密级: _____

中国科学院大学
University of Chinese Academy of Sciences

博士学位论文

若干符号数值混合计算问题的理论和算法研究

作者姓名: 陈经纬

指导教师: 冯勇 研究员

中国科学院成都计算机应用研究所

学位类别: 工学博士

学科专业: 计算机软件与理论

研 究 所: 中国科学院成都计算机应用研究所

二〇一三年五月

Typeset by LaTeX 2$_\varepsilon$ at May 27, 2013

With (modified) package `CASthesis` v0.1j of CTEX.ORG

Study on Theories and Algorithms for Several Problems in Symbolic-Numeric Hybrid Computation

By
Jingwei Chen

A Dissertation Submitted to
The University of Chinese Academy of Sciences
In partial fulfillment of the requirement
For the degree of
Doctor of Philosophy in Computer Software and Theory

Chengdu Institute of Computer Application
Chinese Academy of Sciences

May, 2013

关于学位论文使用权声明

任何收存和保管本论文各种版本的单位和个人，未经著作权人授权，不得将本论文转借他人并复印、抄录、拍照、或以任何方式传播。否则，引起有碍著作权人著作权益之问题，将可能承担法律责任。

.....

关于学位论文使用授权的说明

本人完全了解中国科学院大学和中国科学院成都计算机应用研究所有关保存、使用学位论文的规定，即：中国科学院成都计算机应用研究所有权保留学位论文的副本，允许该论文被查阅；中国科学院成都计算机应用研究所可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定。)

作者签名：　　导师签名：　　日期：

.....

关于学位论文原创性的声明

本人郑重声明：所呈交的学位论文是本人在导师的指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

作者签名：　　导师签名：　　日期：

摘　　要

随着计算机计算能力的迅猛发展, 计算机所能处理问题的规模越来越大, 提高计算的可信性和高效性已成为工业界和科学界的共同目标. 符号计算可以得到问题的精确结果, 但计算复杂度高; 数值计算可以高效地处理实际问题, 但仅能得到近似结果. 因此, 由符号和数值计算各自的优点孕育而生的符号-数值混合计算已成为近年来一个炙手可热的研究方向. 然而, 准确值和近似值之间存在一个天然的鸿沟. 幸运的是, 张景中、冯勇等提出了 “采用近似计算获得准确值” 的思想, 并成功给出了针对有理数域的解决方案. 本文着眼于将上述思想的适用范围扩大至代数数域, 讨论若干相关的计算问题, 设计、分析和实现解决这些问题的高效、可信的算法, 并探讨它们在符号-数值混合计算中的应用.

本文的主要创新在于:

一、 发现了欧几里得格和线性空间的求交问题与欧几里得空间中有限生成加法子群的分解问题之间的对偶关系, 以此为基础对二十世纪十大算法之一的整数关系探测算法 PSLQ 给出了一个新的理解, 并从中得到第一个计算欧几里得空间中有限生成加法子群分解的算法.

二、 克服了 PSLQ 算法对于一组复数只能输出 Gauss 整数关系的不足, 提出 SIRD 同步整数关系探测算法. 并以此为基础提出了解决代数数极小多项式近似重构问题的一个新的、完整的、有效的解决方案.

三、 给出了一个计算有理系数二元多项式有理不可约因子的新算法. 理论分析和实验数据均说明, 该算法非常高效, 尤其是对稀疏的有理系数二元多项式, 效果甚佳.

本文也讨论了一个基于高次代数数近似值的有理系数二元多项式因式分解算法, 该算法对较小规模的实例具有较好的效果.

此外, 对于著名的 LLL 算法的一种特殊情形, 本文采用一个全新的复杂度分析方法得到了一个有趣的结果.

关键词： 欧几里得格, 整数关系, 代数数, 极小多项式, 因式分解

Abstract

With the rapid development of the computing power of the computer, the scale of the problems that computers are able to deal with grows very fast. Meanwhile, it's very urgent and important to compute with high performance and high reliability. Symbolic computation produces exact solutions but has high complexities due to large symbolic expression; in contrast, numeric computation is fast but only produces approximate solutions. Symbolic-numeric hybrid computation takes advantage of symbolic and numeric computation, respectively, hence has been a very hot research topic recently. However, there is a natural gap between approximate and exact values. Fortunately, Zhang and Feng proposed the idea of "obtaining exact value by approximate computations" and an algorithm to deal with the case of the rational number field. In this dissertation, we aim at extending their idea to the algebraic number field, discuss some related computational problems, present, analyze and implement several efficient and reliable algorithms for those problems, and apply these algorithms to polynomial factorization by using symbolic-numeric computation.

The main innovation in this dissertation includes:

1. We disclose the relationship between the problem of computing the intersection between a euclidean lattice and a euclidean vector space and the problem of computing the decomposition of a finitely generated additive subgroup of a eculidean vector space. Based on this relationship, we give a new view on the PSLQ algorithm that is one of the top ten algorithms in the twentieth century. Further, we extract from them the first algorithm for manipulating finitely generated additive subgroups of a euclidean space.
2. We propose the SIRD algorithm for finding simultaneous integer relations for several real vectors. This algorithm overcomes the shortcoming of PSLQ, which only computes a Gaussian integer relation for a complex vector, and leads to a new, complete and efficient algorithm for reconstructing the minimal polynomial of an algebraic number from its approximation.

3. We present a new symbolic-numeric algorithm to factorize bivariate polynomials over the rationals. Both theoretical analysis and experiments show that this algorithm is efficient, especially when the input is sparse.

Also, we give another algorithm which exactly computes bivariate polynomial factorization by approximations of high degree algebraic numbers. It is relevant for many instances with small size.

At last but not least, for the celebrated LLL algorithm, we give a completely new complexity analysis for a special case and obtain an interesting result.

