国家自然科学基金资助

KEXUEJISUANJIYINGYONG

科学计算及其应用

刘春凤 杨爱民
常锦才 龚佃选 ◎著

科学出版社

国家自然科学基金资助

科学计算及其应用

刘春凤　杨爱民
常锦才　龚佃选　著

科学出版社
北京

内 容 简 介

本书共5章，内容包括并行计算法、样条函数、径向基函数、常微分方程的B条解法等领域的理论和数值方法及其应用。

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

科学计算及其应用/刘春凤等著. —北京：科学出版社，2013
ISBN 978-7-03-038981-7

Ⅰ.①科… Ⅱ.①刘… Ⅲ.①科学计算-研究 Ⅳ.①N32

中国版本图书馆CIP数据核字(2013)第254707号

责任编辑：纪 兴 刘文军/责任校对：马英菊
责任印制：吕春珉/封面设计：东方人华平面设计部

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

北京京华虎彩印刷有限公司印刷
科学出版社发行 各地新华书店经销
*
2013年11月第 一 版 开本：787×1092 1/16
2013年11月第一次印刷 印张：10 1/4
字数：230 000

定价：52.00元

（如有印装质量问题，我社负责调换〈京华虎彩〉）
销售部电话 010-62134988 编辑部电话 010-6215763-2003

前　言

科学计算是在现代科技环境下，随着信息技术的飞速发展形成的一种科学研究方法，与理论研究、科学实验并列成为进行科学活动的三大基本手段，目前已经成为内容规模庞大、涵盖范围宽广的一门综合性的新学科。

本书是作者近年来科研工作的整理和总结，内容涉及并行计算、样条函数与计算机辅助几何设计、径向基函数、常微分方程奇异摄动问题等领域的理论和数值方法及其应用。每一个方向的成果都为科学计算领域做出了一些贡献，同时也反映出作者紧跟时代步伐，力争在学科前沿做出成绩的愿望。撰写时，本书补充了部分基础知识，以方便读者阅读。

作者感谢科学出版社的帮助，感谢国家自然基金委员会和河北省自然基金委员会的资助和支持。特别是近年来对科研团队独立承担基金项目（No. 61170317，11126213，A2012209043，E2013209215）的支持，使得作者及其科研团队能够坚持相关的研究工作，并顺利完成本书的写作。本书作者都是国家精品课程“数值计算方法”的骨干教师，作者十分感谢河北联合大学对我们团队的一贯帮助和支持。另外河北联合大学科学计算讨论班的研究生为书稿的录入和校对付出了辛勤的劳动，在此向他们表示诚挚的感谢。

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。

目　录

第1章 引 论

科学计算作为一门综合性的新科学，已经成为人们进行科学活动必不可少的方法和工具。科学计算的核心内容是计算数学，主要研究用计算机求解各类数学问题的数值方法。科学计算也是21世纪高层次人才知识结构中不可缺少的一部分，潜移默化地影响着人们的思维方式和思想方法。科学计算还能解决理论及实验无法解决的问题，并发现一些新的物理现象，增进人们对自然界的认识和理解，从而促进科学的发展。

1.1 科学计算概述

1. 科学计算的内涵与特点

科学计算的核心是数值分析(Numerical Analysis)。数值分析也称计算方法(Computational Method)，是寻求数学问题近似解的方法、过程及其理论的一个数学分支，主要研究适合于在计算机上使用的数值计算方法及其理论与软件实现。传统内容包括解线性方程组的直接法、函数近似计算的插值法与最小二乘法、数值积分和数值微分、常微分方程的数值解法、解线性方程组和非线性方程的迭代法等。现代科学计算已是一个内容庞大的体系，它不仅包括函数逼近(多项式插值、数据拟合、样条函数、径向基函数、小波等)、微分方程数值解(差分法、有限元、边界元、多重网格法、谱方法、无网格法等)、数值代数(直接三角分解法、迭代法、共轭梯度法、同伦算法等)及最优化理论与方法等；而且，伴随着计算机硬件技术的进步及计算能力的提高，面对航空航天、军事国防、经济金融、生命科学、地球科学、现代医药等领域复杂问题的挑战，大规模科学计算的理论与方法越来越成为计算科学关注的热点问题。

