z}^{(k)}\triangleq\tilde{L}_k^{-1}(\beta e_1)\in\mathbb{R}^k.$$

LQ 分解的递推算法

❶ 由于 T_k 是三对角矩阵, 因此 T_k 的 LQ 分解可以通过 Givens 变换实现

❷ T_{k+1} 的 LQ 分解可以在 T_k 的 LQ 分解基础上, 通过一次 Givens 变换得到

- 当 $k=1$ 时, $T_k=\alpha_1$, 令 $L=1$, $Q=\alpha_1$ 即可.
- 当 $k=2$ 时, $T_2=\begin{bmatrix}\alpha_1 & \beta_1\\ \beta_1 & \alpha_2\end{bmatrix}$, 只需一次 Givens 变化即可.
- 设 T_k 的 LQ 分解为

$$T_k=\tilde{L}_k(G_1G_2\cdots G_{k-1})^{\mathsf{T}}\triangleq\tilde{L}_kQ_k,$$

其中 $Q_k=(G_1G_2\cdots G_{k-1})^{\mathsf{T}}$. 这里 G_i 是 Givens 变换.

- 由于 T_k 是三对角, 因此 $\tilde{L}_k = T_k Q_k^{\mathsf{T}}$ 只有三条对角线非零, 即

$$\tilde{L}_k = \begin{bmatrix} l_1^{(1)} & & & & & \\ l_2^{(2)} & l_2^{(1)} & & & & \\ l_3^{(3)} & l_3^{(2)} & l_3^{(1)} & & & \\ & \ddots & \ddots & \ddots & & \\ & & l_{k-1}^{(3)} & l_{k-1}^{(2)} & l_{k-1}^{(1)} & \\ & & & l_k^{(3)} & l_k^{(2)} & \tilde{l}_k^{(1)} \end{bmatrix}.$$

注记

- 这里是对 T_k 做 LQ 分解, 不是对 $T_{k+1,k}$ 做 LQ 分解.
- $\tilde{L}_k$ 最右下角的元素上面有个波浪号, 是用于区分 $\tilde{L}_k$ 和 L_k, 其中 L_k 表示 $\tilde{L}_{k+1}$ 的 k 阶顺序主子矩阵. 后面会看到, $\tilde{L}_k$ 并不是 $\tilde{L}_{k+1}$ 的子矩阵.

令

$$\tilde{Q}_{k+1} \triangleq \begin{bmatrix} Q_k & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{(k+1)\times(k+1)},$$

则

$$T_{k+1}\tilde{Q}_{k+1}^{\mathsf{T}} = \left[\begin{array}{ccc|c} & T_k & & \begin{matrix}0\\ \vdots \\ 0\\ \beta_k\end{matrix} \\ \hline 0 & \cdots & 0 \quad 0 \quad \beta_k & \alpha_{k+1} \end{array}\right] \begin{bmatrix} Q_k^{\mathsf{T}} & 0 \\ 0 & 1 \end{bmatrix} = \left[\begin{array}{ccc|c} & \tilde{L}_k & & \begin{matrix}0\\ \vdots \\ 0\\ \beta_k\end{matrix} \\ \hline 0 & \cdots & 0 \quad l_{k+1}^{(3)} \quad \tilde{\beta}_k & \alpha_{k+1} \end{array}\right],$$

其中

$$\begin{bmatrix} 0 & \cdots & 0 & l_{k+1}^{(3)} & \tilde{\beta}_k \end{bmatrix} = \begin{bmatrix} 0 & \cdots & 0 & 0 & \beta_k \end{bmatrix} Q_k^{\mathsf{T}} = \begin{bmatrix} 0 & \cdots & 0 & 0 & \beta_k \end{bmatrix} G_{k-1}.$$

即 $l_{k+1}^{(3)} = -s_{k-1}\beta_k$, $\tilde{\beta}_k = c_{k-1}\beta_k$.

- 构造 Givens 变换, 作用到 $T_{k+1}\tilde{Q}_{k+1}^{\mathsf{T}}$ 的最后两列上, 消去最后一列中的 β_k, 即

$$G_k^{\mathsf{T}} = \begin{bmatrix} I_{k-1} & & \\ & c_k & s_k \\ & -s_k & c_k \end{bmatrix} \in \mathbb{R}^{(k+1)\times(k+1)}, \quad c_k = \frac{\tilde{l}_k^{(1)}}{l_k^{(1)}},\ s_k = \frac{\beta_k}{l_k^{(1)}},\ l_k^{(1)} = \sqrt{\left(\tilde{l}_k^{(1)}\right)^2 + \beta_k^2}.$$

- 令 $Q_{k+1} = G_k^{\mathsf{T}}\tilde{Q}_{k+1}$. 由于右乘 G_k 时只会影响到最后两列的元素, 所以

$$T_{k+1}Q_{k+1}^{\mathsf{T}} = \left[\begin{array}{ccc|c} & \tilde{L}_k & & \begin{matrix}0\\ \vdots\\ 0\\ \beta_k\end{matrix} \\ \hline 0 & \cdots\ 0\ l_{k+1}^{(3)} & \tilde{\beta}_k & \alpha_{k+1} \end{array}\right] G_k = \left[\begin{array}{ccccc|c} l_1^{(1)} & & & & & 0 \\ l_2^{(2)} & l_2^{(1)} & & & & \vdots \\ l_3^{(3)} & l_3^{(2)} & l_3^{(1)} & & & \vdots \\ & \ddots & \ddots & \ddots & & \vdots \\ & & l_k^{(3)} & l_k^{(2)} & l_k^{(1)} & 0 \\ \hline 0 & \cdots & 0 & l_{k+1}^{(3)} & l_{k+1}^{(2)} & \tilde{l}_{k+1}^{(1)} \end{array}\right] \triangleq \tilde{L}_{k+1}$$

于是, 我们就得到 T_{k+1} 的 LQ 分解 $\boxed{T_{k+1} = \tilde{L}_{k+1}Q_{k+1}}$

- 记 $\tilde{L}_k$ 的 $k-1$ 阶顺序主子矩阵为 L_{k-1}, $\tilde{L}_{k+1}$ 的 k 阶顺序主子矩阵为 L_k.
- 由于 L_k 和 $\tilde{L}_k$ 只有第 (k,k) 元素不同, 因此 L_{k-1} 是 L_k 的 $k-1$ 阶顺序主子矩阵.

> **递推关系:** L_{j-1} 是 L_j 的 $j-1$ 阶顺序主子矩阵, $j=1,2,....$

- 令 $z^{(k)}=L_k^{-1}(\beta e_1)\in\mathbb{R}^k$. 由于 $\tilde{z}^{(k)}=\tilde{L}_k^{-1}(\beta e_1)$, 所以

$$z^{(k)}=\begin{bmatrix} z^{(k-1)} \\ z_k \end{bmatrix}, \quad \tilde{z}^{(k)}=\begin{bmatrix} z^{(k-1)} \\ \tilde{z}_k \end{bmatrix},$$

其中 $z^{(k-1)}=L_{k-1}^{-1}(\beta e_1)$, 因此, 我们可以将 $z^{(k)}$ 和 $\tilde{z}^{(k)}$ 写为

$$z^{(k)}=[z_1,\ldots,z_{k-1},z_k]^{\mathsf{T}}, \quad \tilde{z}^{(k)}=[z_1,\ldots,z_{k-1},\tilde{z}_k]^{\mathsf{T}}, \quad k=1,2,\ldots,$$

其中 $\tilde{z}_k=\dfrac{l_k^{(1)}z_k}{\tilde{l}_k^{(1)}}$.

- 由 $L_k z^{(k)} = \beta e_1$ 可知

$$\begin{cases} z_1 = \beta / l_1^{(1)}, \quad z_2 = -l_2^{(2)} z_1 / l_2^{(1)}, \\ z_i = -\left(l_i^{(3)} z_{i-2} + l_i^{(2)} z_{i-1} \right) / l_i^{(1)}, \quad i = 3, 4, \ldots, k. \end{cases} \tag{2.27}$$

- 分别记 $\tilde{P}_k = V_k Q_k^{\mathsf{T}}$ 和 $\tilde{P}_{k+1} = V_{k+1} Q_{k+1}^{\mathsf{T}}$ 的前 $k-1$ 列和前 k 列为 P_{k-1} 和 P_k, 则

$$\begin{aligned} \tilde{P}_{k+1} = V_{k+1} Q_{k+1}^{\mathsf{T}} &= [V_k, v_{k+1}] \begin{bmatrix} Q_k^{\mathsf{T}} & 0 \\ 0 & 1 \end{bmatrix} G_k \\ &= \left[\tilde{P}_k, v_{k+1} \right] G_k = [P_{k-1}, \tilde{p}_k, v_{k+1}] \begin{bmatrix} I_{k-1} & & \\ & c_k & s_k \\ & -s_k & c_k \end{bmatrix} \\ &= [P_{k-1}, p_k, \tilde{p}_{k+1}] = [P_k, \tilde{p}_{k+1}], \end{aligned} \tag{2.28}$$

- 因此, P_k 和 P_{k-1} 有下面的递推关系 $\boxed{P_k = [P_{k-1}, p_k]}$, 所以, P_k 可统一写为

$$P_k = [p_1, p_2, \ldots, p_k], \quad k = 1, 2, \ldots.$$

- 易知 $\tilde{p}_1 = \tilde{P}_1 = v_1$, 由 (2.28) 可得

$$p_k = c_k\tilde{p}_k - s_k v_{k+1}, \quad \tilde{p}_{k+1} = s_k\tilde{p}_k + c_k v_{k+1}, \quad k = 1, 2, \ldots.$$

- 令 $\tilde{x}^{(k)} = x^{(0)} + P_k z^{(k)}$, $k = 1, 2, \ldots$, 则

$$\tilde{x}^{(k)} = x^{(0)} + P_k z^{(k)} = x^{(0)} + [P_{k-1}, p_k]\begin{bmatrix} z^{(k-1)} \\ z_k \end{bmatrix} = \tilde{x}^{(k-1)} + z_k p_k, \quad k = 1, 2, \ldots. \tag{2.29}$$

- 于是有

$$\begin{aligned} x^{(k+1)} = x^{(0)} + \tilde{P}_{k+1}\tilde{z}^{(k+1)} &= x^{(0)} + [P_k, \tilde{p}_{k+1}]\begin{bmatrix} z^{(k)} \\ \tilde{z}_{k+1} \end{bmatrix} \\ &= x^{(0)} + P_k z^{(k)} + \tilde{z}_{k+1}\tilde{p}_{k+1} = \tilde{x}^{(k)} + \tilde{z}_{k+1}\tilde{p}_{k+1}. \end{aligned} \tag{2.30}$$

有了这个关系式后, 我们在实际计算中只需计算 $\tilde{x}^{(k)}$.

残量范数的计算

与 FOM 类似, 我们可得

$$r_k = b - Ax^{(k)} = -\beta_k (e_k^{\mathsf{T}} y_k) v_{k+1} \quad \Longrightarrow \quad \|r_k\|_2 = |\beta_k (e_k^{\mathsf{T}} y_k)|.$$

☞ 需要计算 $y_k = T_k^{-1}(\beta e_1)$.

但事实上, 我们只需知道 y_k 最后一个分量即可.

注意到 T_k 是对称的, 因此 $T_k^{\mathsf{T}} y_k = T_k y_k = \beta e_1$, 所以 $\tilde{L}_k^{\mathsf{T}} y_k = Q_k(\beta e_1)$.

