计算方法丛书

解数学物理问题的异步并行算法

康立山　孙乐林　陈毓屏　著

科学出版社

计算方法丛书

解数学物理问题的异步并行算法

康立山　孙乐林　陈毓屏 著

科学出版社

1985

内 容 简 介

全书共分三章.第一章引进一般松弛法和混乱松弛法的基本概念;第二章论述区域分裂法的一般理论和解椭圆型偏微分方程边值问题的 Schwarz 算法, Schwarz 混乱松弛法以及它们的收敛性、误差估计和异步并行算法的步骤,并对非定常问题以及某些非线性问题作了类似的处理;第三章提供了多方面的数值例子.

本书可供数值分析工作者、计算机研制工作者以及高等院校有关专业的教师、学生参考.

计算方法丛书

解数学物理问题的异步并行算法

康立山　孙乐林　陈毓屏 著

责任编辑　向安全

科学出版社 出版

北京朝阳门内大街 137 号

中国科学院印刷厂印刷

新华书店北京发行所发行　各地新华书店经售

*

1985 年 9 月第 一 版　开本：850×1168　1/32

1985 年 9 月第一次印刷　印张：4

印数：0001—9,500　字数：101,000

统一书号：13031·2972

本社书号：4517·13—1

定价：1.15 元

《计算方法丛书》编委会

前　言

本书是根据武汉大学数学系并行算法研究小组提供的部分研究成果写成的，主要介绍一类新型的解数学物理问题的异步并行算法.

当今，科学技术的许多领域不断提出一些巨大的计算课题，这些课题要求计算机具有极快的运算速度和极大的信息吞吐量. 不少专家认为，要有效地提高计算机系统性能以解算巨大课题，主要的途径是选择体系结构的并行化. 所有的并行计算机都是以并行算法作基础的. 因此，研究和发展与并行机相适应的并行算法已经成为数值分析工作者面临的十分紧迫而又前景广阔的课题. 事实上，正是由于出现了并行多处理机系统，并行算法才在近十多年里发展成为数值分析的一个活跃的新方向. 并行算法可分为同步并行算法和异步并行算法两大类. 目前同步并行算法成果甚丰，但异步并行算法却还处在萌芽阶段. 前已指出，本书着眼于讨论一类新型的异步并行算法.

什么是异步并行算法？文献[1]认为，它应是一个具有下述性质的并行算法：第一，有一个可为所有过程接触的整体变量的集合；第二，当过程的一个阶段做完时，首先，过程"读"一些整体变量，然后根据这些整体变量之值及上阶段刚得到的结果，过程修改某些整体变量，接着启动下一阶段或结束它本身. 异步并行算法的主要特征是它的过程在任何时候都不需要等待输入，而只根据整体变量里的最新信息来决定自己是继续，还是结束.

本书共分三章. 第一章首先系统地论述解线性代数方程组的松弛方法，自然地引入各种不同的松弛概念，特别是混乱松弛的概念. 然后用物理直观的方式把这些松弛概念应用到数学物理问题的求解中去. 第二章是本书最重要的部分. 在这一章里，我们从

推广 Schwarz 交替法入手建立了区域分裂法的一般理论，着重研究了解线性椭圆型偏微分方程边值问题的 Schwarz 算法与 Schwarz 混乱松弛法，它们的收敛性与误差估计，以及它们的异步并行实现．接着介绍解非线性椭圆边值问题的 Picard-Schwarz 混乱松弛法与 Newton-Schwarz 混乱松弛法等，并进一步将这些算法应用到解线性与非线性抛物型方程的混合问题及定常与非定常的 Navier-Stokes 方程．第三章是一些数值试验结果，介绍在异步并行多处理机上如何应用前两章提供的异步并行算法解弹性裂缝问题(奇异边值问题)，二维定常与非定常的不可压缩粘性流问题，以及物理、化学、生物等学科提出的许多典型的数学物理问题．

本书提供的算法既可用于普通的串行计算机，也可用于并行计算机，特别适用于分布式并行计算机．它使得小型计算机群可能解算大型问题，使得在解同一问题时对不同的子区可分别选用不同的被认为是最合适的计算方法．这样就有益于充分有效地利用资源——包括硬件资源、软件资源和算法资源．

