计算方法丛书

无约束最优化计算方法

邓乃扬 等 著

科学出版社

计算方法丛书

无约束最优化计算方法

邓乃扬 等 著

科 学 出 版 社

1982

内 容 简 介

本书讨论处理无约束最优化问题的数值方法，主要包括 Newton 法、共轭梯度法、拟 Newton 法、Powell 直接方法以及非线性最小二乘法，并且阐明了其理论、应用和发展动向。可供计算数学工作者、工程技术人员、高等院校有关专业高年级学生、研究生及教师参考。

计算方法丛书

无约束最优化计算方法

邓乃扬 等 著

责任编辑 向安全 张鸿林

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

*

1982 年 12 月第 一 版 开本：850×1168 1/32

1982 年 12 月第一次印刷 印张：10 1/16

印数：0001—7,300 字数：264,000

统一书号：13031·2106

本社书号：2877·13—1

定价：1.90 元

序

最优化计算方法是计算数学中的一个十分活跃的分支．随着电子计算机的普及，它已广泛应用于化工、航空、机械、建筑、无线电技术等许多工程技术部门．另外在生产组织、资源分配等管理科学方面，最优化方法也已成为一种重要的决策手段．

最优化计算方法通常可分为两类，即无约束最优化和约束最优化．本书介绍的是前者．无约束最优化计算方法不仅本身有着不少实际应用，而且与约束最优化计算方法也有着紧密的联系：一方面，有些处理无约束问题的方法能够直接推广而且应用于约束问题；另一方面，还可以把一些约束问题转化为无约束问题来处理．因此从这个意义上讲，无约束最优化计算方法也是处理约束最优化问题的基本方法．

本书兼顾实际应用与理论研究两个方面，介绍处理无约束最优化问题的各种方法，其中包括使用目标函数二阶导数值的 Newton 法（第三章）、使用目标函数一阶导数值的共轭梯度法（第四章）和拟 Newton 法（第五章）、仅用目标函数值的直接方法（第六章）以及非线性最小二乘法（第七章）．上述各种方法虽然都要用到第一、二两章的知识，但它们之间基本上是各自独立的．因此仅对其中某种方法感兴趣的读者，可以在读完第一、二章后直接阅读讲述该方法所在的章节．

为了实际应用方便，我们对所推荐的各方法都给出了计算框图；同时为满足关心理论研究的读者的需要，也对各方法的收敛性等理论问题进行了比较深入的探讨．当然这些内容对于仅仅关心实际应用的读者可以略去．

参加本书编写工作的还有诸梅芳、刘德辅、杨振海、王理、唐云等同志．另外当时的研究生陈志、高旅端、史明仁、颜铁成、吴育华、唐恒永、吴振奎、杨燕昌也做了许多工作．北京工业大学应用数学系领导亦曾给予大力支持．但由于作者水平所限，缺点与错误在所难免，请批评指正．

作者

《计算方法丛书》编委会

目　　录

第一章 概 论

§ 1. 无约束最优化

1.1. 无约束最优化问题

设 $f(\boldsymbol{x})$ 是一个定义在 n 维欧氏空间 R^n 上的函数. 我们把寻找 $f(\boldsymbol{x})$ 的极小点的问题称为一个无约束最优化问题. 这个问题可以用下列形式表示:

$$\min f(\boldsymbol{x}), \quad \boldsymbol{x}=(x_1, \cdots, x_n)^T \in R^n, \tag{1.1}$$

其中 $f(\boldsymbol{x})$ 称为目标函数.

我们知道,函数的极小点有两种: 局部极小点和整体极小点. 下面分别给出它们的定义:

定义 1. 若对于 $\boldsymbol{x}^* \in R^n$, 存在着 $\varepsilon>0$, 使得当 $\|\boldsymbol{x}-\boldsymbol{x}^*\|<\varepsilon$ 时,总有

$$f(\boldsymbol{x}) \geqslant f(\boldsymbol{x}^*), \tag{1.2}$$

则称 $\boldsymbol{x}^*$ 是 $f(\boldsymbol{x})$ 的局部极小点. 若当 $\|\boldsymbol{x}-\boldsymbol{x}^*\|<\varepsilon$ 但 $\boldsymbol{x} \neq \boldsymbol{x}^*$ 时, 式 (1.2) 的不等号恒成立, 则称 $\boldsymbol{x}^*$ 是 $f(\boldsymbol{x})$ 的严格局部极小点.