观察上述等式两边的最后一个元素可知

$$\begin{aligned} \tilde{l}_k^{(1)} e_k^{\mathsf{T}} y_k = e_k^{\mathsf{T}} \tilde{L}_k^{\mathsf{T}} y_k &= e_k^{\mathsf{T}} Q_k(\beta e_1) \\ &= \beta e_k^{\mathsf{T}} (G_1 G_2 \cdots G_{k-1})^{\mathsf{T}} e_1 = \beta (G_1 G_2 \cdots G_{k-1} e_k)^{\mathsf{T}} e_1 = \beta s_1 s_2 \cdots s_{k-1}. \end{aligned}$$

由 Givens 变换 G_k 的具体构造公式可知, $s_k\tilde{l}_k^{(1)} + c_k\beta_k = 0$. 所以

$$r_k = -\beta_k(e_k^{\mathsf{T}} y_k)v_{k+1} = -\left(\beta s_1 s_2 \cdots s_{k-1}\beta_k/\tilde{l}_k^{(1)}\right) v_{k+1} = (\beta s_1 s_2 \cdots s_{k-1} s_k/c_k)v_{k+1},$$

因此

$$\|r_k\|_2 = |\beta s_1 s_2 \cdots s_{k-1} s_k/c_k| = |c_{k-1} s_k/c_k| \cdot \|r_{k-1}\|_2.$$

算法 SYMMLQ 方法

1: **给定初值** $x^{(0)}$, **(相对) 精度要求** $\varepsilon > 0$ **和最大迭代步数** IterMax
2: **计算** $r_0 = b - Ax^{(0)}$ **和** $\beta = \|r_0\|_2$
3: $v_1 = r_0/\beta$, $p_1 = v_1$, $\xi_0 = \beta$, $\tilde{x}^{(0)} = x^{(0)}$
4: **for** $k = 1, 2, \ldots,$ **do**
5: $\quad w_k = Av_k - \beta_{k-1}v_{k-1}$ where $\beta_0 = 0$ and $v_0 = 0$
6: $\quad \alpha_k = (w_k, v_k)$
7: $\quad \tilde{w}_k = w_k - \alpha_k v_k$
8: $\quad \beta_k = \|\tilde{w}_k\|_2$
9: $\quad$ **if** $k = 1$ **then**
10: $\quad\quad \tilde{l}_k^{(1)} = \alpha_k$
11: $\quad$ **end if**
12: $\quad$ **if** $k = 2$ **then**
13: $\quad\quad \tilde{\beta}_{k-1} = \beta_{k-1}$
14: $\quad$ **end if**
15: $\quad$ **if** $k > 2$ **then** $\quad$ % apply G_{k-2} to the last row of T_k

16: $\begin{bmatrix} l_k^{(3)} & \tilde{\beta}_{k-1} \end{bmatrix} = \begin{bmatrix} 0 & \beta_{k-1} \end{bmatrix} \begin{bmatrix} c_{k-2} & s_{k-2} \\ -s_{k-2} & c_{k-2} \end{bmatrix}$

17: **end if**

18: **if** $k > 1$ **then** % form the Givens rotation G_{k-1}

19: **if** $|\tilde{l}_{k-1}^{(1)}| > |\beta_{k-1}|$ **then**

20: set $c_{k-1} = \frac{1}{\sqrt{1+\gamma^2}}$, $s_{k-1} = -c_{k-1}\gamma$ where $\gamma = \beta_{k-1}/\tilde{l}_{k-1}^{(1)}$

21: **else**

22: set $s_{k-1} = \frac{1}{\sqrt{1+\gamma^2}}$, $c_{k-1} = -s_{k-1}\gamma$ where $\gamma = \tilde{l}_{k-1}^{(1)}/\beta_{k-1}$

23: **end if**

24: **end if**

25: **if** $k > 1$ **then** % apply G_{k-1} to the last two columns of $T_k \tilde{Q}_{k-1}$

26: $l_{k-1}^{(1)} = \sqrt{\left(\tilde{l}_{k-1}^{(1)}\right)^2 + \beta_{k-1}^2}$

27: $\begin{bmatrix} l_k^{(2)} & \tilde{l}_k^{(1)} \end{bmatrix} = \begin{bmatrix} \tilde{\beta}_{k-1} & \alpha_k \end{bmatrix} \begin{bmatrix} c_{k-1} & s_{k-1} \\ -s_{k-1} & c_{k-1} \end{bmatrix}$

28: **end if**

29: **if** $k = 2$ **then** % compute z_k

30: $z_1 = \beta / l_1^{(1)}$

31: **end if**

32: **if** $k = 3$ **then**

33: $z_2 = -l_2^{(2)} z_1 / l_2^{(1)}$

34: **end if**

35: **if** $k > 3$ **then**

36: $z_{k-1} = -\left(l_{k-1}^{(3)} z_{k-3} + l_{k-1}^{(2)} z_{k-2} \right) / l_{k-1}^{(1)}$

37: **end if**

38: **if** $k = 1$ **then** % compute p_k

39: $\tilde{p}_1 = v_1$

40: **end if**

41: **if** $k > 1$ **then**

42: $p_{k-1} = c_{k-1} \tilde{p}_{k-1} - s_{k-1} v_k$

43: $\tilde{p}_k = s_{k-1} \tilde{p}_{k-1} + c_{k-1} v_k$

44: **end if**

45: **if** $k > 1$ **then** % update $\tilde{x}^{(k)}$

$\tilde{x}^{(k-1)} = \tilde{x}^{(k-2)} + z_{k-1} p_{k-1}$

end if

if $k > 1$ **then** % check convergence

$\xi_{k-1} = (c_{k-2} s_{k-1} / c_{k-1}) \, \xi_{k-2}$

if $|\xi_{k-1}| < \varepsilon$ **then**

$x^{(k)} = \tilde{x}_{k-1} + \left(l_k^{(1)} z_k / \tilde{l}_k^{(1)} \right) \tilde{p}_k$

stop

end if

end if

$v_{k+1} = \tilde{w}_k / \beta_k$

end for

注记

- 由 $r_k = -\beta_k(e_k^{\mathsf{T}} y_k)v_{k+1}$ 可知, 在 SYMMLQ 方法中, 残量是相互正交的.
- 如果 A 是对称正定的, 则 SYMMLQ 与 CG 等价, 此时 SYMMLQ 极小化 $\|x^{(k)} - x_*\|_A$. 如果 A 是不定的, 则没有这个最优性质. 但可以证明, SYMMLQ 在仿射空间 $x^{(0)} + A\mathcal{K}_k(A, r_0)$ 中极小化 $\|x^{(k)} - x_*\|_2$.

2-5 | 收敛性分析

2.5 收敛性分析

https://math.ecnu.edu.cn/~jypan/Teaching/IMP

- 从理论上讲, 如果不考虑舍入误差的影响, Krylov 子空间方法都将在 n 个迭代步内终止. 但在实际应用中, 我们通常希望收敛时的迭代步数远远小于 n. 另一方面, 由于舍入误差的存在, 使得方法不一定都能在 n 步内收敛.
- 收敛性分析不仅仅是给出理论上的结果, 而且还能用来帮助我们改善方法.

2-5-1 CG 方法的收敛性

最优性质

$$\|x^{(k)} - x_*\|_A = \min_{x \in x^{(0)} + \mathcal{K}_k(A, r_0)} \|x - x_*\|_A$$

记 $\mathbb{P}_k$ 为所有次数不超过 k 的实系数多项式组成的集合, 则

$$\mathcal{K}_k(A, r_0) = \{z = p(A)r_0 \ : \ p \in \mathbb{P}_{k-1}\}$$

设 $x \in x^{(0)} + \mathcal{K}_k(A, r_0)$, 则存在多项式 $p(t) \in \mathbb{P}_{k-1}$ 使得

$$x = x^{(0)} + p(A)r_0.$$

误差分析

- 记 $\varepsilon_0 \triangleq x^{(0)} - x_*$, 则

$$x - x_* = \varepsilon_0 + p(A)(b - Ax^{(0)}) = \varepsilon_0 + p(A)(Ax_* - Ax^{(0)}) = \big(I - Ap(A)\big)\varepsilon_0.$$

于是

$$\|x - x_*\|_A^2 = \varepsilon_0^{\mathsf{T}}\big(I - Ap(A)\big)^{\mathsf{T}} A\big(I - Ap(A)\big)\varepsilon_0 \triangleq \varepsilon_0^{\mathsf{T}} q(A)^{\mathsf{T}} A q(A)\varepsilon_0$$

其中 $q(t) = 1 - t\,p(t) \in \mathbb{P}_k$ 满足 $q(0) = 1$.

- 令 $A = Q\Lambda Q^{\mathsf{T}}$, 其中 $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, $\lambda_i > 0$.

记 $y = [y_1, y_2, \ldots, y_n]^{\mathsf{T}} \triangleq Q^{\mathsf{T}}\varepsilon_0$.

$$
\begin{aligned}
\|x^{(k)} - x_*\|_A^2 &= \min_{x \in x^{(0)} + \mathcal{K}_k(A, r_0)} \|x - x_*\|_A^2 \\
&= \min_{q \in \mathbb{P}_k,\ q(0)=1} \varepsilon_0^{\mathsf{T}}\, q(A)^{\mathsf{T}} A\, q(A) \varepsilon_0 \\
&= \min_{q \in \mathbb{P}_k,\ q(0)=1} \varepsilon_0^{\mathsf{T}}\, Q q(\Lambda)^{\mathsf{T}} \Lambda q(\Lambda)\, Q^{\mathsf{T}} \varepsilon_0 \\
&= \min_{q \in \mathbb{P}_k,\ q(0)=1} y^{\mathsf{T}} q(\Lambda)^{\mathsf{T}} \Lambda q(\Lambda) y \\
&= \min_{q \in \mathbb{P}_k,\ q(0)=1} \sum_{i=1}^{n} y_i^2 \lambda_i q(\lambda_i)^2 \\
&\leq \min_{q \in \mathbb{P}_k,\ q(0)=1} \max_{1 \leq i \leq n} \{ q(\lambda_i)^2 \} \sum_{i=1}^{n} y_i^2 \lambda_i \\
&= \min_{q \in \mathbb{P}_k,\ q(0)=1} \max_{1 \leq i \leq n} \{ q(\lambda_i)^2 \}\ y^{\mathsf{T}} \Lambda y \\
&= \min_{q \in \mathbb{P}_k,\ q(0)=1} \max_{1 \leq i \leq n} \{ q(\lambda_i)^2 \}\ \varepsilon_0^{\mathsf{T}} A \varepsilon_0 = \min_{q \in \mathbb{P}_k,\ q(0)=1} \max_{1 \leq i \leq n} \{ q(\lambda_i)^2 \}\ \|\varepsilon_0\|_A^2
\end{aligned}
$$

$$
\frac{\|x^{(k)} - x_*\|_A^2}{\|x^{(0)} - x_*\|_A^2} \leq \min_{q \in \mathbb{P}_k,\ q(0)=1} \max_{1 \leq i \leq n} \{ q(\lambda_i)^2 \}
$$

定理 设 $A \in \mathbb{R}^{n\times n}$ 对称正定, 则由 CG 方法得到的近似解 $x^{(k)}$ 满足

$$\frac{\|x^{(k)} - x_*\|_A}{\|x^{(0)} - x_*\|_A} \leq \min_{q\in\mathbb{P}_k,\ q(0)=1} \max_{1\leq i\leq n} |q(\lambda_i)|. \tag{2.31}$$

注记

- 需要指出的是, (2.31) 中上界是紧凑的, 即对任意 k, 总存在某个右端项 b 或某个初始值 $x^{(0)}$ (与 k 和 A 有关), 使得 (2.31) 中的等号成立. 也就是说, (2.31) 中的上界描述了 CG 方法在最坏情况下的收敛情况.
- 由于 (2.31) 中的上界依赖于 A 的所有特征值. 在通常情况下, A 的特征值是很难计算的, 因此该结论缺乏实用性.

