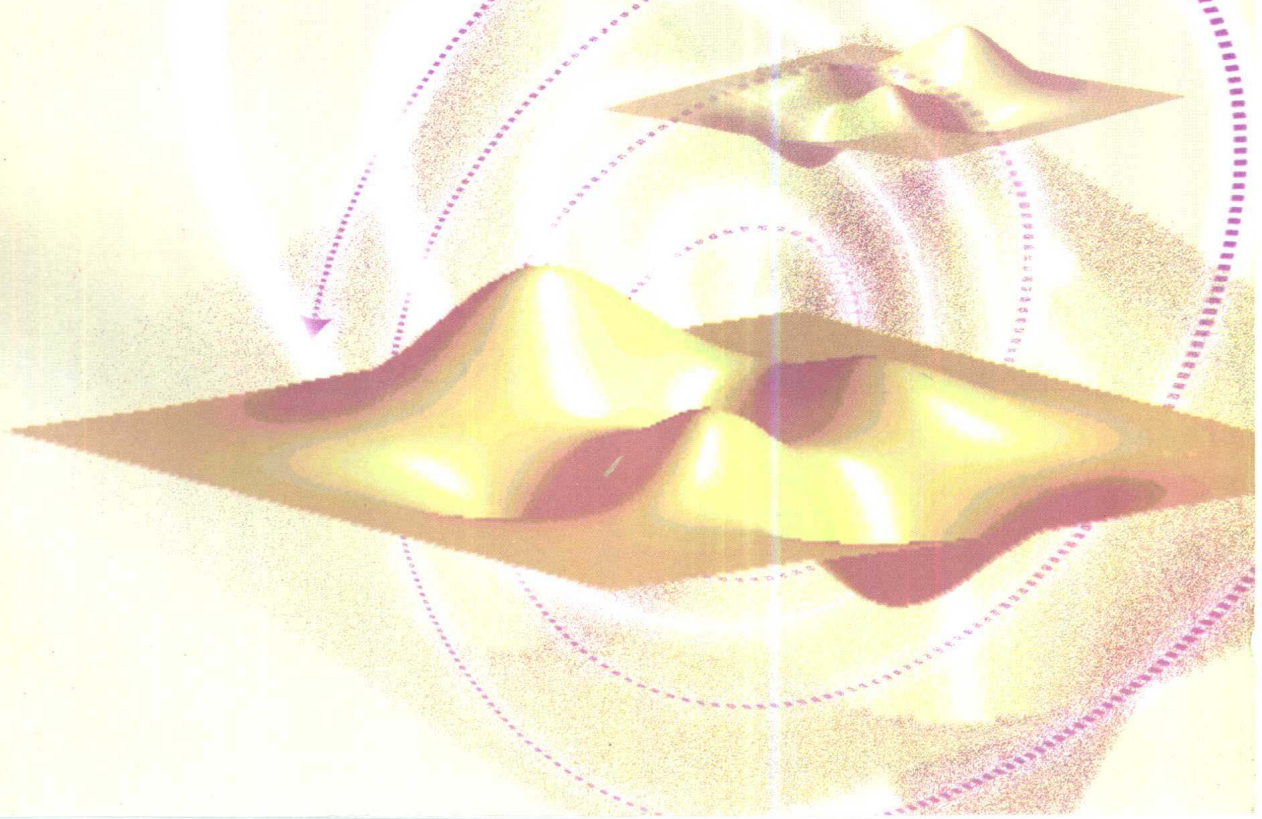
研究生教材

最优化计算原理与算法程序设计

粟塔山 等编著



技大学出版社

最优化计算原理与算法程序设计

粟塔山　彭维杰
周作益　曾之华　编著

国防科技大学出版社
湖南·长沙

内 容 提 要

最优化计算是工程设计和管理决策科学的常用工具。本书包括无约束优化计算、约束优化计算、动态优化计算三部分。书中详细介绍了它们的算法原理和计算步骤。本书强调算法的程序设计，为读者提供了颇有价值的经验和技巧，并对一些公用模块给出了C++源程序。

