第4章 逐次逼近法

4.4 迭代法的加速

4.4 迭代法的加速

使用迭代法的困难所在是计算量难以估计。有时迭代过程虽然收敛，但由于收敛速度缓慢，使计算量变的很大而失去使用价值。因此，迭代过程的加速具有重要意义。 迭代法加速，就是要寻找一种改进迭代法直接产生的序列的收敛速度的方法，使原来不收敛的序列变成收敛，使原来收敛较慢的序列变得收敛快。

4.4.1 基本迭代法的加速

在4.1中介绍了线性方程组求解的**Jacobi**和**Gauss-Seidel**基本迭代法，通过对基本迭代法加速可得其它迭代法。

一、超松弛法（SOR法）

Gauss-Seidel法的迭代格式为：

$$\begin{cases} x_i^{(k+1)} = \dfrac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}\right) \\ a_{ii} \neq 0, \quad i = 1, 2, \cdots, n \end{cases} \qquad \textbf{(4-52)}$$

如果将（**4-52**）的右端记为 $\overline{x}_i^{(k+1)}$，并用 $\overline{x}_i^{(k+1)}$ 和 $x_i^{(k)}$ 的线性组合作迭代加速，得到

$$(1-\omega)x_i^{(k)} + \omega\,\overline{x}_i^{(k+1)} = x_i^{(k+1)} \qquad i = 1, 2, \cdots, n \qquad \textbf{(4-53)}$$

将（4-52）的右端代入（4-53），得

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right) \quad i=1,2,\cdots,n \quad k=1,2,\cdots,$$

这就是逐次超松弛法，简称**SOR**法，

ω 称为松弛因子。**SOR**法的收敛速度与 ω 的取值有关，当 $\omega=1$ 时，它就是**Gauss-Seidel**法。因此，可选取 ω 的值使（**4-55**）的收敛速度较**Gauss-Seidel**法快，从而起到加速作用。

为了讨论 ω 的取值与收敛性的关系，特将（**4-54**）改写成矩阵形式。由（**4-54**）可得

$$a_{ii}x_i^{(k+1)} = (1-\omega)a_{ii}x_i^{(k)} + \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right)$$

设 $\boldsymbol{A}=\boldsymbol{D}-\boldsymbol{L}-\boldsymbol{U}$，其中 $\boldsymbol{D}=\mathrm{diag}\left(a_{11},a_{22},\cdots,a_{nn}\right)$，$\boldsymbol{L},\boldsymbol{U}$同（**4-7**）

则上式可写成矩阵形式

$$\boldsymbol{D}\boldsymbol{x}^{(k+1)}=(1-\omega)\boldsymbol{D}\boldsymbol{x}^{(k)}+\omega\left(\boldsymbol{b}+\boldsymbol{L}\boldsymbol{x}^{(k+1)}+\boldsymbol{U}\boldsymbol{x}^{(k)}\right)$$

整理

$$(\boldsymbol{D}-\omega\boldsymbol{L})\boldsymbol{x}^{(k+1)}=[(1-\omega)\boldsymbol{D}+\omega\boldsymbol{U}]\boldsymbol{x}^{(k)}+\omega\boldsymbol{b}$$

$$\boldsymbol{x}^{(k+1)}=(\boldsymbol{D}-\omega\boldsymbol{L})^{-1}[(1-\omega)D+\omega\boldsymbol{U}]\boldsymbol{x}^{(k)}+\omega(\boldsymbol{D}-\omega\boldsymbol{L})^{-1}\boldsymbol{b}$$

令

$$\boldsymbol{L}_{\omega}=(\boldsymbol{D}-\omega\boldsymbol{L})^{-1}[(1-\omega)\boldsymbol{D}+\omega\boldsymbol{U}] \qquad \boldsymbol{f}_{\omega}=\omega(\boldsymbol{D}-\omega\boldsymbol{L})^{-1}\boldsymbol{b}$$