上界的缩放

如果我们知道 A 的最大和最小特征值, 则可缩放为

$$\frac{\|x^{(k)}-x_*\|_A}{\|x^{(0)}-x_*\|_A} \leq \min_{q\in\mathbb{P}_k,\ q(0)=1} \max_{1\leq i\leq n} |q(\lambda_i)| \leq \min_{q\in\mathbb{P}_k,\ q(0)=1} \max_{\lambda_{\min}\leq\lambda\leq\lambda_{\max}} |q(\lambda)|.$$

利用 Chebyshev 多项式的性质, 我们可以得到下面的结论.

定理 设 $A\in\mathbb{R}^{n\times n}$ 对称正定, 且其最大与最小特征值分别为 $\lambda_{\max}$ 和 $\lambda_{\min}$, 则

$$\frac{\|x^{(k)}-x_*\|_A}{\|x^{(0)}-x_*\|_A} \leq 2\left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^k, \tag{2.32}$$

其中 $\kappa(A)=\lambda_{\max}/\lambda_{\min}$ 是 A 的 (谱) 条件数.

注记

- 需要注意的是, 上界 (2.31) 和 (2.32) 有着本质性差异. 它们的值可能会相差很大. 所以 (2.32) 并不能完全描述 CG 方法的收敛性质.
- A 的条件数越小, 则收敛越快.
  但 A 的条件数很大却并不一定表明 CG 收敛会很慢.
- 在很多实际应用中, 人们观察到 CG 方法往往具有超收敛性, 即方法的平均收敛速度随着迭代步数的增加而不断加快.

CG 方法的超收敛性

定理 设 A 对称正定, 且其特征值为

$$0 < \lambda_n \leq \cdots \leq \lambda_{n+1-i} \leq b_1 \leq \lambda_{n-i} \leq \cdots \leq \lambda_{j+1} \leq b_2 \leq \lambda_j \leq \cdots \leq \lambda_1.$$

则当 $k \geq i+j$ 时有

$$\frac{\|x^{(k)} - x_*\|_A}{\|x^{(0)} - x_*\|_A} \leq 2\left(\frac{b-1}{b+1}\right)^{k-i-j} \max_{\lambda\in[b_1,b_2]} \left\{ \prod_{\ell=n+1-i}^{n} \left(\frac{\lambda-\lambda_\ell}{\lambda_\ell}\right) \prod_{\ell=1}^{j} \left(\frac{\lambda_\ell-\lambda}{\lambda_\ell}\right) \right\},$$

其中

$$b = \left(\frac{b_2}{b_1}\right)^{\frac{1}{2}} \geq 1.$$

若 A 的特征值聚集在 1 附近, 即除了有限多个特征值外, 其余都在 $[1-\varepsilon, 1+\varepsilon]$ 中.

推论 设 A 对称正定, 特征值为

$$0 < \delta \leq \lambda_n \leq \cdots \leq \lambda_{n+1-i} \leq 1-\varepsilon \leq \lambda_{n-i} \leq \cdots \leq \lambda_{j+1} \leq 1+\varepsilon \leq \lambda_j \leq \cdots \leq \lambda_1.$$

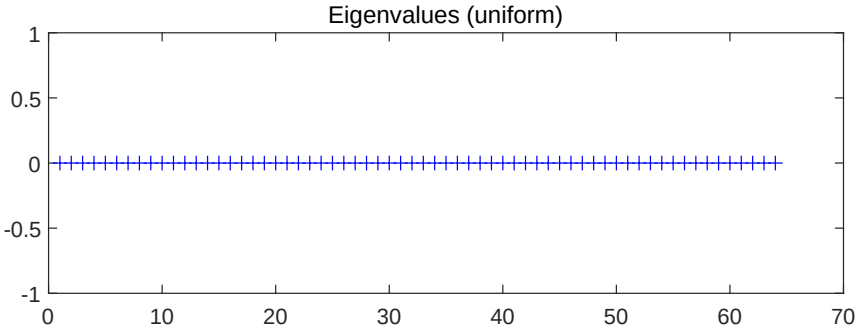
则当 $k \geq i+j$ 时有

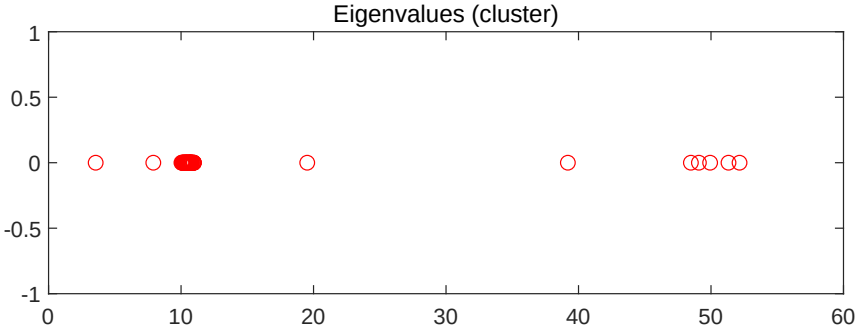
$$\frac{\|x^{(k)} - x_*\|_A}{\|x^{(0)} - x_*\|_A} \leq 2\left(\frac{1+\varepsilon}{\delta}\right)^i \varepsilon^{k-i-j}.$$

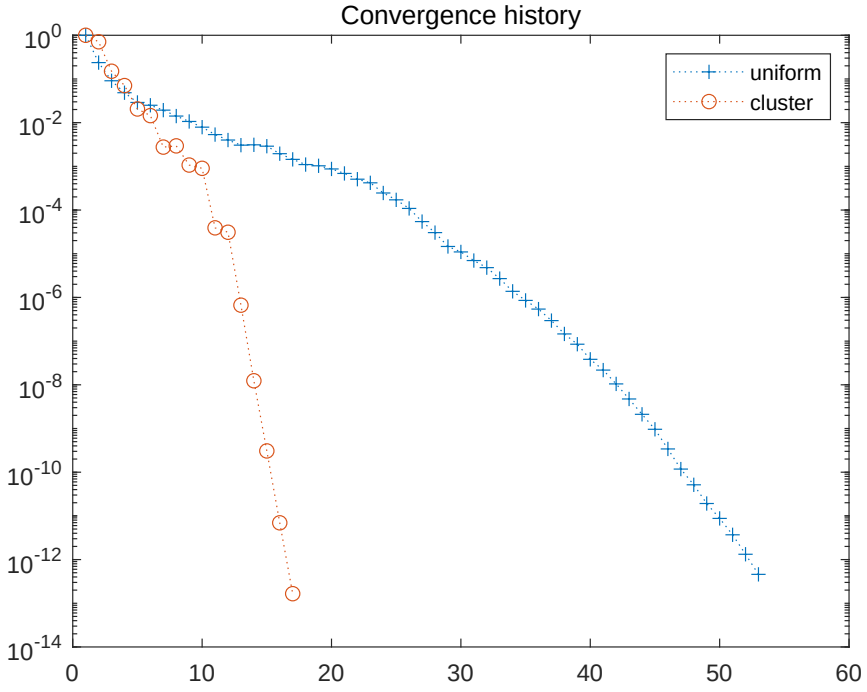
CG 收敛性举例

```
n = 64;
tol = 1e-12;

% 特征值均匀分布
Lambda = linspace(1,100,n);
A = Q*diag(Lambda)*Q';
... ...

% 特征值聚集分布
m = 10;
Lambda = [1+(n-1)*rand(m,1); 10*ones(n-m,1)+ 1*rand(n-m,1)];
A = Q*diag(Lambda)*Q';
... ...
```

(CG_superconvergence.m)


Eigenvalues (uniform)
1
0.5
0
-0.5
-1
0
10
20
30
40
50
60
70


Eigenvalues (cluster)
1
0.5
0
-0.5
-1
0
10
20
30
40
50
60


Convergence history
uniform
cluster
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
0
10
20
30
40
50
60

2-5-2 GMRES 方法的收敛性

GMRES 最优性质

$$\|b - Ax^{(k)}\|_2 = \min_{x \in x^{(0)} + \mathcal{K}_k(A, r_0)} \|b - Ax\|_2$$

正规矩阵情形

- 设 A 是正规矩阵, 即 $A = U\Lambda U^*$, 其中 $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, $\lambda_i \in \mathbb{C}$.
- 设 $x \in x^{(0)} + \mathcal{K}_k(A, r_0)$, 则存在多项式 $p(t) \in \mathbb{P}_{k-1}$ 使得 $x = x^{(0)} + p(A)r_0$. 于是

$$b - Ax = b - Ax^{(0)} - Ap(A)r_0 = (I - Ap(A))r_0 \triangleq q(A)r_0, \tag{2.33}$$

其中 $q(t) = 1 - tp(t) \in \mathbb{P}_k$ 满足 $q(0) = 1$. 直接计算可知

$$\|b - Ax\|_2 = \|q(A)r_0\|_2 = \|Uq(\Lambda)U^*r_0\|_2 \le \|U\|_2\|U^*\|_2\|q(\Lambda)\|_2\|r_0\|_2 = \|q(\Lambda)\|_2\|r_0\|_2.$$

- 又 Λ 是对角矩阵, 所以

$$\|q(\Lambda)\|_2 = \max_{1\le i\le n} |q(\lambda_i)|.$$

- 设 $x^{(k)}$ 是由 GMRES 方法得到的近似解. 由 GMRES 方法的最优性可知

$$\begin{aligned}\|b-Ax^{(k)}\|_2 &= \min_{x\in x^{(0)}+\mathcal{K}_k(A,r_0)} \|b-Ax\|_2\\ &= \min_{q\in\mathbb{P}_k,\ q(0)=1} \|q(A)r_0\|_2\\ &\le \min_{q\in\mathbb{P}_k,\ q(0)=1} \|q(\Lambda)\|_2\|r_0\|_2\\ &= \|r_0\|_2 \min_{q\in\mathbb{P}_k,\ q(0)=1}\ \max_{1\le i\le n} |q(\lambda_i)|.\end{aligned}$$

- 于是, 我们有下面的结论.

定理 设 $A\in\mathbb{R}^{n\times n}$ 是正规矩阵, $x^{(k)}$ 是 GMRES 方法得到的近似解, 则

$$\frac{\|b-Ax^{(k)}\|_2}{\|r_0\|_2} \le \min_{q\in\mathbb{P}_k,\ q(0)=1}\ \max_{1\le i\le n} |q(\lambda_i)|. \tag{2.34}$$

✍ 需要指出的是, 上界 (2.34) 是紧凑的.

误差界的缩放

选取一个包含 A 的所有特征值的区域 $\Omega \subset \mathbb{C}$, 上界缩放为

$$\min_{q\in\mathbb{P}_k,\ q(0)=1} \max_{\lambda\in\Omega} |q(\lambda)|. \tag{2.35}$$

通常 Ω 必须是连通的, 否则 (2.35) 的求解非常困难.

✍ Ω 不能包含原点.