Keywords: euclidean lattice, integer relation, algebraic number, minimal polynomial, polynomial factorization

目　录

表　格

插　　图

第一章　引言

自 1946 年第一台电子计算机 ENIAC 问世以来, 其应用已深入到政治、经济、军事、文化等社会生活的各个方面. 随着计算机计算能力的迅猛发展, 计算机所能处理问题的规模越来越大. 然而, 如何在现代计算机上高性能、低能耗、高可靠性地完成计算任务已成为近年来研究的热门内容. 而这自然地对高性能计算 [1]、绿色计算 [4] 和可信计算 [7] 等领域的研究提出了新的要求和挑战. 符号计算可以得到问题的精确解, 但容易发生中间过程膨胀 (符号表达式在计算过程中越来越庞大), 计算复杂度高; 数值计算可以高效地处理实际问题, 但仅能得到局部的近似解. 因此, 由符号和数值计算各自的优点孕育而生的符号-数值混合计算 [11, 45, 94, 151] 已成为近年来一个炙手可热的研究方向. 然而, 准确值和近似值之间存在一个天然的鸿沟. 幸运的是, 张景中、冯勇等提出了 “采用近似计算获得准确值” 的思想, 并成功给出了针对有理数域的解决方案 [150]. 本文着眼于将上述思想的适用范围扩大至代数数域, 讨论若干相关的计算问题, 设计、分析和实现解决这些问题的高效、可信的算法, 并探讨它们在符号-数值混合计算中的应用.

1.1　研究背景及意义

由于计算机中仅能够准确的表示整数 (从而,有理数), 而不能够精确地表示所有实数 (如 $\sqrt{2}$), 因此计算误差无处不在. 在可信计算需求日益增加的背景下, 如何用整数来表示实数就成了近年来一个非常热门的研究课题. 自 2008 年以来 [5], 国家自然科学基金委员会已经连续六年将 “实数的整数化表示理论与算法” 列为信息与数学交叉类项目面上项目指南的头条, 其研究内容为 “设计用整数正确表示实数的理论与算法, 并在计算机中实现该算法, 给出该算法的复杂性分析.” 本文第四章代数数极小多项式的近似重构为代数数的整数化表示提供了一种新的完整的解决方案.

这个解决方案所采用的基本方法是误差可控的符号-数值混合计算. 符号-数值混合计算不仅是应用数学和纯数学的交叉学科, 而且也是数值分析和计算机代数的交叉学科 [11, 45, 94, 151], 并且在数学与信息科学的研究中扮演着越来

越重要的角色. 2007 年, 美国国家自然科学基金委员会将符号、数值、代数计算的交叉和合作列为计算机时代科学研究创新的重要挑战性领域 (见 [11]). 欧盟也大力支持符号-数值混合计算的基础和应用研究, 法国 INRIA 开发了混合计算软件库 SYNAPS [76], 德国、意大利、荷兰的 Algebraic Oil 项目组开发的 ApCoCoA 符号-数值混合计算软件包等 [124]. 我国著名数学家、首届国家最高科学技术奖获得者吴文俊先生也将符号-数值混合计算列为 “21 世纪 100 个交叉学科难题” 之一 [10].

代数数极小多项式的近似重构所采用的基本思想是 “采用近似计算获得准确值”. 这一思想源于我国计算机科学家洪家威对近似计算有效位的讨论 [6], 后被张景中和冯勇精确提出并给出了针对有理数情形的算法 [150]. 随后, 推广到采用近似方法获得准确有理系数插值多项式 [48]. 这一思想不仅切合可信计算的目标, 而且也是 “符号-数值混合计算” 研究中的一个重要方向. 因此, 本文作者所在团队先后参与了国家自然科学基金重大研究计划 (项目编号: 91118001) 和国家 973 项目 (项目编号: 2011CB302402), 致力于 “基于符号-数值混合计算的误差可控算法” 的研究.

上述解决方案所使用的基本工具是二十世纪十大算法之一 (见 [44]) ——整数关系探测算法 PSLQ. 该算法比较完整地解决了 $\mathbb{R}^n$ 上的整数关系的探测问题, 同时其应用包括重构实代数数的极小多项式 [23, 127]、发现了一个计算圆周率 π 的新公式 [22]、发现了诸多物理常数之间的关系 [27], 进而形成了一门新的学科 “实验数学” [21, 28]. 但是, 正如 Cipra 所评论的一样: 尽管该算法功能强大, 但却十分难于理解和实施 [38]. 因此, 对 PSLQ 算法的较为直观的理解也是一个非常有意义的工作. 本文在这方面做了一些尝试, 得到一些有益的结果 (第二、三章).

对于上述解决方案来讲, 最直接的应用便是计算多项式的不可约分解. 多项式的因式分解不仅一直是符号计算的中心课题, 而且也是符号计算领域解决最彻底的问题之一 [89]. 尽管如此, 由于该问题广泛应用于符号化简、交换环中理想准素分解、多项式方程求解、代数编码和密码学等诸多方面, 所以这方面的研究仍是持续的热点. 尤其是基于符号-数值混合计算的准确多项式因式分解领域, 十分活跃. 在符号计算顶级会议 The International Symposium on Symbolic and Algebraic Computation (ISSAC) [77] 上, 每年都有学者报告这方面最新的研究进展. 在 2012 年于法国 Grenoble 召开的第 37 届 ISSAC 上, 本文

作者也将本文第五章的部分内容在会议期间作了展示.

综上所述, 本文所研究的内容是科学计算和可信软件研究中重要的基础课题, 并且属于信息科学与数学交叉学科中亟待解决的难题. 本文的工作发挥了我国在符号-数值混合计算领域的优势, 发展了基于符号-数值混合计算的若干误差可控算法, 为进一步探索和解决相关的实际问题提供了理论基础和技术支持.

