计算方法及MATLAB实现

郑勋烨　编著

主审：高世臣　褚宝增　王祖朝　王翠香

副审：李少琪　李明霞　赵琳琳　赵俊芳

国防工业出版社

第六章　线性方程组的迭代法

§6.1　线性方程组的古典迭代法

【 定义 1. 线性方程组的 (一阶定常) 迭代法 】

将方程组 $Ax = b$ 变形化为不动点映照 $x = Bx + f$ 的形式 (类比将求根方程 $f(x) = 0$ 等价变形化为 $x = \varphi(x)$), 据此构造迭代序列 $x^{(k+1)} = Bx^{(k)} + f$ 逐步代入初始值求近似解的方法称为 线性方程组的 (一阶定常) 简单迭代法. 若极限 $\lim\limits_{k\to+\infty} x^{(k)} = x^*$ 存在有限 (事实上这一不动点即为精确解向量), 则称迭代法 收敛(收敛意义可视为依向量范数收敛), 否则称为 发散 的.

作系数矩阵 A 的矩阵分解

$A = M - N = D - L - U, Mx = Nx + b, x = M^{-1}Nx + M^{-1}b.$

称 A 的近似矩阵 M 为迭代法的 分裂矩阵. 而称矩阵

$M^{-1}N = M^{-1}(M - A) = I - M^{-1}A$ 为迭代法的 迭代矩阵.

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

$$= \begin{pmatrix} a_{11} & & \cdots & \\ & a_{22} & \cdots & \\ \vdots & \vdots & \ddots & \vdots \\ & & \cdots & a_{nn} \end{pmatrix} + \begin{pmatrix} 0 & & \cdots & \\ a_{21} & 0 & \cdots & \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{pmatrix}$$

$$+\begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ & & \cdots & 0 \end{pmatrix}.$$

$$= \begin{pmatrix} a_{11} & & \cdots & \\ & a_{22} & \cdots & \\ \vdots & \vdots & \ddots & \vdots \\ & & \cdots & a_{nn} \end{pmatrix} - \begin{pmatrix} 0 & & \cdots & \\ -a_{21} & 0 & \cdots & \\ \vdots & \vdots & \vdots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & 0 \end{pmatrix}$$

$$- \begin{pmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ & 0 & \cdots & -a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ & & \cdots & 0 \end{pmatrix}$$

定义对角阵和上下三角形矩阵分别为

$$D = \begin{pmatrix} a_{11} & & \cdots & \\ & a_{22} & \cdots & \\ \vdots & \vdots & \ddots & \vdots \\ & & \cdots & a_{nn} \end{pmatrix}$$

$$L = \begin{pmatrix} 0 & & \cdots & \\ -a_{21} & 0 & \cdots & \\ \vdots & \vdots & \vdots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & 0 \end{pmatrix}$$

$$U = \begin{pmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ & 0 & \cdots & -a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ & & \cdots & 0 \end{pmatrix}$$

则有

$$L+U=\begin{pmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & 0 & \cdots & -a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & 0 \end{pmatrix}$$

为零主对角元的矩阵 (对角元被 D 拿走了).

而

$$D-L=\begin{pmatrix} a_{11} & & \cdots & \\ a_{21} & a_{22} & \cdots & \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

为原来的系数矩阵 $A=D-L-U$ 的下三角部分.

【 命题 1.　3 阶非齐次线性方程组的 Jacobi 迭代法设计 】

考虑形如下式的 3 阶非齐次线性方程组：

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases}$$

将非零的主对角元系数 $a_{ii} \neq 0 (i, j = 1, 2, 3)$ 剥离出来 (事实上要想迭代法收敛，我们对矩阵 A 有更强的要求，比如严格对角占优)，放在左边，改写成如下形式：

$$\begin{cases} a_{11}x_1 &= -(a_{12}x_2 + a_{13}x_3) + b_1 \\ a_{22}x_2 &= -(a_{21}x_1 + a_{23}x_3) + b_2 \\ a_{33}x_3 &= -(a_{31}x_1 + a_{32}x_2) + b_3 \end{cases}$$