情形一: Ω 是复平面上的一个圆盘

- 由于 A 是实的, 其特征值关于实轴对称, 可设 Ω 以 $C(c,0)$ 为圆心, 半径为 $r>0$ (假定 $r<|c|$, 即原点不在 Ω 内)
- 定义多项式 $q_k(z)=\left(\dfrac{c-z}{c}\right)^k$, 则 $q_k(z)\in\mathbb{P}_k$ 且 $q_k(0)=1$. 所以

$$\min_{q\in\mathbb{P}_k,\ q(0)=1}\ \max_{1\le i\le n}|q(\lambda_i)|\le\max_{1\le i\le n}|q_k(\lambda_i)|=\max_{1\le i\le n}\left|\frac{c-\lambda_i}{c}\right|^k\le\frac{r^k}{|c|^k}.$$

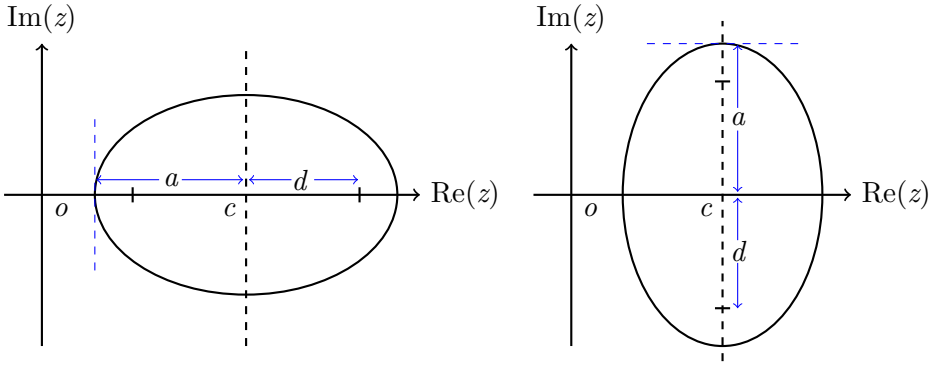
由此可见, 当 r 越小或 c 越大时, 右式趋向于 0 的速度就越快.

结论

设 A 是正规矩阵, 其特征值包含在一个圆盘内, 则圆盘的半径越小或者圆心离原点越远, 则 GMRES 收敛越快.

情形二: Ω 是复平面上的一个椭圆盘

- 如果用椭圆盘来代替圆盘, 则可得出更紧凑的上界.
- 设 A 的特征值全部包含在椭圆盘 $E(c,d,a)$ 内, 其中 $d>0$ 为焦距, $a>0$ 为半长轴长, $C(c,0)$ 为椭圆中心. 并且假定原点不在椭圆盘内. 如下图所示.



假定椭圆盘 $E(c,d,a)$ 的长轴在实轴上.

令 $T_k(z)$ 为复 Chebyshev 多项式, 则

$$\tilde{T}_k(z)=\frac{T_k(\frac{c-z}{d})}{T_k(\frac{c}{d})}$$

就是极小极大问题

$$\min_{q\in\mathbb{P}_k,\ q(0)=1}\ \max_{\lambda\in E(c,d,a)}|q(\lambda)| \tag{2.36}$$

的渐进最优解.

定理 设 $A\in\mathbb{R}^{n\times n}$ 是正规矩阵, $x^{(k)}$ 是由 GMRES 得到的近似解, 则

$$\frac{\|b-Ax^{(k)}\|_2}{\|r_0\|_2}\leq\frac{T_k(\frac{a}{d})}{|T_k(\frac{c}{d})|}\approx\left(\frac{a+\sqrt{a^2-d^2}}{c+\sqrt{c^2-d^2}}\right)^k$$

非正规情形

当 A 不是正规矩阵时, 在一般情况下, GMRES 方法的收敛性很难分析.

- 如果 $A \in \mathbb{R}^{n\times n}$ 是可对角化的, 即 $A = X\Lambda X^{-1}$, 则

$$\|b - Ax^{(k)}\|_2 = \min_{x\in x^{(0)}+\mathcal{K}_k(A,r_0)} \|b - Ax\|_2 = \min_{q\in\mathbb{P}_k,\ q(0)=1} \|q(A)r_0\|_2.$$

相类似地, 我们可以得到下面的结论.

定理 设 $A = X\Lambda X^{-1}$, $x^{(k)}$ 是 GMRES 方法得到的近似解, 则

$$\frac{\|b - Ax^{(k)}\|_2}{\|r_0\|_2} \leq \kappa(X) \min_{q\in\mathbb{P}_k,\ q(0)=1} \max_{1\leq i\leq n} |q(\lambda_i)|,$$

其中 $\kappa(X)$ 是 X 的谱条件数.

注记

- 如果 A 接近正规, 则 $\kappa(X) \approx 1$, 此时定理中的上界在一定程度上能描述 GMRES 的收敛速度. 但如果 X 远非正交, 则 $\kappa(X)$ 会很大, 此时的上界就失去实际意义了.
- 需要指出的是, 上面的分析 **并不意味着非正规矩阵就一定比正规矩阵收敛慢**. 事实上, 对任意一个非正规矩阵, 总存在一个相应的正规矩阵, 使得 GMRES 方法的收敛速度是一样的 (假定初始残量相同)

虽然 GMRES 的收敛性与 A 的特征值有关, 但并不仅仅取决于特征值的分布.

定理 对任意给定的特征值分布和一条不增的曲线, 总存在一个矩阵 A 和一个右端项 b, 使得 A 具有指定的特征值分布, 且 GMRES 的收敛曲线与给定的曲线相同.

如果 A 不可以对角化, 通常会用一个新的极小化问题来近似原来的极小化问题

- 事实上, 我们有

$$\frac{\|b - Ax^{(k)}\|_2}{\|r_0\|_2} = \frac{\min\limits_{q\in\mathbb{P}_k,\ q(0)=1} \|q(A)r_0\|_2}{\|r_0\|_2} \leq \max_{\|v\|_2=1} \min_{q\in\mathbb{P}_k,\ q(0)=1} \|q(A)v\|_2 \tag{2.37}$$

$$\leq \min_{q\in\mathbb{P}_k,\ q(0)=1} \|q(A)\|_2. \tag{2.38}$$

不等式 (2.37) 右端代表的是在最坏情况下的 GMRES 收敛性, 而且是紧凑的.

- 可以证明, 当 A 是正规矩阵时, 上界 (2.37) 和 (2.38) 是相等的.

注记

最后需要指出的是, 收敛性也依赖于迭代初值和右端项. 以上结论中的上界描述的都是最坏情况下的收敛速度. 在实际计算中, 算法的收敛速度可能会比预想的要快得多.

2-6 | 基于双正交化过程的迭代方法

2.6 基于双正交化过程的迭代方法

2-6-1 双正交化过程

什么是双正交

双正交化过程 (biorthogonalization) **也称为双边 Lanczos 过程** (two-sided Lanczos Process), **是指同时寻找子空间** $\mathcal{K}(A, r_0)$ **的一组基** $\{v_1, v_2, \ldots\}$ **和子空间** $\mathcal{K}(A^{\mathsf{T}}, \tilde{r}_0)$ **的一组基** $\{w_1, w_2, \ldots\}$, **使得这两组基之间相互正交, 即** $(v_i, w_j) = 0$, $i \neq j$. **这样做的优点是可以利用三项递推公式, 从而节省运算量.**

双正交化过程也可以看作是将对称矩阵的 Lanczos 过程推广到非对称情形.

- 在 Lanczos 过程中, v_{k+1} 是通过将 Av_k 与 v_k 和 v_{k-1} 正交化后得到的.
- 在双正交化过程中, v_{k+1} 则是由 Av_k 与 w_k 和 w_{k-1} 正交化后所得到.
- 同时, w_{k+1} 由 $A^{\mathsf{T}}w_k$ 与 v_k 和 v_{k-1} 正交化后得到.

算法 双正交化过程

给定两个非零向量 r_0, $\tilde{r}_0$ 满足 $(\tilde{r}_0, r_0) \neq 0$

计算 $\beta = \|r_0\|_2$, 令 $\beta_0 = 0$, $\gamma_0 = 0$

$v_1 = r_0/\beta$, $w_1 = \beta\tilde{r}_0/(\tilde{r}_0, r_0)$

% make sure that $w_1^{\mathsf{T}} v_1 = 1$

for $k = 1, 2, \ldots,$ **do**

  $\hat{v}_{k+1} = Av_k - \beta_{k-1} v_{k-1}$

  $\hat{w}_{k+1} = A^{\mathsf{T}} w_k - \gamma_{k-1} w_{k-1}$

  $\alpha_k = (v_k, \hat{w}_{k+1})$ % $\alpha_k = w_k^{\mathsf{T}} A v_k$

  $\tilde{v}_{k+1} = \hat{v}_{k+1} - \alpha_k v_k$

  $\tilde{w}_{k+1} = \hat{w}_{k+1} - \alpha_k w_k$

  $\gamma_k = \sqrt{|(\tilde{v}_{k+1}, \tilde{w}_{k+1})|}$

  if $\gamma_k = 0$, **then** break, **end if** % ghost break down

  $\beta_k = (\tilde{v}_{k+1}, \tilde{w}_{k+1})/\gamma_k$

  % so that $\beta_k \gamma_k = (\tilde{v}_{k+1}, \tilde{w}_{k+1})$

  $v_{k+1} = \tilde{v}_{k+1}/\gamma_k$

  $w_{k+1} = \tilde{w}_{k+1}/\beta_k$

end for

双正交性

由双正交化过程可知

$$\gamma_k v_{k+1} = Av_k - \alpha_k v_k - \beta_{k-1} v_{k-1}, \tag{2.39}$$

$$\beta_k w_{k+1} = A^{\mathsf{T}} w_k - \alpha_k w_k - \gamma_{k-1} w_{k-1}, \quad k = 1, 2, \ldots, m. \tag{2.40}$$

引理 设双正交化算法前 $m-1$ 步不中断, 则向量组 $\{v_k\}_{k=1}^m$ 和 $\{w_k\}_{k=1}^m$ 双正交, 即

$$W_m^{\mathsf{T}} V_m = I, \tag{2.41}$$

其中 $V_m = [v_1, v_2, \ldots, v_m]$, $W_m = [w_1, w_2, \ldots, w_m]$.

(板书)

☞ 注: β_k 和 γ_k 可自由选取, 只需满足 $\beta_k \gamma_k = (\tilde{v}_{k+1}, \tilde{w}_{k+1})$, 以确保 $(v_{k+1}, w_{k+1}) = 1$.

双正交化过程的性质

将耦合三项递推过程 (2.39) 和 (2.40) 写成矩阵形式即为

$$\begin{aligned} A\,V_k &= V_{k+1}\,T_{k+1,k} = V_k T_k + \gamma_k v_{k+1} e_k^{\mathsf{T}}, \\ A^{\mathsf{T}}\,W_k &= W_{k+1}\,T_{k+1,k}^{\mathsf{T}} = W_k T_k^{\mathsf{T}} + \beta_k w_{k+1} e_k^{\mathsf{T}}, \end{aligned} \qquad k = 1, 2, \ldots, m, \tag{2.42}$$

其中 $e_k = [0, \ldots, 0, 1]^{\mathsf{T}} \in \mathbb{R}^k$,

$$T_{k+1,k} = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \gamma_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \gamma_{k-1} & \alpha_k \\ & & & \gamma_k \end{bmatrix}_{(k+1)\times k}.$$

这里 $T_k \in \mathbb{R}^{k\times k}$ 是由 $T_{k+1,k}$ 的前 k 行组成的矩阵.

由 (2.41) 和 (2.42) 可知

$$W_k^{\mathsf{T}} A V_k = W_k^{\mathsf{T}} V_k T_k + \gamma_k (W_k^{\mathsf{T}} v_{k+1}) e_k^{\mathsf{T}} = T_k, \quad k = 1, 2, \ldots, m. \tag{2.43}$$

注意, T_k 是三对角矩阵, 但不是对称的. 与 Lanczos 过程类似, 我们有下面的结论.