数值计算作为近代数学的一个重要分支，既有数学类课程理论上的抽象性和严谨性，又有实用性和实验性的技术特征，是一门理论性和实践性都很强的学科。其特点概括起来有以下四点。

1) 面向计算机，要根据计算机的特点提供切实可行的有效算法，即算法只能包括加、减、乘、除运算和逻辑运算。

2) 有可靠的理论分析，能任意逼近并达到精度要求，对近似算法要保证收敛性和数值稳定性，还要对误差进行分析。

3) 要有好的计算复杂性。时间复杂性好是指节省时间，空间复杂性好是指节省存储量。

4) 要有数值实验。科学计算本质上是一种近似计算，因此计算过程中必然存在误差。误差的来源包括模型误差、观测误差、公式误差和舍入误差等。针对一类问题，一般

会有多种解决方案，科学计算中算法的选择也是科技工作者必然要面对的问题。算法的选择一般有如下标准。

① 精确性：一个算法的计算结果要达到一定的精度要求，也就是结果的误差在问题允许的范围内。

② 快速性：如果精确性能够达到要求，算法越快越好，即时间复杂度问题。

③ 存储量：算法的存储量是指计算过程中变量所占计算机内存的大小。好的算法希望存储的中间变量越少越好，即空间复杂度问题。

2. 科学计算的应用

当代科学计算已渗透到极其广泛的专业领域，形成了许多新的边缘学科，如计算物理学、计算力学、计算流体力学、计算化学、计算生物学、计算材料学、计算经济学等，而计算数学正是联系它们的纽带和共同的基础。

1.2 本书主要内容

本书以科学计算中若干领域如并行计算、样条函数、径向基函数、常微分方程奇异摄动问题等为研究方向，将作者的一些研究结果整理、汇总，集结成册，供同行参考。本书主要内容如下。

第 2 章，首先结合并行计算技术，给出了并行 Lagrange 和 Newton 插值算法，曲线拟合的最小二乘并行算法，求解方程组的并行 Gauss 消元法、Gauss-Jordan 消元法、LU 分解法等数值算法，并给出了在相关领域的应用。本章旨在为经典数值算法的改造提供一种新思路。

第 3 章，首先介绍通常的样条函数和 B 样条基函数，定义了均匀 T-B 样条基函数、均匀 T-B 样条曲线和 T-B 样条曲面，归纳了均匀 T-B 样条曲线、T-B 样条曲面的基本性质。其次，根据曲线、曲面光滑拼接的基本条件，给出了两条相邻的 T-B 样条曲线光滑拼接的条件、不相邻的两条 T-B 样条曲线间过渡曲线的构造方法及 T-B 样条曲线与 NURBS 曲线的光滑拼接条件。计算得出了两张相邻的双三次 T-B 样条曲面的光滑拼接条件，进一步构造了连接不相邻的 T-B 样条曲面的过渡曲面。最后，给出了 C-B 样条曲线曲面的定义和基本性质、C-B 样条曲线的光滑拼接条件及 C-B 样条曲线间过渡曲线的构造方法及两张相邻的 C-B 样条曲面间的光滑拼接条件。本章旨在为复杂曲线曲面的光滑拼接提供一种新的思想和方法。

第 4 章，首先简要介绍径向基函数方法的基本理论，指出径向基函数方法是一个功能强大的无网格方法，在科学研究及工程计算中得到了广泛的应用。其次，对径向基函数插值中心点集的选取方法进行了一些讨论，该方面的研究成果将对径向基理论以及径向基函数方法产生深远的影响。该问题目前还没有得到很好的解决，需要更多感兴趣的读者继续深入研究。最后，将紧支径向基函数作为传递函数应用于神经网络进行曲面修补，并进行了模拟实验。误差曲线和网络训练时间表明，用紧支径向基神经网络修补曲面的性能优越，可以到达预期的效果。