本书可作为工程、管理类研究生、应用数学专业高年级学生的教材或教学参考书。

图书在版编目(CIP)数据

最优化计算原理与算法程序设计/粟塔山等编著. 长沙：国防科技大学出版社，2001.2
ISBN 7-81024-716-6

Ⅰ.最… Ⅱ.粟… Ⅲ.①最优化算法-高等学校-教学参考资料②最优化算法-算法设计-高等学校-教学参考资料 Ⅳ.0242.23

中国版本图书馆CIP数据核字(2000)第73980号

国防科技大学出版社出版发行
电话：(0731)4555681 邮政编码：410073
E-mail：gfkdcbs@public.cs.hn.cn
责任编辑：文 慧 责任校对：何 晋
新华书店总店北京发行所经销
国防科技大学印刷厂印装
*
787×1092 1/16 印张：14.75 字数：341千
2001年1月第1版第1次印刷 印数：1—3000册
*
定价：20.00元

前　言

最优化方法是一门古老而又年青的学科。这门学科的源头可以追溯到法国数学家拉格朗日关于一个函数在一组等式约束条件下的极值问题。伴随着工业、军事技术和管理决策科学的发展，这门学科也在不断丰富发展它的内涵，衍生出组合优化，线性规划，非线性规划，动态规划，最优控制等分枝。拉格朗日乘子法则、库恩塔克条件、庞特里雅金极大值原理、贝尔曼最优化方程，奠定了优化理论研究发展的里程碑。这些经典的优化理论着重描述了最优解的特征。但是，直到有了高速计算机，人们才能够对各类较大规模的优化问题利用计算机实施求解，使最优化方法成为工程设计、决策管理的一种实用工具。

几乎所有类型的优化问题都可概括为这样的数学模型：给定一个集合（称为可行集）和该集合上定义的实值函数（称为目标函数），要计算函数在集合上的极值。通常，人们按照可行集的性质对优化问题进行分类：如果可行集中的元素是有限的，则归结为“组合优化”或“网络规划”，如图论中最短路径、最小费用最大流、最大权匹配等；如果可行集是有限维空间中的一个连续子集，则归结为线性或非线性规划问题；如果可行集中的元素是依赖于时间的决策序列，则归结为“动态规划”；如果可行集是无穷维空间中的连续子集（集合中的元素是有限维空间中的一条曲线，由一组常微分方程描述，而目标函数为一定积分），则归结为“最优控制问题”。当然，这样的划分不是绝对的，不论是描述问题或是计算求解，这些分支都有一定的联系。网络规划的许多问题都可表示为线性规划；而当今流行的“内点算法”则用非线性规划的方法来求解线性规划。最优控制中的许多算法都可以在非线性规划中找到它们的影子。

一般说来，各优化分支有其相应的应用领域（但不是绝对的）。线性规划、网络规划、动态规划更多地用于管理与决策科学；非线性规划更多地用于工程优化设计；最优控制常用于控制工程。作为一本主要面向工程类研究生的教材，囿于40学时的教学时数，《最优化计算原理与算法程序设计》主要介绍了非线性规划的理论和算法，并扼要地介绍了动态规划的基本原理以及最优控制问题的数值方法。

非线性规划是在一组等式和不等式约束条件下，求一个函数的极值问题。1951年，库恩（H.W.Kuhn）和塔克（A.W.Tucker）等人提出了非线性规划的最优性条件，为其发展奠定了理论基础。随着计算机的发展和应用，各种非线性规划算法应运而生。最著名的算法包括DFP（Davidon-Fletcher-Powell）和BFGS（Brogdew-Fletcher-Goldfarb-Shanno）无约束变尺度法、HP（Hestenes-Powell）广义乘子法、WHP（Wilson-Han-Powell）约束变尺度法。上述这些算法都是针对计算目标函数的局部极小点。近十年来，全局优化算法渐露头角，提出了较为成功的填充函数法。当然，全局优化的理论目前还很不成熟，算法也只是处于实验性的阶段。本书对上述诸算法均给出了比较详细的介绍，其中，关于全局优化方面的内容，目前国内的教材很少涉及。