1.2　国内外研究现状

一个**欧几里得格** (euclidean lattice) $\Lambda \subseteq \mathbb{R}^m$ 是欧几里得空间 $\mathbb{R}^m$ 中的一个离散的加法子群, 简称为**格**. 格 Λ 的一组**基** (basis) $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n \in \mathbb{R}^m$ 是一组满足 $\Lambda = \sum_i \mathbb{Z}\boldsymbol{b}_i$ 的线性无关的向量. 自 1896 年 Minkowski 的 *Geometrie der Zahlen*[1] [115] 问世以来, 欧几里得格已成为数论中的一个标准的工具. 欧几里得格中的基本问题是寻找格中的最短向量, 但该问题已被证明是 NP 完全的 [17, 46]. 而在诸多的实际问题中, 求得格中的较短向量就能满足需求. 由 Lenstra, Lenstra 和 Lovász 于 1982 年提出的 LLL 算法 [102] 便能求得格的一组基, 这组基由格中相对较短并接近正交的一些向量组成. 在 LLL 算法的原始论文中, 利用 LLL 算法给出了一个计算有理系数单变元多项式因式分解的算法, 该算法是第一个多项式时间复杂度的有理系数多项式因式分解算法. 此后, LLL 算法的应用犹如雨后春笋般涌现. 至今, LLL 算法已经在 Diophantine 近似 [64]、多项式的因式分解 [68]、整数规划 [13, 106] 、GPS [65]、编码理论 [105] 和密码学 [113] 等领域得到了成功的应用. 同时, 对 LLL 算法本身的改进也持续不断, 如 [69, 83, 97, 117, 121, 123, 138, 139, 141, 142].

LLL 算法的又一个典型的应用是代数数极小多项式的近似重构. 该问题最早是由著名的理论计算机科学家, 1995 年 Turing 奖获得者 M. Blum 于上世纪八十年代在研究伪随机序列时提出的. 应用 LLL 算法, Kannan, Lenstra 和 Lovász [95] 通过代数数的近似值成功获得了其对应的准确代数数的极小多项式 (第 4.1 节有此算法的简介). 该算法回答了 M. Blum 所提出的问题: 代数数的二进制展开不形成安全的伪随机序列. 计算机代数系统 Maple [108] 中的 `PolynomialTools:-MinimalPolynomial` 命令便基于 [95] 中的算法. 另外, Just 在文献 [80] 中给出了一个寻找多个代数数整数关系的算法, 该算法也能运用于代数数极小多项式的近似重构. 但是, 该算法所需要的精度太高 (详见表 4.1).

[1] "Geometrie der Zahlen" 中文意思是 "数的几何".

2009 年, 本文作者所在团队利用整数关系探测算法 PSLQ, 给出了实代数数极小多项式的近似重构方案 [8, 9, 127, 128]. 但是这些方案都仅对实代数数有效, 因为对于复数的情况, PSLQ 算法只能输出 Gauss 整数关系. 本文第四章将给出一个基于同步整数关系探测的有效算法来弥补以上工作的不足.

设 $\boldsymbol{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n$. 如果存在一个非零向量 $\boldsymbol{m} \in \mathbb{Z}^n$, 使得 $\boldsymbol{x} \cdot \boldsymbol{m}^T = 0$, 那么就说 $\boldsymbol{m}$ 是 $\boldsymbol{x}$ 的一个**整数关系** (integer relation). 整数关系的探测是一个古老的数学问题. 《几何原本》中的 Euclid 算法就可以解决有理数域上的 $n = 2$ 的整数关系探测问题, 这等价于求 x_1/x_2 的连分数表示. 在漫长的数学历史长河中, 包括 Jacobi、Hermite、Poincaré、Perron、Brun 和 Szekeres 在内的一大批数学家都试图攻克 $n > 2$ 的情形, 但是效果都不理想. 他们所采用的解决方案都集中于多维 Euclid 算法或多维连分数算法. 对 $n > 2$ 的情形, 直到上世纪 70 年代末才由 Ferguson 和 Forcade 给出了第一个行之有效的方法 [51, 52]: 多维 Euclid 算法. 自此之后的诸多该算法的推广中, Håstad, Just, Lagarias 和 Schnorr 给出的 HJLS 算法 [66, §3] 是第一个关于整数关系探测的多项式时间的算法, 而且该算法还可推广到线性独立的整数关系探测 (得到给定向量的多个线性无关的整数关系) 和同步整数关系探测 (得到给定多个向量的整数关系) 两个多项式时间算法. Just 在文献 [80] 中给出了一个针对代数数的整数关系探测算法. HJLS 算法和 [80] 中的算法都采用了部分量度约化 (见第 2.1 节), 同时应用了经典的 Gram-Schmidt 正交化过程, 因此造成了数值不稳定.[2]. HJLS 算法不稳定的例子可以参考文献 [49, 50]. 此后又经过了进一步地完善, Ferguson 等基于多维 Euclid 算法提出的 PSLQ 算法 [49, 50] 是一个较为完善的高效的整数关系探测算法. 尽管在精确的实数计算模型下, Meichsner 证明了 PSLQ 算法和 HJLS 算法之间在本质上是等价的[111, § 2.3.1] (亦见 [29, Appendix B, Theorem 7]), 但是 PSLQ 算法不仅在实际计算时间上优于之前的算法, 而且能够被推广到复数和 Hamilton 四元数的情形, 此时分别对应 Gauss 整数关系和 Hamilton 整数关系, 在现实生活中被广泛应用. 因此, *Computing in Science and Engineering* 杂志于 2000 年将其评为二十世纪十大算法之一 [44].

尽管 PSLQ 算法功能强大, 但也有不足之处. 一方面是由于其算法本身难于理解和实施, 另一方面则是由于算法本身存在一定的局限性. 比如, 若对复数向量 $\boldsymbol{x} = (1 + I, 1 + 2I, 2 + I)$ 应用 PSLQ 算法, 则只能得到一组 Gauss 整数向

[2]关于部分量度约化和完全量度约化的数值性质的差异, 可以参考文献 [140]; 关于经典的 Gram-Schmidt 正交化的数值性质可以参考文献 [63, 67].