定义 2. 若存在着 $\boldsymbol{x}^* \in R^n$, 使得对任意的 $\boldsymbol{x} \in R^n$, 式 (1.2) 都成立,则称 $\boldsymbol{x}^*$ 是 $f(\boldsymbol{x})$ 的整体极小点. 若当 $\boldsymbol{x} \neq \boldsymbol{x}^*$ 时,式(1.2)的不等号恒成立,则称 $\boldsymbol{x}^*$ 是 $f(\boldsymbol{x})$ 的严格整体极小点.

虽然实际计算中所关心的往往是整体极小点, 但是现有的算法常常只能保证近似地求出局部极小点, 所以我们规定求这两种极小点都属于无约束最优化问题. 对于寻求整体极小点的问题, 目前最流行的方法是多次使用通常求解无约束最优化问题的算法,设法多找出几个局部极小点,然后把其中取值最小的那个极小

点近似地作为整体极小点．本书仅限于讨论这些通常求解无约束最优化问题的算法．但是应该指出，关于寻求整体极小点的问题，已经有了不少专门的研究，有兴趣的读者可参看文献[14]—[17]．

顺便说明一点，由于 $f(\boldsymbol{x})$ 的极大点对应于 $-f(\boldsymbol{x})$ 的极小点，因而无约束最优化计算方法同样适用于求函数的极大点问题．

1.2. 极小点的基本性质

我们在这里给出极小点的必要及充分条件(证明从略)．

定理 1(极小点的一阶必要条件)．设 $f(\boldsymbol{x})$ 是 R^n 上的连续可微函数．若 $\boldsymbol{x}^*$ 为其局部极小点，则 $\boldsymbol{x}^*$ 必为 $f(\boldsymbol{x})$ 的稳定点，即

$$\boldsymbol{g}(\boldsymbol{x}^*)=\nabla f(\boldsymbol{x}^*)=\boldsymbol{0},$$

其中 $\boldsymbol{g}(\boldsymbol{x})=\nabla f(\boldsymbol{x})$ 为 $f(\boldsymbol{x})$ 的梯度

$$\boldsymbol{g}(\boldsymbol{x})=\nabla f(\boldsymbol{x})=\left(\frac{\partial f(\boldsymbol{x})}{\partial x_1},\cdots,\frac{\partial f(\boldsymbol{x})}{\partial x_n}\right)^T.$$

定理 2(极小点的二阶必要条件)．设 $f(\boldsymbol{x})$ 是 R^n 上的二次连续可微函数．若 $\boldsymbol{x}^*$ 为其局部极小点，则它必为 $f(\boldsymbol{x})$ 的稳定点，且 $f(\boldsymbol{x})$ 在点 $\boldsymbol{x}^*$ 处的 Hessian 矩阵 $G(\boldsymbol{x}^*)=\nabla^2 f(\boldsymbol{x}^*)=\left(\frac{\partial^2 f(\boldsymbol{x}^*)}{\partial x_i\,\partial x_j}\right)$ 半正定．即 $\boldsymbol{g}(\boldsymbol{x}^*)=\nabla f(\boldsymbol{x}^*)=\boldsymbol{0}$ 且对任意的 $\boldsymbol{p}\in R^n$，有

$$\boldsymbol{p}^T G(\boldsymbol{x}^*)\boldsymbol{p}\geqslant 0.$$

定理 3(极小点的二阶充分条件)．设 $f(\boldsymbol{x})$ 是 R^n 上的二次连续可微函数．若在点 $\boldsymbol{x}^*$ 处有

i) $\boldsymbol{g}(\boldsymbol{x}^*)=\nabla f(\boldsymbol{x}^*)=\boldsymbol{0}$；

ii) $G(\boldsymbol{x}^*)=\nabla^2 f(\boldsymbol{x}^*)$ 正定，

则 $\boldsymbol{x}^*$ 是 $f(\boldsymbol{x})$ 的严格局部极小点．

这几个定理表明，极小点和稳定点之间有着十分密切的关系．由于很难直接验证一个点是不是极小点，我们常常通过检验稳定点的办法来鉴别它．

1.3. 历史简述

无约束最优化是一个十分古老的课题，至少可以追溯到 Newton 发明微积分的时代. 根据定理 1，能够把问题(1.1)化为求解方程组

$$\frac{\partial f(x)}{\partial x_1} = \cdots = \frac{\partial f(x)}{\partial x_n} = 0, \tag{1.3}$$