第5章，首先介绍奇异摄动问题的相关概念及常微分方程的B样条解法。其次，将B样条函数应用于几类常见的奇异摄动问题，包括一般的奇异摄动问题、摄动参数为平方的奇异摄动问题、时滞奇异摄动问题、含有两个参数的奇异摄动问题等，并给出其详细求解过程、方法收敛性分析和数值算例。基本思想是，根据方程的阶数，选择相应次数的B样条基函数，并以其线性组合作为数值解的表达式，令其满足方程和边界条件，列出关于组合系数的线性方程组来求解。方法简单有效，可供工程技术人员参考。

第2章 数值并行算法及其应用

数值计算方法既是一门古老的学科，又是一门新兴的学科，电子计算机的产生和发展大大地促进了数值计算方法的发展，现在数值计算方法已经深入到各个领域。程序设计方法学是随着计算机的产生和发展而发展起的一门学科，只有把数值计算方法和程序设计紧密地结合起来，才能真正使用计算机帮助人们解决极其复杂的任务。近年来，已经有许多不同的解决数值计算问题的并行算法。机群系统是目前较流行的高性能计算平台，在高性能计算机中占越来越大的比例，其系统的规模可以从单机、少数几台连网的微机直到上千个结点的大规模并行系统，该系统廉价而且容易普及，因而应用非常广泛。互联网中的广播传递就是指从一个处理结点向其他处理结点发送相同的基本数据单位。根据并行的思想对经典的插值和拟合方法、经典的数值代数方法进行有效的改造将进一步推动计算科学的迅速发展，并扩展其应用领域。

2.1 并行插值算法及其应用

2.1.1 并行 Lagrange 插值算法及其应用[1]

并行 Lagrange 插值算法是以 n 个节点 $(x_0, y_0), (x_1, y_1), \cdots, (x_{n-1}, y_{n-1})$ 的 Lagrange 插值多项式公式为基础，提出在机群系统并行环境下的构造 Lagrange 插值多项式的一种并行算法。当处理机数量为 n^2 时，该算法的时间复杂度为 $3\log(n)+O(1)$；当处理机数量为 $p^2(p<n)$ 时，该算法的时间复杂度为 $O(n^2/p^2)\log(n)$。

下面给出在 n^2 台处理机条件下插值多项式的两种并行算法的构造，首先算法的计算模型如下。

1) 为方便，将 n^2 台处理机排列成 n 行，n 列的正方形，并用记号 $p(i,j)(0\leqslant i,j\leqslant n-1)$ 来标记第 $i+1$ 行，第 $j+1$ 列的处理机。

2) 为 $p(0,0)$ 分配 3 个寄存器，分别记为 $A(0,0)$，$B(0,0)$ 和 $D(0,0)$；为 $p(k,k)(0<k\leqslant n-1)$ 分配寄存器 $A(k,k)$ 和 $B(k,k)$。给其余的处理机 $p(i,j)$ 只分配一个寄存器 $A(i,j)(0\leqslant i,j\leqslant n-1$ 且 $i\neq j)$。

对应的算法如下。

1. 并行算法 A

Step 1：对 1.1 和 1.2 并行处理。

1.1：$A(0,j)=x_j$，其中 $0\leqslant j\leqslant n-1$；

1.2：$B(i,i)=x$，其中 $0\leqslant j\leqslant n-1$。

Step 2：对于每个 $j(0\leqslant j\leqslant n-1)$ 并行做如下广播传递。

将 $A(0,j)$ 的值广播至同一列的其余处理机的 $A(i,j)$ 中，其中 $0\leqslant j\leqslant n-1$。

Step 3：对 3.1 和 3.2 并行处理。

3.1：$A(i,i)=B(i,i)-A(i,i)$，其中 $0\leqslant j\leqslant n-1$；

3.2：$A(i,j)=A(i,j)-A(j,i)$，其中 $0\leqslant j<i\leqslant n-1$。

Step 4：对 4.1 和 4.2 并行处理。

4.1：$A(i,i)=-A(j,i)$，其中 $0\leqslant i<j\leqslant n-1$；

4.2：$B(i,i)=A(i,i)$，其中 $0\leqslant j\leqslant n-1$。

Step 5：对 5.1 和 5.2 并行处理。

5.1：$B(0,0)=\prod_{k=0}^{n-1}B(k,k)$；

5.2：对每个 j 同时做乘积运算，即 $A(j,j)=\prod_{k=0}^{n-1}A(k,j)$，其中 $0\leqslant j\leqslant n-1$；