量 $\boldsymbol{m} = (1, I, -I)$ 满足 $\boldsymbol{x}\boldsymbol{m}^T = 0$, 其中 $I = \sqrt{-1}$ 表示虚数单位. 然而在实际应用中, 往往需要一个整数向量, 而不是一个 Gauss 整数向量, 此时 PSLQ 算法便无能为力了. 比如, 应用整数关系探测算法由复代数数的近似值获得其准确的极小多项式. 因此, 文 [8, 9, 127, 128] 中运用 PSLQ 算法由代数数的近似值获得其准确的极小多项式的思路仅对实代数数有效. 基于改进的 HJLS 算法, Rössner 和 Schnorr 在 [130] 中试图给出一个计算两个实向量 $\boldsymbol{x}_1$ 和 $\boldsymbol{x}_2$ 的同步整数关系探测算法, 但是由于存在一些问题没有解决, 至今尚未公开发表. 本文第三章将给出一个稳定的同步整数关系探测算法 SIRD, 该算法将使得基于整数关系探测的代数数极小多项式近似重构方法成为一个完备的方法.

本文的另一个主题——多项式的因式分解——是符号计算 (计算机代数) 领域最迷人的研究方向之一. 其历史可以追溯到公元前 1900 年到公元前 1600 年间古巴比伦人的二次方程求根公式 [57]. 如前所述, 对于单变量的有理系数多项式的因式分解, 第一个多项式时间复杂度的算法归功于 Lenstra, Lenstra 和 Lovász [102]. 对于多变元的有理系数多项式, 多项式时间复杂度的因式分解算法是由 Kaltofen 最先给出的 [81, 82, 85]. 这方面的文献浩如烟海, 在此只简要介绍与本文工作有关的部分, 有兴趣的读者可以参考 [37, 57, 87–89, 92] 了解更多信息.

日本学者 Sasaki 等人不仅在 [135–137] 中引进了*扩展的 Hensel 构造*, 也引入了因子重组技术. 扩展的 Hensel 构造随后被应用于求解多元代数系统以及多元多项式的 Puiseux 级数分解 [74, 75, 78, 79, 132–134]. 本文将改进他们的工作, 并弥补其中的疏漏, 提出一个保持稀疏性的*推广的 Hensel 提升* (详见第 5.2.2 节).

另外, 因子重组技术已被成功应用到因式分解中, 例如用于单变元多项式因式分解的 van Hoeij 重组算法 [68], Belabas 等提出的适用于全局域上因式分解的对数-微分重组算法 [24]. 本文的算法 5.5 也用到了因子重组技术, 与已有方法的不同之处在于其建立方程的方式是通过近似赋值, 从而可以利用数值线性代数 (如 [67]) 的工具进行求解.

对于数域上的多元多项式因式分解有很多成熟的技术可以应用,如 [30, 54, 70, 87, 88, 98–101, 119, 145, 146]. 但是, 这些方法都依赖于经典的 Hensel 提升. 而经典的 Hensel 提升在提升因子的过程中会破坏原来多项式的稀疏性, 从而导致中间过程膨胀. 尽管 Bernardin 在二元的 Hensel 提升和并行算法方面做了一些工作 [25], 但其研究也没有考虑稀疏性. 为了解决这个问题, Zippel [152, 153]

进行了初步的尝试. 通过概率方法, 将随机点上赋值为 0 的系数就看成 0, 从而将提升步骤转化为求解一个线性代数方程. Von zur Gathen 和 Kaltofen 随后也研究了这一问题, 并对该方法中涉及的概率进行了严格的分析 [56, 59, 86].

基于文 [42] 中的结果, Lenstra 于 1999 年给出了一个计算代数扩张中的多项式的固定次数因子的快速算法 [103, 104]. 该算法对低次因子十分有效, 可以解决次数很高项数很少的多项式因式分解问题. 随后, 有学者将此方法推广到多元情形 [18, 90, 91]. 这方面的最新进展可以参考 [33].

关于稀疏多项式因式分解的另一个重要方向是利用多项式的支撑集的 *Newton* 多胞体 (Newton polytope) 的信息. 2001 年, Gao 采用此方法讨论了多项式的绝对不可约性 [53]. 一系列的方法随之出现, 如[14, 16] 等. 在 [15] 中, Abu Salem 给出了一个基于多胞体的稀疏二元多项式在有理数域上的分解算法. 该算法的基本思路是遍历 Newton 多胞体的 Minkowski 分解 [40, Chap. 7], 因此, 该方法对那些 Newton 多胞体的 Minkowski 分解较少的稀疏多项式具有很高的效率.

对于稠密的有理二元多项式的因式分解, 当前最好的结果归功于 Lecerf [101]. 尽管 Weimann [148] 给出了 Lecerf 算法的一个稀疏版本, 但是 Weimman 的算法要求输入多项式具有常数项和一次项, 减小了算法的适用范围. 文 [26] 给出了一个约减 Newton 多胞体积的方法也可以用于二元多项式的因式分解中.

1.3 本文工作

本文是作者攻读博士学位期间的部分工作 [2, 3, 34, 35, 149] 的整理集结. 主要内容和创新性可以归纳为如下几个方面:

- 通过引入更为一般化、功能更为强大的代数表述, 对求解一组实数的整数关系的标准算法 HJLS 和 PSLQ 重新解读, 给出了这两个算法的一个新的理解. 进而, 从这两个算法中提取出第一个处理欧氏空间中有限生成加法子群 (包括有限个欧几里得格的和、欧几里得格到线性空间的投影) 分解问题的算法, 并在新的代数意义下, 分析了该算法的收敛性和复杂度.

- 针对有限生成加法子群的分解问题, 也考查了基于 LLL 算法的方法. 对于整数情形的分析, 本文得到了一个十分有趣的结果. 该结果不同于文献中已有的 LLL 分析方法, 或有更大的价值有待发掘.

- 克服了 PSLQ 算法对于一组复数只能输出 Gauss 整数关系的不足, 提出 SIRD 同步整数关系探测算法. 并深入研究了如何在计算机代数系统 Maple 高效地实现该算法. 实验数据显示, 与文献中已有的算法相比, SIRD 算法优势明显.

- 给出了代数数极小多项式近似重构的误差控制条件, 进而基于同步整数关系探测算法 SIRD, 得到一个从代数数近似值重构其准确极小多项式的完备的新算法, 从而将 “采用近似计算获得准确值” 这一思想的适用范围从有理数扩展到代数数.