这是微积分中常用的方法. 另外早在 1847 年，Cauchy[18] 就提出了最速下降法. 也许这些就是最早的求解无约束最优化问题的方法. 对于变量不多的某些问题，这些方法是可行的. 但对于变量较多的一般问题，就常常不适用了. 其原因是除某些特殊情形外，方程组(1.3)是非线性的，对它求解十分困难；而最速下降法往往又收敛得很慢. 但是在以后的很长一段时间内，这一古老的课题一直没有取得实质性的进展. 只是近二十多年来，由于电子计算机的应用和实际需要的增长，才使这一古老的课题获得了新生. 人们除了使用最速下降法[19]外，还使用并发展了 Newton 法(例如文献[20,21]等). 同时也出现了一些从直观几何图象导出的搜索方法(例如文献[22—24]等). 由 Davidon[25] 发明的变度量法（本书称为拟 Newton 法)，是无约束最优化计算方法中最杰出的、最富有创造性的工作. Broyden, Powell, Fletcher 等人沿着这一方向做了大量的工作. 另外应该提及的两个方法是 Powell 直接方法[26]和共轭梯度法[27]，它们在无约束最优化计算方法中也占有十分重要的地位. 以上所述的方法都是针对一般目标函数设计的，对于平方和这种特殊形式的目标函数，Marquardt[28] 做了很好的工作. 总之，现在已经有了不少行之有效的新方法(对于这些方法的简要介绍可参看文献[29]).

在几十年前，人们简直不可能想像能求解具有几百个变量乃至上千个变量的问题，但在今天，这已经是司空见惯的了. 然而同时应该指出，这一领域中还有不少悬而未决的问题有待解决. 特

别是由于其应用范围日益广泛，更迫切地要求创造出更加有效、更加可靠的新方法．因此无约束最优化计算方法，至今仍是一个相当活跃的课题．

§2. 下降算法

2.1. 下降算法

求解无约束最优化问题的方法大都属于迭代法．许多迭代法都是根据下述思想建立的：我们并不期望一下子就找到函数的极小点，而是从某一点 x_1 出发，先找一个比 x_1 取值小一些的点 x_2，然后再找一个取更小函数值的点 x_3，……．最终希望能得到极小点或者能接近极小点．这类算法的特点是，每进行一步都要求函数值有所下降，因此称为下降算法．这里的关键问题是要根据目标函数 $f(x)$ 的某些信息，确立一个由 x_k 得到下一点 x_{k+1} 的规则，或者确定一个从 R^n 到 R^n 的映射 $a = a(x)$，令

$$x_{k+1} = a(x_k)$$

来给出这种规则．这样，只要有了映射 a，就可以构造如下的算法．

算法 1（下降算法 I）

1. 取初始点 x_1，置 $k = 1$．
2. 置 $x_{k+1} = a(x_k)$．
3. 若 $f(x_{k+1}) \geqslant f(x_k)$，则停止计算；否则置 $k = k + 1$，转 2．

在上述算法中，由 x_k 到 x_{k+1} 的规则是由点到点的映射 a 给出的．有时我们也用点到点的集合的映射 $A = A(x)$ 给出这种规则——这里 A 把 R^n 中的点映射到 R^n 中的非空子集．与此对应，可以建立下列算法模型．

算法 2（下降算法 II）

1. 取初始点 x_1，置 $k = 1$．
2. 从集合 $A(x_k)$ 中取出一点 y．
3. 置 $x_{k+1} = y$．

4. 若 $f(\boldsymbol{x}_{k+1}) \geqslant f(\boldsymbol{x}_k)$，则停止计算；否则置 $k=k+1$，转 2.

