迭代逼近 - GG大婶 - 博客园

在许多实际问题中，常常需要求解这样的线性方程组：他们的系数矩阵数很高，但非零元素很少，也就是大型稀疏线性代数方程组。对于这类方程组，如果不具备带状性，那么用直接发求解就不会很有效。因为用直接法进行消元或矩阵的三角分解时，没有考虑到系数矩阵的稀疏性，破坏了系数矩阵的形状，导致计算量的增加和存储单元的浪费。

迭代法是通过逐次迭代来逼近方程组的解，因此，收敛性和收敛速度是构造迭代方法时要注意的问题。那么，是否可以构造一种适用于一般情况的迭代法？答案是否定的，这时因为不同的稀疏矩阵具有不同的性态，一般的，每一种迭代法都具有一定的适用范围。

一，雅克比迭代

考虑线性方程组，如下：

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \qquad \vdots \qquad \qquad \vdots \qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \tag{2.1}$$

采用矩阵和向量记号，可以把（2.1）写成：

$$Ax = b, \tag{2.2}$$

为了方便给出矩阵表示式，引进下列矩阵分裂：

$$A = -L + D - U, \tag{2.3}$$

其中：

$$L = \begin{bmatrix} 0 & & & \\ -a_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ -a_{n1} & \cdots & -a_{n,n-1} & 0 \end{bmatrix},$$

$$D = \begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{bmatrix},$$

$$U = \begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ & 0 & \ddots & \vdots \\ & & \ddots & -a_{n-1,n} \\ & & & 0 \end{bmatrix}.$$

从式（2.1）的第i个方程中解出xi：

$$\begin{cases} x_1 = \dfrac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3 \cdots - a_{1n}x_n) \\ \vdots \qquad \vdots \qquad \qquad \vdots \\ x_i = \dfrac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i=1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}x_j\right) \\ \vdots \qquad \vdots \qquad \qquad \vdots \\ x_n = \dfrac{1}{a_{nn}}(b_n - a_{n1}x_1 - a_{n2}x_2 \cdots - a_{n,n-1}x_{n-1}). \end{cases}$$

我们把迭代前面的值代入上式右边，从而得到等式左边的值作为新一次迭代的新值。如此反复就得到了雅克比迭代公式：

$$\begin{cases} x_1^{(m)} = \dfrac{1}{a_{11}}(b_1 - a_{12}x_2^{(m-1)} \cdots - a_{1n}x_n^{(m-1)}) \\ \vdots \qquad \vdots \qquad \qquad \vdots \\ x_i^{(m)} = \dfrac{1}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(m-1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(m-1)}) \\ \vdots \qquad \vdots \qquad \qquad \vdots \\ x_n^{(m)} = \dfrac{1}{a_{nn}}(b_n - a_{n1}x_1^{(m-1)} \cdots - a_{n,n-1}x_{n-1}^{(m-1)}). \end{cases}$$

由式（2.4）及采用矩阵A的分裂记号（2.3），可以得到：

$$Dx^{(m)} = b + Lx^{(m-1)} + Ux^{(m-1)}$$

于是雅克比迭代的矩阵表示形式为：

$$x^{(m)} = (D-L)^{-1}Ux^{(m-1)} + (D-L)^{-1}b. \tag{2.7}$$

现在回过过头来想想为什么这么做会收敛。在新一轮的迭代完成后，得到的新值与解之间的误差其实是其他所有值的综合，因为有的值会比解大一点，有的值会比解小一点，这样得到的数值可能就会更接近解，当然我们可以构造出发散的例子，也就是通过迭代，得到的值与解之间的偏差越来越大。

二、高斯-赛德尔迭代

在雅克比迭代过程中都是用一组新值来计算得到旧的值，因为一般情况下新的结果会比旧的结果更加精确，所以在计算的过程中能用新值就用新的，这样能加速收敛的过程。这就是高斯-赛德尔迭代：

$$\begin{cases} x_1^{(m)} = \dfrac{1}{a_{11}}(b_1 - a_{12}x_2^{(m-1)} \cdots - a_{1n}x_n^{(m-1)}) \\ \vdots \qquad \vdots \qquad \vdots \\ x_i^{(m)} = \dfrac{1}{a_{ii}}(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(m)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(m-1)}) \\ \vdots \qquad \vdots \qquad \vdots \\ x_n^{(m)} = \dfrac{1}{a_{nn}}(b_n - a_{n1}x_1^{(m)} \cdots - a_{n,n-1}x_{n-1}^{(m)}). \end{cases}$$

三、SOR迭代

在迭代的过程中可能出现反复的现象：也就是说迭代得到的结果和真实的解之间的差距没有变小，而是有时比真实解大有时小而已。可以引入松弛因子来防止这种现象并能起到加速收敛的作用：

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

SOR迭代的矩阵表示形式：

$$x^{(m)} = (D - \omega L)^{-1}\{(1-\omega)D + \omega U\}x^{(m-1)} + \omega(D - \omega L)^{-1}b$$

四、SSOR迭代

在SOR迭代过程中，新分量计算是依次从第1个到第n个逐个进行的，这个次序也可以反过来，即得到SSOR迭代：

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