则

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{L}_{\omega}\boldsymbol{x}^{(k)}+\boldsymbol{f} \qquad \text{（4-56）}$$

其中 $\boldsymbol{L}_{\omega}$ 为**SOR**法的迭代矩阵。

显然，**SOR**法收敛 $\Leftrightarrow \rho\left(\boldsymbol{L}_{\omega}\right)<1$。

①可以证明 $\rho(\boldsymbol{L}_\omega) \geq |\omega - 1|$，故若**SOR**法收敛，则

$$|\omega - 1| \leq \rho(\boldsymbol{L}_\omega) < 1$$

即 $0 < \omega < 2$ 是（4−56）收敛的必要条件。

②证明如果A是对称正定矩阵，则满足 $0 < \omega < 2$ 的 ω 和任意初始向量$\boldsymbol{x}^{(0)}$，**SOR**法均收敛。

特别地，取 $\omega = 1$，则是**Gauss-Seidel**法。 从而得到结论：

如果A为对称正定阵时，**Gauss-Seidel**迭代法必收敛。（充分条件）

使***SOR***法收敛最快的松弛因子 ω 称最优松弛因子， 一般用表示 ω_{opt} 。

但是在实际计算时，ω_{opt}是很难事先确定。一般可用试算法取近似最优值。在有些数学软件平台中有取ω_{opt}近似值的算法。

在设计SOR算法时，应注意 **Jacobi**法、**Gauss-Seidel**法 和 **SOR**法的异同点，即（4-6），（4-11）和（4-55）的异、同点，使设计的算法更具一般性。**算例2，见表格(书中第149页)**。

松弛因子ω	迭代次数k	松弛因子ω	迭代次数k
0.8	28	1.2	20
0.9	23	1.3	24
1.0	18	1.4	28
1.1	16	1.5	34

4.4.2 Aitken加速

若某迭代法产生的数列数列$\{x_k\}$为线性收敛，即

$$\lim_{k\to\infty}\frac{e_{k+1}}{e_k}=\lim_{k\to\infty}\frac{x_{k+1}-\alpha}{x_k-\alpha}=c\neq 0$$

此时, 当$\boldsymbol{k}$从分大时,

$$\frac{x_{k+1}-\alpha}{x_k-\alpha}\approx c\,,\qquad \frac{x_k-\alpha}{x_{k-1}-\alpha}\approx c$$

则有

$$\frac{x_k-\alpha}{x_{k-1}-\alpha}\approx\frac{x_{k+1}-\alpha}{x_k-\alpha}$$

$$(\qquad\qquad)^2\approx(\qquad\qquad)(\qquad\qquad)$$

$$x_k^2-2\alpha x_k\cancel{+\alpha^2}\approx x_{k+1}x_{k-1}-\alpha\left(x_{k-1}+x_{k+1}\right)\cancel{+\alpha^2}$$

$$x_k^2-x_{k+1}x_{k-1}\approx\alpha\left(2x_{k-1}-x_{k+1}-x_{k-1}\right)$$

由此推出

$$\alpha \approx \frac{x_{k-1}x_{k+1}-x_k^2}{x_{k-1}-2x_k+x_{k+1}} = x_{k+1} - \frac{\left(x_{k+1}-x_k\right)^2}{x_{k-1}-2x_k+x_{k+1}}$$

令

$$\bar{x}_k = x_{k+1} - \frac{\left(x_{k+1}-x_k\right)^2}{x_{k-1}-2x_k+x_{k+1}} \qquad \text{（4-57）}$$

由（**4-57**）对数列$\{x_k\}$进行加速的方法称**Aitken**加速。

如果原数列$\{x_k\}$为线性收敛，可以证明：

$$\lim_{k\to\infty}\frac{\bar{x}_k-\alpha}{x_k-\alpha}=0$$

故由（**4-57**）进行加速后得到的新数列$\left\{\bar{x}_k\right\}$，其收敛速度一般都比原数列快。　将这种加速方法用于具体的迭代法上，可对原迭代法进行有效的加速，有时甚至能将发散的具体迭代格式通过这种加速后变成收敛。

