自动微分方法与最优化

张海斌　高　欢　著

科学出版社

自动微分方法与最优化

张海斌　高　欢　著

科学出版社

北京

内 容 简 介

自动微分方法是计算函数导数的有效工具. 传统观念认为,计算 n 元函数的一个偏导数所需要的计算量与计算该函数的一个函数值的计算量大致相当. 因此, 计算 n 元函数的梯度(n 个偏导数), 所需计算量相当于函数值计算量的 n 倍. 通常的方法, 如数值微分(差商近似)和符号微分, 都是如此. 然而自动微分颠覆了这一传统观念. 它计算函数梯度的计算量只相当于计算函数本身的数倍, 而与自变量个数 n 无关. 这一令人吃惊的结果, 激发了人们对自动微分的强烈兴趣.

近二十年来, 自动微分已成为国际上人们关注的热点, 但在国内的研究依然不足. 据作者所知, 本书是国内第一本对自动微分方法及其在最优化中的应用进行介绍和论述的书籍.

本书由浅入深, 系统地介绍自动微分的基本理论、算法设计和实现的软件工具, 包括低阶和高阶微分方法. 作为应用范例, 本书还给出了基于自动微分的最优化方法和特征值的数值计算. 阅读本书除相关应用(第 4、5 章)外, 只需具备高等数学和线性代数的基础知识.

本书可作为数值计算相关专业的高年级本科生、研究生的教学用书, 也可作为科研及工程技术人员的参考书.

图书在版编目(CIP)数据

自动微分方法与最优化/张海斌, 高欢著. —北京: 科学出版社, 2016.1

ISBN 978-7-03-047101-7

Ⅰ. ①自… Ⅱ. ①张… ②高… Ⅲ. ①微分–最优化算法–研究

Ⅳ. ①O172.1 ②O242.23

中国版本图书馆 CIP 数据核字(2016) 第 014817 号

责任编辑: 李 欣 / 责任校对: 彭 涛

责任印制: 张 倩 / 封面设计: 陈 敬

科学出版社 出版

北京东黄城根北街 16 号

邮政编码: 100717

http://www.sciencep.com

三河市骏杰印刷有限公司 印刷

科学出版社发行 各地新华书店经销

*

2016 年 1 月第 一 版 开本: 720×1000 1/16

2016 年 1 月第一次印刷 印张: 8 1/4

字数: 166 000

定价: 58.00 元

(如有印装质量问题, 我社负责调换)

前　言

函数导数的计算是科学与工程计算领域最基本的运算之一, 自动微分是计算多元函数导数的一种技术或方法, 它利用链式法则等微分原理以及算子重载和代码转换等计算技术, 以与函数本身的计算量大致相当的成本, 能够“自动”高效地计算出该函数的导数, 如梯度、方向导数等. 自动微分的思想出现在二十世纪五六十年代, 甚至更早, 但其理论的系统化、工具软件的大量开发和应用是近二十年的事. 以自动微分为主题的国际会议已经举行了 6 次, 这些会议的召开无疑推动了自动微分的发展. 2016 年的自动微分国际会议即将在英国牛津举行.

科学计算和工程分析中往往涉及大规模问题, 相应函数的自变量个数可能有数万甚至数百万之多, 而且常常需要计算梯度或雅可比矩阵及高阶导数等, 在这种情况下, 通常的微分方法是低效甚至是不可行的. 自动微分给出了高效的微分计算, 因而应用非常广泛, 是人们关注的热点. 在 2008 年的自动微分官方网站登录了 14 个相关的软件工具, 而现在已有 56 个之多. 在百度网页上以 “automatic differentiation” 为关键词搜索, 会返回近 100 万条搜索结果. 在我国, 自动微分于二十世纪八十年代最早应用于气象科学中基于变分资料同化的数值模拟问题.

本书的内容包含作者在博士期间学习整理的自动微分理论、算法以及部分研究工作. 作者尽量增强本书的可读性, 读者只需具备高等数学和线性代数的基础知识就可顺利阅读本书的基础部分内容, 并可以很容易着手运用自动微分方法. 关于阅读路线：第 1, 2, 3 章是基础, 按照顺序阅读, 之后可以直接跳到第 6 章, 关注软件实现；也可以直接到最后的附录, 阅读关于自动微分的复杂性；中间的第 4 章和第 5 章, 主要是关于自动微分在最优化方面的应用.