- 基于代数数域上代数数极小多项式的近似重构, 提出了一个分解有理系数二元多项式的算法 (算法 5.1), 该算法针对小规模的问题有较好的效果. 另外, 受 Sasaki 等提出的扩展的 Hensel 构造的启发, 建立了推广的 Hensel 提升. 在对有理系数二元多项式的应用中, 推广的 Hensel 提升克服了传统构造的中间过程膨胀问题, 保持了输入多项式的稀疏性. 结合基于数值线性代数的因子组合方法, 得到另一个有理系数二元多项式的因式分解算法 (算法 5.5). 实验显示, 该算法效率较高, 尤其是对稀疏的有理系数二元多项式.

本文组织. 本文共六章. 第一章简要介绍研究背景、研究动机和研究内容. 第二章揭示整数关系探测与欧几里得空间中的有限生成加法子群的联系, 从而给出 HJLS 算法和 PSLQ 算法的新的理解. 第三章建立、分析并实现同步整数关系探测算法 SIRD. 第四章基于 SIRD 同步整数关系探测算法建立代数数极小多项式近似重构的完备的新方法. 第五章提出并分析两个基于符号-数值混合计算的有理系数二元多项式的因式分解算法. 第六章总结全文.

说明. 为避免冗长的重复叙述, 本文对一些基本概念和术语都不加定义的加以使用. 那些未作说明的概念和术语均可以在 [31, 58, 143] 中找到. 如无特殊说明, 本文的所有向量均是行向量, 用黑体字母表示. 用 $\langle \boldsymbol{b}, \boldsymbol{c} \rangle$ 表示两个向量 $\boldsymbol{b}$ 和 $\boldsymbol{c}$ 的内积. 向量 $\boldsymbol{b}$ 的 2-范数为 $\|\boldsymbol{b}\| = \sqrt{\langle \boldsymbol{b}, \boldsymbol{b} \rangle}$.

本文涉及的所有程序和数据都可以在如下网址下载:

`https://sites.google.com/site/jingweichen84/`

第二章　整数关系与欧氏空间中的有限生成加法子群

称一个向量 $\boldsymbol{m} \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$ 为向量 $\boldsymbol{x} \in \mathbb{R}^n$ 的**整数关系** (integer relation) 若 $\boldsymbol{x} \cdot \boldsymbol{m}^T = 0$. 1986 年, 由 Håstad, Just, Lagarias 和 Schnorr 给出的 HJLS 算法 [66, §3] 是第一个多项式时间的算法. 该算法要么找到一个整数关系, 要么证明整数关系具有小的 2-范数. 其时间复杂度是 n 和整数关系界的多项式函数. 1992 年, Ferguson 和 Bailey 给出了另一个标准算法 PSLQ [49]. 文献 [50] 给出了 PSLQ 算法的简化分析. 有关整数关系探测的历史注记, 可以参考 [66, §1] 和 [50, Sec. 9]. 本章的计算模型将采用精确的实数计算模型. 在此模型下, Meichsner 已经证明了 PSLQ 算法在本质上等价于 HJLS 算法 [111, §2.3.1], 或见 [29, Appendix B, Theorem 7]. 因此, 本章中如无特殊说明, 将以 HJLS 算法代指 HJLS 和 PSLQ 算法.

对于 $\mathbb{R}^n$ 中的向量 $\boldsymbol{x}$, HJLS 算法旨在找到格 $\Lambda = \mathbb{Z}^n$ 与 $(n-1)$-维向量空间 $E = \operatorname{span}(\boldsymbol{x})^\perp \subseteq \mathbb{R}^n$ 的交集中的一个非零元素. 该算法可大致分为如下三个步骤: (1) 首先计算单位矩阵的行 (形成了格 Λ 的一组基) 到 E 的投影. 这便生成了 n 个属于一个 $(n-1)$-维向量空间的向量. 这 n 个向量所有的整系数线性组合所构成的集合可能不再是一个欧几里得格. 一般地, 这个集合仅仅是一个 $\mathbb{R}^n$ 中的**有限生成加法子群** (finitely generated additive subgroup). (2) 对这 n 个向量进行类似于 (尽管是不同的) LLL 算法的行幺模操作 (交换两行、一行加上另一行的整数倍). 这一步的主要目的是消除这 n 个向量的线性相关性. (3) 如果在这 n 个向量中找到了 $n-1$ 个属于同一个 $(n-2)$-维空间, 并且剩下的第 n 个向量与这 $n-1$ 个向量线性无关, 那么算法停止. 此时, 如上的第 n 个向量将不能再被其余的 $n-1$ 个向量的任意整系数线性组合约化而变得更短. 而且, 此时的幺模矩阵的逆矩阵中便包含了一个 $\boldsymbol{x}$ 的非平凡的整数关系. 这个过程中, 最大的计算开销发生在第二步, 即对有限生成加法子群的操作.

本章通过研究 $\mathbb{R}^n$ 中的有限生成加法子群的结构给出了一个理解 HJLS 算法和 PSLQ 算法的新视角. 整数关系探测算法 (部分地) 解决了如下**格与向量空间的求交问题** (Intersect) 的一种特殊情形: 给定一个欧几里得格 Λ 的一组基和一个向量空间 E 的基, 如何找到格 $\Lambda \cap E$ 的一组基 (对于整数关系探测算法 HJLS 和 PSLQ, 格 $\Lambda \cap E$ 便是所有的整数关系). 然而, HJLS 算法的主要步骤

与 $\mathbb{R}^n$ 中的有限生成加法子群的结构紧密相连. 对任意的 $\mathbb{R}^n$ 中的有限生成加法子群 $\mathcal{S}$, 其拓扑闭包 $\overline{\mathcal{S}}$ 是唯一一个欧几里得格和一个向量空间的正交和, 即 $\overline{\mathcal{S}} = \Lambda \oplus E$. $\mathbb{R}^n$ 中的*有限生成加法子群的分解问题* (Decomp) 便是对于给定的 $\mathbb{R}^n$ 中的有限生成加法子群计算其拓扑闭包的格分支和线性空间分支的基. 从这个新的角度出发, 本章将揭示 Intersect 问题和 Decomp 问题之间的对偶关系.

