Parallel Algorithm of Partial Differential Equation and Numerical Method of Inverse Problem

偏微分方程并行算法及反问题数值解法

刘春凤　彭亚绵　著

清华大学出版社

Parallel Algorithm of Partial Differential Equation and Numerical Method of Inverse Problem

偏微分方程并行算法及反问题数值解法

刘春凤　彭亚绵　著

清华大学出版社
北京

内 容 简 介

本书系统介绍了偏微分方程并行数值求解方法，及其偏微分方程反问题的数值求解方法和应用。主要包括偏微分方程的多重网格方法并行理论和差分方法的并行化，非线性不适定问题的基本概念，求解不适定问题的正则化法及其改进，求解反问题的并行遗传算法理论及应用，并在最后一部分介绍了环境水力学反问题以及应用求解。书中内容包含了作者及其学生近几年来的相关工作。

本书可作为应用数学、计算数学专业研究生的教学参考书，也可供从事科学与工程计算的科学技术工作者参考。

图书在版编目(CIP)数据

偏微分方程并行算法及反问题数值解法/刘春凤，彭亚绵著. —北京：清华大学出版社，2015

ISBN 978-7-302-42313-3

Ⅰ. ①偏… Ⅱ. ①刘… ②彭… Ⅲ. ①偏微分方程－并行算法 ②偏微分方程－数值计算 Ⅳ. ①O241.82

中国版本图书馆 CIP 数据核字(2015)第 287063 号

责任编辑：付弘宇　柴文强
封面设计：何凤霞
责任校对：梁　毅
责任印制：李红英

出版发行：清华大学出版社
网　址：http://www.tup.com.cn，http://www.wqbook.com
地　址：北京清华大学学研大厦 A 座　　邮　编：100084
社总机：010-62770175　　邮　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课　件　下　载：http://www.tup.com.cn，010-62795954

印 刷 者：三河市君旺印务有限公司
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：170mm×230mm　　**印　张**：15.25　　**字　数**：240 千字
版　　次：2015 年 12 月第 1 版　　**印　次**：2015 年 12 月第1 次印刷
印　　数：1～1000
定　　价：59.80 元

产品编号：063854-01

前　言

在工程技术中经常用数值方法来获得非线性偏微分方程及其反问题的近似解，因而对非线性偏微分方程给出一些相应的数值分析，已经成为计算数学领域的一个重要部分。这样就使得偏微分方程的数值解法在数值分析中占有了更加重要的地位。自然界很多的数学模型都是用数值方法来求解的，即使是连续的数学模型，要使其在实际生活中加以实现，一般也要离散化，然后再进行计算。偏微分方程的数值解法很多，如有限差分法、有限元法、有限体积法、多重网格法等。

本文的研究目标是在传统的串行算法的基础上，对偏微分方程及其反问题的数值解法进行深入研究把数值方法并行化，并对其算法进行上机实现；把并行算法与串行算法的结果进行比较，并且通过分析并行算法性能度量（如时间、规模、并行度、加速比等），真实地反应出并行算法的优越性。偏微分方程反问题是非线性和不适定的，所以研究不适定问题的解法很有必要。本书主要综述了常用的算法：正则化算法，Landweber 迭代法，投影算法。

本书是作者近年来科研工作的整理和总结，内容涉及到：偏微分方程的多重网格方法并行理论和差分方法的并行化，非线性不适定问题的基本概念，求解不适定问题的正则化法及其改进，求解反问题的最佳摄动量法理论的改进及应用，并在最后一部分介绍了环境水力学反问题以及应用求解。每一个方向的成果都为科学计算领域做出了一些贡献，同时也反映出作者在学科前沿做出成绩的愿望。

我们感谢清华大学出版社的帮助，感谢河北省重点学科（河北联合大学应用数学重点学科）的资助和我们科研团队独立承担基金项目（No. 61170317）的资助，使得我们能够坚持相关的研究工作并顺利完成本书的写

作。本书作者是国家精品课程《数值计算方法》的负责人和骨干教师，我们要感谢河北联合大学对我们团队的一贯帮助和支持。没有以上支持和帮助，本书是难以面世的，作者对此表示诚挚的感谢。