5.3：$D(0,0)=B(0,0)$。

Step 6：对每个 $i(0\leqslant j\leqslant n-1)$ 同时做 $B(i,j)=y_i$ 操作。

Step 7：对每个 $i(0\leqslant j\leqslant n-1)$ 做并行处理，即 $A(i,i)=\dfrac{B(i,i)}{A(i,i)}$。

Step 8：将 n 个节点的 Lagrange 插值多项式的结果存储在 $A(0,0)$。

8.1：$A(0,0)=\sum_{k=0}^{n-1}A(i,i)$；

8.2：$A(0,0)=A(0,0)\times D(0,0)$。

经分析知道，并行算法 A 中的 Step 2，Step 5 和 Step 8 各需 $\log(n)$ 时间步，其余的所有步骤的总运算量的时间复杂度为 $O(1)$，因而并行算法 A 共需 $3\log(n)+O(1)$ 时间步。

2. 并行算法 B

当处理机的台数为 $p^2(p<n)$ 时，对并行算法 A 进行修正得到并行算法 B。为了简单起见，假设 $n=kp$，其中 k 为整数，这样将 n 个节点划分为 $k=n/p$ 个部分：$\{x_0,x_1,\cdots,x_{p-1}\},\{x_p,x_{p+1},\cdots,x_{2p-1}\},\cdots,\{x_{(k-1)p},x_{(k-1)p+1},\cdots,x_{kp-1}\}$。

为实现并行算法 B 还需要为每个处理机分配两个寄存器 temp1 和 temp2。

Step 1：对 $\{x_0,x_1,\cdots,x_{p-1}\}$ 根据算法 A 的 Step1 到 Step5 依次操作，并将结果 $\prod_{i=0}^{p-1}(x-x_i)$ 存储到 $D(0,0)$ 中；同时将结果 $(x-x_i)\prod_{j=0且j\neq i}^{n-1}(x_i-x_j)$ 存储到 temp1(0,0)，其中 $0<i\leqslant p-1$。

Step 2：将 $\{x_p,x_{p+1},\cdots,x_{2p-1}\}$ 同时存放在第 1 行，第 2 行，…，第 n 行后，再对 2.1 和 2.2 进行处理。

2.1：求出 $\prod_{i=0}^{2p-1}(x-x_i)$ 后，将其和 $D(0,0)$ 的乘积存储在 $D(0,0)$，结果为

$$D(0,0)=\prod_{i=0}^{2p-1}(x-x_i)$$

2.2:对每个 $i(0\leqslant i\leqslant p-1)$,同时计算出乘积 $\prod_{j=p}^{2p-1}(x_i-x_j)$,并将其和的乘积存储在 temp1$(i,i)$。

Step 3:现在依次对 $\{x_{2p},x_{2p+1},\cdots,x_{3p-1}\},\{x_{3p},x_{3p-1},\cdots,x_{4p-1}\},\cdots,\{x_{(k-1)p},x_{(k-1)p+1},\cdots,x_{kp-1}\}$ 进行 Step 2 的操作步骤,经过 $k-2$ 次操作后将得到 $D(0,0)=\prod_{i=o}^{n-1}(x-x_i)$ 和 temp1 $(i,i)\prod_{j=0且j\neq i}^{n-1}(x_i-x_j)(0\leqslant i\leqslant p-1)$。

Step 4:依次完成如下操作。

4.1:$B(i,i)=y_i$,其中 $0\leqslant i\leqslant p-1$;

4.2:对每个 $i(0\leqslant i\leqslant p-1)$ 同时进行操作,即 $\text{temp1}(i,i)=\dfrac{B(i,i)}{\text{temp1}(i,i)}$;