除了导出上述两问题之间的对偶关系, 这个新的角度还蕴含了一个分解欧式空间中的有限生成加法子群的算法 Decomp_HJLS. 在这之前, 仅有特殊的情形被专门讨论过: Pohst 的 MLLL 算法 [125] 和 [66, §2] 中所给出的算法, 从给定的一组线性相关的格向量能够计算出该格的一组基. 本章将详细地给出 Decomp_HJLS 算法, 通过改进 [50] 中的分析给出其正确性证明, 并分析算法的收敛性. 对于秩为 r 的 $\mathbb{R}^m$ 中的有限生成加法子群, 若其生成元的 2-范数不超过 X, 则 Decomp_HJLS 算法需要的迭代次数不超过 $\mathcal{O}(r^3 + r^2 \log \frac{X}{\lambda_1(\Lambda)})$, 其中 $\lambda_1(\Lambda)$ 表示输入的有限生成加法子群的格分支 Λ 的最短向量的长度. 在 Decomp_HJLS 中, 一次迭代需要的算术操作数不超过 $\mathcal{O}(nm)$, 其中 n 是有限生成加法子群生成元的个数. 同时可以证明, Decomp_HJLS 算法输出的格分支的基是*弱约化的* (weakly reduced).

本章也将考查另外一个解决 Decomp 问题的算法 Decomp_LLL, 该方法可以追溯到原始的 LLL 文献中 [102, pp. 525]. 该方法首先将输入实例嵌入到更高维的格中, 然后再调用 LLL 格约化算法. 为了确保能够从 LLL 算法的输出中得到原问题的解, 需要对嵌入的部分乘以一个很小的量. 具体地讲, 若想要分解矩阵 $A \in \mathbb{R}^{n\times m}$ 的行所生成的有限生成加法子群, Decomp_LLL 算法将对 $(c^{-1} \cdot I_n | A)$ 调用 LLL 算法, 其中 I_n 为 n 阶单位阵, 参数 $c > 0$. 对于充分大的 c, 格分支的一组基将出现在 LLL 算法输出结果的右下方. 然而, 参数 c 可能无限大, 因此 Decomp_LLL 的分析要比 Decomp_HJLS 困难得多. 幸运的是, 对于整数情形, 对充分大的参数 c, 能够证明 Decomp_LLL 所需的迭代次数独立于所选择的参数 c. 这个有趣的结果是对标准的 LLL 分析的进一步优化.

本章中, 假设所有的运算都是精确的实数计算, 即实数的加、减、乘、除、比较、取整等运算均能在单位时间内完成. 若 $\boldsymbol{b}$ 为一向量, E 为向量空间, 用 $\pi(\boldsymbol{b}, E)$ 表示 $\boldsymbol{b}$ 到 E 上的正交投影.

2.1　背景知识

本节介绍关于线性代数、欧几里得格、HJLS 算法和 PSLQ 算法的必要的背景知识. 有兴趣的读者可参考文献 [129] 了解更多关于格的基本知识.

设 $A \in \mathbb{R}^{n\times m}$ 为一个秩为 r 的矩阵. 则其有唯一的 LQ 分解 $A = L \cdot Q$, 其中 Q-因子 $Q \in \mathbb{R}^{r\times m}$ 具有正交的行 (即 $QQ^T = I_r$), L-因子 $L = (l_{i,j}) \in \mathbb{R}^{n\times r}$ 满足如下性质: 存在对角元下标 (diagonal indices) $1 \le k_1 \le \cdots \le k_r \le m$ 使得 $l_{i,j} = 0$ 对所有的 $i < k_j$ 成立, 且 $l_{k,j} > 0$ 对所有的 $j \le r$ 成立. 当 $n = r$ 时, L-因子为对角线元素全为正的下三角阵. 对矩阵 A 作 LQ 分解等价于对矩阵 A^T 作经典的 QR 分解.

定义 2.1. 设 $L = (l_{i,j}) \in \mathbb{R}^{n\times r}$ 是一个秩为 r 的下梯形矩阵 (对角线以上的元素全为零), 对角元下标为 $k_1 \le \cdots \le k_r$. 若 $|l_{i,j}| \le \frac{1}{2}\left|l_{k_j,j}\right|$ 对任意的 $i > k_j$ 成立, 则称 L 是量度约化的 (size-reduced).

给定 $L \in \mathbb{R}^{n\times r}$, 存在幺模矩阵 $U \in \mathrm{GL}_n(\mathbb{Z})$ 使得 $U \cdot L$ 是量度约化的. 计算 U 并更新 $U \cdot L$ 能够在 $\mathcal{O}(n^3)$ 次算术操作内完成.

一个欧几里得格 (euclidean lattice) $\Lambda \subseteq \mathbb{R}^m$ 是欧几里得空间 $\mathbb{R}^m$ 中的一个离散的加法子群. 在不引起混淆的情况下, Λ 也被简称作格. 格 Λ 的一组基 (basis) $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n \in \mathbb{R}^m$ 是一组满足 $\Lambda = \sum_i \mathbb{Z}\boldsymbol{b}_i$ 的线性无关的向量. 称 $B = (\boldsymbol{b}_1^T, \ldots, \boldsymbol{b}_n^T)^T \in \mathbb{R}^{n\times m}$ 为 Λ 的基矩阵 (basis matrix). 整数 n 被称作格 Λ 的维数 (dimension). 若 $n \ge 2$, 则 Λ 有无穷多组基. 若 B 为任意的 Λ 的基矩阵, 则对任意的 $U \in \mathrm{GL}_n(\mathbb{Z})$, $U \cdot B$ 也是 Λ 的基矩阵. 格 Λ 的第 i 个 *Minkowski* 极小值 (the i-th Minkowski successive minimum) $\lambda_i(\Lambda)$ $(i \le n)$ 被定义为最小的包含 Λ 中 i 个线性无关向量的球的半径. 格 Λ 的行列式 (determinant) 被定义作 $\det(\Lambda) = \sqrt{\det(BB^T)}$. 格 Λ 的对偶格 (dual lattice) $\widehat{\Lambda}$ 被定义为 $\widehat{\Lambda} = \{\boldsymbol{x} \in \mathrm{span}(\Lambda) : \forall \boldsymbol{b} \in \Lambda, \langle \boldsymbol{b}, \boldsymbol{x}\rangle \in \mathbb{Z}\}$. 若 B 是格 Λ 的一个基矩阵, 则 $(BB^T)^{-1}B$ 是 $\widehat{\Lambda}$ 的一个基矩阵, 称为 B 的对偶基 (dual basis).