为了形象地说明上述两个算法，我们讨论只有两个自变量的目标函数的情形．设

$$z = f(\boldsymbol{x}),\ \boldsymbol{x} = (x_1, x_2)^T. \tag{1.4}$$

它代表一个曲面．这个曲面及其等高线的性态如图 1.1 所示．我们的目标是找出该曲面的最低点．设已经取定了某个初始点 $\boldsymbol{x}_1$．可以考虑在 (x_1, x_2) 平面上选一适当方向 $\boldsymbol{p}_1$．试着沿这个方向找出取值更低一些的点 $\boldsymbol{x}_2 = \boldsymbol{x}_1 + \lambda \boldsymbol{p}_1$（这里 λ 是一个纯量）．如果真能找到这样的 $\boldsymbol{x}_2$，使得

$$f(\boldsymbol{x}_2) < f(\boldsymbol{x}_1), \tag{1.5}$$

我们就前进了一步．以下可以继续从 $\boldsymbol{x}_2$ 出发，再选适当方向 $\boldsymbol{p}_2$，求得 $\boldsymbol{x}_3$．一般来说，从 $\boldsymbol{x}_k$ 出发，选择适当方向 $\boldsymbol{p}_k$，即可求得 $\boldsymbol{x}_{k+1}$．方向 $\boldsymbol{p}_k$ 称为搜索方向．容易想到，在沿搜索方向 $\boldsymbol{p}_k$ 寻找 $\boldsymbol{x}_{k+1}$ 时，不应仅仅满足于使函数值下降，而应设法使函数值下降得尽可能多一些．换句话说，最好以 $f(\boldsymbol{x})$ 在方向 $\boldsymbol{p}_k$ 上的极小点作为 $\boldsymbol{x}_{k+1}$．这样的做法实际上是算法 1 的一个特殊情况．其中的映射 $a = a(\boldsymbol{x})$ 是按下述方式给出的：对 R^n 中任一点 $\boldsymbol{x}$，选定一适当方向 $\boldsymbol{p} = \boldsymbol{p}(\boldsymbol{x})$，然后从点 $\boldsymbol{x}$ 出发，找出 $f(\boldsymbol{x})$ 在方向 $\boldsymbol{p}$ 上的（第一个）极小点，以此作为 $\boldsymbol{x}$ 的映象 $a(\boldsymbol{x})$．

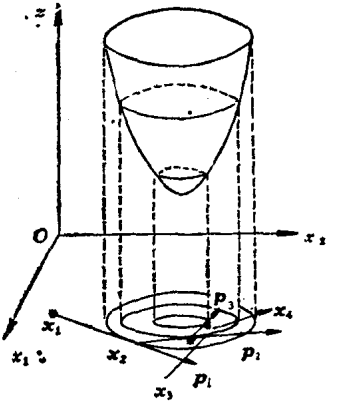

图 1.1　对应于二元函数的下降算法

如果对 R^n 中的点 $\boldsymbol{x}$ 来说，所选的方向 $\boldsymbol{p} = \boldsymbol{p}(\boldsymbol{x})$ 可能有多个，即 $\boldsymbol{p} = \boldsymbol{p}(\boldsymbol{x})$ 可取某一 n 维向量集合 $P = P(\boldsymbol{x})$ 中的任意元素，则上述做法对应于算法 2．一般来说，从点 $\boldsymbol{x}$ 出发，沿 $P(\boldsymbol{x})$ 中的每一个方向 $\boldsymbol{p}(\boldsymbol{x})$ 都能找到 $f(\boldsymbol{x})$ 在该方向上的极小点，这些极小点组成的集合就是 $A(\boldsymbol{x})$．因此 $A(\boldsymbol{x})$ 可形式地记作

$$A(\boldsymbol{x}) = \{\boldsymbol{y} = \boldsymbol{x} + \lambda^* \boldsymbol{p} \mid \boldsymbol{p} \in P(\boldsymbol{x});\ f(\boldsymbol{x} + \lambda^* \boldsymbol{p}) = \min_{\lambda > 0} f(\boldsymbol{x} + \lambda \boldsymbol{p})\}$$

所谓"从集合 $A(\boldsymbol{x}_k)$ 中取出一点 $\boldsymbol{y}$" 就相当于从 $P(\boldsymbol{x}_k)$ 中选取一个方向 $\boldsymbol{p}_k = \boldsymbol{p}(\boldsymbol{x}_k)$，然后求出 $f(\boldsymbol{x})$ 沿方向 $\boldsymbol{p}_k$ 的极小点，以该极小点作为 $\boldsymbol{y}$.