定理 设算法前 $m-1$ 步不中断, 则向量组 $\{v_k\}_{k=1}^m$ 和 $\{w_k\}_{k=1}^m$ 分别是子空间 $\mathcal{K}_m(A, r_0)$ 和 $\mathcal{K}_m(A^{\mathsf{T}}, \tilde{r}_0)$ 的一组基, 且

$$\begin{aligned} A V_m &= V_m T_m + \gamma_m v_{m+1} e_m^{\mathsf{T}}, \\ A^{\mathsf{T}} W_m &= W_m T_m^{\mathsf{T}} + \beta_m w_{m+1} e_m^{\mathsf{T}}, \\ W_m^{\mathsf{T}} A V_m &= T_m. \end{aligned}$$

✍ 需要指出的是, 在通常情况下, $\{v_k\}_{k=1}^m$ 或 $\{w_k\}_{k=1}^m$ 本身并不相互正交.

注记

双正交化算法给出了构造 Krylov 子空间的一组基的另一种方法.

- 优点是可以利用三项递推公式, 从而能有效地节省运算量.
- 缺点是构造出来的基是非正交的, 而且方法更容易中断.
  一个常用的补救措施就是所谓的 **look-ahead** 技术, 此时不要求 $W_m^{\mathsf{T}} V_m = I$, 而只要求 $W_m^{\mathsf{T}} V_m$ 是块对角矩阵, 从而尽可能地避免出现中断.

2-6-2 BiCG 方法

BiCG, Bi-Conjugate Gradient

选取 $\mathcal{K}(A^{\mathsf{T}}, \tilde{r}_0)$ 作为约束空间

$$\text{find} \quad \tilde{x} \in x^{(0)} + \mathcal{K}(A, r_0) \quad \text{such that} \quad b - A\tilde{x} \perp \mathcal{K}(A^{\mathsf{T}}, \tilde{r}_0).$$

设 $x^{(k)} = x^{(0)} + V_k y_k$ 是 BiCG 方法在 $x^{(0)} + \mathcal{K}_k(A, r_0)$ 中找到的近似解, 则

$$0 = W_k^{\mathsf{T}}(b - Ax^{(k)}) = W_k^{\mathsf{T}}(r_0 - A\,V_k y_k) = W_k^{\mathsf{T}} r_0 - W_k^{\mathsf{T}} A\,V_k y_k = \beta\, W_k^{\mathsf{T}} v_1 - T_k y_k.$$

因此, y_k 是下列三对角线性方程组的解 $T_k y = \beta\, W_k^{\mathsf{T}} v_1 = \beta e_1$ → 采用 LU 分解求解

算法 BiCG 方法

1: **给定初值** $x^{(0)}$ **和 (相对) 精度要求** $\varepsilon > 0$
2: **计算** $r_0 = b - Ax^{(0)}$ **和** $\beta = \|r_0\|_2$
3: **选取向量** $\tilde{r}_0$, **满足** $(r_0, \tilde{r}_0) \neq 0$ % **常用的选择是** $\tilde{r}_0 = r_0$
4: **令** $p_1 = r_0$, $\tilde{p}_1 = \tilde{r}_0$
5: **for** $k = 1, 2, \ldots$ **do**
6: $\alpha_k = (r_{k-1}, \tilde{r}_{k-1})/(Ap_k, \tilde{p}_k)$
7: $x^{(k)} = x^{(k-1)} + \alpha_k p_k$
8: $r_k = r_{k-1} - \alpha_k A p_k$
9: **if** $\|r_k\|_2/\beta < \varepsilon$ **then** % check convergence
10: stop
11: **end if**
12: $\tilde{r}_k = \tilde{r}_{k-1} - \alpha_k A^{\mathsf{T}} \tilde{p}_k$
13: $\beta_k = (r_k, \tilde{r}_k)/(r_{k-1}, \tilde{r}_{k-1})$
14: $p_{k+1} = r_k + \beta_k p_k$
15: $\tilde{p}_{k+1} = \tilde{r}_k + \beta_k \tilde{p}_k$
16: **end for**

注记

- 如果 A 对称正定, 且 $\tilde{r}_0 = r_0$, 则 BiCG 方法就是 CG 方法.
- BiCG 可同时求解 $Ax = b$ 和 $A^{\mathsf{T}}\tilde{x} = \tilde{b}$ (添加一条语言: $\tilde{x}^{(k)} = \tilde{x}^{(k-1)} + \alpha_k \tilde{p}_k$)
- 需要指出的是, 如果 T_k 不存在 LU 分解, 则 BiCG 方法就失效.
- 如果在方法收敛之前, 存在某个 k 使得 T_k 是奇异的, 则方法同样会失败.
- BiCG 是不稳定的, 同时, 由于不具备最优性质, 收敛曲线也不规则.

下面是 BiCG 方法的一个重要性质.

定理 设 r_k, $\tilde{r}_k$, p_k 和 $\tilde{p}_k$ 是由 BiCG 方法所产生的, 则当 $i \neq j$ 时有

$$(r_i, \tilde{r}_j) = 0 \quad \text{和} \quad (Ap_i, \tilde{p}_j) = 0.$$

(留作练习)

2-6-3 QMR 方法

QMR, Quasi-Minimal Residual

借鉴 GMRES 的思想, 将求解线性方程组 $T_k y = \beta e_1$ 改为求解最小二乘问题:

$$\min_{x \in x^{(0)} + \mathcal{K}_k(A, r_0)} \|b - Ax\|_2. \tag{2.44}$$

这样就能保证残量的范数是递减的.

设 $x = x^{(0)} + V_k y$, 则

$$b - Ax = r_0 - A V_k y = \beta v_1 - V_{k+1} T_{k+1,k} y = V_{k+1}(\beta e_1 - T_{k+1,k} y).$$

于是 (2.44) 就转化为 $\min\limits_{y \in \mathbb{R}^k} \| V_{k+1}(\beta e_1 - T_{k+1,k} y)\|_2$

与 GMRES 方法不同的是, 这里的 V_{k+1} 不是列正交的. 所以求解比较困难.

$$\| V_{k+1}(\beta e_1 - T_{k+1,k}y)\|_2 \leq \| V_{k+1}\|_2 \cdot \|\beta e_1 - T_{k+1,k}y\|_2,$$

因此我们可以通过极小化 $\|\beta e_1 - T_{k+1,k}y\|_2$ 来给出近似解. 这就是 **QMR 方法**

$$\text{find} \quad \tilde{x} \in x^{(0)} + \mathcal{K}(A, r_0) \quad \text{such that} \quad \|\beta e_1 - T_{k+1,k}\tilde{y}\|_2 = \min, \tag{2.45}$$

注记

- 需要注意的是, 在 QMR 方法中, 极小化的是**拟残量范数**.
- QMR 与 GMRES 非常类似, 只需将 Arnoldi 过程改为双正交化过程即可. 同时, 由于 $T_{k+1,k}$ 是三对角的, 因此最小二乘问题更容易求解.

2-7 | 免转置迭代方法

2.7 免转置迭代方法

https://math.ecnu.edu.cn/~jypan/Teaching/IMP

在 BiCG 和 QMR 方法中, 都需要利用 A^{T}. 但在许多实际问题中, A^{T} 可能无法获得, 或者计算 A^{T} 与向量的乘积比较困难. 因此我们希望在计算过程中避免使用 A^{T}. 这就是 **免转置方法** (Transpose-Free Methods).

2-7-1 CGS 方法

我们注意到, 当 BiCG 方法迭代 k 步后, 残量可表示为

$$r_k = \varphi_k(A)r_0, \tag{2.46}$$

其中 $\varphi_k \in \mathbb{P}_k$, 且 $\varphi_k(0) = 1$, 称为 **残量多项式**.

- 同时, 代表搜索方向的向量 p_{k+1} 可表示为

$$p_{k+1} = \psi_k(A)r_0, \quad \psi_k \in \mathbb{P}_k. \tag{2.47}$$

- 我们还注意到, 除了 A 与 A^{T} 的区别外, $\tilde{r}_k$ 的更新公式与 r_k 的更新公式是一样的, 而 $\tilde{p}_{k+1}$ 的更新公式则与 p_{k+1} 的更新公式是一样的, 即

$$\tilde{r}_k = \varphi_k(A^{\mathsf{T}})\tilde{r}_0, \qquad \tilde{p}_{k+1} = \psi_k(A^{\mathsf{T}})\tilde{r}_0, \quad k = 1, 2, \ldots. \tag{2.48}$$

- BiCG 中 α_k 和 β_k 的计算公式可改写为

$$\alpha_k = \frac{(r_{k-1}, \tilde{r}_{k-1})}{(Ap_k, \tilde{p}_k)} = \frac{(\varphi_{k-1}(A)r_0, \varphi_{k-1}(A^{\mathsf{T}})\tilde{r}_0)}{(A\psi_{k-1}(A)r_0, \psi_{k-1}(A^{\mathsf{T}})\tilde{r}_0)} = \frac{(\varphi_{k-1}^2(A)r_0, \tilde{r}_0)}{(A\psi_{k-1}^2(A)r_0, \tilde{r}_0)},$$

$$\beta_k = \frac{(r_k, \tilde{r}_k)}{(r_{k-1}, \tilde{r}_{k-1})} = \frac{(\varphi_k(A)r_0, \varphi_k(A^{\mathsf{T}})\tilde{r}_0)}{(\varphi_{k-1}(A)r_0, \varphi_{k-1}(A^{\mathsf{T}})\tilde{r}_0)} = \frac{(\varphi_k^2(A)r_0, \tilde{r}_0)}{(\varphi_{k-1}^2(A)r_0, \tilde{r}_0)}.$$

- 因此, 计算 α_k 和 β_k 时不再需要 A^{T}.
  但需要给出 $\varphi_k^2(A)r_0$ 和 $\psi_k^2(A)r_0$ 的 (递推) 计算方法.

$\varphi_k^2(A)r_0$ 和 $\psi_k^2(A)r_0$ 的递推算法

- 由 $r_k = r_{k-1} - \alpha_k A p_k$ 和 $p_{k+1} = r_k + \beta_k p_k$ 可得

$$\varphi_k(A) = \varphi_{k-1}(A) - \alpha_k A\psi_{k-1}(A), \tag{2.49}$$

$$\psi_k(A) = \varphi_k(A) + \beta_k\psi_{k-1}(A), \quad k = 1, 2, \ldots, \tag{2.50}$$

其中 $\varphi_0(A) = I$, $\psi_0(A) = I$.