例 2.1. 图 2.1 中的黑色圆点就是以向量 $\boldsymbol{b}_1 = (4, 2)$ 和 $\boldsymbol{b}_2 = (-3, 4)$ 为基的 2-维格的部分格点. 向量 $\boldsymbol{b}_3 = (-2, 10)$ 和 $\boldsymbol{b}_4 = (1, 6)$ 也是该格的一组基, 并且

$$\begin{pmatrix}\boldsymbol{b}_3\\ \boldsymbol{b}_4\end{pmatrix} = \begin{pmatrix}1 & 2\\ 1 & 1\end{pmatrix} \cdot \begin{pmatrix}\boldsymbol{b}_1\\ \boldsymbol{b}_2\end{pmatrix}.$$

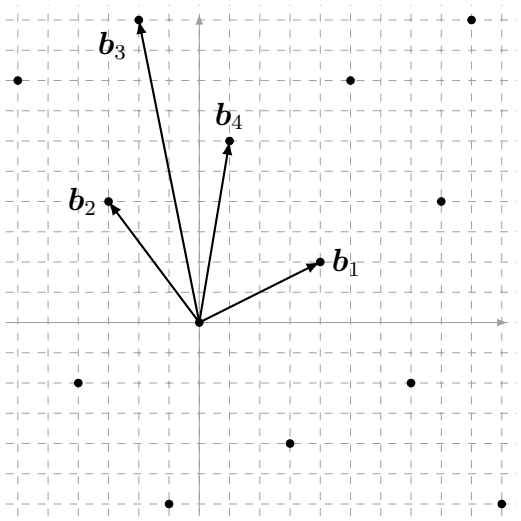

图 2.1: 欧几里得格

设 $B = (\boldsymbol{b}_1^T, \cdots, \boldsymbol{b}_n^T)^T \in \mathbb{R}^{n\times m}$ 为格 Λ 的基矩阵, $L = (l_{i,j})$ 是其 L-因子. 设 $\delta \in (1/4, 1)$. 称这组基 $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n$ 是 *LLL*-约化的 (LLL-reduced) 若 L 是量度约化的并且 *Lovász* 条件 $\delta \cdot l_{i,i}^2 \le l_{i+1,i+1}^2 + l_{i+1,i}^2$ 对 $i \le n$ 成立. 设 $\gamma = 1/\sqrt{\delta^2 - 1/4} > 2/\sqrt{3}$. 若基 $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n$ 是 LLL-约化的, 则对任意的 $1 \le j \le i \le n$ 有 [102]:

$$\begin{aligned} l_{j,j} &\le \gamma^{i-j} \cdot l_{i,i}, \\ \|\boldsymbol{b}_j\| &\le \gamma^{i-1} \cdot l_{i,i}, \\ \gamma^{-i+1} \cdot \lambda_i(\Lambda) \le \|\boldsymbol{b}_i\| &\le \gamma^{n-1} \cdot \lambda_i(\Lambda). \end{aligned} \tag{2.1}$$

弱约化是经典的 LLL-约化的弱化, 其定义类似于 [141] 中的半约化. 设 $\gamma > 2/\sqrt{3}$, $C \ge 1$. 称基 $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n$ 是弱约化的 (weakly reduced) 若 L 是量度约化的并且 (推广的) *Schönhage* 条件 $l_{j,j} \le C \cdot \gamma^i \cdot l_{i,i}$ 对 $1 \le j \le i \le n$ 成立. 按此定义, 一组基若是 LLL-约化的, 则也是以 $C = 1$ 弱约化的. 并且, 若基 $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n$ 是弱约化的, 则

$$\begin{aligned} \|\boldsymbol{b}_i\| &\le \sqrt{n} C \gamma^i \cdot l_{i,i}, \\ (\sqrt{n} C^2 \gamma^{2i})^{-1} \cdot \lambda_i(\Lambda) \le \|\boldsymbol{b}_i\| &\le \sqrt{n} C^2 \gamma^{2n} \cdot \lambda_i(\Lambda). \end{aligned} \tag{2.2}$$

事实上, 若格 Λ 的一组基 $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n$ 是弱约化的, 则

$$\begin{aligned} \|\boldsymbol{b}_i\|^2 &= l_{i,i}^2 + \sum_{j<i} l_{i,j}^2 \\ &\leq l_{i,i}^2 + \sum_{j<i} \left(\frac{1}{4}C^2 \cdot \gamma^{2i}\right) l_{i,i}^2 \\ &= \left(1 + \frac{i-1}{4}C^2 \cdot \gamma^{2i}\right) l_{i,i}^2 \\ &\leq nC^2\gamma^{2i} \cdot l_{i,i}^2. \end{aligned} \tag{2.3}$$

因此对任意的 $1 \leq j \leq i \leq n$,

$$\|\boldsymbol{b}_j\|^2 \leq nC^2\gamma^{2j} \cdot l_{j,j}^2 \leq nC^4\gamma^{2(i+j)} \cdot l_{i,i}^2. \tag{2.4}$$

又 $l_{i,i}^2 \leq \|\boldsymbol{b}_i\|^2$, 所以

$$(\sqrt{n}C^2\gamma^{2i})^{-1} \cdot \lambda_i(\Lambda) \leq \|\boldsymbol{b}_i\|. \tag{2.5}$$