在本书的编写过程中, 作者得到了中国农业大学邓乃扬教授、北京交通大学修乃华教授的指导和帮助, 西南大学的程强教授、北京工业大学薛毅教授和徐大川教授等国内同行也提出了许多宝贵意见, 研究生李飞和吕亚斐等在阅读书稿和修改中做了大量的工作, 在此作者谨向他们表示诚挚的感谢.

感谢国家自然科学基金项目 (61179033)、首都社会建设和社会管理协同创新中心以及北京工业大学基础研究基金提供的经费支持.

由于时间仓促, 作者水平有限, 书中肯定有许多不足和需改进之处, 恳请广大读者不吝赐教, 不胜感激. 我们的邮箱 zhanghaibin@bjut.edu.cn 或 huanhuan135213@163.com.

张海斌　高欢

2015 年 8 月 16 日

目　录

第1章 引 论

在科学计算和工程分析中, 很多情况都需要应用函数的各种导数. 通常我们会考虑使用直接法、符号微分[101] 或差商方法来求解函数的导数. 直接法即手写微分对于经过仔细分析后的小规模或简单问题效率是较高的, 但对于大规模或复杂问题将会耗费大量的人力来开发, 这几乎是不可能实现的. 符号微分是对符号进行操作运算得到导数的方法, 它的优点是人力介入较少, 但所需惊人的计算代价甚至使得仅仅数十维的问题都无法忍受. 差商方法是一种近似方法, 面临增量两难选择的问题. 增量过大会放大截断误差, 过小会放大舍入误差. 与这些传统的微分方法相比, 自动微分方法能够有效地克服以上缺点, 具有理想的计算复杂性, 同时能够达到机器精度. 在给定原函数赋值程序的情况下, 利用自动微分方法可以有效地获得函数导数相关信息的计算程序. 例如, 航空航天、大气和海洋以及化工等领域常常会出现数万甚至数十万行的原函数计算程序, 对其的参数灵敏度分析或参数优化相关目标函数等都需要计算这个“大型”函数的导数相关信息, 随着自动微分方法的提出和发展, 这类问题已经得以有效地解决. 同时, 自动微分可以改进最优化算法, 这里强调的改进不仅是将其中的“微分器”由传统的符号微分、差分近似或手写微分置换为自动微分, 而且将梯度、海塞矩阵等导数项的自动微分计算与优化算法有机地结合, 考虑其改进后算法的效率.

本章接下来介绍自动微分的发展历史、基本知识和简单应用, 我们力求通过本章能使读者基本了解自动微分, 甚至可以直接用这一方法去解决一些相关问题.

1.1 自动微分的发展历史

自动微分的起源可以追溯到二十世纪五六十年代, 甚至认为更早前就有相关的基本思想, 只是没有正式提出或明确化. 自动微分包括正向模式和逆向模式, 最早正式发表的使用正向模式的应该是 1964 年 Wengert 的工作[97] 以及同年 Wilkins 的论文[98], Rall 在 1969 年的工作[84] 显示他们可以将简单的正向模式应用到很多问题中. Wilkins 预言自动微分将会作为一个调试工具以及独立的计算技术, 且有很光明的前景. 然而, 由于受计算机软硬件的局限, 自动微分的发展一度停滞不前, 在随后的很多年, Wengert 与 Wilkins 均没有进一步推进他们在自动微分方面的研究.

1971 年, Ostrovskii 及其合作者发表了关于自动微分逆向模式的论文[78], 这被

认为是逆向模式的首次提出, 但依然没有引起人们进一步研究的兴趣. 直到 1980 年, Speelpenning 的博士论文[93] 以及高级程序语言的发展, 人们才对自动微分的研究重新重视起来. 后来, 在美国阿贡 (Argonne) 国家实验室工作的 Griewank 及其同事对自动微分方法做了大量的研究工作, 并于 1989 年给出了自动微分的复杂性分析, 证明了梯度计算可以是函数的至多 5 倍的计算量, 且与自变量的维数无关, 这一惊人的结果激发了人们对自动微分的研究及其应用的浓厚兴趣. 当然我们相信在此之前, 已有不少人发现这一结果.