- 两边平方可得

$$\varphi_k^2(A) = \varphi_{k-1}^2(A) + \alpha_k^2 A^2\psi_{k-1}^2(A) - 2\alpha_k A\varphi_{k-1}(A)\psi_{k-1}(A),$$

$$\psi_k^2(A) = \varphi_k^2(A) + \beta_k^2\psi_{k-1}^2(A) + 2\beta_k\varphi_k(A)\psi_{k-1}(A).$$

对于交叉项 $\varphi_{k-1}(A)\psi_{k-1}(A)$ 和 $\varphi_k(A)\psi_{k-1}(A)$, 我们有

$$\begin{aligned}\varphi_{k-1}(A)\psi_{k-1}(A) &= \varphi_{k-1}(A)(\varphi_{k-1}(A)+\beta_{k-1}\psi_{k-2}(A))\\ &= \varphi_{k-1}^2(A)+\beta_{k-1}\varphi_{k-1}(A)\psi_{k-2}(A), \quad k=2,3,\ldots\end{aligned}$$

$$\begin{aligned}\varphi_k(A)\psi_{k-1}(A) &= (\varphi_{k-1}(A)-\alpha_k A\psi_{k-1}(A))\psi_{k-1}(A)\\ &= \varphi_{k-1}(A)\psi_{k-1}(A)-\alpha_k A\psi_{k-1}^2(A)\\ &= \varphi_{k-1}^2(A)+\beta_{k-1}\varphi_{k-1}(A)\psi_{k-2}(A)-\alpha_k A\psi_{k-1}^2(A), \quad k=2,3,\ldots.\end{aligned}$$

所以, 我们有下面的递推关系式 (为了书写方便, 我们省去变量 A)

$$\begin{aligned} \varphi_1^2 &= (\varphi_0 - \alpha_1 A\psi_0)^2 = (I - \alpha_1 A)^2, \\ \varphi_1\psi_0 &= \varphi_1 = I - \alpha_1 A, \\ \psi_1^2 &= (\varphi_1 + \beta_1\psi_0)^2 = (\varphi_1 + \beta_1 I)^2, \end{aligned} \tag{2.51}$$

和

$$\begin{aligned} \varphi_k^2 &= \varphi_{k-1}^2 + \alpha_k^2 A^2\psi_{k-1}^2 - 2\alpha_k A(\varphi_{k-1}^2 + \beta_{k-1}\varphi_{k-1}\psi_{k-2}), \\ \varphi_k\psi_{k-1} &= \varphi_{k-1}^2 + \beta_{k-1}\varphi_{k-1}\psi_{k-2} - \alpha_k A\psi_{k-1}^2, \\ \psi_k^2 &= \varphi_k^2 + \beta_k^2\psi_{k-1}^2 + 2\beta_k\varphi_k\psi_{k-1}, \\ & \qquad\qquad k = 2, 3, \ldots. \end{aligned} \tag{2.52}$$

记 $\hat{r}_k = \varphi_k^2(A)r_0$, $\hat{p}_{k+1} = \psi_k^2(A)r_0$, $\hat{q}_k = \varphi_k(A)\psi_{k-1}(A)r_0$, 则

$$\begin{cases} \hat{r}_k = \hat{r}_{k-1} + \alpha_k^2 A^2 \hat{p}_k - 2\alpha_k A(\hat{r}_{k-1} + \beta_{k-1}\hat{q}_{k-1}) \\ \quad = \hat{r}_{k-1} + \alpha_k A(\alpha_k A\hat{p}_k - 2\hat{r}_{k-1} - 2\beta_{k-1}\hat{q}_{k-1}), \\ \hat{q}_k = \hat{r}_{k-1} + \beta_{k-1}\hat{q}_{k-1} - \alpha_k A\hat{p}_k, \\ \hat{p}_{k+1} = \hat{r}_k + \beta_k^2 \hat{p}_k + 2\beta_k \hat{q}_k, \end{cases} \quad k = 1, 2, \ldots,$$

其中 $\hat{r}_0 = r_0$, $\hat{p}_1 = r_0$, $\hat{q}_0 = 0$, $\beta_0 = 0$,

$$\alpha_k = \frac{(\varphi_{k-1}^2(A)r_0, \tilde{r}_0)}{(A\psi_{k-1}^2(A)r_0, \tilde{r}_0)} = \frac{(\hat{r}_{k-1}, \tilde{r}_0)}{(A\hat{p}_k, \tilde{r}_0)},$$

$$\beta_k = \frac{(\varphi_k^2(A)r_0, \tilde{r}_0)}{(\varphi_{k-1}^2(A)r_0, \tilde{r}_0)} = \frac{(\hat{r}_k, \tilde{r}_0)}{(\hat{r}_{k-1}, \tilde{r}_0)}. \quad k = 1, 2, \ldots.$$

为了简化计算, 我们引入辅助变量 $u_k \triangleq \hat{r}_k + \beta_k \hat{q}_k$. 于是迭代公式变为

$$\begin{cases} \hat{q}_k = u_{k-1} - \alpha_k A\hat{p}_k, \\ \hat{r}_k = \hat{r}_{k-1} + \alpha_k A(\alpha_k A\hat{p}_k - 2u_{k-1}) \\ \quad = \hat{r}_{k-1} - \alpha_k A(\hat{q}_k + u_{k-1}), \\ \hat{p}_{k+1} = u_k + \beta_k^2 \hat{p}_k + \beta_k \hat{q}_k. \end{cases} \quad k = 1, 2, \ldots. \tag{2.53}$$

CGS 算法

由于 $\hat{r}_k = \varphi_k^2(A)r_0$, 如果 BiCG 方法收敛, 即 $\|r_k\| = \|\varphi_k(A)r_0\|$ 趋向于 0 (对任意初始向量都成立), 则 $\|\hat{r}_k\| = \|\varphi_k^2(A)r_0\|$ 也趋向于 0, 而且速度可能会更快.

- 因此, 我们将 $\hat{r}_k$ 对应的向量 $x^{(k)}$ 作为近似解, 即 $x^{(k)}$ 满足

$$\hat{r}_k = b - Ax^{(k)}.$$

- 由 (2.53) 可以直接得到 $x^{(k)}$ 的递推公式

$$x^{(k)} = x^{(k-1)} + \alpha_k(\hat{q}_k + u_{k-1}), \quad k = 1, 2, \ldots.$$

- 将上面的迭代公式整理后即得 **CGS 方法** (Conjugate Gradient Squared)

算法 CGS 方法

给定初值 $x^{(0)}$ **和 (相对) 精度要求** $\varepsilon > 0$
计算 $r_0 = b - Ax^{(0)}$, **令** $\hat{p}_1 = r_0$, $\hat{r}_0 = r_0$, $u_0 = r_0$
选取 $\tilde{r}_0$ **满足** $(r_0, \tilde{r}_0) \neq 0$ % one common choice is $\tilde{r}_0 = r_0$
for $k = 1, 2, \ldots$ **do**
    $\alpha_k = (\hat{r}_{k-1}, \tilde{r}_0)/(A\hat{p}_k, \tilde{r}_0)$
    $\hat{q}_k = u_{k-1} - \alpha_k A\hat{p}_k$
    $x^{(k)} = x^{(k-1)} + \alpha_k(\hat{q}_k + u_{k-1})$
    $\hat{r}_k = \hat{r}_{k-1} - \alpha_k A(\hat{q}_k + u_{k-1})$
    if $\|\hat{r}_k\|_2 < \varepsilon$ **then** % check convergence
        stop
    end if
    $\beta_k = (\hat{r}_k, \tilde{r}_0)/(\hat{r}_{k-1}, \tilde{r}_0)$
    $u_k = \hat{r}_k + \beta_k \hat{q}_k$
    $\hat{p}_{k+1} = u_k + \beta_k^2 \hat{p}_k + \beta_k \hat{q}_k$
end for

注记

- 在 CGS 方法中, 需要计算两次矩阵向量乘积, 因此每个迭代步的运算量与 BiCG 方法大致相同, 但这里不再需要 A^{T}.
- 由于 CGS 方法的残量多项式是 BiCG 方法残量多项式的平方, 因此 CGS 方法的收敛速度大约是 BiCG 的两倍. 但我们知道, 即使 BiCG 方法收敛, 其残量的范数也不一定是递减的. 因此, 如果在某个迭代步, BiCG 方法的残量范数变大, 则 CGS 方法的残量范数也可能会变大, 而且增幅会更大. 因此, CGS 方法的收敛曲线可能会出现非常剧烈的震荡. 这很可能会导致方法的不稳定.

2-7-2 BiCGSTAB 方法

为了克服 CGS 方法可能会出现的剧烈震荡, van der Vorst 提出了 **BiCGSTAB** 方法 (Bi-Conjugate Gradient Stabilized). 其基本想法是将残量表示为

$$r_k = \phi_k(A)\varphi_k(A)r_0, \quad k = 1, 2, \ldots,$$

其中 φ_k 是 BiCG 方法的残量多项式, 而 ϕ_k 则是一个新的 k 次多项式, 用来修正残量范数的震荡现象.

如果取 $\phi_k = \varphi_k$, 则 BiCGSTAB 就是 CGS.

ϕ_k 的选取

在 BiCGSTAB 方法中, 多项式 ϕ_k 是通过下面的递推方法来定义的

$$\phi_0(A) = I, \quad \phi_k(A) = (I - \omega_k A)\phi_{k-1}(A), \quad k = 1, 2, \ldots,$$

其中 ω_k 是待定系数, 用来极小化残量范数.

- 于是, 残量和搜索方向可定义为

$$r_k = \phi_k(A)\varphi_k(A)r_0, \quad p_{k+1} = \phi_k(A)\psi_k(A)r_0, \quad k = 1, 2, \ldots,$$

r_k 和 p_k 的递推公式

由 $\phi_k(A)$ 的递推公式和

$$\begin{aligned}\varphi_k(A) &= \varphi_{k-1}(A) - \alpha_k A\psi_{k-1}(A),\\ \psi_k(A) &= \varphi_k(A) + \beta_k\psi_{k-1}(A), \quad k = 1, 2, \ldots,\end{aligned}$$

可知

$$\begin{aligned}\phi_k(A)\varphi_k(A) &= (I - \omega_k A)\Big(\phi_{k-1}(A)\varphi_{k-1}(A) - \alpha_k A\phi_{k-1}(A)\psi_{k-1}(A)\Big)\\ \phi_k(A)\psi_k(A) &= \phi_k(A)\varphi_k(A) + \beta_k(I - \omega_k A)\phi_{k-1}(A)\psi_{k-1}(A).\end{aligned}$$

所以

$$\begin{aligned}r_k = \phi_k(A)\varphi_k(A)r_0 &= (I - \omega_k A)(\phi_{k-1}(A)\varphi_{k-1}(A)r_0 - \alpha_k A\phi_{k-1}(A)\psi_{k-1}(A)r_0)\\ &= (I - \omega_k A)(r_{k-1} - \alpha_k A p_k),\\ p_{k+1} = \phi_k(A)\psi_k(A)r_0 &= \phi_k(A)\varphi_k(A)r_0 + \beta_k(I - \omega_k A)\phi_{k-1}(A)\psi_{k-1}(A)r_0\\ &= r_k + \beta_k(I - \omega_k A)p_k.\end{aligned}$$

参数 α_k 和 β_k 仍然定义为

$$\alpha_k = \frac{(r_{k-1}, \tilde{r}_{k-1})}{(Ap_k, \tilde{p}_k)} = \frac{(\varphi_{k-1}(A)r_0, \varphi_{k-1}(A^{\mathsf{T}})\tilde{r}_0)}{(A\psi_{k-1}(A)r_0, \psi_{k-1}(A^{\mathsf{T}})\tilde{r}_0)} = \frac{(\varphi_{k-1}^2(A)r_0, \tilde{r}_0)}{(A\psi_{k-1}^2(A)r_0, \tilde{r}_0)}$$

$$\beta_k = \frac{(r_k, \tilde{r}_k)}{(r_{k-1}, \tilde{r}_{k-1})} = \frac{(\varphi_k(A)r_0, \varphi_k(A^{\mathsf{T}})\tilde{r}_0)}{(\varphi_{k-1}(A)r_0, \varphi_{k-1}(A^{\mathsf{T}})\tilde{r}_0)} = \frac{(\varphi_k^2(A)r_0, \tilde{r}_0)}{(\varphi_{k-1}^2(A)r_0, \tilde{r}_0)}.$$

由于我们不再计算 $\varphi_k^2(A)$ 或 $\psi_k^2(A)$, 因此, 需要寻找 **其它方式** 计算 α_k 和 β_k.

又 $\varphi_i(A)r_0$ 和 $\varphi_i(A^{\mathsf{T}})\tilde{r}_0$ 是 BiCG 的残量, 由 BiCG 性质可知

$$(\varphi_i(A)r_0, \varphi_j(A^{\mathsf{T}})\tilde{r}_0) = 0, \quad \text{for } i > j,$$

即

$$(\varphi_i(A)r_0, \left(A^{\mathsf{T}}\right)^j \tilde{r}_0) = 0, \quad \text{for } i > j.$$

因此, 在计算 $(\varphi_{k-1}(A)r_0, \varphi_{k-1}(A^{\mathsf{T}})\tilde{r}_0)$ 时, 只需考虑 $\varphi_{k-1}(A^{\mathsf{T}})$ 的最高次项.