引入矩阵向量记号，可以改写为 $Dx = (L + U)x + b$. 于是

$$\begin{cases} x_1 = \dfrac{-a_{12}x_2 - a_{13}x_3 + b_1}{a_{11}} \\ x_2 = \dfrac{-a_{21}x_1 - a_{23}x_3 + b_2}{a_{22}} \\ x_3 = \dfrac{-a_{31}x_1 - a_{32}x_2 + b_3}{a_{33}} \end{cases}$$

引入矩阵向量记号，可以改写为 $x = D^{-1}(L+U)x + D^{-1}b$. 这就是我们期望的“不动点”格式 $x = Jx + f$ 的形式，其中矩阵 $J = D^{-1}(L+U)$ 称为 Jacobi 迭代阵. 扰动项是 $f = D^{-1}b$. 据此构造迭代序列 $x^{(k+1)} = Jx^{(k)} + f$ 如下：

$$\begin{cases} x_1^{(k+1)} = \dfrac{-a_{12}x_2^{(k)} - a_{13}x_3^{(k)} + b_1}{a_{11}} \\ x_2^{(k+1)} = \dfrac{-a_{21}x_1^{(k)} - a_{23}x_3^{(k)} + b_2}{a_{22}} \\ x_3^{(k+1)} = \dfrac{-a_{31}x_1^{(k)} - a_{32}x_2^{(k)} + b_3}{a_{33}} \end{cases}$$

称为 Jacobi 迭代序列. 显然，这里的 Jacobi 迭代法为“横向显式迭代”，直接由 $x^{(k)}$ 迭代到 $x^{(k+1)}$ 而不使用分量的新信息（比如算 $x_2^{(k+1)}$ 时不用 $x_1^{(k+1)}$ 的计算结果）. 各个迭代方程的关系是平行的，先求哪个分量都行.

【 定义 2. Jacobi 迭代法 】

作 A 的分裂矩阵 $A = M - N = D - L - U$, 保留非奇异的主对角阵 D 在左方，而主对角元为 0 的矩阵 $L+U$ 在右方，获得等价变形 $Dx = (L+U)x + b$ ，则
$x = D^{-1}(L+U)x + D^{-1}b$. 由此构造迭代序列
$x^{(k+1)} = D^{-1}(L+U)x^{(k)} + D^{-1}b$. 称为 Jacobi 迭代法. 相应矩阵 $J = D^{-1}(L+U)$ 称为 Jacobi 迭代阵.

对应分量形式为

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

$$\Rightarrow x_i^{(k)} = \frac{b_i - \sum\limits_{j\neq i, j=1}^{n} a_{ij}x_j^{(k-1)}}{a_{ii}}.$$

【 **算法** 1. Jacobi **迭代算法** (Jacobi Iterative) 】

输入（ INPUT ）: n 阶矩阵 A 的阶数 n，所有元素 a_{ij}，非齐次扰动向量元素 b_i. 初值向量元素 $x0_i$. 精度上界 (Tolerance) TOL, 最大迭代步数 (Maximum number of iterations)N.

输出（ OUTPUT ）: 近似值向量元素 x_i 或者迭代溢出信息 (Maximum number of iterations exceeded).

STEP1. Set $k = 1$.

STEP2. While $k \leq N$,Do steps 3-6.

STEP3. For $i = 1, 2, \cdots, n$

$$\text{Set } x_i = \frac{b_i - \sum_{j \neq i, j=1}^{n} a_{ij} x0_j}{a_{ii}}.$$

STEP4. If $\|x - x0\| < TOL$,Then OUTPUT $(x_1, x_2, \cdots, x_n)$. STOP.

STEP5. Set $k = k + 1$.

STEP6. For $i = 1, 2, \cdots, n$, Set $x0_i = x_i$.

STEP7. OUTPUT('Maximum number of iterations exceeded').

STOP.

【 命题 2.　3 阶非齐次线性方程组的 Gauss-Seidel 迭代法设计 】

考虑形如下式的 3 阶非齐次线性方程组：

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{cases}$$

现在将 A 的“下三角”部分 $D-L$ 剥离出来，放在左边，改写成如下形式：

$$\begin{cases} a_{11}x_1 & = -(a_{12}x_2 + a_{13}x_3) + b_1 \\ a_{21}x_1 + a_{22}x_2 & = -a_{23}x_3 + b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 & = b_3 \end{cases}$$

即获得等价变形 $(D-L)x = Ux + b$.