随着 1980 年前后自动微分的复兴和计算机技术的发展, 尽管逆向模式的开发和应用要比正向模式复杂得多, 但上述复杂性分析的惊人结果使得逆向模式处于非常突出的地位, 此时自动微分的应用领域也迅速扩展到数学及其应用中, 包括最优化、微分方程等数值计算, 计算工程如空气动力学中的敏感度分析与参数估计以及非线性优化实验设计等, 也进一步促进了自动微分工具软件的开发和研究, 如使用 C 语言的 ADOL-C, 使用 Fortran 语言的 ADIFOR 以及针对 ANSI-C 语言开发的 ADIC 等.

第一次关于自动微分的国际会议于 1991 年在 Breckenridge 举行, 它将自动微分的研究者和不同领域中的应用者联系在了一起, 他们之间的交流从横向和纵向促进着自动微分的研究和应用. 随后的三次国际学术会议 Santa Fe(1996), Nice(2000), Chicago(2004), Bonn(2008) 以及 Fort Collins(2012) 的成功举行不断地推动着自动微分的飞速发展, 即将在英国牛津举行的 AD(2016) 也将进一步推动自动微分的发展和应用. 在自动微分理论及算法应用等方面有大量的研究工作, 在此我们只能列举一部分, 如参考文献 [11–13], [18], [30], [59], [85]. 关于自动微分在金融计算、仿真计算、化学工程、气象预报、机器学习等方面的应用, 可参考相关文献 [4], [14], [15], [17], [37], [71], [80], [87], [95], [102], [113] 等.

接下来我们将对自动微分的基础知识、基本方法以及在非线性最优化方法中的简单应用加以探讨.

1.2　函数的计算框架

自动微分技术处理的对象是一个任意给定的 (有点不确切) 向量值函数

$$y = F(x) : D \subset \mathbf{R}^n \to \mathbf{R}^m, \tag{1.1}$$

其中定义域 D 是 $\mathbf{R}^n$ 的一个开集, 它的值域 $F(D)$ 是 $\mathbf{R}^m$ 中的一个子集. 求函数的各种导数不外乎是求与雅可比矩阵

$$F^{'}(x)=\left(\frac{\partial F_i}{\partial x_j}\right):D\subset\mathbf{R}^n\to\mathbf{R}^{m\times n} \tag{1.2}$$

相关的式子或相关项, 当然其本身是一个矩阵值函数.

由于所研究的是如何以实用的和构造性的方式求一个已知的函数 F 的导数, 那么该函数 F 必定是显式给定的, 我们应该清楚如何去计算或对其每一个组成部分进行求导. 现假设可以通过代数表达式具体化相应的函数. 我们必须通过计算机程序给定和计算一个函数, 所以很难完全区分函数的符号表示和数值表示. 用恰当的计算程式计算一个函数要比冗长的表达式更易于分析而且更加经济有效.

本章首先给出的一个重要概念就是函数的计算程式.

一个带有独立变量和依赖变量的输入参数集合的计算程序形成了一个计算程式, 相应的控制流就确定了. 给定独立变量的具体数值, 执行这个计算程式就产生了一个踪迹, 而这个踪迹可以被视为它的一个计算图, 并且其每个节点常常被视为一个变量.

一个计算程式可以覆盖变量值也可以调用子程式.

考虑向量值函数 $F(x):\mathbf{R}^3\to\mathbf{R}^2$, 其中 $F(x)$ 的定义如下:

$$\begin{cases} F_1(x_1,x_2,x_3)=\mathrm{e}^{x_1x_2}(\mathrm{e}^{x_1x_2}+x_1x_2\sin x_3), \\ F_2(x_1,x_2,x_3)=(x_1x_2\sin x_3+\mathrm{e}^{x_1x_2})/x_3. \end{cases} \tag{1.3}$$

因为 $x_3\mathrm{e}^{x_1x_2}=\dfrac{F_1(x_1,x_2,x_3)}{F_2(x_1,x_2,x_3)}$, 所以分别存储和计算 $F_1(x_1,x_2,x_3)$ 和 $F_2(x_1,x_2,x_3)$ 是不经济的. 共同的子表达式 $\mathrm{e}^{x_1x_2}$ 和 $\mathrm{e}^{x_1x_2}+x_1x_2\sin x_3$ 是计算 $F_1(x_1,x_2,x_3)$ 和 $F_2(x_1,x_2,x_3)$ 的主要计算量. 因此, 从这个例子可以看出, 如果一个表达式中有多个共同子表达式, 为了避免多次重复, 可以将其作为中间变量. 因此, 对于式 (1.3) 而言, 可以采取以下计算方式:

$$\mu_1=\mathrm{e}^{x_1x_2};\quad \mu_2=x_1x_2\sin x_3;\quad \mu=\mu_1+\mu_2;$$

$$F_1(x_1,x_2,x_3)=\mu\mu_1;\quad F_2(x_1,x_2,x_3)=\frac{\mu}{x_3}.$$

特别地, 以 $y=F_1(x_1,x_2,x_3)$ 为例来详细说明其计算过程, 即

$$y=\mathrm{e}^{x_1x_2}(\mathrm{e}^{x_1x_2}+x_1x_2\sin x_3). \tag{1.4}$$

同时, 我们希望计算 y 的函数值.

计算过程见表 1.2.1, 该表中包含了许多的数学变量定义. 我们仅仅用一个基本例子来说明通过链式法则来进行数值计算. 当进行以上计算时, 我们会想如果这个问题有成千上万的变量和数千次循环, 结果会怎样? 表中的第一部分是输入变量,

接着是中间变量 (记为 x_i, 其中 $i > n$), 最后一行就是输出变量 (本例中只有一个输出变量). 每一个变量都是通过先前定义的变量经过一些简单运算 $+$, $-$, $\times$, sin, exp 等来进行计算. 同时可以通过一个计算图 (图 1.2.1) 表示其计算过程. 虽然这个计算图不能直接用来计算, 但可以更清晰地看到过程和更有效地分析结构. 后面将具体介绍计算过程.

表 1.2.1　函数 (1.4) 的计算过程

$x_1 = 1.0000$
$x_2 = 0.2000$
$x_3 = 3.0000$
$x_4 = x_1 * x_2 = 1.0000 * 0.2000 = 0.2000$
$x_5 = \sin(x_3) = \sin(3.0000) = 0.1411$
$x_6 = \exp(x_4) = \exp(0.2000) = 1.2214$
$x_7 = x_4 * x_5 = 0.2000 * 0.1411 = 0.0282$
$x_8 = x_6 + x_7 = 1.2214 + 0.0282 = 1.2496$
$x_9 = x_6 * x_8 = 1.2214 * 1.2496 = 1.5263$
$y = x_9 = 1.5263$

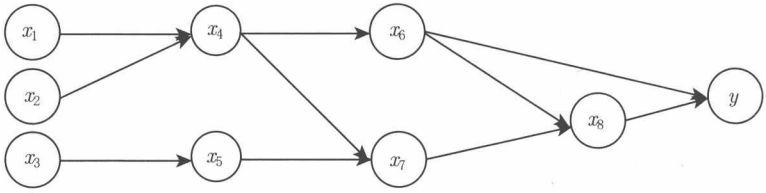

图 1.2.1　表 1.2.1 的计算图

假设我们进一步将函数 (1.3) 分解为更小的原子操作. 实际上, 当函数在计算机上计算时, 这种分解方式是经常存在的. 表 1.2.2 给出了函数 F 具体三段计算框架, 这个过程第一部分首先将自变量的当前值赋值给输入变量 (x_1, x_2, x_3) 和第三部分将最终值 (x_9, x_{10}) 赋给输出变量 y_1, y_2, 它们位于表格的底端. 最终计算过程是表 1.2.2 的中间部分.

一般来说, 我们假定在特定情形时所有变量 x_i 在函数计算时数目是给定的, 使得

$$[\underbrace{x_1, \cdots, x_n}, x_{n+1}, x_{n+2}, \cdots, x_{l-m}, \underbrace{x_{l-m+1}, \cdots, x_l}].$$

通过对变量 x_j(其中 $j \prec i$) 应用基本函数 φ_i (后面有详细介绍) 得到 x_i(其中 $i > 0$), 从而有

$$x_i = \varphi_i(\{x_j\}_{j \prec i}), \tag{1.5}$$

其中 $\prec$ 表示 x_i 直接与 x_j 有关, 通常是一个指标或者两个指标 $j<i$ 的情况.

在这本书中, 我们应该关注在特定参数下的函数, 可以通过表 1.2.3 给出一般计算过程. 因为该计算过程提供了 F 作为它的基本函数 φ_i 的复合函数的数学表达式, 且它们的计算规则是明确的, 所以这个计算过程对于我们的目标是非常有用的. 各种各样的微分算子可能被应用, 产生与其紧密相关结构的衍生和伴随计算过程.