作为工程类的研究生学习最优化方法，主要着重两方面——最优性条件与算法步骤。

如果说最优性条件指明了一次旅行要到达的目的地，那么，算法步骤则指导我们如何一步一个脚印向目的地进发。本书在描述这些内容时，时刻考虑到大部分工程类研究生的数学基础，为读者作了尽可能细致的铺垫。图文并茂的叙述方式，生动、直观而不失严谨。这是一本既可用于课堂讲授又适宜于自学的教材。

最优化方法是一门工具性的课程，仅仅理解它的内容是不够的。只有将那些算法变成高质量的计算机程序，这类工具才能为人们广泛利用。按照一张精美的家俱图纸打造出美观实用的家俱，要靠木匠师傅的技艺；由算法步骤到高质量的模块化结构的计算机程序同样需要创造性的劳动。本书向读者提供了许多值得借鉴的编程经验、教训和技巧，这些经验能让人少走弯路。对某些关键性的“算法构件”，本书还为读者提供了值得参考的源程序。

这是一本有特色的教材，我乐意将它推荐给广大读者。

沙基昌

2000 年 10 月 30 日

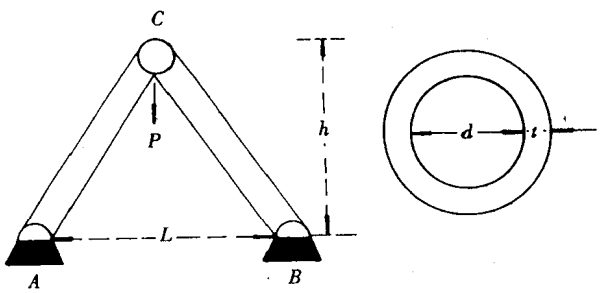
导　读

· 什么是优化计算？
· 本书是否适合你？
· 为什么要自己编程？
· 本书内容的安排及风格。
· 致使用 C++ 程序语言的读者.

什么是优化计算？

当人们刚开始做一件不太熟悉的事情，他或许只要能把事情办成就感到满意了。当他做这件事有了些经验，就会考虑如何把事情做得更好(假设他是爱岗敬业的)。追求完美是人类社会进步的动力之一。早年设计一辆轿车，主要考虑能安全运行；后来进一步要求如何运行得更平稳，如何节能；现在还希望内部装修舒适，外部造型优美。一言以蔽之，“把事情做得更好”——这就是优化。

优化计算则是工程优化设计和社会经济活动优化管理中使用的一种数学工具。在一项工程设计中，可以由人选择的设计因素称为**设计变量**，用 $X=(x_1\ x_2\cdots\ x_n)^T$ 表示。例如，两杆桁架是建筑工程的常用构件，它由两根等长的空心钢管在 C 处铰结，再固定 A，B 两端。如下图(右边是空心钢管剖面图)所示。



技术要求是：当 C 点承受压力 P 时，钢管不出现屈曲和塑性变形。桁架的跨度 L 已经被工作环境所确定，不可选择。能够选择的是桁架的高度 h、钢管的内径 d 和厚度 t，这里 h、d 和 t 就是设计变量。为使 C 点能承受拉力 P，t 和 d 不能太小，要使桁架实用，h 也不能太低(假设最低限制为 H_0)。对设计变量的限制称为**约束条件**。对于两杆桁架问题，由材料力学的知识，钢管不出现屈曲和塑性变形可以表示为关于 t、d、h 的两个非线性不等式(具体的表达式就不列出了)，再考虑到自然的非负约束，于是，对 t、d、h 的限制可表达为

$$g_1(t\ d\ h)\leqslant 0$$
$$g_2(t\ d\ h)\leqslant 0$$

$$t \geqslant 0, \quad d \geqslant 0, \quad h \geqslant H_0$$

满足这一组不等式的 t、d、h 是不惟一的。实际上，只要让钢管尽量粗大，桁架适当地高，总是能满足技术要求的。但是，这只是一种可行的设计，而不是优化设计。“优化”是针对某种“愿望”(或称为目标)而言的。假设我们的“愿望”是在满足技术要求的前提下让桁架的重量最小。桁架的重量完全由 t, d, h 确定，可表示为 t, d, h 的函数