由此构造迭代序列

$$\begin{cases} a_{11}x_1^{(k+1)} = -(a_{12}x_2^{(k)} + a_{13}x_3^{(k)}) + b_1 \\ a_{21}x_1^{(k+1)} + a_{22}x_2^{(k+1)} = -a_{23}x_3^{(k)} + b_2 \\ a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)} + a_{33}x_3^{(k+1)} = b_3 \end{cases}$$

即 $(D-L)x^{(k+1)} = Ux^{(k)} + b.$

或者，类似于 Jacobi 迭代法， 依然保留非奇异的主对角阵 D 在左方，则有

$$\begin{cases} a_{11}x_1^{(k+1)} &= -a_{12}x_2^{(k)} - a_{13}x_3^{(k)} + b_1 \\ a_{22}x_2^{(k+1)} &= -a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)} + b_2 \\ a_{33}x_3^{(k+1)} &= -a_{31}x_1^{(k+1)} - a_{32}x_2^{(k+1)} + b_3 \end{cases}$$

即 $Dx^{(k+1)} = Lx^{(k+1)} + Ux^{(k)} + b$. 称之为 Gauss-Seidel 迭代法.

显然， Gauss-Seidel 迭代法为既有横向过程又有纵向过程的“双向隐式”迭代，由 $x^{(k)}$ 迭代到 $x^{(k+1)}$，计算 $x_2^{(k+1)}$ 时需要使用第一个迭代序列算出的新信息 $x_1^{(k+1)}$，计算 $x_3^{(k+1)}$ 时需要使用第一和第二个迭代序列算出的新信息 $x_1^{(k+1)}, x_2^{(k+1)}$. 一般地，计算分量 $x_i^{(k+1)}$ 时需要使用新信息 $x_j^{(k+1)}, j = 1, 2, \cdots, i-1.$

具体计算分量时， Gauss-Seidel 迭代法也可写为

$$\begin{cases} x_1^{(k+1)} = \dfrac{-a_{12}x_2^{(k)} - a_{13}x_3^{(k)} + b_1}{a_{11}} \\ x_2^{(k+1)} = \dfrac{-a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)} + b_2}{a_{22}} \\ x_3^{(k+1)} = \dfrac{-a_{31}x_1^{(k+1)} - a_{32}x_2^{(k+1)} + b_3}{a_{33}} \end{cases}$$

形式上看，它和 Jacobi 迭代法完全相似，只是分子上的迭代序列不同， 计算分量 $x_i^{(k+1)}$ 时 Jacobi 迭代法不用而 Gauss-Seidel 迭代法要用分量的新信息 $x_j^{(k+1)}, j = 1, 2, \cdots, i-1.$

【 定义 3.Gauss-Seidel 迭代法 】

作 A 的分裂矩阵 $A = M - N = D - L - U$, 保留非奇异的下三角阵 $D - L$ 在左方，获得等价变形 $(D - L)x = Ux + b$. 则 $x = (D - L)^{-1}Ux + (D - L)^{-1}b$. 由此构造迭代序列 $(D - L)x^{(k+1)} = Ux^{(k)} + b$ 或仍由形式 $Dx = Lx + Ux + b$, 构造迭代序列 $Dx^{(k+1)} = Lx^{(k+1)} + Ux^{(k)} + b$. 称为 Gauss-Seidel 迭代法. 相应矩阵 $G = (D - L)^{-1}U$ 称为 Gauss-Seidel 迭代阵.

对应分量形式为

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

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

【 **算法** 2. Gauss-Seidel **迭代算法** (Gauss-Seidel Iterative) 】

输入（ INPUT ）: n 阶矩阵 A 的阶数 n，所有元素 a_{ij}，非齐次扰动向量元素 b_i. 初值向量元素 $x0_i$. 精度上界 (Tolerance) TOL, 最大迭代步数 (Maximum number of iterations)N.

输出（ OUTPUT ）: 近似值向量元素 x_i 或者迭代溢出信息 (Maximum number of iterations exceeded).

STEP1. Set $k = 1$.

STEP2. While $k \leq N$,Do steps 3-6.