有必要指出，作者在书中比较注重算法的数学描述，属于算法复杂性的某些问题则很少提及．我们在由四台微处机组成的"WUPP-80"分布式并行处理系统上所作的大量数值试验结果表明：一般说来，解算同一问题，用本书提供的异步并行算法在并行多处理机上计算，比用不增加运算量的串行算法在同型的单机上计算，可获得二倍乃至更高的并行加速．

在开展异步并行算法研究的过程中，我们得到了吉林大学的冯果忱同志，兰州大学的王德人同志，武汉水利电力学院的郑邦民同志和国内外其他许多朋友的支持与鼓励．这里特向他们致谢．

作者诚挚地感谢华中工学院的王能超同志．他欣然审阅了本书原稿并提出了许多宝贵意见．根据他的意见，作者修改了原稿．

由于作者水平有限，书中难免还有许多错误，敬请读者批评指正．

作者

1983 年 8 月于武汉大学

目　　录

第一章　一般松弛法与混乱松弛法

本书将反复提到“混乱松弛法”(Chaotic Relaxation) 这个目前还没有广泛流行的概念. 为了介绍混乱松弛法，我们先从一般的松弛法讲起.

要离开任何具体的计算过程而给数值分析中“松弛”一词的涵义作简洁而确切的解释或许还有些难处. 但众所周知，在我们接触过的各类松弛法中都有这样的迭代程序：给定问题的解的满足某些条件的一个猜测，根据某种标准检查它是否是问题的“合乎要求”的解；若不是，则依某一原则对它执行一次调整(松弛)，再以调整后的结果作为新的猜测，如此循环直至再无调整的必要. 在本章的最后一节里读者可以看到，松弛法的这种反复调整的迭代过程，有着清楚的物理背景.

事实上，“松弛法”是由牛津大学的 R. V. Southwell 在 1935 年首先发展起来的. 他从弹性力学的观点出发来解数学物理问题中的差分方程，特别是解由 *Laplace* 方程和双调和方程导出的差分方程. 由于当时计算主要依赖于手工，所以他们的松弛法是一种带有一定随机性的松弛法，即松弛的顺序是由计算者通过对残量的观察临时选择的.

电子数字计算机的出现，使松弛法进入了一个崭新的阶段. 特别重要的进展是 1954 年 D. Young 发表了他的著名论文《解椭圆型偏差分方程的迭代法》[2]，把 Southwell-Fox 的观察松弛法发展成为一种系统的迭代法 (Systematic Iteration)，他称之为逐次超松弛法SOR (Successive Over Relaxation). 随后又出现了所谓逐次低松弛法 SUR，对称超松弛法 SSOR，逐块超松弛法 BSOR 和隐交替方向法 ADI 等. 计算技术的迅速发展，特别是出现了 MIMD (Multiple-Instruction Stream Multiple-Data Stream) 这样的计算机

系统，有力地推动了松弛法的研究．也正是为了适应这种可以进行异步并行计算的计算机系统，D. Chazan 和 W. L. Miranker[3] 1969 年提出了混乱松弛法，使松弛法具有更广泛的意义．此后一些法国学者如 J. C. Miellou, F. Robert, G. M. Baudet 和 M. N. El. Tarazi[4~7] 在这方面又做了许多工作．

§ 1.1 解线性代数方程组的松弛法

我们采用 Фаддеев 和 Фаддеева[8] 的叙述方法．

考虑线性代数方程组

$$Ax = f, \tag{1.1}$$

这里 $x, f \in R^n$, $A: R^n \to R^n$. 设 $A > 0$, 即

$$(Ax, x) \geqslant 0, \quad \forall x \in R^n,$$

而且只有当 $x = 0$ 时上式中的等号才成立．这时方程组 (1.1) 有唯一解．把它记作 x^*.