4.3: $\text{temp2}(0,0)=\sum_{i=1}^{p-1}\text{temp1}(i,i)$。

Step 5:依次对 $\{x_p,x_{p+1},\cdots,x_{2p-1}\},\{x_{2p},x_{2p+1},\cdots,x_{3p-1}\},\cdots,\{x_{(k-1)p},x_{(k-1)p+1},\cdots,x_{kp-1}\}$ 类似进行上述的 Step 1 到 Step 4 的操作步骤。

Step 6:对 $D(0,0)$ 和 temp2$(0,0)$ 求积,并将最后的结果存储在 $D(0,0)$,此时得到的 $D(0,0)$ 就是要求的 Lagrange 插值多项式。

由于算法 B 的 Step 1 是根据算法 A 的 Step 1 到 Step 5 步骤进行操作的,这样由算法 A 的 Step 1 到 Step 5 的运算量分析很容易得到算法 B 的 Step 1 需要 $2\log(n)+O(1)$ 时间步;根据 2.1 和 2.2 各需要 $\log(p)+O(1)$ 时间步知 Step 2 需要 $\log(p)+O(1)$;而 Step 3 是重复 Step 2 的操作 $n-2$ 次,故需要 $(k-2)\times[2\log(p)+O(1)]$ 时间步,Step 4 需要 $\log(p)+O(1)$ 时间步,Step 5 需要 $(k-1)\times[(2k+1)\log(p)+O(k)]$ 时间步,因此算法 B 共需要的时间步为 $k[(2k+1)\log(p)+O(k)]=O((n^2/p^2)\log(n))$。

加速比和效率分析:

用运算次数的多少来衡量算法,设加法和减法所需时间为单位时间 T_1,乘法和除法所需时间为单位时间 T_2。对于串行 Largrange 插值算法,运算 1 个插值基函数 $l_i(x)$ 的计算时间为 $8T_1+7T_2$,再乘以 y_i,则工作时间为 $8T_1+8T_2$,一共 5 个 $l_i(x)$,则工作时间为 $5(8T_1+8T_2)$,然后将这 5 个数相加,则工作时间为 $5(8T_1+8T_2)+4T_1$,则一组的串行时间为 $5(8T_1+8T_2)+4T_1$。而运用并行算法时,设第一组数据为总的 CPU,运用了 5 个CPU,则运算 5 个 $l_i(x)$ 的工作时间为 $8T_1+7T_2$,将得到的数据返回总的 CPU,代入 Lagrange 插值法公式之后,运算了 4 次加法和 5 次乘法,则并行时间为

$$8T_1+7T_2+4T_1+5T_2=12T_1+12T_2$$

根据加速比的计算公式,得

$$S_p=\frac{T_1}{T_p}=\frac{5(8T_1+8T_2)+4T_1}{12T_1+12T_2}$$

$$E_p=\frac{S_p}{p}=\frac{5(8T_1+8T_2)+4T_1}{5(12T_1+12T_2)}$$

假如有 n 个插值节点,设 $n=4k+1$(k 为整数),将 n 个插值节点分成 k 组,每组有 m 个数,则

$$P_n(x)=l_0(x)y_0+l_1(x)y_1+\cdots+l_n(x)y_n$$

$$l_k(x)=\prod_{\substack{i=0\\ i\neq k}}\frac{(x-x_i)}{(x_k-x_i)},k=0,1,\cdots,n$$

设加法和减法所需时间为单位时间 T_1，乘法和除法所需时间为单位时间 T_2。对于串行算法，运算一个 $l_i(x)$的计算时间为 $8T_1+7T_2$，再乘以 y_i，则工作时间为 $8T_1+8T_2$，一共 m 个 $l_i(x)$，则工作时间为 $m(8T_1+8T_2)$，然后将这 m 个数相加，则工作时间为 $m(8T_1+8T_2)+(m-1)T_1$，则一组的串行时间为 $m(8T_1+8T_2)+(m-1)T_1=(9m-1)T_1+8mT_2$。而运用并行算法时，设第一组数据为总的 CPU，运用了 m 个 CPU，则运算 m 个 $l_i(x)$的工作时间为 $8T_1+7T_2$，将得到的数据返回到总的 CPU，代入 Lagrange 插值法公式中之后，运算了 $m-1$ 次加法和 m 次乘法，则并行时间为