STEP3. For $i = 1, 2, \cdots, n$

Set $x_i = \dfrac{b_i - \displaystyle\sum_{j=1}^{i-1} a_{ij} x_j - \sum_{j=i+1}^{n} a_{ij} x0_j}{a_{ii}}$.

STEP4. If $\|x - x0\| < TOL$,Then OUTPUT $(x_1, x_2, \cdots, x_n)$. STOP.

STEP5. Set $k = k + 1$.

STEP6. For $i = 1, 2, \cdots, n$, Set $x0_i = x_i$.

STEP7. OUTPUT('Maximum number of iterations exceeded').

STOP.

【 注记 1. Jacobi 迭代法与 Gauss-Seidel 迭代法的比较 】

1 、二者均为单步定常迭代法;

2 、 Jacobi 迭代法为横向显式迭代，由 $x^{(k)}$ 迭代到 $x^{(k+1)}$ 不使用新信息，原始矩阵 A 不变;

3 、 Gauss-Seidel 迭代法为双向隐式迭代，由 $x^{(k)}$ 迭代到 $x^{(k+1)}$ ，计算 $x_i^{(k+1)}$ 时需要使用新信息 $x_j^{(k+1)}, j = 1, 2, \cdots, i-1.$

4 、收敛速度 Gauss-Seidel 迭代法通常快于 Jacobi 迭代法，但并非尽然.

【 注记 2. 构造迭代序列的实用操作 】

构造迭代序列时，我们通常只作方程组等价变形形式 $Dx = (L+U)x + b$，因为它事实上同时适用于 Jacobi 迭代法和 Gauss-Seidel 迭代法，不同之处在于 Gauss-Seidel 迭代序列计算 $x_i^{(k+1)}$ 时需要使用新信息 $x_j^{(k+1)}, j = 1, 2, \cdots, i-1$.

【 引例 1 】 非齐次线性方程组

$$\begin{cases} 10x_1 + 3x_2 + x_3 & = 14 \\ 2x_1 - 10x_2 + 3x_3 & = -5 \\ x_1 + 3x_2 + 10x_3 & = 14 \end{cases}$$

变形为 $x = Bx + f$ 的形式并构造迭代序列，作 Jacobi 迭代法与 Gauss-Seidel 迭代法的比较.

解 增广矩阵 $(A|b) = \begin{pmatrix} 10 & 3 & 1 & 14 \\ 2 & -10 & 3 & -5 \\ 1 & 3 & 10 & 14 \end{pmatrix}$, 精确解为 $x = A^{-1}b = (1, 1, 1)^T$.

将方程组变形化为基本形式 $Dx = (L + U)x + b$, 对应于方程组即为

$$\begin{cases} 10x_1 & = 14 - 3x_2 - x_3 \\ -10x_2 & = -5 - 2x_1 - 3x_3 \\ 10x_3 & = 14 - x_1 - 3x_2 \end{cases}$$

除去左端系数即 $x = D^{-1}(L+U)x + D^{-1}b$，即至此已经变形为 $x = Bx + f$ 的形式. 对应于方程组就是

$$\begin{cases} x_1 &= \dfrac{1}{10}(-3x_2 - x_3 + 14) \\ x_2 &= -\dfrac{1}{10}(-2x_1 - 3x_3 - 5) \\ x_3 &= \dfrac{1}{10}(-x_1 - 3x_2 + 14) \end{cases}$$

即

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 & -\frac{3}{10} & -\frac{1}{10} \\ \frac{1}{5} & 0 & \frac{3}{10} \\ -\frac{1}{10} & -\frac{3}{10} & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} \frac{7}{5} \\ \frac{1}{2} \\ \frac{7}{5} \end{pmatrix}$$

由此构造 Jacobi 迭代序列仅作横向显式迭代,

$$\begin{pmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ x_3^{(k+1)} \end{pmatrix} = \begin{pmatrix} 0 & -\frac{3}{10} & -\frac{1}{10} \\ \frac{1}{5} & 0 & \frac{3}{10} \\ -\frac{1}{10} & -\frac{3}{10} & 0 \end{pmatrix} \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ x_3^{(k)} \end{pmatrix} + \begin{pmatrix} \frac{7}{5} \\ \frac{1}{2} \\ \frac{7}{5} \end{pmatrix}$$