我们用近似方法求(1.1)的解：

给一初始猜测 x^0, 得到

$$f - Ax^0 = r^0 = \begin{pmatrix} r_1^0 \\ r_2^0 \\ \vdots \\ r_n^0 \end{pmatrix},$$

式中 r^0 称为残向量．若 $x^0 \neq x^*$, 则 $r^0 \neq 0$．我们的目的是要逐次调整近似解向量 x^i, 使得残向量 $r^i (i = 0,1,2,\cdots)$ 逐渐变成零向量．例如，我们调整 x^0 的第 j 个分量：

$$x^1 = x^0 + \alpha e_j,$$

其中 e_j 为单位向量，它的第 j 个分量为 1，其余分量皆为 0, α 是某一未定实参数．我们选择 α 使调整后的残向量 r^1 的第 j 个分量 $r_j^1 = 0$，这是容易办到的．因为 $A > 0$，所以 A 的对角元素 $a_{jj} > 0$. 取 $\alpha = r_j^0 / a_{jj}$, 则有

$$f - Ax^1 = r^1 = (r_1^1, r_2^1, \cdots, r_{j-1}^1, 0, r_{j+1}^1, \cdots r_n^1)^T.$$

故当

$$\boldsymbol{x}^1 = \boldsymbol{x}^0 + \frac{r_i^0}{a_{ii}}\boldsymbol{e}_i$$

时有 $r_i^1 = 0$.

若记误差向量为

$$\boldsymbol{y}^1 = \boldsymbol{x}^* - \boldsymbol{x}^1,$$

则有

$$A\boldsymbol{y}^1 = \boldsymbol{r}^1.$$

因为 $A > 0$, 故泛函

$$J(\boldsymbol{y}^1) = (A\boldsymbol{y}^1, \boldsymbol{y}^1) \geqslant 0, \tag{1.2}$$

且只有当 $\boldsymbol{y}^1 = 0$ 即 $\boldsymbol{x}^* = \boldsymbol{x}^1$ 时才有 $J(\boldsymbol{y}^1) = 0$.

又由

$$\boldsymbol{x}^1 = \boldsymbol{x}^0 + \alpha\boldsymbol{e}_i,$$

$$\boldsymbol{y}^1 = \boldsymbol{x}^* - \boldsymbol{x}^1 = \boldsymbol{x}^* - \boldsymbol{x}^0 - \alpha\boldsymbol{e}_i = \boldsymbol{y}^0 - \alpha\boldsymbol{e}_i$$

推得

$$\begin{aligned} J(\boldsymbol{y}^1) &= (A\boldsymbol{y}^1, \boldsymbol{y}^1) = J(\boldsymbol{y}^0) - 2\alpha r_i^0 + \alpha^2 a_{ii} \\ &= J(\boldsymbol{y}^0) + \frac{1}{a_{ii}}(\alpha a_{ii} - r_i^0)^2 - \frac{(r_i^0)^2}{a_{ii}}. \end{aligned} \tag{1.3}$$

因 $a_{ii} > 0$, 故当 $\alpha = r_i^0/a_{ii}$ 时 $J(\boldsymbol{y}^1)$ 下降得最快. 这时有

$$J(\boldsymbol{y}^1) = J(\boldsymbol{y}^0) - \frac{(r_i^0)^2}{a_{ii}}. \tag{1.4}$$

如此推知,只要我们每次(一般地, 第 i 次)对相应于 $r_i^{i-1} \neq 0$ 的那个方程(第 i 个方程)作上述调整,就可得到向量序列 $\{\boldsymbol{x}^i\}$, 它使$\{J(\boldsymbol{y}^i)\}$成为一个严格单调下降的数列. 由(1.2)式知这数列有下界,故必收敛到它的极小值零, 相应地,序列$\{\boldsymbol{x}^i\}$收敛到(1.1)的解 $\boldsymbol{x}^*$. 因为若不然, 设 $\tilde{\boldsymbol{x}}$ 使泛函 J 取极小值 $J(\tilde{\boldsymbol{y}}) = J(\boldsymbol{x}^* - \tilde{\boldsymbol{x}})$, 但是 $\boldsymbol{x}^i \to \tilde{\boldsymbol{x}} \neq \boldsymbol{x}^*$, 则推出

$$\boldsymbol{f} - A\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{r}} \neq \boldsymbol{0}.$$

这意味着还存在 $\tilde{r}_i \neq 0$, 即还可调整 $\tilde{\boldsymbol{x}}$,使得

$$\hat{\boldsymbol{x}}^* = \hat{\boldsymbol{x}} + \frac{(\tilde{r}_i)^2}{a_{ii}}\boldsymbol{e}_i,$$

$$J(\tilde{y}^*) = J(\tilde{y}) - \frac{(\tilde{r}_j)^2}{a_{jj}}.$$

这与 $J(\tilde{y})$ 为泛函之极小值的假设矛盾.