$$8T_1+7T_2+(m-1)T_1+mT_2=(7+m)(T_1+T_2)$$

根据加速比的计算公式，得

$$S_p=\frac{T_1}{T_p}=\frac{(9m-1)T_1+8mT_2}{(7+m)(T_1+T_2)},\ E_p=\frac{S_p}{p}=\frac{(9m-1)T_1+8mT_2}{m(7+m)(T_1+T_2)}$$

例 2.1　并行 Lagrange 插值法在低湿区露点检测中的应用。

在科学研究和工业生产过程中，很多场合对气体的纯度要求很高，而气体中含有的微量水分可能会对科研和生产过程带来不利影响，属于有害杂质，因此需要对气体中的微量水分进行测定，其实质就是对湿度进行测量。对于绝对湿度或气体含水量的表达，国际上逐步向露点温度上统一。露点是指气体在恒定压力下，气体中水蒸气在达到饱和时的温度。露点温度越低，则气体的湿度越小。在低湿区环境下的露点检测中，电容式露点传感器测量的频率与露点值呈现非线性关系，且受环境温度影响很大。此外，湿度越低，测量的分辨力也越低。采用一种恒定低温方法来检测低湿区露点，即传感器的环境温度恒保持在 0℃以下，可大大提高低湿区的测量分辨力，再根据露点值与频率值间具有的确定的非线性关系，将采集的频率经 Lagrange 插值计算后就可得到露点值。

解：利用标准露点气体发生器产生特定湿度气体，通入(−5±0.1)℃低温恒温腔，用高精度频率计测量传感器电容量经 C/F 变换后的频率值，从露点值−80～+0℃每隔 5℃标定一次，共得到 17 组标定值，如表 2.1 所示。

表 2.1　−5℃环境温度下频率与露点标定值表

露点/℃	频率/kHz	露点/℃	频率/kHz	露点/℃	频率/kHz
−80	29.59	−50	25.90	−20	19.28
−75	29.06	−45	24.89	−15	18.68
−70	28.62	−40	23.74	−10	18.18
−65	28.09	−35	22.51	−5	17.76
−60	27.49	−30	21.22	0	17.34
−55	26.80	−25	20.14		

表 2.1 为实际测得的一组数据，下面对以上数据采用并行 Lagrange 插值法对数据进行插值，得到相应的露点值。经误差分析表明，采用四阶插值即可保证足够的精度，预先

将表 2.1 中区间[17.34,29.59]上的 17 组频率标定值分为 4 组:{29.59, 29. 06,28.62, 28.09, 27.49},{27.49, 26.80, 25.90, 24.89, 23.74},{23.74, 22.51, 21.22, 20.14, 19.28},{19.28, 18.68, 18.18, 17.76, 17.34}。

根据 Lagrange 插值公式

$$p(x)=\sum_{k=0}^{n} l_k(x)y_k, k=0,1,\cdots,n$$

有

$$p_4(x)=l_0(x)y_0+l_1(x)y_1+l_2(x)y_2+l_3(x)y_3+l_4(x)y_4$$

首先计算第一组数据,第一组中有 x_1,x_2,x_3,x_4,x_5 共 5 个数据,将 x_0,x_1,x_2,x_3,x_4 和 y_0,y_1,y_2,y_3,y_4 的值代入,有