由于时间仓促，加之作者水平所限，本书难免会有错误和不妥之处，敬请专家、读者批评指正，我们将不胜感激。

作者

2015 年 6 月

目　录

第1章 引 论

并行计算机的发展，使大型科学与工程计算成为可能，使得科学计算作为科学研究的一种有效手段，已上升为与科学理论和科学实验并重的三大科学方法之一。在自然科学与工程技术领域中有许多问题都可以用偏微分方程来描述，研究偏微分方程的数值解是解决上述问题的有力工具。当偏微分方程中的算子、右端项、边界条件、初始条件从过去的已知变成未知，而原方程的解仍然未知时，就构成了偏微分方程的反问题。由于反问题的不适定性与非线性，使得它的理论与求解成为广大数学工作者、自然科学工作者及工程技术人员努力开拓的学科领域。

1.1 并行算法简介

并行计算在许多计算机应用领域都产生了巨大的影响，使原来无法解决的应用问题成为可能。

例如：天气预报。

考虑3000×3000平方公里的范围，垂直方向的考虑高度为11公里。将3000×3000×11立方公里的区域分成0.1×0.1×0.1立方公里的小区域，则将近有10^{11}个不同的小区域。另外还需考虑时间因素，将时间参数量化。假定考虑48小时天气预报。

每一小区域的计算包括参数的初始化及与其他区域的数据交换。若每一小区域计算的操作指令为100条，则整个范围一次计算的指令为$10^{11}\times 100=10^{13}$，两天的计算次数将近100次，因此，指令总数为$10^{15}$条。用一台10亿次/秒(PⅢ500)的计算机计算，将大约需要280小时。若我们用100个

10 亿次/秒的处理器构成一台并行处理机，每个处理器计算的区域为 10^8 个，不同的处理器通过通信来传输参数，若个处理器的计算能力得到充分利用，则整个问题的计算时间不超过 3 小时。说明两点：①并行计算机可以解决原先不能解决的问题；②可进行更准确的天气预报。

其他应用包括：卫星数据处理、石油数据处理（连续优化问题），调度问题、平面性问题及 VLSI 设计（离散优化问题）。

随着计算机硬件、软件及算法的进步，在并行计算领域，并行体系结构、并行软件和并行算法缺一不可，而其中并行算法则是核心技术。

并行算法是一些可同时执行的进程的集合，这些进程相互作用和协调工作，从而达到求解给定问题的目的。也就是指在各种并行机上求解问题和处理数据的算法，其本质是把多任务映射到多处理机中执行。

算法（Algorithm）是解题方法的精确描述，是一组有穷的规则，它们规定了解决某一特定问题的一系列运算。

并行算法（Parallel Algorithm）是一些可同时执行的多个进程的集合，这些进程相互作用和协调工作，从而达到对给定问题的求解。从不同的角度，并行算法可以分为不同的类别：数值并行算法和非数值并行算法；同步的、异步的和分布式的并行算法；共享存储的和分布存储的并行算法；确定的和随机的并行算法等。

数值计算（Numerical Computing）是指基于代数关系运算的一类诸如矩阵计算、多项式求值、求解线性方程组等数字计算问题。求解数值计算问题的算法称为数值算法（Numerical Algorithm）。

非数值计算（Non-numerical Computing）是指基于比较关系运算的一类计算问题，比如排序、选择、搜索和匹配等符号处理问题。求解非数值计算问题的算法称为非数值算法（Non-numerical Algorithm）。

同步算法（Synchronized Algorithm）是指算法的各个进程的执行必须相互等待的一类算法。

异步算法（Asynchronized Algorithm）是指算法的各个进程的执行不必相互等待的一类算法。

分布算法(Distributed Algorithm)是指由通信链路连接的多个节点协同完成问题求解的一类算法。

确定性算法(Deterministic Algorithm)是指算法的每一步都能明确的指明下一步的动作的一类算法。

随机算法(Randomized Algorithm)是指算法的每一步都随机的从指定范围内选取若干参数,由此确定算法的下一动作的一类算法。

同步(Synchronization)是在时间上强制使一组执行中的进程在某一点相互等待。在并行算法的各进程异步执行过程中,为了确保各处理器的正确工作顺序以及对共享资源的正确访问(资源的互斥访问),程序员需要在算法中恰当的位置设置同步点。同步可以用软件、硬件或固件的方法来实现。