$$f(t\ d\ h)$$

称为**目标函数**。我们的任务就是在满足约束条件(不等式组)的所有设计方案($t\ d\ h$)中找出一组方案($t^*\ d^*\ h^*$)，它使得目标函数取最小值。表示为

$$\begin{aligned} &\min f(t\ d\ h) \\ &\text{s.t}\ \ g_1(t\ d\ h) \leqslant 0 \\ &\qquad g_2(t\ d\ h) \leqslant 0 \\ &\qquad t \geqslant 0, \quad d \geqslant 0, \quad h \geqslant H_0 \end{aligned}$$

“s.t”的意思是“受约束于”。称($t^*\ d^*\ h^*$)为“最优设计方案”或最优解。

设计变量、约束条件、目标函数构成了优化设计的三要素。大部分工程优化设计问题都可表示为在一组约束条件下求一个目标函数的极值问题

$$\begin{aligned} &\text{opt}\ f(x_1\ x_2 \cdots\ x_n) \\ &\text{s.t}\ \ g_i(x_1\ x_2 \cdots\ x_n) \leqslant 0 \quad (i = 1, \cdots, m) \\ &\qquad h_j(x_1\ x_2 \cdots\ x_n) = 0 \quad (j = 1, \cdots, l) \end{aligned}$$

“opt”意即“优化(optimal)”，可能是“max”，也可能是“min”，依实际问题而定(对目标函数添加负号即可相互转化)。m 个不等式 $g_i(x_1\ x_2 \cdots\ x_n) \leqslant 0$ $(i = 1, \cdots, m)$称为不等式约束条件，l 个等式 $h_j(x_1\ x_2 \cdots\ x_n) = 0$ $(j = 1, \cdots, l)$称为等式约束条件。满足全体约束条件的 n 维向量 $X = (x_1\ x_2 \cdots\ x_n)^{\mathrm{T}}$构成 n 维空间 R^n 中的子集。

$$\Omega = \{X \in R^n \mid g_i(X) \leqslant 0, h_j(X) = 0, i = 1, \cdots, m; j = 1, \cdots, l\}$$

称为可行集。可行集中使目标函数达到“最优”的点称为该问题的最优解。如果 $\Omega = R^n$，称为无约束优化问题，否则称为约束优化问题。所谓优化计算，就是求出这个问题的最优解。读者能够意识到，不使用计算机优化计算寸步难行。

本书是否适合你?

希望掌握优化计算这个数学工具的读者(通常是工程类研究生)，可能分为三类：

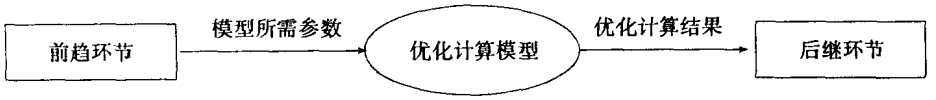
- 不愿意劳神费力去了解优化计算的数学原理，只要能熟练使用一些优化计算的数值软件，如 MATLAB 优化工具箱，或者 LINGO 等(使用盗版不算过错)，现学现用，轻松上手，解决自己的问题。
- 希望大致明白优化计算的数学原理，了解各种算法的长短处及适用范围，对计算结果有较敏锐的洞察力，通过对计算结果的合理分析从中得到更多的信息，让自己成为一个素养更高的优化工具使用者。但是，不打算自己去编制算法程序。
- 希望透彻地了解优化计算的数学原理，详细掌握各种算法的计算步骤，由自己编制出质量上乘的优化计算软件。

如果你是第三类读者，这本书正是为你写的。如果你是第一类读者，另有一个优化计算软件供你选择：这是作者自己编制的一个优化计算软件，计算效果优于 MATLAB 优化工具箱和 LINGO。你无需读任何文字资料，5 分钟即可上手。运行环境是 Windows95 以上。