非线性方程迭代求根的加速

设x^*是$f(x)=0$的根。 作简单迭代法 $x_k = \varphi(x_{k-1})$

1）任取初始值x_0; **2**）取k=1,2, ···

2.1 计算

$$\begin{cases} y_k = \varphi(x_k) \\ z_k = \varphi(y_k) \end{cases} \qquad \textbf{(4-58)}$$

2.2 用（**4-57**）对（**4-58**）进行加速，得到

$$x_{k+1} = z_k - \frac{(z_k - y_k)^2}{z_k - 2y_k + x_k} \qquad \textbf{(4-59)}$$

这就是对迭代格式 $x_k = \varphi(x_{k-1})$，使用**Aitken**加速后的新迭代公式。

可合并为

$$x_{k+1} = \varphi(\varphi(x_k)) - \frac{[\varphi(\varphi(x_k)) - \varphi(x_k)]^2}{\varphi(\varphi(x_k)) - 2\varphi(x_k) + x_k} \qquad \textbf{(4-60)}$$

这种加速后的迭代法（**4-60**）称**Steffensen**迭代法。

例1　用Steffensen迭代法计算 $x=x_3-1$在$x_0=1.5$附近的近似解。

解　1）**直接使用迭代格式：**$x_{k+1}=x_k^3-1$，则

$$\varphi(x)=x^3-1,\quad \text{且有}\quad \left|\varphi'(1.5)\right|=2\times1.5^2-1=2.5>1$$

显然该迭代过程发散。

2）使用**Steffensen**法，即

$$y_k=\varphi(x_k),\quad z_k=\varphi(y_k),\qquad x_{k+1}=z_k-\frac{(z_k-y_k)^2}{z_k-2y_k+x_k}\qquad k=1,2,\cdots$$

$$x_1=z_0-\frac{(z_0-y_0)^2}{z_0-2y_0+x_0}\approx12.3965-\frac{100.4305}{9.1465}\approx1.41629$$

$$y_1=x_1^3-1=(1.41629)^3-1\approx1.84091$$

$$z_1=y_1^3-1=(1.84091)^3-1\approx5.23875$$

$$x_2 = z_1 - \frac{(z_1 - y_1)^2}{z_1 - 2y_1 + x_1} = 5.23875 - \frac{11.54532}{2.97384} \approx 1.35646$$

$$y_2 = x_2^3 - 1 = (1.35646)^3 - 1 \approx 1.49586$$

$$z_2 = y_2^3 - 1 = (1.49586)^3 - 1 \approx 2.34714$$

$$x_3 = z_2 - \frac{(z_2 - y_2)^2}{z_2 - 2y_2 + x_2} \approx 2.34714 - \frac{0.72468}{0.71188} \approx 1.32916$$

$$\vdots$$

从上面的计算可以看出，加速后的迭代是收敛的。对于原线性收敛或不收敛的数列，通过加速后可以达到更快的收敛, 在一定条件下, 甚至可达到二阶收敛。

定理 设函数 $\psi(x)$ 由 $\varphi(x)$ 按如下定义

$$\psi(x)=\varphi(\varphi(x))-\frac{\left[\varphi(\varphi(x))-\varphi(x)\right]^2}{\varphi(\varphi(x))-2\varphi(x)+x}$$

① 若x^*是 $\varphi(x)$ 的不动点, $\varphi'(x)$ 在x^*处连续, 且 $\varphi'(x^*)\neq 1$, 则x^*也是 $\psi(x)$ 的不动点; 反之, 若x^*是 $\psi(x)$ 的不动点, 则 x^*也是 $\varphi(x)$ 的不动点。

② 若x^*是 $\varphi(x)$ 的不动点, $\varphi'''(x)$在x^*处连续, 且 $\varphi'(x^*)\neq 1$, 则**Steffensen**迭代法

$$y_k=\varphi(x_k),\ z_k=\varphi(y_k),\quad x_{k+1}=z_k-\frac{(z_k-y_k)^2}{z_k-2y_k+x_k},$$

k=1, 2, ⋯

至少具有局部平方收敛。

二、幂法的加速

使用幂法求矩阵$\boldsymbol{A}$的主特征值时，其收敛速度取决于比值

$r=\left|\dfrac{\lambda_2}{\lambda_1}\right|$ 的大小，r越小，收敛速度越快， 如果$r\approx 1$，则收敛速度就很慢，需要采用加速技术。 此时，可以采用**Aitken**加速.