今记第 i 步调整第 j 个分量得到的向量为 $x^{i(j)}$. 我们把 $i(j)$ 的每一择定方案视为一种策略. 一个很自然的问题是: 在什么策略意义下使 $i(j)$ 满足什么条件时序列 $\{x^{i(j)}\}$ 收敛最快?

R. V. Southwell 的调整方案是: 若第 $i-1$ 步有

$$\max_{1\leqslant k\leqslant n} |r_k^{i-1}| = |r_j^{i-1}|,$$

则下一步就调整第 j 个分量,从而得到 $x^{i(j)}$.

至此,我们可以对上述近似过程作如下概括:

1) 给一初始猜测 x^0, 从而可设 $x^{i-1(l)}$ 为已知;

2) 作 $r^{i-1} = f - Ax^{i-1(l)}$;

3) 若 $\max\limits_{1\leqslant k\leqslant n} |r_k^{i-1}| = |r_j^{i-1}| > 0$, 则作调整

$$x^{i(j)} = x^{i-1(l)} + \frac{r_j^{i-1}}{a_{jj}} e_j,$$

$$i = 1, 2, \cdots; \ j, l \in \{1, 2, \cdots, n\}.$$

这就是 Southwell-Fox 的逐点观察松弛法.

由(1.4)可看出,要使 $J(y)$ 下降最快,则需采取这样的调整方案: 如果

$$\max_{1\leqslant k\leqslant n} \frac{(r_k^{i-1})^2}{a_{kk}} = \frac{(r_j^{i-1})^2}{a_{jj}},$$

那么第 i 步就调整第 j 个分量. 但这里所说的"最快"也只不过是在"走一步看一步"的策略意义下的"下降最快"罢了. 考虑调整的时序与调整量的大小却如下棋一样, 一个聪明的棋手总是运筹全局, 深谋远虑. 愈能如此, 其策略愈见高明. 从有益全局(计算过程的高度规格化与自动化,减少信息量和形成信息的运算量,从而最终加快收敛速度)的意义上讲, Young 的 *SOR* 算法是一个更高明的调整策略.

我们进一步分析上述逐点调整法的收敛条件. 由(1.3)知, 只要 α 满足

$$(r_j^0)^2 - (\alpha a_{jj} - r_j^0)^2 > 0,$$

更一般地，只要 α 满足

$$(r_j^{i-1})^2 - (\alpha a_{jj} - r_j^{i-1})^2 > 0,$$

泛函 $J(\boldsymbol{y})$ 就会下降，序列 $\{\boldsymbol{x}^{i(j)}\}$ 就收敛. 这条件等价于

$$\left|\alpha \frac{a_{jj}}{r_j^{i-1}} - 1\right| < 1.$$

令

$$\alpha = \omega \frac{r_j^{i-1}}{a_{jj}},$$

上述条件又转化成

$$|\omega - 1| < 1,$$

即

$$0 < \omega < 2. \tag{1.5}$$

当 ω 满足条件(1.5)时，有

$$J(\boldsymbol{y}^i) = J(\boldsymbol{y}^{i-1}) - \omega(2-\omega)\frac{(r_j^{i-1})^2}{a_{jj}},$$