给定初值向量可迭代求解. 如取 $x^{(0)} = (0, 0, 0)^T$, 迭代一步得

$$\begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{pmatrix} = f = \begin{pmatrix} \frac{7}{5} \\ \frac{1}{2} \\ \frac{7}{5} \end{pmatrix} = \begin{pmatrix} 1.4 \\ 0.5 \\ 1.4 \end{pmatrix}$$

由方程组变形为 $x = D^{-1}(L+U)x + D^{-1}b$ 的形式直接构造 Gauss-Seidel 迭代序列，需要同时作双向隐式迭代，

$$\begin{cases} x_1^{(k+1)} &= \dfrac{1}{10}(-3x_2^{(k)} - x_3^{(k)} + 14) \\ x_2^{(k+1)} &= -\dfrac{1}{10}(-2x_1^{(k+1)} - 3x_3^{(k)} - 5) \\ x_3^{(k+1)} &= \dfrac{1}{10}(-x_1^{(k+1)} - 3x_2^{(k+1)} + 14) \end{cases}$$

如取 $x^{(0)} = (0, 0, 0)^T$, 迭代一步得

$$\begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{pmatrix} = \begin{pmatrix} \frac{7}{5} \\ \frac{1}{5} \times \frac{7}{5} + \frac{1}{2} \\ -\frac{1}{10} \times (\frac{7}{5} + 3 \times \frac{39}{50}) + \frac{7}{5} \end{pmatrix} = \begin{pmatrix} 1.4 \\ 0.78 \\ 1.026 \end{pmatrix}$$

更接近精确解. 大致可见对于本问题 Gauss-Seidel 迭代法收敛快于 Jacobi 迭代法.

【 **定义** 4. SOR **迭代法 (逐次超松弛迭代法)**(Successive Over Relaxation Method) 】

根据等价变形 $(D-L)x = Ux + b$ 利用凸线性组合 (加权平均) 的技巧作适当改造 (类比我们对非线性方程求根的加速方法)，引入一个加权因子 $\omega, 0 < \omega < 2$, 把主对角阵 D 和上三角阵 U 作凸线性组合，将方程组变形成为

$$(D - \omega L)x = ((1-\omega)D + \omega U)x + \omega b$$

即

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

其中迭代阵

$$L_\omega = (D - \omega L)^{-1}((1-\omega)D + \omega U)$$

称为 逐次松弛迭代阵. 由此构造迭代序列

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

或

$$Dx^{(k+1)} = (1-\omega)Dx^{(k)} + \omega(Lx^{(k+1)} + Ux^{(k)} + b)$$

称为 SOR 迭代法. 这里特别把加权因子 $\omega, 0 < \omega < 2$ 称为 松弛因子(下文将证明，其取值区间 $0 < \omega < 2$ 是保证 SOR 迭代法收敛的必要条件).

$$x^{(k+1)} = x^{(k)} + \omega D^{-1}(-Dx^{(k)} + Lx^{(k+1)} + Ux^{(k)} + b)$$

$$= (1-\omega)x^{(k)} + \omega D^{-1}(Lx^{(k+1)} + Ux^{(k)} + b).$$

写为分量迭代格式即

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

或

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

SOR 迭代法是求解大型稀疏线性方程组的基本方法，实用中通常取 $\omega, 1 < \omega < 2$ 称为 超松弛因子，若 $0 < \omega < 1$ 则称为低松弛因子. 相应迭代法分别称为 SOR 超松弛法 和 低松弛法. 松弛因子 ω 的选取对于迭代法的收敛速度具有决定性的影响.

【 注记 3. SOR 迭代法与 Gauss-Seidel 迭代法的比较 】

1 、 SOR 迭代法是 Gauss-Seidel 迭代法通过凸线性组合 (加权平均) 的加速修正，当 $\omega = 1$ 时 SOR 迭代法退化为 Gauss-Seidel 迭代法;

2 、 SOR 迭代法与 Gauss-Seidel 迭代法均为双向隐式迭代，由 $x^{(k)}$ 迭代到 $x^{(k+1)}$ ，计算 $x_i^{(k+1)}$ 时需要使用新信息 $x_j^{(k+1)}, j = 1, 2, \cdots, i-1$.