由
$$\begin{aligned}&\varphi_k(A)=\varphi_{k-1}(A)-\alpha_k A\psi_{k-1}(A),\\&\psi_k(A)=\varphi_k(A)+\beta_k\psi_{k-1}(A),\quad k=1,2,\ldots,\end{aligned}$$
可知

$$\begin{aligned}\varphi_k(A^{\mathsf{T}})&=\varphi_{k-1}(A^{\mathsf{T}})-\alpha_k A^{\mathsf{T}}\psi_{k-1}(A^{\mathsf{T}})\\&=\varphi_{k-1}(A^{\mathsf{T}})-\alpha_k A^{\mathsf{T}}(\varphi_{k-1}(A^{\mathsf{T}})+\beta_k\psi_{k-2}(A^{\mathsf{T}}))\\&=-\alpha_k A^{\mathsf{T}}\varphi_{k-1}(A^{\mathsf{T}})+\varphi_{k-1}(A^{\mathsf{T}})-\alpha_k\beta_k A^{\mathsf{T}}\psi_{k-2}(A^{\mathsf{T}}).\end{aligned}$$

因此, $\varphi_k(A^{\mathsf{T}})$ 的最高次项的系数与 $-\alpha_k A^{\mathsf{T}}\varphi_{k-1}(A^{\mathsf{T}})$ 的最高次项的系数是相等的.

由于 $\varphi_0(A^{\mathsf{T}})=I$, 所以 $-\alpha_k A^{\mathsf{T}}\varphi_{k-1}(A^{\mathsf{T}})$ 的最高次项系数为 $(-1)^k\alpha_k\alpha_{k-1}\cdots\alpha_1$

所以

$$\Big(\varphi_{k-1}(A)r_0,\varphi_{k-1}(A^{\mathsf{T}})\tilde{r}_0\Big)=(-1)^{k-1}\alpha_{k-1}\cdots\alpha_1\Big(\varphi_{k-1}(A)r_0,(A^{\mathsf{T}})^{k-1}\tilde{r}_0\Big)$$

另一方面, 由 $\phi_k(A)$ 的递推公式可知 $\phi_{k-1}(A^{\mathsf{T}})$ 的最高次项是

$$(-1)^{k-1}\omega_{k-1}\omega_{k-2}\cdots\omega_1(A^{\mathsf{T}})^{k-1}.$$

所以

$$\Big(\varphi_{k-1}(A)r_0,\phi_{k-1}(A^{\mathsf{T}})\tilde{r}_0\Big)=(-1)^{k-1}\omega_{k-1}\omega_{k-2}\cdots\omega_1\Big(\varphi_{k-1}(A)r_0,(A^{\mathsf{T}})^{k-1}\tilde{r}_0\Big).$$

因此, 我们可以得到下面的等式

$$\frac{\Big(\varphi_{k-1}(A)r_0,\varphi_{k-1}(A^{\mathsf{T}})\tilde{r}_0\Big)}{\Big(\varphi_{k-1}(A)r_0,\phi_{k-1}(A^{\mathsf{T}})\tilde{r}_0\Big)}=\frac{\alpha_{k-1}\alpha_{k-2}\cdots\alpha_1}{\omega_{k-1}\omega_{k-2}\cdots\omega_1},\quad k=1,2,\ldots,$$

也就是说, 我们可以通过 $\Big(\varphi_k(A)r_0,\phi_k(A^{\mathsf{T}})\tilde{r}_0\Big)$ 来计算 $\Big(\varphi_k(A)r_0,\varphi_k(A^{\mathsf{T}})\tilde{r}_0\Big)$.

下面考虑如何计算 $\left(A\psi_{k-1}(A)r_0, \psi_{k-1}(A^{\mathsf{T}})\tilde{r}_0\right)$.

- 由 $\psi_k(A) = \varphi_k(A) + \beta_k\psi_{k-1}(A)$ 可知, $\psi_k(A^{\mathsf{T}})$ 的最高次项与 $\varphi_k(A^{\mathsf{T}})$ 相同.
- 根据 BiCG 的性质, 我们有

$$\left(A\psi_i(A)r_0, \psi_j(A^{\mathsf{T}})\tilde{r}_0\right) = 0, \quad \text{for } i > j,$$

即

$$\left(A\psi_i(A)r_0, (A^{\mathsf{T}})^j\tilde{r}_0\right) = 0, \quad \text{for } i > j.$$

- 所以计算 $\left(A\psi_{k-1}(A)r_0, \psi_{k-1}(A^{\mathsf{T}})\tilde{r}_0\right)$ 时只需考虑 $\psi_{k-1}(A^{\mathsf{T}})$ 的最高次项, 即

$$\left(A\psi_{k-1}(A)r_0, \psi_{k-1}(A^{\mathsf{T}})\tilde{r}_0\right) = (-1)^{k-1}\alpha_{k-1}\alpha_{k-2}\cdots\alpha_1\left(A\psi_{k-1}(A)r_0, (A^{\mathsf{T}})^{k-1}\tilde{r}_0\right)$$

相类似地, 我们有

$$\Big(A\psi_{k-1}(A)r_0, \phi_{k-1}(A^{\mathsf{T}})\tilde{r}_0\Big) = (-1)^{k-1}\omega_{k-1}\omega_{k-2}\cdots\omega_1\Big(A\psi_{k-1}(A)r_0, (A^{\mathsf{T}})^{k-1}\tilde{r}_0\Big)$$

因此

$$\frac{(A\psi_{k-1}(A)r_0, \psi_{k-1}(A^{\mathsf{T}})\tilde{r}_0)}{(A\psi_{k-1}(A)r_0, \phi_{k-1}(A^{\mathsf{T}})\tilde{r}_0)} = \frac{\alpha_{k-1}\alpha_{k-2}\cdots\alpha_1}{\omega_{k-1}\omega_{k-2}\cdots\omega_1}, \quad k = 1, 2, \ldots.$$

所以

$$\begin{aligned}\alpha_k = \frac{(\varphi_{k-1}(A)r_0, \varphi_{k-1}(A^{\mathsf{T}})\tilde{r}_0)}{(A\psi_{k-1}(A)r_0, \psi_{k-1}(A^{\mathsf{T}})\tilde{r}_0)} &= \frac{(\varphi_{k-1}(A)r_0, \phi_{k-1}(A^{\mathsf{T}})\tilde{r}_0)}{(A\psi_{k-1}(A)r_0, \phi_{k-1}(A^{\mathsf{T}})\tilde{r}_0)} \\ &= \frac{(\phi_{k-1}(A)\varphi_{k-1}(A)r_0, \tilde{r}_0)}{(A\phi_{k-1}(A)\psi_{k-1}(A)r_0, \tilde{r}_0)} = \frac{(r_{k-1}, \tilde{r}_0)}{(Ap_k, \tilde{r}_0)},\end{aligned}$$

$$\begin{aligned}\beta_k = \frac{(\varphi_k(A)r_0, \varphi_k(A^{\mathsf{T}})\tilde{r}_0)}{(\varphi_{k-1}(A)r_0, \varphi_{k-1}(A^{\mathsf{T}})\tilde{r}_0)} &= \frac{\alpha_k}{\omega_k} \cdot \frac{(\varphi_k(A)r_0, \phi_k(A^{\mathsf{T}})\tilde{r}_0)}{(\varphi_{k-1}(A)r_0, \phi_{k-1}(A^{\mathsf{T}})\tilde{r}_0)} \\ &= \frac{\alpha_k}{\omega_k} \cdot \frac{(\phi_k(A)\varphi_k(A)r_0, \tilde{r}_0)}{(\phi_{k-1}(A)\varphi_{k-1}(A)r_0, \tilde{r}_0)} = \frac{\alpha_k}{\omega_k} \cdot \frac{(r_k, \tilde{r}_0)}{(r_{k-1}, \tilde{r}_0)}.\end{aligned}$$

整理后可得

$$\begin{cases} r_k = (I - \omega_k A)(r_{k-1} - \alpha_k A p_k) \quad \text{with } \alpha_k = \dfrac{(r_{k-1}, \tilde{r}_0)}{(Ap_k, \tilde{r}_0)}, \\ p_{k+1} = r_k + \beta_k (I - \omega_k A) p_k \quad \text{with } \beta_k = \dfrac{\alpha_k}{\omega_k} \cdot \dfrac{(r_k, \tilde{r}_0)}{(r_{k-1}, \tilde{r}_0)}, \end{cases} \qquad k = 1, 2, \ldots,$$

其中 $p_1 = r_0$.

近似解 $x^{(k)}$ 的更新公式

由表达式 $r_k = (I - \omega_k A)(r_{k-1} - \alpha_k A p_k)$ 可知

$$\begin{aligned} b - Ax^{(k)} = r_k &= (I - \omega_k A)(r_{k-1} - \alpha_k A p_k) \\ &= r_{k-1} - \alpha_k A p_k - \omega_k A(r_{k-1} - \alpha_k A p_k) \\ &= b - Ax^{(k-1)} - \alpha_k A p_k - \omega_k A(r_{k-1} - \alpha_k A p_k), \end{aligned}$$

因此

$$x^{(k)} = x^{(k-1)} + \alpha_k p_k + \omega_k (r_{k-1} - \alpha_k A p_k)$$

参数 ω_k 的选取

前面已经提到, 参数 ω_k 的选取准则是极小化残量范数 $\|r_k\|_2$, 即

$$\omega_k = \arg\min_{\omega} \|(I - \omega A)(r_{k-1} - \alpha_k A p_k)\|_2.$$

记

$$q_k \triangleq r_{k-1} - \alpha_k A p_k.$$

则上述最小二乘问题的解为

$$\omega_k = \frac{(q_k, Aq_k)}{(Aq_k, Aq_k)}$$

算法 BiCGSTAB 算法

1: **给定初值** $x^{(0)}$ **和 (相对) 精度要求** $\varepsilon > 0$
2: **计算** $r_0 = b - Ax^{(0)}$
3: **选取** $\tilde{r}_0$ **满足** $(r_0, \tilde{r}_0) \neq 0$ % one common choice is $\tilde{r}_0 = r_0$
4: **令** $p_1 = r_0$
5: **for** $k = 1, 2, \ldots$ **do**
6: $\alpha_k = (r_{k-1}, \tilde{r}_0)/(Ap_k, \tilde{r}_0)$ % if $(r_{k-1}, \tilde{r}_0) = 0$ or $(Ap_k, \tilde{r}_0) = 0$, then breakdown
7: $q_k = r_{k-1} - \alpha_k A p_k$
8: $x^{(k)} = x^{(k-1)} + \alpha_k p_k$ % if $\|q_k\| < \varepsilon$, then converged and stop
9: $\omega_k = (q_k, Aq_k)/(Aq_k, Aq_k)$ % if $\omega_k = 0$, then breakdown
10: $x^{(k)} = x^{(k)} + \omega_k q_k$
11: $r_k = q_k - \omega_k A q_k$
12: **if** $\|r_k\|_2 < \varepsilon$, **then** stop, **end if** % check convergence
13: $\beta_k = \alpha_k/\omega_k \cdot (r_k, \tilde{r}_0)/(r_{k-1}, \tilde{r}_0)$
14: $p_{k+1} = r_k + \beta_k(p_k - \omega_k A p_k)$
15: **end for**

注记

Bi-CGSTAB 是当前常用的求解非对称线性方程组的方法之一, 但在某些情况下效率较低, 比如那种系数矩阵的特征值虚部占优时的情形, 这种方程通常来自于那些对流项占优的偏微分方程. 此时可以考虑 BiCGStab2 或 BiCGstab(ℓ).