以上对于二维问题的分析告诉我们，借助于求 $f(\boldsymbol{x})$ 在某个方向上的极小点有可能加快整个计算的过程. 而求 $f(\boldsymbol{x})$ 在某个方向上的极小点是求一维线性流形[1]上的极小点问题，这样的问题称为一维搜索. 下面给出一个使用一维搜索的算法模型.

算法 3（使用一维搜索的下降算法）

1. 取初始点 $\boldsymbol{x}_1$，置 $k = 1$.

2. 选择适当的搜索方向 $\boldsymbol{p}_k$.

3. 一维搜索：求 $f(\boldsymbol{x})$ 在一维线性流形

$$\mathscr{B} = \{\boldsymbol{x} | \boldsymbol{x} = \boldsymbol{x}_k + \lambda \boldsymbol{p}_k, \ -\infty < \lambda < \infty\}$$

上的极小点. 以该极小点作为 $\boldsymbol{x}_{k+1}$.

4. 若 $f(\boldsymbol{x}_{k+1}) \geqslant f(\boldsymbol{x}_k)$，则停止计算；否则置 $k = k + 1$，转 2.

这类算法的基点是把高维极小问题转化为一维极小问题. 当然并不一定要把后者局限于一维，也可以用寻求某个低维线性流形上的极小点来代替求一维极小. 例如要把 n 维空间的问题 (1.1) 转化为 m 维线性流形上的问题 $(1 < m < n)$，可用下列步骤取代算法 3 中的 3.

3′. 低维搜索：当 $k < m$ 时，以 $f(\boldsymbol{x})$ 在 k 维线性流形

---

1) 设 $\boldsymbol{x}_1 \in R^n$, $\boldsymbol{d}_1, \cdots, \boldsymbol{d}_m \in R^n$ $(m \leqslant n)$，则称集合

$$\mathscr{B}_m = \{\boldsymbol{x} | \boldsymbol{x} = \boldsymbol{x}_1 + \sum_{i=1}^{m} \lambda_i \boldsymbol{d}_i, \ -\infty < \lambda_i < \infty\}$$

为经过 $\boldsymbol{x}_1$ 由 $\boldsymbol{d}_1, \cdots, \boldsymbol{d}_m$ 张成的线性流形，它的维数就是 $\boldsymbol{d}_1, \cdots, \boldsymbol{d}_m$ 中线性无关向量的最大个数. 特别地，当 $\mathscr{B}_m$ 包含原点且 $\boldsymbol{d}_1, \cdots, \boldsymbol{d}_m$ 线性无关时，它就是 R^n 中的一个 m 维子空间.

仿定义 1，若对于 $\boldsymbol{x}^* \in \mathscr{B}_m$ 存在着 $\varepsilon > 0$，使得当 $\boldsymbol{x} \in \mathscr{B}_m$ 且 $\|\boldsymbol{x} - \boldsymbol{x}^*\| < \varepsilon$ 时，总有 $f(\boldsymbol{x}) \geqslant f(\boldsymbol{x}^*)$，则称 $\boldsymbol{x}^*$ 是 $f(\boldsymbol{x})$ 在 $\mathscr{B}_m$ 上的局部极小点. 若当 $\boldsymbol{x} \in \mathscr{B}_m$ 且 $\|\boldsymbol{x} - \boldsymbol{x}^*\| < \varepsilon$ 但 $\boldsymbol{x} \neq \boldsymbol{x}^*$ 时，恒成立着 $f(\boldsymbol{x}) > f(\boldsymbol{x}^*)$，则称 $\boldsymbol{x}^*$ 是 $f(\boldsymbol{x})$ 在 $\mathscr{B}_m$ 上的严格局部极小点.

同样地，$f(\boldsymbol{x})$ 在 $\mathscr{B}_m$ 上的整体极小点和严格整体极小点的定义也与定义 2 相仿.

$$\mathscr{B}_k = \left\{ x \mid x = x_k + \sum_{j=1}^{k} \lambda_j p_j, \ -\infty < \lambda_j < \infty \right\}$$

上的极小点为 x_{k+1}；当 $k \geqslant m$ 时，以 $f(x)$ 在 m 维线性流形

$$\mathscr{B}_{m(k)} = \left\{ x \mid x = x_k + \sum_{j=k-m+1}^{k} \lambda_j p_j, -\infty < \lambda_j < \infty \right\}$$

上的极小点为 x_{k+1}.