$$i = 1, 2, \cdots; \ j \in \{1, 2, \cdots, n\}.$$

我们暂且不关心调整的次序，而对前述近似算法作进一步的归纳：

为解线性代数方程组(1, 1)，我们作

1) 给一初始猜测 $\boldsymbol{x}^0$，从而可设 $\boldsymbol{x}^{i-1(l)}$ 为已知；

2) 计算 $\boldsymbol{r}^{i-1} = \boldsymbol{f} - A\boldsymbol{x}^{i-1(l)}$；

3) 若 $\max\limits_{1\leqslant k\leqslant n} |r_k^{i-1}| = |r_j^{i-1}| > 0$，则作调整

$$\boldsymbol{x}^{i(j)} = \boldsymbol{x}^{i-1(l)} + \omega \frac{r_j^{i-1}}{a_{jj}} \boldsymbol{e}_j,$$

$$\omega \in (0, 2); \ j, l \in \{1, 2, \cdots, n\}; \ i = 1, 2, \cdots.$$

定义 1.1 上述逐点松弛法称为

(i) 完全松弛法，当 $\omega = 1$；

(ii) 低松弛法，当 $0 < \omega < 1$；

(iii) 超松弛法，当 $1 < \omega < 2$，

其中 ω 称为松弛因子. 相对于 (i), 也称 (ii), (iii) 为不完全松弛法.

如果松弛顺序是依次地取 $j=k, k+1, \cdots, n-1, n, 1, 2, \cdots, k-1 (1 \leqslant k \leqslant n)$, 然后循环下去, 则称为系统松弛. 由前面的论证可知, 当 $\omega \in (0, 2)$ 时, 系统松弛法是收敛的.

对系统松弛法而言, 当 $\omega = 1$ 时, 就是通常的 Seidel 迭代法. Seidel 迭代法是一种系统的完全松弛法, Young 提出的 *SOR* 法则是一种系统的超松弛法.

前面提到的 Southwell-Fox 的松弛法, 依 $\max\limits_{1 \leqslant k \leqslant n} |r_k| = |r_j|$ 或 $\max\limits_{1 \leqslant k \leqslant n} (r_k)^2/a_{kk} = (r_j)^2/a_{jj}$ 来决定下一步松弛第 j 个分量, 都是非系统的完全松弛法. 根据已作的论证, 它们都是收敛的. 如果在这类松弛法里也引入超或低松弛因子使之成为非系统的不完全松弛法, 算法仍然是收敛的.

下面讨论逐块松弛法. 设 A 有分块的形式:

$$A = \begin{bmatrix} A_{11} A_{12} \cdots A_{1m} \\ A_{21} A_{22} \cdots A_{2m} \\ \vdots \\ A_{m1} A_{m2} \cdots A_{mm} \end{bmatrix},$$

其中 A_{ij} 为 $m_i \times m_j$ 方阵, $\sum\limits_{j=1}^{m} m_j = n$. 又设 $A > 0, A_{ij} > 0$. 于方程组

$$AX = F \tag{1.6}$$

中, 将 F 与 X 对应于 A 分块:

$$F = (f_1, f_2, \cdots, f_m)^T;$$

$$X = (x_1, x_2, \cdots, x_m)^T,$$

其中 f_i 与 x_i 都是 m_i 维向量. 解(1.6)的逐块松弛法可描述如下:

给一初始猜测

$$X^0 = (x_1^0, x_2^0, \cdots, x_m^0)^T,$$

若残向量 $r^0 = (r_1^0, r_2^0, \cdots, r_j^0, \cdots, r_m^0)^T$ 的第 j 组分量 $r_j^0 \neq 0$, 则

松弛第 i 组分量

$$X^1 = X^0 + \omega \begin{pmatrix} 0 \\ \vdots \\ 0 \\ A_{ii}^{-1} r_i^0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

因 $A_{ii} > 0$，故 A_{ii}^{-1} 存在．第 i 次松弛第 j 组分量的算法是

$$X^{i(j)} = X^{i-1(l)} + \omega \begin{pmatrix} 0 \\ \vdots \\ 0 \\ A_{ii}^{-1} r_i^{i-1} \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

这两个式子表明松弛第 j 组分量一次就须求一次逆矩阵 A_{ii}^{-1}，或者解一次 m_i 阶线性方程组

$$A_{ii} x_i^i = A_{ii} x_i^{i-1} + \omega \left(f_i - \sum_{k=1}^{m} A_{ik} x_k^{i-1} \right), \tag{1.7}$$

得到 x_i^i 并用以替换 $X^{i-1(l)}$ 中的 x_i^{i-1} 而形成 $X^{i(j)}$.