为什么要自己编程？

既然有了 MATLAB 优化工具箱和 LINGO 这样的计算软件，为什么还要自己劳神费力地编制优化计算程序呢？基于下面的两点理由，作者认为你最好有自己的计算程序。

(1) 在一些科研项目中，优化计算是“实时计算”，而不是在你认为需要的时候从键盘上输入目标函数和约束条件坐等计算结果。一个系统的运行由下图所示的一系列环节构成。



所谓“实时计算”是指，优化计算模型从前一环节获取模型参数，计算结果立即送给后一环节，这些交换都是在内存中进行的。因为系统必须连续运行，一气呵成。显然，那些带窗口界面的优化计算软件对此无能为力。你必须将自编的程序作为一个模块嵌入到系统中。作者确实遇到过这种情况。一位研究生从事一项光学测量方面的研究项目，他需要计算一个无约束优化问题。设计变量不多，只有 8 个，但目标函数比较复杂：

$$\min f(x_1 \cdots x_8) = \sum_{j=1}^{1142}\left[\frac{a_1^{(j)}x_1 + \cdots + a_8^{(j)}x_8}{b_1^{(j)}x_1 + \cdots + b_8^{(j)}x_8}\right]^2$$

其中 $a_1^{(j)},\cdots,a_8^{(j)},b_1^{(j)},\cdots,b_8^{(j)}(j=1,\cdots,1142)$事前并不知道，而是由系统的前一环节计算出来的。这位研究生开始并没有打算作优化计算，他利用本专业的知识和经验，对这个问题的解给出了一个“较为合理”的猜测 $\tilde{X}$。不幸的是，$\tilde{X}$ 对应的目标值$f(\tilde{X})$太大了：

$$f(\tilde{X}) \approx 10^{10}$$

“系统无法正常运行下去。”他如是说。而将作者编制的无约束优化计算动态链接库嵌入他的系统，得到了计算结果 X^*，使得

$$f(X^*) \approx 10^{-16}$$

因为目标函数 $f(X)\geqslant 0$，再考虑到计算机的数值误差，X^* 显然已经是最优解了。如此高的计算精度，作者自己也感到有些惊讶。

(2) 诸如 MATLAB 优化工具箱和 LINGO 这样的数值软件，可能会让你失望。作者对它们有过多次测试，只要目标函数和约束条件稍微复杂一点，这些数值软件都求不出最优解。

国家级的大型计算中心当然有很好的优化计算软件，只有大型的科研项目(几百上千个变量)才不得不求助于它。对于小型项目，杀鸡何须用牛刀。有了自己的计算软件，召之即来，多么方便。只要愿意花时间研读本书，更愿意花时间自己动手编程，你应该能编制出一套质量上乘的优化计算软件。

本书内容的安排及风格

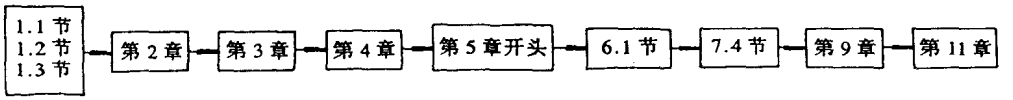
本书分为无约束优化计算(第 1 章至第 8 章)、约束优化计算(第 9 章至第 12 章)、动态优化计算(第 13,14 两章)三部分。每一章配置了少量的习题,让读者加深对算法及原理的认识。鼓励读者把主要的精力和时间用于编程。