$$\begin{aligned}I_1=p_4(x)=&\frac{(x-29.06)(x-28.62)(x-28.09)(x-27.49)}{(29.59-29.06)(29.59-28.62)(29.59-28.09)(29.59-27.49)}\times(-80)\\&+\frac{(x-29.59)(x-28.62)(x-28.09)(x-27.49)}{(29.06-29.59)(29.06-28.62)(29.06-28.09)(29.06-27.49)}\times(-75)\\&+\frac{(x-29.59)(x-29.06)(x-28.09)(x-27.49)}{(28.62-29.59)(28.62-29.06)(28.62-28.09)(28.62-27.49)}\times(-70)\\&+\frac{(x-29.59)(x-29.06)(x-28.62)(x-27.49)}{(28.09-29.59)(28.09-29.06)(28.09-28.62)(28.09-27.49)}\times(-65)\\&+\frac{(x-29.59)(x-29.06)(x-28.62)(x-28.09)}{(27.49-29.59)(27.49-29.06)(27.49-28.62)(27.49-28.09)}\times(-60)\end{aligned}$$

将 $l_0(x),l_1(x),l_2(x),l_3(x),l_4(x)$并行,设有 5 个 CPU,第一个 CPU 为总的 CPU,每个 CPU 将控制一个 $l_i(x)$,这样 5 组数一起工作;同理第二组($I_2=p_4(x)$)、第三组($I_3=p_4(x)$)和第四组($I_4=p_4(x)$)也可像第一组那样并行,这样大大减少了工作时间。加速比和效率分别为

$$S_p=\frac{T_1}{T_p}=\frac{5(8T_1+8T_2)+4T_1}{12T_1+12T_2},\quad E_p=\frac{S_p}{p}=\frac{5(8T_1+8T_2)+4T_1}{5(12T_1+12T_2)}$$

本节实例对并行 Lagrange 插值法进行了全面的阐述与推导,并从具体实例推广到了一般的情形。在实例中,通过与串行算法比较,证明此算法有良好的并行性,另外还从理论上分析了加速比与效率,并可以看出 Lagrange 插值法的优越性。

2.1.2 并行 Newton 插值算法及其应用[2]

为了便于方法的引入,结合实例对传统 Newton 插值法与并行 Newton 插值法进行比较,说明与推导。

例 2.2 已知有函数表如表 2.2 所示。表中有 13 个节点,试用 Newton 插值法构造插值多项式。

解:考虑到精度,选取四阶插值来构造函数,故先将表 2.2 中的 13 组数据分成 3 组:{28,18.68,188,17.76,17.34},{17.34,31,35,38,39.62},{39.62,78,156.6,280,520}。下面先来计算第一组数据的均差表,如表 2.3 所示。

表 2.2　函数表

i	x_i	y_i	i	x_i	y_i
1	28	20	8	38	43
2	18.68	65	9	39.62	28.44
3	188	100	10	78	35
4	17.76	25	11	156.6	36
5	17.34	90	12	280	35
6	31	94	13	520	0
7	35	80			

表 2.3　第一组数据的均差表

x_i	$f(x_i)$	一阶	二阶	三阶	四阶
28	28				
18.68	65	−4.828326			
188	100	0.206709	0.031469		
17.76	25	0.44055	−0.25418	0.027895	
17.34	90	−154.7619	0.909425	−0.868362	0.084077

将以上所得均差代入 Newton 插值公式，即可求得插值多项式，其中 Newton 插值公式为

$$N_n(x)=f(x_0)+f[x_0,x_1](x-x_0)+f[x_0,x_1,x_2](x-x_0)(x-x_1)+\cdots$$
$$+f[x_0,x_1,\cdots,x_n](x-x_0)(x-x_1)\cdots(x-x_{n-1})$$

计算出其他两组均差后，代入公式亦可得所求多项式。通过对以上串行算法的分析可以发现计算过程中存在的并行性：

1) 3 组数据之间的计算过程完全是可以并行的。

2) 在计算其均差的过程中存在着并行，而这期间的并行性的挖掘可以对算法进行很大程度的改进。

下面对 1)中的并行性采用分而治之的思想进行处理。

对于分而治之的思想，其方法就是把一个问题分成若干子问题，然后并发求解子问题，最后合并这些子问题的结果，得到整个问题的解，也就是分而治之包含如下步骤。

第一步：把问题分成子问题 $P_1,P_2,\cdots,P_s$，并发求解，得到解 $S_1,S_2,\cdots,S_s$。

第二步：合并这些子问题的结果，得到最后的结果。

其实串行算法中的分组就透露出了一种非常明显的并行性。在串行算法中对以上三组数据进行处理，必须依次进行计算。而在并行算法中，利用多处理器系统，完全可以对其进行并行处理，也就是说，3 组数据可以用 3 台处理器的系统进行并行处理。同样针对不同问题选取多处理器系统，可以对不同问题进行并行处理。