通信(Communication)是多个并发执行的进程在空间上进行数据交换。

从计算复杂性的角度来考虑,一个算法的复杂性可分为空间和时间复杂性两个方面。并行算法的目标是尽可能减少时间复杂性,通常这是通过增加空间复杂性(如增加空间的维数及增加处理器台数)来实现的。并行算法采用"浅而宽"的结构,即让每个时刻可容纳的计算量相应增加,使整个算法的步数尽可能减少,或者说通过增加每个时间步的算法复杂性来减少整体的时间复杂性。

可以从不同的角度将并行算法分为数值并行算法和非数值并行算法;同步并行算法、异步并行算法和分布并行算法;SIMD 并行算法 MIMD 并行算法和 VLSI 并行算法;确定的和随机的并行算法等。

描述一个算法,可以使用自然语言进行物理描述;也可以使用某种程序设计语言进行形式化描述。语言的选用应避免二义性,且力图直观、易懂而不苛求严格的语法格式。像描述串行算法所选用的语言一样,类-Algol、Pidgin-Aglol、类 Pascal 等语言均可选用。在这些语言中,允许使用任何类型的数学描述,通常也无数据类型的说明部分,但只要需要,任何数据类型都可以引进。

在描述并行算法时,所有描述串行算法的语句及过程调用等均可使用,

而只是为了表达并行性而引入几条并行语句：

Par-do 语句 当算法的若干步要并行执行时，我们可以使用“do in parallel”语句，简记为“par-do”进行描述：

```
For i = 1 to n par-do
...
End for
```

其中“End for”也可以代之以“drop”。

For all 语句 当几个处理器同时执行相同的操作时，可以使用“For all”语句描述之：

```
For all P_i, where 0≤i≤k do
...
End for
```

1.1.1 并行算法的设计

通常算法设计者针对同一问题可设计出多种不同算法以适应在不同模型上对该问题的求解，并分析和评价并行算法的优劣。并行算法的设计与分析依赖于并行计算模型。需要给并行算法的设计与分析提供一个简单、方便的框架，并行计算模型抽象了一类并行计算机的基本特征，避开了硬件结构过多的繁琐细节限制，保证了它在相当范围内的通用性，同时又能反映出不同算法的主要特征，为算法的设计提供启发、指导和评价依据。

另外，并行算法的设计具有一定的生命力。算法设计者避开了多种多样的具体的并行计算结构，依据并行计算模型来设计算法。一方面可使研制者集中精力开发应用问题本身固有的并行性，分析算法性能；另一方面设计出的算法具有通用性，从而使并行算法的研究成为一项相对独立的活动。

利用并行处理机系统求解一个给定的问题，需要根据系统的类型和特征设计并行算法。通常有三种途径：

(1) 检测和开发现有串行算法中固有的并行性而直接将其并行化；

(2) 从问题的本身的特征出发，设计一个新的并行算法；

(3) 修改已有的并行算法使其可求解另一类相似问题。

但对一类具有内在顺序性的串行算法很难于并行化,修改已有的并行算法有赖于特定的一类问题,设计全新的并行算法,技术上尚不成熟且似乎又有些技巧。

目前普遍使用的几种并行算法的设计技术主要有:流水线技术、分治策略、平衡树方法、倍增技术以及加速级连策略等,这里将主要介绍以后章节要用到的分治策略。

分治策略(Divide-and-Conquer-Technique)即分而治之策略,其思想是将原问题分解成若干个特征相同或相似的子问题分而治之,将计算负载分摊在各个计算机结点上,以减少单个节点的计算量,从而达到加速整体运行效率的目的。若所得的子问题规模仍然太大,可反复使用分治策略直至很易求解诸子问题为止。使用分治法时,子问题的类型通常和原问题的类型相同,因此很自然的导致递归过程。该策略在不少并行算法文献中已被公认为设计并行算法的根本准则,常见的分治方法主要有以下两种:

(1) 数据分割:数据分割是指各节点基本执行相同的任务,只是数据不同。此方法常用在计算的各数据之间具有对称性的情况。例如对于向量的加法,可以分割为各个分量的加法。

(2) 任务分割:任务分割是指总任务由自然数个子任务组成,将其分摊到各个节点上。例如对于连续的数据处理任务,可以将其分割成数据采集、数据计算和结果输出三个子任务。

根据设计并行算法的特点,可以注意到无论采用什么方法设计并行算法,以下几点都是决定并行效率的重要因素:

① 挖掘并行性。针对一个具体问题设计其并行算法时,可根据求解该问题的串行算法出发,分析其明显的并行性,加以并行化,并应该利用相关知识,深刻挖掘问题内在的可并行性。