第 1 章既是复习,也是深化。重点是多元二阶台劳展式的矩阵形式,它是理解算法的基础,读者应该对各种形式的台劳展开滚瓜烂熟。梯度计算是优化算法的基本构件,读者应该将它编制成独立的模块。Cholesky 分解在优化计算中用途很广,最好也将它编制成独立的模块。第 2 章描述了无约束优化计算的统一迭代格式及迭代的停止准则。第 3 章描述了两种线搜索方法,它们也是优化算法的基本构件,读者应该将它们编制成独立的模块。第 4 章描述了共轭梯度法,这种方法的优点是结构简单、稳定可靠,占内存少,适宜于大规模问题。第 5 章描述了两种牛顿型法——Gill-Murray 改进牛顿法和信赖域法。从理论上说,它们是收敛速率最快的方法,但计算工作量随变量的个数按平方级增长,占内存大,只适宜于目标函数性态优良的小规模问题。第 6 章描述了变尺度法,综合衡量算法的各项性能指标,对它的评价最高。第 7 章描述了四种不使用梯度的直接法。从作者对它们的大量数值实验来看,很难褒彼贬此,但普遍认为方向加速法(Powell)性能更好。第 8 章介绍了近十几年来优化算法研究的新成果——全局优化计算,供读者参考。第 9 章描述了约束优化的最优性条件。这一部分某些定理的证明稍微有点难度,需要读者有耐心和毅力。读者应该熟悉几个主要的结论并理解它们的几何意义。第 10 章描述了投影梯度法,这种方法适应于线性约束问题。第 11 章描述了广义乘子法,这是一种稳定可靠、适用范围广的方法。它需要调用无约束优化计算子模块。第 12 章描述了约束变尺度法,它是无约束变尺度法在约束问题中的推广。该方法计算效率高,但程序设计颇费功夫。第 13 章描述了动态规划(多阶段决策过程),读者将见识到与前面类型迥异的优化问题。第 14 章描述了一类常见的最优控制问题,并探讨了它的数值算法。

附录 1 为读者编制模块化的计算软件提供了很有价值的参考建议。

附录 2 为读者提供了普遍采用的 10 个测试问题,读者可用这些问题测试自己的软件。

如果读者希望用较少的时间掌握一套无约束和约束优化算法,建议按下图所示的顺序学习。



由此掌握的算法足以应付绝大部分非线性优化问题,并对优化理论有相当的认识。

作者多年为工程类研究生讲授"最优化方法"课程。这些研究生对优化计算表现出的兴趣是作者写这本书的主要动力。作者希望写一本既容易读懂又读而有所得的教材。下面是作者的一些观点。

· 必须明白的就彻底明白;只需了解的则一笔带过。

非线性优化计算的灵魂是迭代(其他类型的优化计算皆如此)。即首先任取一个初始解 X_0,按照某种规则,产生一系列迭代点 $X_k(k=1,2,\cdots)$,使得 $X_k \to X^*(k \to +\infty)$($X^*$为问题的最优解)。这引出下面四个问题:

(1) 如何由 X_k 产生 X_{k+1}? 这就是迭代规则,或称为算法。

(2) 迭代不能永无终止,什么时候可以停下来? 这就是停止准则问题,或称为最优性条件。

(3) 由迭代规则产生的点列 $\{X_k\}_{k=1}^{+\infty}$ 能收敛到最优解 X^* 吗? 这就是算法的收敛性问题。

(4) 算法如果是收敛的,收敛得快还是慢? 这就是收敛速率问题。

作者认为,对于工程类专业的研究生,(1)和(2)是必须明白的;(3)和(4)是只需了解的。因此,作者对每一个算法的原始思想及来龙去脉作了尽可能细致的描述。不但细致地描述了算法的每一个动作,而且讲清楚为什么要执行这个动作。

本书介绍的各类算法都能够在普通的条件下保证收敛,且有较满意的收敛速率。所以,正文不再提及此类问题。

· 提倡"白话",贴近读者

数学中概念、定理的描述讲究精炼,习惯于用符号代替文字。如果用之过当,则不利于读者的理解和领会。一本微积分教科书中对同号级数是这样定义的:

$$\forall\, i,j,a_i \cdot a_j \geqslant 0$$

虽然精炼,但有点"弯弯绕"的感觉。说白了就是:级数的各项符号相同。作者更愿意用后一种风格,并将此风格贯穿全书。

· 一幅图画胜过千言万语

读者一定体会过,自己看书总不如听教师的讲授来得轻松。原因之一就是,教师在黑板上随手画出的一个插图让人豁然开朗(虽然他画得并不优美)。本书对那些需要稍加思考领会的概念都配上一幅插图,读来令人倍感轻松。