超记忆下降法[30]就属于这类方法，看起来是有前途的. 但是目前尚未引起人们充分的注意，我们不准备进一步讨论它了. 本书主要介绍算法 3 所描述的算法类.

要把算法 3 发展成为切实可行的算法，至少还要解决两个问题：如何选择较好的搜索方向，以及怎样进行一维搜索. 在第二章中我们将专门讨论后一问题. 而对于前一问题的处理，则是最优化计算方法中的一个核心问题，解决这一问题的不同方式就形成了不同的方法，这正是后面各章要仔细研究的.

2.2. 下降算法的基本理论问题

对于任意一个求解无约束最优化问题的算法，我们自然要问：它所构造的序列 $\{x_k\}$ 能不能收敛到目标函数的极小点，以及其收敛的速度如何？这是一些最基本的理论问题. 现在引进几个与此有关的概念，以便给这些问题以精确的数学描述，为今后的讨论作些准备.

收敛性

定义 3. 若一算法对于某类目标函数来说，任给初始点 $x_1 \in R^n$，按该算法构造的序列 $\{x_k\}$ 总停止或者收敛到目标函数的一个极小点，则称该算法对该类函数具有全局收敛性.

定义 4. 若一算法对于某类目标函数来说，在其定义域的某个区域 C 上任取一点做为初始点 x_1，按算法构造的序列 $\{x_k\}$ 属于区域 C，并总停止或者收敛到这个区域内的一个极小点，则称该算法对该类函数具有区域 C 上的收敛性.

显然，全局的或者在某区域上的收敛性，是算法应该具有的一个重要性质．另外我们再引进一个希望算法具有的性质——二次终止性．考虑正定二次函数

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T G\boldsymbol{x} + \boldsymbol{r}^T\boldsymbol{x} + \delta \tag{1.6}$$

（其中G是$n \times n$阶正定对称矩阵，$\boldsymbol{r}$和δ分别是n维常向量和常数）．一方面，由于这类函数是具有极小点的最简单的函数类；另一方面，一般的函数在极小点附近大都可以用这类函数来近似，所以作为一个好的算法，常常要求它能够很有效地处理这类函数，最好经过有限步就能达到其极小点．因此我们有下列定义．

定义 5. 若某算法对于任意形如式（1.6）所示的二次函数来说，从任意初始点出发，都能在有限步内达到其极小点，则称该算法具有二次终止性．

有的文献上也把二次终止性称作二次收敛性．但本书不用这个名称．

收敛速率 在引进有关收敛速率的某些概念时，我们始终假定算法构造的序列$\{\boldsymbol{x}_k\}$是收敛的，即

$$\lim_{k\to\infty} \boldsymbol{x}_k = \boldsymbol{x}^*. \tag{1.7}$$

定义 6. 若对于序列$\{\boldsymbol{x}_k\}$来说，存在着$p \geqslant 0$，并存在着常数N和L，使得当$k \geqslant N$时，有

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leqslant L\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^p,$$

（其中$\|\cdot\|$可以是R^n中的任意范数，但为使用方便，不妨认为它指的是向量的l_2范数），则称序列$\{\boldsymbol{x}_k\}$收敛的级是p，或者说$\{\boldsymbol{x}_k\}$为p级收敛．

显然上述定义并不保证序列收敛的级唯一；事实上，如果某序列收敛的级是$p > 0$，那么任何不超过p的非负数，也都是它收敛的级．另外容易看出，收敛的级所描述的是一次迭代的进展情况．下列定义则对应于连续进行m次迭代的进展情况．

定义 7. 若对于序列$\{\boldsymbol{x}_k\}$来说，存在着常数N和L，使得当$k \geqslant N$时，有

$$\|x_{k+m}-x^*\| \leqslant L\|x_k-x^*\|^2,$$

则称序列 $\{x_k\}$ m 步二级收敛.

显然，一步二级收敛就是通常的二级收敛.

定义 8. 若对于序列 $\{x_k\}$ 来说，存在着 $\theta\in(0,1)$，并存在着 N 和 L，使得当 $k\geqslant N$ 时，有

$$\|x_k-x^*\| < L\theta^k,$$

则称序列 $\{x_k\}$ 线性收敛.