（1） 取初始向量$\boldsymbol{v}^{(0)}$,

（2） $\boldsymbol{v}^{(k)}=\boldsymbol{A}^k\boldsymbol{v}^{(0)}/\max\left(\boldsymbol{A}^{k-1}\boldsymbol{v}^{(0)}\right) \quad k=1,2,\cdots$

且令 $m_k=\max\left(\boldsymbol{v}^{(k)}\right)$

（3）**Aitken**加速，即

$$\overline{m}_k=m_{k+2}-\frac{\left(m_{k+2}-m_{k+1}\right)^2}{m_{k+2}-2m_{k+1}+m_k} \qquad (4\text{-}61)$$

（4-61）就是用幂法求主特征值的加速公式。

例2 用幂法计算矩阵 $A=\begin{pmatrix}2&3&2\\10&3&4\\3&6&1\end{pmatrix}$ 的主特征值与特征向量。并对计算主特征值的迭代进行**Aitken**加速

解 取 $v^{(0)}=(0,0,1)^T$

$$v^{(1)}=Av^{(0)}=\begin{pmatrix}2&3&2\\10&3&4\\3&6&1\end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix}=\begin{pmatrix}2\\4\\1\end{pmatrix}\qquad m_1=\max\left(v^{(1)}\right)=4$$

$$u^{(1)}=\frac{1}{4}(2,4,1)^T=(0.5,1,0.25)^T$$

$$v^{(2)}=Av^{(1)}=\begin{pmatrix}2&3&2\\10&3&4\\3&6&1\end{pmatrix}\begin{pmatrix}0.5\\1\\0.25\end{pmatrix}=\begin{pmatrix}4.5\\9\\7.76\end{pmatrix}\qquad m_2=\max\left(v^{(2)}\right)=9$$

$$u^{(2)}=\frac{1}{9}(4.5,9,7.75)^T=(0.5,1,0.8611)^T$$

$m_3,m_4,\cdots,m_8$ 见表。

进行Aitken加速

$$\bar{m}_1 = m_3 - \frac{(m_3 - m_2)^2}{m_3 - 2m_2 + m_1} = 11.4444 + 2.3380 = 13.7824$$

$$\bar{m}_2 = m_4 - \frac{(m_4 - m_3)^2}{m_4 - 2m_3 + m_2} = 10.9224 + 0.1417 = 11.0641$$

$$\bar{m}_3 = m_5 - \frac{(m_5 - m_4)^2}{m_5 - 2m_4 + m_3} = 11.0140 - 0.0139 = 11.0001$$

k	m_k	$\left(\boldsymbol{u}^{(k)}\right)^T$	$\bar{m}_k$
0	**1**	**(0, 0, 1.0000)**	
1	**4**	**(0.5000, 1.0000, 0.2500)**	
2	**9**	**(0.5000, 1.0000, 0.8611)**	
3	**11.4444**	**(0.5000, 1.0000, 0.7306)**	**13.7324**
4	**10.9224**	**(0.5000, 1.0000, 0.7535)**	**11.0641**
5	**11.0140**	**(0.5000, 1.0000, 0.7493)**	**11.0001**
6	**10.9927**	**(0.5000, 1.0000, 0.7501)**	
7	**11.0004**	**(0.5000, 1.0000, 0.7500)**	
8	$\boxed{\mathbf{11.0000}}$	$\boxed{\mathbf{(0.5000, 1.0000, 0.7500)}}$	

完了