· 与读者一起编程

从算法步骤到程序还有一段距离。算法步骤只是告诉你要做什么,如何做则需要你自己创造性的劳动。试举两例:

一个优化算法需要解决这样一个问题:现有 k 个 n 维向量

$$\boldsymbol{p}_1,\boldsymbol{p}_2,\cdots,\boldsymbol{p}_k$$

它们可能是线性相关的,也可能线性无关;可能 $k \geqslant n$,也可能 $k < n$。无论何种情况,要从中挑出一个最大无关组。

另一个优化算法需要模拟这样的过程:n 个人排成一行,每人轮流抛掷硬币,第 n 人抛完后再从第一人开始,如此反复……直到每个人都抛出一个正面接着抛出一个反面,过程终止(每个人的这种情况不必同时出现)。

诸如此类的问题,都需要读者稍微费点心思,也引发你接受挑战的激情。作者在本书中对这些程序设计中的细节问题准备了必须的知识和工具。

致使用 C++程序语言的读者

如果你习惯用 C++语言编程,本书尤其适合你。优化算法中有三个要反复执行的动作:计算梯度;计算二阶导数矩阵;一维搜索。可以说编制了这三个模块就完成了算法程序设计的一半工作。本书给出了这三个模块的 C++源程序,并给出了如何调用它们的示例。

与其他数学问题的算法一样,优化算法基本上就是矩阵和向量之间的运算。在长期的编程实践中,作者体会到很多代码都是可重用的。感谢 C++为我们提供了代码重用的极好机制——创建类(class)。为此,作者编制了一个矩阵向量类,它提供了大量的矩阵与向量的操作(100 多个功能),诸如任意阶矩阵的动态生成,+,-,* 运算,矩阵求秩,矩阵求逆,矩阵分解,线性方程组求解,求特征根,线性规划计算等等,均可用一条语句完成。使用这一类能极大地提高工作效率。如果读者感兴趣,欢迎与作者交流、切磋。

本书的初稿完成后,国防科技大学人文与管理学院院长沙基昌教授对初稿作了全面细致的审阅,提出了很好的修改意见,使本书的结构、内容的剪裁更趋合理。作者对沙基昌教授的帮助和指导深表感谢。

国防科技大学出版社的文慧同志和其他工作人员为本书的出版付出了艰辛的努力,他们严谨细致的工作使本书增色不少,作者向他们一并致谢。

作者尽了最大的努力避免错误和疏漏,但仍不敢保证白玉无瑕,诚恳希望读者批评指正。

作者

2000 年 8 月 30 日于国防科技大学

目　录

第二部分 约束优化计算

第三部分　动态优化计算

第一部分

无约束优化计算

第 1 章　预备知识

欲善工事先利其器。学习最优化方法也要准备好一些数学工具。必备的工具是:梯度、二阶导数矩阵、台劳展式。凸集和凸函数是最优化理论的基础。而 cholesky 分解和 Householrd 变换为某些算法所必需。这些内容读者并不陌生,在微积分和线性代数、矩阵分析的教科书中都涉及过。本章的目的是复习、深化它们,把这些工具磨光、磨亮。

梯度是大部分优化算法中必须计算的量,有些算法还要计算二阶导数矩阵。本章要讨论如何编制计算梯度和二阶导数矩阵的程序模块,并示例在算法中如何调用这些模块。考虑到现在流行使用 C + + 语言,这两个模块都用 C + + 编制(只要稍作修改,也适用于 Turbo C2.0 编译器)。

1.1　梯度、二阶导数矩阵、台劳展式

1. 梯度

设有可微的 n 元函数

$$f(X) = f(x_1, x_2, \cdots, x_n)$$

f 对各变量的偏导($\frac{\partial f}{\partial x_i}$或者$f_{x_i}$)构成的列向量称为 f 在 X 处的梯度,记作 $\nabla f(X)$:

$$\nabla f(X) = (\frac{\partial f}{\partial x_1} \frac{\partial f}{\partial x_2} \cdots \frac{\partial f}{\partial x_n})^{\mathrm{T}}$$