为了研究逐块松弛法的收敛性，仿逐点松弛法，可得到

$$\begin{aligned} J(\boldsymbol{y}^i) &= (A(X^* - X^{i(j)}),\ X^* - X^{i(j)}) \\ &= J(\boldsymbol{y}^{i-1}) - \omega(2-\omega)(r_i^{i-1}, A_{ii}^{-1} r_i^{i-1}). \end{aligned}$$

因 $A_{ii} > 0$，所以 $(r_i^{i-1}, A_{ii}^{-1} r_i^{i-1}) \geqslant 0$，且仅当 r_i^{i-1} 为零向量时才取等号．故知只要松弛因子 $\omega \in (0,2)$，就有

$$J(\boldsymbol{y}^i) < J(\boldsymbol{y}^{i-1}),$$

即泛函 $J(\boldsymbol{y})$ 下降．完全类似地，我们给出

定义 1.2 称上述算法为

(i) 逐块完全松弛法，当 $\omega = 1$ 时；

(ii) 逐块低松弛法，当 $0 < \omega < 1$ 时；

(iii) 逐块超松弛法，当 $1 < \omega < 2$ 时.

后两者亦可并称为逐块不完全松弛法.逐点松弛法中的系统松弛、非系统松弛等概念都可以平行地在逐块松弛法中定义. 也容易证明，系统的和非系统的逐块松弛法都是收敛的. 顺便指出，系统的逐块松弛法 (*BSOR*) 是解椭圆型偏微分差分方程和有限元离散方程的常用工具.

若解方程组(1.7)也用松弛法，则形成了"嵌套"的松弛.

§1.2 解线性代数方程组的混乱松弛法

按照上节给出的算法作松弛运算：

$$x^{i(j)} = x^{i-1(l)} + \omega \frac{r_j^{i-1}}{a_{jj}} e_j,$$

$$\omega \in (0, 2);\ j, l \in \{1, 2, \cdots, n\};\ i = 1, 2, \cdots,$$

得到序列 $\{x^{i(j)}\}$. 我们可以将它的标号序列$\{i(j)\}$分成 n 个子序列

$$\{i_1(1)\}, \{i_2(2)\}, \cdots, \{i_j(j)\}, \cdots, \{i_n(n)\}.$$

这些标号子序列一一对应地记录着对向量 x 的每个分量的调整. 如果不对 $i(j)$ 作任何限制，则当 $i \to \infty$ 时，上述 n 个子序列中有的可能是无限集，有的可能是有限集，有的甚至可能是空集，但其中至少有一个是无限集，也就是说，当 $i \to \infty$ 时，标号 $i_1(1)$, $i_2(2), \cdots, i_j(j), \cdots, i_n(n)$ 中至少有一个趋向无穷大.

定义 1.3 称上节定义的逐点松弛法为逐点混乱松弛法(简称 *CR* 算法)，如果当 $i \to \infty$ 时有

$$i_j(j) \to \infty,\ \forall j \in \{1, 2, \cdots, n\};$$

相应地，称一个 *CR* 算法为

(i) 混乱完全松弛法 (*CCR*)，当 $\omega = 1$ 时；

(ii) 混乱低松弛法 (*CUR*)，当 $0 < \omega < 1$ 时；

(iii) 混乱超松弛法 (COR)，当 $1 < \omega < 2$ 时.

这样一来，前述各种逐点松弛法都是 CR 算法的特殊情形.

定理 1.1 若 $A > 0, \omega \in (0,2)$，则 CR 算法收敛.