2-8 | 正规方程迭代方法

2.8 正规方程迭代方法

https://math.ecnu.edu.cn/~jypan/Teaching/IMP

求解非对称线性方程组的另外一种选择: 转化为等价的正规方程, 然后用 CG 求解.

- **缺点:** 条件数是原问题的平方, 不适合病态问题.
- **优点:** 在某些场合, 这种方法很有吸引力, 比如 A 具有非常聚集的奇异值.

2-8-1 CGNR 方法和 CGNE 方法

CGNR: CG Normal Residual

用 CG 求解与原方程组等价的正规方程组 (法方程组): $A^{\mathsf{T}}Ax = A^{\mathsf{T}}b$

1: 给定初值 $x^{(0)}$, 计算 $r_0 = b - A^{\mathsf{T}}Ax^{(0)}$, 令 $p_1 = r_0$
2: **for** $k = 1, 2, \ldots$ until convergence **do**
3: $\quad \xi_k = (r_{k-1}, r_{k-1})/(A^{\mathsf{T}}Ap_k, p_k)$
4: $\quad x^{(k)} = x^{(k-1)} + \xi_k p_k$
5: $\quad r_k = r_{k-1} - \xi_k A^{\mathsf{T}}Ap_k$
6: $\quad \mu_k = (r_k, r_k)/(r_{k-1}, r_{k-1})$
7: $\quad p_{k+1} = r_k + \mu_k p_k$
8: **end for**

- 优点: 容易实现
- 缺点: 条件数是原来的平方, 即 $\kappa(A^{\mathsf{T}}A) = \kappa(A)$

例 考虑 $Ax = b$, 其中 $A = \begin{bmatrix} 1 & & \\ 1 & \varepsilon & \\ 1 & 0 & \varepsilon \end{bmatrix}$, 则 $A^{\mathsf{T}}A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1+\varepsilon^2 & 1 \\ 1 & 1 & 1+\varepsilon^2 \end{bmatrix}$.

若 $\varepsilon \in (\sqrt{\varepsilon_u}, \varepsilon_u)$, 其中 ε_u 表示机器精度, 则由于舍入误差, 实际计算得到 $A^{\mathsf{T}}A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$.

→ 应尽量避免直接计算 $A^{\mathsf{T}}A$.

CGNR 算法

引入变量 $z_k = A^{\mathsf{T}} r_k$ (正规方程的残量), 其中 $r_k = b - Ax^{(k)}$ (原方程组的残量)

算法 CGNR 方法

给定初值 $x^{(0)}$, 计算 $r_0 = b - A^{\mathsf{T}} A x^{(0)}$ 和 $z_0 = A^{\mathsf{T}} r_0$
令 $p_1 = z_0$
for $k = 1, 2, \ldots$ until convergence **do**
    $\xi_k = (z_{k-1}, z_{k-1})/(Ap_k, Ap_k)$
    $x^{(k)} = x^{(k-1)} + \xi_k p_k$
    $r_k = r_{k-1} - \xi_k A p_k$
    $z_k = A^{\mathsf{T}} r_k$
    $\mu_k = (z_k, z_k)/(z_{k-1}, z_{k-1})$
    $p_{k+1} = z_k + \mu_k p_k$
end for

CGNR 的最优性质

根据 CG 方法的最优性质可知, 由 CGNR 方法得到的近似解 $x^{(k)}$ 在仿射空间 $x^{(0)} + \mathcal{K}_k(A^{\mathsf{T}}A,\ A^{\mathsf{T}}r_0)$ 中极小化残量 $b - Ax$ 的 2-范数.

定理 设 $x^{(k)}$ 是由 CGNR 方法迭代 k 步后得到的近似解, 则

$$\|b - Ax^{(k)}\|_2 = \min_{x \in x^{(0)} + \mathcal{K}_k(A^{\mathsf{T}}A, A^{\mathsf{T}}r_0)} \|b - Ax\|_2.$$

(留作练习)

CGNE: CG Normal Error

引入辅助变量 y, 即令 $x = A^{\mathsf{T}} y$, 则原方程组等价于

$$AA^{\mathsf{T}} y = b.$$

将 CG 方法用于求解上述方程, 就得到 **CGNE 方法** 或 **Craig 方法**.
(该方法的具体推导过程与算法描述留作练习)

类似地, 我们有下面的最优性质.

定理 设 $x^{(k)}$ 是由 Craig 方法迭代 k 步后得到的近似解, 则

$$\|x^{(k)} - x_*\|_2 = \min_{x \in x^{(0)} + \mathcal{K}_k(A^{\mathsf{T}}A, A^{\mathsf{T}} r_0)} \|x - x_*\|_2,$$

其中 x_* 表示精确解. (留作练习)

注记

- 根据 CGNR 和 CGNE 的最优性质, 我们有趣地发现, CGNR 方法和 CGNE 方法都是在同一个仿射空间寻找近似解, 但找到的却是具有不同的最优性质的近似解.
- 需要指出的是, 上面两个方法也可以用于求解超定或欠的定线性方程组, 即线性最小二乘问题.

2-8-2 LSQR 方法

如果系数矩阵条件数比较大, 则不建议求解正规方程, 可以考虑使用 **LSQR 方法**, 即求解等价**鞍点问题** (或增广方程组, augmented system)

$$\mathcal{A}f \triangleq \begin{bmatrix} I & A \\ A^{\mathsf{T}} & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} \triangleq g,$$

LSQR

选取初值 $[0,0]^{\mathsf{T}}$, 用 Lanczos 算法构造 $\mathcal{K}(\mathcal{A}, g)$ 的一组标准正交基.

通过直接计算可知

$$v_1 = \frac{1}{\|b\|_2}\begin{bmatrix} b \\ 0 \end{bmatrix}, \quad v_2 = \frac{1}{\|A^{\mathsf{T}}b\|_2}\begin{bmatrix} 0 \\ A^{\mathsf{T}}b \end{bmatrix}.$$

依次类推, 我们可以发现这样一个规律: 基向量 v_{2k-1} 和 v_{2k} 分别具有下面的形式

$$v_{2k-1} = \begin{bmatrix} * \\ 0 \end{bmatrix} \quad \text{和} \quad v_{2k} = \begin{bmatrix} 0 \\ * \end{bmatrix}.$$

因此, 我们引入另外两组单位向量 $\{u_k\}$ 和 $\{w_k\}$ 来表示这组基, 即

$$v_{2k-1} = \begin{bmatrix} u_k \\ 0 \end{bmatrix}, \quad v_{2k} = \begin{bmatrix} 0 \\ w_k \end{bmatrix}, \quad k = 1, 2, \ldots.$$

记 $\beta = \|b\|_2$, 则鞍点问题的 Lanczos 过程可描述为:

$$\beta u_1 = b, \quad \alpha_1 w_1 = A^{\mathsf{T}} u_1, \quad \begin{cases} \beta_k u_{k+1} = A w_k - \alpha_k u_k, \\ \alpha_{k+1} w_{k+1} = A^{\mathsf{T}} u_{k+1} - \beta_k w_k, \end{cases} \quad k = 1, 2, \ldots,$$

其中 $\beta_k > 0$ 和 $\alpha_{k+1} > 0$ 的选取是确保 $\|u_{k+1}\|_2 = \|w_{k+1}\|_2 = 1$, 即

$$\beta_k = \|A w_k - \alpha_k u_k\|_2, \quad \alpha_{k+1} = \|A^{\mathsf{T}} u_{k+1} - \beta_k w_k\|_2.$$

需要指出的是, 这里的 α_k 和 β_k 与一般 Lanczos 过程中的 α_k 和 β_k 是不一样的.

上面的这个过程实际上就是一个 **双对角化** (bidiagonalization) 过程.

记 $U_m = [u_1, u_2, \ldots, u_m]$, $W_m = [w_1, w_2, \ldots, w_m]$, 则有

$$\begin{aligned} A\,W_m &= U_{m+1}\tilde{B}_m, \\ A^{\mathsf{T}}\,U_{m+1} &= W_m\tilde{B}_m^{\mathsf{T}} + \alpha_m w_m e_{m+1}^{\mathsf{T}}, \end{aligned} \tag{2.54}$$

其中 $e_{m+1} = [0, \ldots, 0, 1]^{\mathsf{T}} \in \mathbb{R}^{m+1}$, $\tilde{B}_m = \begin{bmatrix} \alpha_1 & & \\ \beta_1 & \ddots & \\ & \ddots & \alpha_m \\ & & \beta_m \end{bmatrix} \in \mathbb{R}^{(m+1)\times m}$,

且 $U_{m+1}^{\mathsf{T}}U_{m+1} = I_{m+1}$, $W_m^{\mathsf{T}}W_m = I_m$. 所以, 矩阵

$$\begin{bmatrix} u_1 & 0 & u_2 & 0 & \cdots & 0 & u_m \\ 0 & w_1 & 0 & w_2 & \cdots & w_{m-1} & 0 \end{bmatrix} \quad \text{或} \quad \begin{bmatrix} U_{m+1} & 0 \\ 0 & W_m \end{bmatrix}$$

的列向量组构成了 $\mathcal{K}_{2m+1}(\mathcal{A}, g_0)$ 的一组基.

LSQR 数值求解

在 $\mathcal{K}_{2m+1}(\mathcal{A}, g_0)$ 中寻找近似解, 使得原方程组的残量范数 $\|b - A\tilde{x}\|_2$ 达到最小.
设近似解为 $[\tilde{x}, \tilde{r}]^{\mathsf{T}}$, 可写成

$$\begin{bmatrix} \tilde{r} \\ \tilde{x} \end{bmatrix} = \begin{bmatrix} U_{m+1} & 0 \\ 0 & W_m \end{bmatrix} \begin{bmatrix} t_{m+1} \\ y_m \end{bmatrix} \quad \text{或者} \quad \begin{cases} \tilde{r} = U_{m+1} t_{m+1}, \\ \tilde{x} = W_m y_m. \end{cases}$$

由 (2.54) 可知

$$\begin{aligned} b - A\tilde{x} &= \beta u_1 - A W_m y_m \\ &= U_{m+1}(\beta e_1) - U_{m+1} \tilde{B}_m y_m \\ &= U_{m+1}(\beta e_1 - \tilde{B}_m y_m). \end{aligned}$$

设 T_m 是 Lanczos 算法作用到正规方程 $A^{\mathsf{T}}Ax = A^{\mathsf{T}}b$ 上得到的对称三对角矩阵, 则可以证明: $\tilde{B}_m$ 就是 T_m 的双对角化因子, 即 $T_m = \tilde{B}_m^{\mathsf{T}} \tilde{B}_m$. (证明留作练习)

由于 $U_{m+1}^{\mathsf{T}} U_{m+1} = I_{m+1}$, 因此

$$\|b - A\tilde{x}\|_2 = \|U_{m+1}(\beta e_1 - \tilde{B}_m y_m)\|_2 = \|\beta e_1 - \tilde{B}_m y_m\|_2.$$

所以, 极小化残量范数就等价于求解下面的最小二乘问题:

$$\min_{y \in \mathbb{R}^m} \|\beta e_1 - \tilde{B}_m y\|_2.$$

我们可以采用 QR 分解来求解该问题. 这就形成了完整的 LSQR 方法.

注记

相比于正规方程的 CG 方法, LSQR 方法具有更好的稳定性, 对于坏条件问题, 可以更有效地控制舍入误差的增长.


谢 谢
THANK YOU