将梯度向量看成 n 维空间中的一个方向,它的意义是:f 在 X 处沿梯度方向有最快的上升趋势。图 1 - 1 是二元函数梯度的示意图,平面上椭圆代表函数的等值线,函数值由内向外增大。

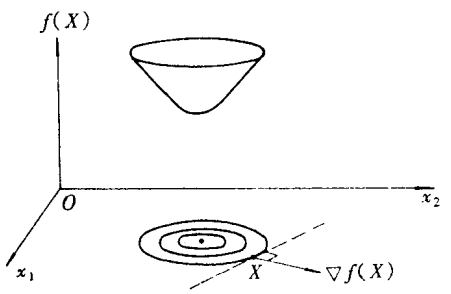


图 1 - 1

2. **二阶导数矩阵(Hessian 矩阵)**

设 n 元函数 $f(X)$ 二阶连续可微。f 的所有二阶偏导数构成的 $n \times n$ 矩阵称为 f 在 X 处的二阶导数矩阵,或称为 Hessain 矩阵,记作 $\nabla^2 f(X)$:

$$\nabla^2 f(X) = \begin{bmatrix} f''_{x_1^2} & f''_{x_1 x_2} & \cdots & f''_{x_1 x_n} \\ f''_{x_2 x_1} & f''_{x_2^2} & \cdots & f''_{x_2 x_n} \\ \vdots & \vdots & \vdots & \vdots \\ f''_{x_n x_1} & f''_{x_n x_2} & \cdots & f''_{x_n^2} \end{bmatrix}$$

因为 $f(X)$ 二阶连续可微,混合偏导与变量的次序无关,即

$$f''_{x_i x_j} = f''_{x_j x_i}$$

所以 $\nabla^2 f(X)$ 是一个对称矩阵。这一点很重要。

下面,以例题的形式给出一个重要结论,希望读者能自己验证这个结论,至少要牢记这个结论,以后将多次使用它。

例 1-1-1 设 A 是一个 $n \times n$ 的对称矩阵,b 是 n 元列向量,X 是 n 元列向量,c 是一个标量,称如下形式的 n 元函数为二次函数

$$f(X) = \frac{1}{2} X^{\mathrm{T}} A X + b^{\mathrm{T}} X + c$$

如果 A 是正定矩阵,则称 $f(X)$ 是正定二次函数。重要的结论是

$$\nabla f(X) = AX + b$$
$$\nabla^2 f(X) = A$$

这两个式子很好记忆,只要计算 $f(x) = \frac{1}{2} ax^2 + bx + c$ 的一阶和二阶导数就明白了。

3. **n 元二阶台劳展式**

在微积分教科书中,读者比较熟悉的是一元函数的二阶台劳展式。例如一元函数 $\varphi(\lambda)$ 在 $\lambda = 0$ 处的二阶台劳展式为(带 Piano 余项或 Cauchy 余项):

$$\begin{cases} \varphi(\lambda) = \varphi(0) + \varphi'(0)\lambda + \frac{1}{2}\varphi''(0)\lambda^2 + \circ(\lambda^2) \\ \varphi(\lambda) = \varphi(0) + \varphi'(0)\lambda + \frac{1}{2}\varphi''(\xi)\lambda^2 \end{cases} \tag{1-1-1}$$

现在要导出 n 元函数 $f(X)$ 的二阶台劳展式。给定一点 X 及一个方向 p,考虑一元函数

$$\varphi(\lambda) = f(X + \lambda p) = f(x_1 + \lambda p_1, \cdots x_i + \lambda p_i, \cdots, x_n + \lambda p_n)$$

根据复合函数求导法则,可计算出 $\varphi'(\lambda)$ 和 $\varphi''(\lambda)$。

定理 1-1-1 若记 $\varphi(\lambda) = f(X + \lambda p)$,则

$$\varphi'(\lambda) = \nabla^{\mathrm{T}} f(X + \lambda p) p$$
$$\varphi''(\lambda) = p^{\mathrm{T}} \nabla^2 f(X + \lambda p) p$$

[证明]