然而，通过仔细分析可见，在计算均差的过程中，采用二叉树模型同样可以对其进行并行处理，并且通过对均差计算的并行处理可以大大地提高算法的效率。所以，这里对

2)中的并行性采用二叉树模型进行改进,并且本文也将主要针对均差计算采用二叉树模型对 Newton 插值法进行研究与改进。

接下来,对问题中的第一组数据中的 5 个数进行并行计算,并将其推广到一般情况。如图 2.1 所示,为 5 个数建立二叉树。

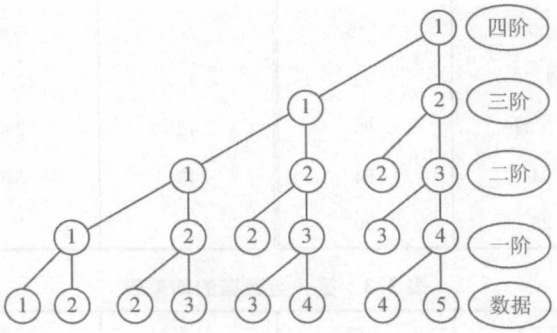

图 2.1 为 5 个数建立二叉树

为清楚表述,树中均以数字代替真实数据,最底层为所给数据,以上为相应各阶均差,每一层重复数字表示需要用两次,并且除最底层外,其他各层数字为 1 的节点为均差表中对角线上所对应的均差。

假定现在有 4 个可用的处理器,分别记为 P_1,P_2,P_3,P_4,则并行运算过程如下。

一阶均差:经处理器 P_1 计算得$(F_{11})f[x_0,x_1]$,经处理器 P_2 计算得$(F_{21})f[x_1,x_2]$,经处理器 P_3 计算得$(F_{31})f[x_2,x_3]$,经处理器 P_4 计算得$(F_{41})f[x_3,x_4]$。

二阶均差:经处理器 P_1 计算得$(F_{22})f[x_0,x_1,x_2]$,经处理器 P_2 计算得$(F_{32})f[x_1,x_2,x_3]$,经处理器 P_3 计算得$(F_{42})f[x_2,x_3,x_4]$。

三阶均差:经处理器 P_1 计算得$(F_{33})f[x_0,x_1,x_2,x_3]$,经处理器 P_2 计算得$(F_{43})f[x_1,x_2,x_3,x_4]$。

四阶均差:经处理器 P_1 计算得$(F_{44})f[x_0,x_1,x_2,x_3,x_4]$。

可以清楚地看到,每计算一阶均差,处理器的数目就减少一个,这样将 3 组数据的均差计算完毕后,再次结合组与组之间的并行性关系,此时用 3 个处理器 P_1,P_2,P_3 并行运算就可求出各组的 Newton 插值多项式,最后将子问题结果合并得最终所要结果。

加速比与效率分析:

现在来分析上述算法的加速比和效率。至于算法运算速度的快慢,在很多参考书中都是以算法中进行基本运算的次数的多少,即所需乘、除法运算步和加、减法运算步次数的多少来衡量的,而多数情况下又主要取决于乘、除运算次数的多少,做乘除法运算次数少的算法,即运算速度快的算法,是较好的算法。

对于本文中的算法,同样可以用运算次数的多少来衡量,但为了更加明显地体现本文算法的并行性和优越性,在此,不需考虑单个数学运算的单位时间,而只把进行一次均差的计算看做一个时间单位 T,便能对问题进行很好的说明。所以,基于以上分析,在不考虑通信开销和软件开销时,可进行以下计算。

对于串行算法,一组数据均差的计算时间为 $1T+2T+3T+4T=10T$,则总的运算时