证. 根据定义，当 $i \to \infty$ 时，对于 $\boldsymbol{x}$ 的每一分量 $x_j (j = 1, 2, \cdots, n)$，松弛的标号 $i_j(j)$ 都趋向无穷大，所以

$$\lim_{i \to \infty} J(\boldsymbol{y}^{i(j)}) = J(0).$$

若不然，设 $\lim_{i \to \infty} J(\boldsymbol{y}^{i(j)}) = J(\tilde{\boldsymbol{y}})$，但 $\tilde{\boldsymbol{y}} = \boldsymbol{x}^* - \tilde{\boldsymbol{x}} \neq 0$，那么残向量 $\tilde{\boldsymbol{r}}$ 必至少有一个分量 $\tilde{r}_j \neq 0$. 因为 $i_j(j) \to \infty$，故这时必在相应于 $\tilde{r}_j$ 的某个 $i_j(j)$ 处还可作松弛

$$\boldsymbol{x}^{i_j(j)} = \tilde{\boldsymbol{x}} + \omega \frac{\tilde{r}_j}{a_{jj}} \boldsymbol{e}_j,$$

使得

$$J(\boldsymbol{y}^{i_j(j)}) < J(\tilde{\boldsymbol{y}}).$$

且因 $\boldsymbol{x}^{i_j(j)} \in \{\boldsymbol{x}^{i(j)}\}$，故推出矛盾.

对于逐块松弛法，仿照上面的作法将其近似解序列 $\{X^{i(j)}\}$ 的标号序列 $\{i(j)\}$ 分成 m 个子序列

$$\{i_1(1)\}, \{i_2(2)\}, \cdots, \{i_j(j)\}, \cdots, \{i_m(m)\}.$$

它们与子序列

$$\{X^{i_1(1)}\}, \{X^{i_2(2)}\}, \cdots, \{X^{i_j(j)}\}, \cdots, \{X^{i_m(m)}\}$$

是一一对应的.

定义 1.4 逐块松弛法称为逐块混乱松弛法 (BCR)，如果当 $i \to \infty$ 时，

$$i_j(j) \to \infty, \forall j \in \{1, 2, \cdots, m\}.$$

类似地也有

定理 1.2 若 $A > 0, \omega \in (0, 2)$，则 BCR 算法收敛.

值得指出的是，BCR 算法是概括性很广的一类算法. 由于三个任意性——分块的任意性、松弛因子的任意性(既可固定，又可在区间(0,2)内随着松弛次数 i 的变化而变化)及松弛顺序的任意性，所以现有的解具有正定系数矩阵的线性代数方程组的各种

迭代法，几乎均可看作 *BCR* 算法之特例. 特别是松弛顺序的任意性，恰好为建立一类异步并行算法提供了理论依据.

作为例子，我们试根据 *BCR* 算法设计出解方程组 (1.6) 的一个异步并行程序：

设由 m 台处理机 (m 个过程) $P_1, P_2, \cdots, P_j, \cdots, P_m$ 来解问题 (1.6). 整体变量定义为

$$\begin{cases} X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_j, \cdots, \boldsymbol{x}_m); \\ EPS = (\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_j, \cdots, \varepsilon_m); \\ E. \end{cases}$$

$\boldsymbol{x}_j$ 与 ε_j 由 $P_j (j = 1, 2, \cdots, m)$ 负责更新数据，E 指定由 P_m 负责更新数据. E 之初值可取为 1 (不得为零)，其余整体变量之初值可全置零.

process P_j $\quad (j = 1, 2, \cdots, m)$;

begin

if $E > \varepsilon$ **then**

begin

$$\omega * A_{jj}^{-1}\left(f_j - \sum_{k=1}^{m} A_{jk}\boldsymbol{x}_k\right) \to T_j;$$

$$\|T_j\| \to \varepsilon_j;$$

$$\boldsymbol{x}_j + T_j \to \boldsymbol{x}_j$$

$$(\|EPS\| \to E, if \quad j = m)$$

end

end

m 个过程 $P_j (j = 1, 2, \cdots, m)$ 异步并行运行，各过程的运行速度可以不同，但定理 1.2 保证了它们的收敛性.

混乱松弛法的定义很多[3-7]，我们认为本书给出的定义要比其他文献上见到的定义简单、明了而且实用一些. 细心的读者不难从上面的例子及其说明中进一步察觉我们的定义里所要求的条件的实质：所谓当 $i \to \infty$ 时有 $i_j(j) \to \infty$, $\forall j \in \{1, 2, \cdots, m\}$，无非是说只要在计算过程中任一处理机 (过程) 都不至于中断到"死