表 1.2.2 函数 (1.3) 的计算过程

$x_1 = x_1$
$x_2 = x_2$
$x_3 = x_3$
$x_4 = x_1 * x_2$
$x_5 = \sin(x_3)$
$x_6 = \exp(x_4)$
$x_7 = x_4 * x_5$
$x_8 = x_6 + x_7$
$x_9 = x_6 * x_8$
$x_{10} = x_8/x_3$
$y_1 = x_9$
$y_2 = x_{10}$

表 1.2.3 函数的一般计算过程

$x_i = x_i$	$i = 1, \cdots, n$
$x_i = \varphi_i(\{x_j\}_{j \prec i})$	$i = n+1, \cdots, l$
$y_i = x_{l-m+i}$	$i = 1, \cdots, m$

自动微分将微分运算的计算过程提升到基本的计算程序层面上来. 任意一个计算过程唯一确定一个向量函数 $y = F(x)$, 它的性质由 x_i 和基本元素 φ_i 的性质所确定. 当 $i > n$ 时, 每一个 $x_i = x_i(x)$ 的值也可以解释为独立向量 $x \in \mathbf{R}^n$ 的一个中间函数 $x_i(x), i > n$. 下面我们可以发现, 大部分基本元素往往是很基本的初等函数, 但是我们的框架允许列入计算更复杂的元素.

1.2.1 向量元素的一般化

一般地, 令 x_i 是一个向量, 其中 $i > n$, 且维数定义如下

$$m_i = \dim(x_i) \geqslant 1, \quad \text{对于 } i = n+1, \cdots, l-m.$$

将它们视为标量可以让我们更容易理解自动微分的基本方法. 基本函数 φ_i 的维数由 m_i 所定义, 其中 $m_i > 1$ 意味着我们可能把基本代数子程序或者其他子程序作

为单个基本函数. 这种解释是有实际意义的. 特别地, 它可解释整个计算过程作为一个 (超级) 基本函数, 这个基本函数可以看作在更高层面上计算过程的一部分. 这种方式的计算过程可以看作是分层结构, 这些分层结构可以在各种层面上进行不同的微分技术. 大部分情况我们仅仅考虑两个层面上的情形, 但是一些技术和结果表明在任意许多层面上完全是递归的. 独立变量和依赖变量总是假定是标量, 使得

$$i \leqslant n \text{ 或者 } i > l - m, \text{ 意味着 } m_i = 1.$$

为了避免记号的复杂性, 我们把每个 x_i 看作是一个数据项, 那么基本函数 φ_i 的定义域维数由式 (1.6) 给定

$$n_i = \sum_{j \prec i} m_j, \quad \text{对于 } i = 1, \cdots, l. \tag{1.6}$$

相应地, 我们可能把 x_j 联立起来, 即 φ_i 依赖以下参数变量

$$u_i = \{x_j\}_{j \prec i} \in \mathbf{R}^{n_i}. \tag{1.7}$$

这个定义可能增加基本函数的参数变量的数目, 该基本函数仅仅依赖中间变量 x_j $(j > n)$ 元素的选取.

1.2.2　代码的独立性

在某种程度上, 我们习惯考虑函数为一个简单的映射, 但它可能是不恰当的, 因为自动微分作为某些基本函数的复合, 它与这些复合函数关系密切. 同时这种分解方式不是唯一的, 我们需要面对一些任意性: 计算同一数学函数的两个不同过程可能会对微分的计算产生不一样的稳定性和有效性. 然而, 并不只对具体的微分计算产生这样的效应; 而且对在任意给定参数前提下, 计算函数的基本任务也发生影响. 代码好坏问题和自动微分很大程度上是相关的. 也就是说, 代码的质量对函数和微分具有相同的影响.

更专业地来说, 得到的计算过程和程序的好 (或者坏) 取决于原始给定函数的计算过程和程序. 通过这本书, 我们将讨论在准则下的代码质量的好坏, 特别是计算复杂性和数值稳定性.