定义 9. 若对于序列 $\{x_k\}$ 来说，任给 $\beta>0$，都存在着 $N>0$，使当 $k\geqslant N$ 时，有

$$\|x_{k+1}-x^*\| \leqslant \beta\|x_k-x^*\|,$$

则称序列 $\{x_k\}$ 超线性收敛.

显然，线性收敛是超线性收敛的必要条件，而超线性收敛又是二级收敛的必要条件.

另外需要说明的一点是，以上的收敛速率都是对序列 $\{x_k\}$ 定义的. 我们规定，如果一算法对于某类目标函数所构造的任意序列都具有某种收敛性质，就称该算法对于那一类目标函数具有该收敛性质. 在这个意义上，我们可以说某算法收敛的级、某算法线性收敛等等.

为了帮助读者理解上述几个定义，我们看三个一维空间中的例子.

例 1. 考虑 $x_k=aq^k$ $(0<|q|<1,\ a\neq 0)$. 这是一个收敛到 $x^*=0$ 的序列. 因为

$$|x_{k+1}-0|/|x_k-0| = |q| < 1,$$

所以它是一级收敛的. 同时易见它也线性收敛.

例 2. 考虑 $x_k=(1/k)^k$. 显然 $\lim\limits_{k\to\infty} x_k=0$. 注意到

$$|x_{k+1}-0|/|x_k-0| = \left(\frac{k}{k+1}\right)^k\cdot\frac{1}{k+1}\to 0\ (k\to\infty),$$

即知它超线性收敛. 同时不难验证它收敛的级不高于 1.

例 3. 考虑 $x_k=q^{(2^k)}(0<|q|<1)$. 由

$$|x_{k+1}-0|/|x_k-0|^2 = 1,$$

可见它二级收敛.

很明显,上述关于收敛速率的概念仅仅涉及到当 $k \to \infty$ 时的渐近性质. 但实际使用任一算法进行计算时,都只能进行有限步. 这似乎会使人觉得它们并不能真正描述算法的实际效果. 然而事实并非如此. 根据上述渐近性质判断的收敛速率常与实际计算十分一致. 这表明它们确实是一些值得研究的重要性质.

2.3. 关于收敛性的两个定理

与收敛速率的研究不同,对于收敛性问题已经有了比较一般的理论和方法,即点集映射法和强函数法.要了解这方面的概貌可参阅[31]及该文所引的有关文献. 为给以后证明具体算法的收敛性作准备,这里介绍两个比较一般的定理. 它们都属于强函数法.

在讨论收敛性问题时,往往不是直接证明算法构造的序列 $\{x_k\}$ 收敛到极小点,而是先证明序列 $\{x_k\}$ 的极限点都是目标函数 $f(x)$ 的稳定点;然后再根据目标函数的性质,证明序列 $\{x_k\}$ 收敛到 $f(x)$ 的极小点. 下列两个定理可用于实现第一步. 事实上,当目标函数 $f(x)$ 连续可微时,只要把定理中的 Ω^* 取为 $f(x)$ 的稳定点组成的集合,就能达到上述目的.

定理 4. 考虑把算法 2 应用于 R^n 上的连续函数 $f(x)$. 设 Ω^* 是 R^n 中的一个子集. 若对于 R^n 中不属于 Ω^* 的任意点 x 来说,都存在着 $\varepsilon = \varepsilon(x) > 0$ 和 $\delta = \delta(x) < 0$,使得当 $\|x' - x\| < \varepsilon(x)$, $y' \in A(x')$ 时,

$$f(y') - f(x') \leqslant \delta(x),$$

则算法 2 构造的序列 $\{x_k\}$ 满足

i) 当 $\{x_k\}$ 为有穷序列 $\{x_1, \cdots, x_{m-1}, x_m\}$ 时, $x_{m-1} \in \Omega^*$;

ii) 当 $\{x_k\}$ 为无穷序列时,它的任一极限点 $\hat{x} \in \Omega^*$.

证明. 先考虑 $\{x_k\}$ 为有穷序列 $\{x_1, \cdots, x_{m-1}, x_m\}$ 的情形. 此时按算法构造有

$$x_m \in A(x_{m-1}),\ f(x_m) \geqslant f(x_{m-1}).$$