3 、 SOR 迭代法是求解大型稀疏线性方程组的基本方法；此外还有 SSOR 迭代法 (对称超松弛迭代法).

【 **算法** 3. SOR **迭代算法** (SOR Iterative) 】

输入（ INPUT ）: n 阶矩阵 A 的阶数 n，所有元素 a_{ij}，非齐次扰动向量元素 b_i. 初值向量元素 $x0_i$. 精度上界 (Tolerance) TOL, 最大迭代步数 (Maximum number of iterations)N. 松弛因子（ Relaxation Parameter)ω.

输出（ OUTPUT ）: 近似值向量元素 x_i 或者迭代溢出信息 (Maximum number of iterations exceeded).

STEP1. Set $k = 1$.

STEP2. While $k \leq N$,Do steps 3-6.

STEP3. For $i = 1, 2, \cdots, n$

Set

$$x_i = (1-\omega)x0_i + \omega \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}x0_j}{a_{ii}}$$

STEP4. If $\|x - x0\| < TOL$,Then OUTPUT $(x_1, x_2, \cdots, x_n)$. STOP.

STEP5. Set $k = k + 1$.

STEP6. For $i = 1, 2, \cdots, n$, Set $x0_i = x_i$.

STEP7. OUTPUT('Maximum number of iterations exceeded').

STOP.

【 注记 5. 构造 SOR 迭代序列实用操作 】

SOR 迭代序列实用的构造方式是依据变形 $Ax = b$ 从而 $0 = b - Ax$ ，由此 $0 = \omega D^{-1}(b - Ax)$ ，于是 $x = x + \omega D^{-1}(b - Ax)$. 对应方程组同时作双向隐式迭代即获得 SOR 迭代序列.

【 引例 2 】 讨论非齐次线性方程组

$$\begin{cases} 4x_1 + 3x_2 & = 24 \\ 3x_1 + 4x_2 - x_3 & = 30 \\ -x_2 + 4x_3 & = -24 \end{cases}$$

的 SOR 迭代法.

解 增广矩阵 $(A|b) = \begin{pmatrix} 4 & 3 & 0 & 24 \\ 3 & 4 & -1 & 30 \\ 0 & -1 & 4 & -24 \end{pmatrix}$, 精确解为 $x = A^{-1}b = (3, 4, -5)^T$.

将方程组变形化为形式 $x = x + D^{-1}(b - Ax)$.

$$\begin{cases} x_1 &= x_1 + \dfrac{1}{4}(24 - 4x_1 - 3x_2) \\ x_2 &= x_2 + \dfrac{1}{4}(30 - 3x_1 - 4x_2 + x_3) \\ x_3 &= x_3 + \dfrac{1}{4}(-24 + x_2 - 4x_3) \end{cases}$$

由此构造 SOR 迭代序列 (松弛因子为 ω)

$$\begin{cases} x_1^{(k+1)} &= x_1^{(k)} + \dfrac{\omega}{4}(24 - 4x_1^{(k)} - 3x_2^{(k)}) \\ x_2^{(k+1)} &= x_2^{(k)} + \dfrac{\omega}{4}(30 - 3x_1^{(k+1)} - 4x_2^{(k)} + x_3^{(k)}) \\ x_3^{(k+1)} &= x_3^{(k)} + \dfrac{\omega}{4}(-24 + x_2^{(k+1)} - 4x_3^{(k)}) \end{cases}$$

给定初值向量可迭代求解. 如取松弛因子为 $\omega = 1$, 初值 $x^{(0)} = (1, 1, 1)^T$, SOR 迭代序列退化为 Gauss-Seidel 迭代法:

$$\begin{cases} x_1^{(k+1)} &= -\frac{3}{4}x_2^{(k)} + 6 \\ x_2^{(k+1)} &= -\frac{3}{4}x_1^{(k+1)} + \frac{1}{4}x_3^{(k)} + 7.5 \\ x_3^{(k+1)} &= \frac{1}{4}x_2^{(k+1)} - 6 \end{cases}$$

迭代 7 步得

$$\begin{pmatrix} x_1^{(7)} \\ x_2^{(7)} \\ x_3^{(7)} \end{pmatrix} = \begin{pmatrix} 3.0134110 \\ 3.9888241 \\ -5.0027940 \end{pmatrix}$$