表 1.2.4 列出了在大多数高级编程语言中可用的算术运算和一些初等函数. 虽然基本算术运算符 $+$, $-$ 和 $*$ 是必不可少的, 但是构成基本库的其他基本函数的选择是非常灵活的. 基本函数集 Φ 至少包含 $+$ 和 $*$ 两种运算, 一个一元符号转换符 $-$ 和对常标量 c 的初始化. 我们称这个最小的集合为多项式核, 因为通过适当的求值追踪, 我们可以求任何多项式的值. 表 1.2.4 有如下规定: k 是整数; u, v, u_k, v_k 是变量; c, c_k 是常数.

表 1.2.4 光滑基本函数

数量运算	向量运算
$u+v,\ u*v$	$\sum_k u_k * v_k$
$-u,\ c$	$\sum_k c_k * u_k$
$\mathrm{rec}(u)=1/u$	
$\exp(u),\ \log(u)$	
$\sin(u),\ \cos(u)$	
$u^c,\ c>1, u>0$	
$\cdots$	

有一些可以扩展为基本函数的光滑函数有

$$u-v,\quad u/v,\quad c*u,\quad c\pm u,$$
$$u^k,\quad u^c,\quad \arcsin(u),\quad \tan(u).$$

由于所有单变量的初等函数事实上都是通过一系列的多项式运算来求值的, 如 $\sin(u)$, $\mathrm{rec}(u)=1/u$ 等. 这使我们认为除多项式核之外, 不需要引入其他基本函数. 然而, 这一结论是错误的. 例如, 求值程序中常常用到的分支运算. 它们可能是符号函数, 也可能是三元条件赋值等形式, 这些函数在计算和应用中非常重要, 我们将它们列举如下:

$$\mathrm{abs}(u)=|u|,\quad \max(u,v),\quad \min(u,v),\quad \mathrm{sign}(u),$$

$$\mathrm{H}(u)=(u>0)?u,w,\quad \text{即 } \mathrm{H}(u)=\begin{cases} u, & \text{如果 } u>0,\\ w, & \text{其他.}\end{cases}$$

向量情况如

$$\|u\|_\infty=\max_k\{|u_k|\},\quad \|u\|_1=\sum_k|u_k|,\quad \|u\|_2=\sqrt{\sum_k u_k^2}.$$

$\|u\|_\infty$ 在一致逼近问题、精密仪器数据拟合等方面应用很多, $\|u\|_1$ 常用在目前研究很热门的稀疏优化、压缩感知以及数据处理等问题中, $\|u\|_2$ 更是有着广泛的应用, 如稀疏块 LASSO 问题. 不幸的是, 这些基本函数显然是不可微的. 因此通过应用链式法则计算此类复合函数的微分是行不通的. 本书前面部分只考虑光滑基本函数. 所以, 一开始我们将如上的非光滑函数排除在外.

接下来, 我们假定基本函数 φ_i 取自于基本函数库

$$\Phi=\{c,\pm,*,\exp,\sin,\cdots\},\tag{1.8}$$

其元素 $\varphi \in \Phi$ 满足 d 阶可微 $(d \geqslant 1)$. 因此表 1.2.4 中的后两类一般是排除在外以确保所有的复合函数 F 至少一阶连续可微. 给定基本函数库 Φ, 记

$$\mathcal{F} = \mathrm{Span}[\Phi] \tag{1.9}$$

为基本函数张成的可微函数空间, 它们可以用 Φ 中基本函数构成的计算程序来定义.

1.3　自动微分的基本理论

本节将通过简单例子说明自动微分的基本方法是什么, 它是如何工作的. 首先基于例子 (1.4), 即

$$y = F_1(x_1, x_2, x_3) = \mathrm{e}^{x_1 x_2}(\mathrm{e}^{x_1 x_2} + x_1 x_2 \sin x_3)$$

来描述自动微分中的正向模式和逆向模式. 我们将在第 2 章进行详细深入地探讨.

1.3.1　通过正向模式求函数的方向导数

为了计算一个 n 元函数沿方向 $\mathrm{d}x \in \mathbf{R}^n$ 的方向导数, 我们使用正向模式, 基本方法是从自变量、中间变量到函数, 利用链式法则微分函数计算过程中的每一步, 我们把这种从上到下、从左到右的计算过程称为**正向模式**.