② 并行算法依赖于并行的环境。并行算法与其实现的并行环境密切联系,一个并行算法在不同的并行系统上的差别很大,所以说,要根据并行的环境有针对性的设计并行算法。

③ 考虑通信开销。通信开销是一个必须考虑的因素，因为有时通信在整个算法中占用时间的比例很大，特别是在分布存储的并行环境下，要尽量减少通信开销，才能设计出高效率的并行算法。

1.1.2 并行算法的性能度量

对于一个给定的问题，如果我们设计了一个新的并行算法，就必须对该算法的性能进行评价。性能度量的目的就是为并行算法的设计与分析提供一个统一的衡量标准。本小节我们将对度量并行算法性能的一些基本概念，如算法复杂性度量、加速比和效率等进行介绍，以方便后面的算法分析。

串行算法的研究中，一个重要的课题是研究算法的时间复杂性和空间复杂性。在并行算法中，复杂性问题同样是重要的研究课题。

对于并行算法，除了研究所需的运行时间外还需要研究算法所需处理器的数目，以及研究两者在最坏情形下与问题规模的变化关系。

运行时间通常包含两部分的时间，一是数据从一个处理器经由互联网络或共享存储器到达另一个处理器的时间；二是数据在一个处理器内做运算所需的计算数目。

处理器数目就是求解给定的问题所需要的处理器的数目，可以固定大小，也可以随问题的规模变化而变化。

一个算法的复杂性(Complexity)也叫复杂度，是指它所含的工作量。例如，串行算法的复杂性是指算法的运算量与存储量。它们都是求解问题的规模函数。

在分析算法时，如果对算法的所有输入分析其平均形态时的复杂度则称之为期望复杂度(Expected Complexity)，为此往往需要对输入的分布作某种假定，这在大多情况下并非容易的事。

分析并行算法时，我们将研究算法的运行时间和所需的处理机的台数 P 与问题规模 n 的变化关系。对于并行算法，通常需要分析的指标有：

(1) 运行时间 $t(n)$：就是指算法运行在给定的模型下求解问题所需的时间(它主要是输入规模 n 的函数)，如果不同的处理机不能同时结束它们

的计算，则并行算法的运行时间定义为：从第一台处理机开始执行到最后一台处理机执行完成所经过的时间。它通常包含计算时间和通信时间，分别用计算时间步和选路时间步作为单位。

(2) 处理器数 $p(n)$：它是求解给定问题所用的处理器数。做算法分析时可假设求解给定问题所需的处理器数目是问题规模 n 的函数。有时，处理机台数 p 是不依赖于 n 的常数，并且 $p<<n$。

(3) 并行算法的成本 $c(n)$：它定义为并行算法的运行时间 $t(n)$ 与其所需的处理器数 $p(n)$ 的乘积，即 $c(n)=t(n)\times p(n)$。

(4) 总运算量 $W(n)$：指的是并行算法所完成的总的操作数量。

加速比和效率：加速比与效率是评估一种并行算法得失的两项重要指标，它们是最传统的并行算法评价标准，体现了在并行机上运用并行算法求解实际问题所获得的好处。

定义：并行系统的加速比是指对于一个给定的应用，并行算法或并行程序的执行速度相对于串行算法或串行程序的执行速度加快了的倍数，可以定义为：

$$S_p = \frac{T_1}{T_p}$$

定义：并行算法的效率，是一个与加速比密切相关的概念。它定义为：

$$E_p = \frac{S_p}{P}$$

式中，T_1 表示串行算法在单处理机上的运行时间，T_p 表示并行算法在 P 台处理机上的运行时间，P 为处理机台数。

不难看出，P 越大越好。理想情况下 $S_p=P$。但实际只有类似于向量加法等非常特殊的情况才能达到此完全加速比。一般情况下是 $S_p\leqslant P$。一个并行算法虽然有好的加速比，但处理机的利用率可能很低，特别是当处理机的台数 $P(n)$ 不固定的情况下，S_p 不是一个最好的评价标准。引入并行算法的效率后，就可度量并行系统中处理机能力发挥的程度。显然 $0\leqslant E_p\leqslant 1$。

如果按照某种条件，并保持每台处理机的计算规模，并行算法加速比 S_p 与处理机台数 P 成正比，则称该并行算法在该条件下，在该并行机上具有线