以求 $y = F_1(x_1, x_2, x_3)$ 的一阶方向导数 (切向微分) 为例, 我们需要在计算过程中对表 1.2.1 每个基本函数微分每一个变量, 得到关于式 (1.4) 的正向模式, 见表 1.3.1. 很明显, 从原始的计算过程表 1.2.1 转换得到正向模式表 1.3.1 是完全机械性的.

当方向 $\mathrm{d}x \in \mathbf{R}^n$ 确定时, 这些值和运算仅仅依赖于 x. 现假设函数变量 y 想对 x_1 求微分 (同时我们也可以对 x_2 或者 x_3 求微分). 事实上, 我们仅把 x_1 看作独立变量, y 看作依赖变量. 利用链式法则, 并结合表 1.2.1, 可以计算 $\mathrm{d}x_i = \partial x_i/\partial x_1 (i > 3)$. 显然, $\mathrm{d}x_1 = 1.0000$, $\mathrm{d}x_2 = 0.0000$, $\mathrm{d}x_3 = 0.0000$, 因为 $x_4 = x_1 * x_2$, 从而有

$$\begin{aligned}\mathrm{d}x_4 =& (\partial x_4/\partial x_1)\mathrm{d}x_1 + (\partial x_4/\partial x_2)\mathrm{d}x_2 = \mathrm{d}x_1 * x_2 + \mathrm{d}x_2 * x_1 \\ =& (1.0000 * 0.2000 + 0.0000 * 1.0000) = 0.2000.\end{aligned}$$

同样, 由 $x_5 = \sin(x_3)$, 有

$$\mathrm{d}x_5 = (\partial x_5/\partial x_3)\mathrm{d}x_3 = \cos(x_3) * \mathrm{d}x_3 = \cos(3.0000) * 0.0000 = 0.0000.$$

表 1.3.1 函数 (1.4) 的正向模式计算过程

$x_1 = 1.0000$	$\mathrm{d}x_1 = 1.0000$
$x_2 = 0.2000$	$\mathrm{d}x_2 = 0.0000$
$x_3 = 3.0000$	$\mathrm{d}x_3 = 0.0000$
$x_4 = x_1 * x_2 = 1.0000 * 0.2000 = 0.2000$	
$\mathrm{d}x_4 = \mathrm{d}x_1 * x_2 + x_1 * \mathrm{d}x_2 = 1.0000$*$0.2000+1.0000$*$0.0000=0.2000$	
$x_5 = \sin(x_3) = \sin(3.0000) = 0.1411$	
$\mathrm{d}x_5 = \cos(x_3) * \mathrm{d}x_3 = \cos(3.0000) * 0.0000 = 0.0000$	
$x_6 = \exp(x_4) = \exp(0.2000) = 1.2214$	
$\mathrm{d}x_6 = x_6 * \mathrm{d}x_4 = 1.2214 * 0.2000 = 0.2443$	
$x_7 = x_4 * x_5 = 0.2000 * 0.1411 = 0.0282$	
$\mathrm{d}x_7 = \mathrm{d}x_4 * x_5 + x_4 * \mathrm{d}x_5 = 0.2000$ *$0.1411 + 0.2000$*$0.0000= 0.0282$	
$x_8 = x_6 + x_7 = 1.2214 + 0.0282 = 1.2496$	
$\mathrm{d}x_8 = \mathrm{d}x_6 + \mathrm{d}x_7 = 0.2443 + 0.0282 = 0.2725$	
$x_9 = x_6 * x_8 = 1.2214 * 1.2496 = 1.5263$	
$\mathrm{d}x_9 = \mathrm{d}x_6 * x_8 + \mathrm{d}x_8 * x_6 = 0.2443$*$1.2496 + 0.2725$*$1.2214= 0.6381$	
$y = x_9 = 1.5263$	$\mathrm{d}y = \mathrm{d}x_9 = 0.6381$

具体的见表 1.3.1. 该表是自动微分的基本正向模式, 在以后我们还会详细地解释自动微分的正向模式. 同时我们也可以精确计算 $\partial y/\partial x_2$ 和 $\partial y/\partial x_3$, 前者初值为 $\mathrm{d}x_1 = 0.0000, \mathrm{d}x_2 = 1.0000, \mathrm{d}x_3 = 0.0000$, 后者初值为 $\mathrm{d}x_1 = 0.0000, \mathrm{d}x_2 = 0.0000, \mathrm{d}x_3 = 1.0000$, 按照表 1.3.1 进行计算可以得到相应的 $\mathrm{d}y = 3.1904$ 和 $\mathrm{d}y = -0.2417$.

从表 1.3.1 可以看出: 正向模式简单方便, 存储量少, 它的计算量大约相当于函数值计算量的 2.5 倍, 且与 n 无关, 详细的复杂性讨论见后面的章节. 但如果我们求函数的梯度, 则需要求 n 个方向的方向导数, 这样显然是不合算的, 如果求一个 n 元函数的梯度则需要下面的逆向模式.

1.3.2 通过逆向模式求函数的梯度

利用自动微分在计算函数的梯度或向量值函数的加权梯度时, 我们需要在函数计算结束之后, 逆着函数计算的轨迹求对等式右端变量的导数, 我们称之为伴随值, 或伴随变量, 利用链式法则逆向传递, 即采取从下到上, 自右向左的顺序进行, 最后到达开始端得到对各个自变量的导数, 即梯度的每一个分量. 我们称这个计算过程为**逆向模式**.

例子 (1.4) 的计算过程见表 1.2.1, 通过它产生一个增量伴随过程见表 1.3.2. 原始过程的执行在最左边, 伴随变量 bx_9 初始化给具体值 by. 之后, 所有的原始变量采用逆序的方式, 同时依赖于它们是否有一个或者两个变量被相应的一个或者两个增量符号替代. 对于 $i = 1, 2, 3$, 在每一个表的最后聚集 bx_{i-3} 的值, 它与独立变量

x_i 的伴随值 bx_i 一致. 增量返回扫描位于表 1.3.2 的中间; 相应的非增量形式位于表 1.3.3.

表 1.3.2　函数 (1.4) 的逆向模式伴随过程

正向扫描
$x_1 = x_1 = 1.0000$
$x_2 = x_2 = 0.2000$
$x_3 = x_3 = 3.0000$
$x_4 = x_1 * x_2 = 1.0000 * 0.2000 = 0.2000$
$x_5 = \sin(x_3) = \sin(3.0000) = 0.1411$
$x_6 = \exp(x_4) = \exp(0.2000) = 1.2214$
$x_7 = x_4 * x_5 = 0.2000 * 0.1411 = 0.0282$
$x_8 = x_6 + x_7 = 1.2214 + 0.0282 = 1.2496$
$x_9 = x_6 * x_8 = 1.2214 * 1.2496 = 1.5263$
$y = x_9 = 1.5263$
增量返回
$bx_9 = by = 1$
$bx_8 += bx_9 * x_6 = 1 * 1.2214 = 1.2214$
$bx_6 += bx_9 * x_8 = 1 * 1.2496 = 1.2496$
$bx_7 += bx_8 = 1.2214$
$bx_6 += bx_8 = 1.2496 + 1.2214 = 2.4710$
$bx_4 += bx_7 * x_5 = 1.2214 * 0.1411 = 0.1723$
$bx_5 += bx_7 * x_4 = 1.2214 * 0.2000 = 0.2443$
$bx_4 += bx_6 * x_6 = 0.1723 + 2.4710 * 1.2214 = 3.1904$
$bx_3 += bx_5 * \cos(x_3) = 0.2443 * \cos(3.0000) = -0.2419$
$bx_2 += bx_4 * x_1 = 3.1904 * 1.0000 = 3.1904$
$bx_1 += bx_4 * x_2 = 3.1904 * 0.2000 = 0.6381$
$bx_3 = bx_3 = -0.2419$
$bx_2 = bx_2 = 3.1904$
$bx_1 = bx_1 = 0.6381$

表 1.3.3　函数 (1.4) 的逆向模式伴随过程

正向扫描
$x_1 = x_1 = 1.0000$
$x_2 = x_2 = 0.2000$
$x_3 = x_3 = 3.0000$
$x_4 = x_1 * x_2 = 1.0000 * 0.2000 = 0.2000$
$x_5 = \sin(x_3) = \sin(3.0000) = 0.1411$
$x_6 = \exp(x_4) = \exp(0.2000) = 1.2214$
$x_7 = x_4 * x_5 = 0.2000 * 0.1411 = 0.0282$
$x_8 = x_6 + x_7 = 1.2214 + 0.0282 = 1.2496$
$x_9 = x_6 * x_8 = 1.2214 * 1.2496 = 1.5263$
$y = x_9 = 1.5263$