现设 $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_m \in \Lambda$ 是 Λ 中线性无关的一组向量. 则 $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_m$ 可表为 $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n$ 的整系数线性组合:

$$\boldsymbol{c}_i = \sum_{k \leq n} x_{k,i}\boldsymbol{b}_k \ \ (x_{k,i} \in \mathbb{Z}, 1 \leq k \leq n, 1 \leq i \leq m).$$

设 $k(i)$ 是满足 $x_{k,i} \neq 0$ 的最大下标 k. 不失一般性, 可设 $k(1) \leq k(2) \leq \cdots \leq k(m)$. 则 $i \leq k(i)$ (因为 $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_m$ 是线性无关的), 并且

$$\|\boldsymbol{c}_i\|^2 \geq l_{k(i),k(i)}^2. \tag{2.6}$$

于是, 由弱约化的定义, 并结合式 (2.4) 和式 (2.6), 可得

$$\begin{aligned} \|\boldsymbol{b}_i\|^2 &\leq nC^4\gamma^{2(k(i)+i)} \cdot l_{k(i),k(i)}^2 \\ &\leq nC^4\gamma^{4n} \cdot l_{k(i),k(i)}^2 \\ &\leq nC^4\gamma^{4n} \cdot \|\boldsymbol{c}_i\|^2. \end{aligned}$$

现假设 $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_i$ 是线性无关的格向量, 且恰好达到第 i 个 Minkowski 极小值. 则对 $1 \leq j \leq i$ 有 $\|\boldsymbol{c}_j\| \leq \lambda_i(\Lambda)$ 成立, 从而式 (2.2) 中的第二个方程成立.

现在, 采用 PSLQ 算法 [50] 的形式来描述 HJLS 算法 [66, §3]. 称此算法为 HJLS-PSLQ 算法 (算法 2.1). 给定 $\boldsymbol{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n$, HJLS-PSLQ 要么返

回一个 $\boldsymbol{x}$ 的整数关系, 要么返回一个 $\lambda_1(\Lambda_x)$ 的下界, 其中 Λ_x 表示 $\boldsymbol{x}$ 的所有的整数关系形成的格, $\lambda_1(\Lambda_x)$ 表示格 Λ_x 中非零最短向量的 2-范数. 在步骤 1b, 2a, 2b 和 2c 中, 更新矩阵 U 和 Q 以使得 $UL_x = LQ$ 总是成立. [1]

算法 2.1 (HJLS-PSLQ). 输入: $\boldsymbol{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n$ 满足 $x_i \neq 0$ $(i \leq n)$; $M > 0$; $\gamma > 2/\sqrt{3}$.

输出: 要么输出 $\boldsymbol{x}$ 的一个整数关系, 要么断言 $\lambda_1(\Lambda_x) > M$.

1. (a) 令 $\boldsymbol{x} := \boldsymbol{x}/\|\boldsymbol{x}\|$; 令 $U := I_n$, $Q := I_{n-1}$.
   (b) 计算矩阵 $(\boldsymbol{x}|I_n)^T$ 的 Q-因子 $(\boldsymbol{x}^T|L_x)^T$; 令 $L := L_x$; 量度约化 L 并更新矩阵 U.
2. 当 $l_{n-1,n-1} \neq 0$ 且 $\max_i l_{i,i} \geq 1/M$ 时执行以下步骤:
   (a) 选择下标 k 满足 $\gamma^k \cdot l_{k,k} = \max_{j<n} \gamma^j \cdot l_{j,j}$; 交换矩阵 L 的第 k 行和第 $k+1$ 行并更新矩阵 U;
   (b) 计算矩阵 L 的 LQ 分解; 令 L 为新得到的 L-因子, 更新 Q.
   (c) 量度约化 L 并更新矩阵 U.
3. 若 $l_{n-1,n-1} \neq 0$, 则返回 "$\lambda_1(\Lambda_x) > M$". 否则, 返回矩阵 U^{-1} 的最后一列.

为了证明算法的终止性, HJLS 算法只需要在交换之前满足部分量度约化条件 $|l_{k+1,k}| \leq \frac{1}{2} l_{k,k}$, 而无需采用 PSLQ 所采用的完全量度约化. 由于更强的约化条件, 且 PSLQ 算法中额外加入了 $(\boldsymbol{x} \cdot U^{-1})_j$ 是否为零的测试 (若为零, 则 U^{-1} 的第 j 列就是一个 $\boldsymbol{x}$ 的整数关系), 所以对于某些特殊情形可能比 HJLS 提前终止. 除此之外, 如果 HJLS 也采用完全量度约化, 那么 PSLQ 算法等价于 HJLS [111, § 2.3.1] (亦见 [29, Appendix B, Theorem 7]). 在精确的实数计算模型下, 完全量度约化与部分量度约化并不会影响算法本身所需要的迭代次数. 事实上, HJLS 算法需要的迭代次数不超过 $\mathcal{O}(n^3 + n^2\lambda_1(\Lambda_x))$, PSLQ 算法也建立了相同的迭代次数上界. 不失一般性, HJLS 的原始描述中采用了 $\gamma = \sqrt{2}$. 量度约化的种类和参数 γ 在位复杂度模型下可能有着显著的影响, 比如完全量度约化可以让矩阵的位数保持得比较相当 [66, 102]; 若采用浮点运算, 则可能也会影响近似的精度 [50]. 然而, 这几个方面都已超出了本章讨论的范围.

[1] 实际上, 存储和更新 Q 对算法的执行来说不是必须的, 也没有出现在 [50] 中. 这里之所以更新是为了第 2.3 节中描述的方便.

2.2　Decomp 问题和 Intersect 问题

本节将在精确实数计算模型下给出 Decomp 问题和 Intersect 问题的等价性, 即可以调用解决其中一个问题的预言机 (oracle) 去解决另外一个问题.

2.2.1　$\mathbb{R}^m$ 中的有限生成加法子群

由向量 $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n \in \mathbb{R}^m$ 生成的*有限生成加法子群* (finitely generated additive subgroup) 是 $\boldsymbol{a}_i$ 所有整系数线性组合形成的集合:

$$\mathcal{S} = \sum_{i=1}^{n} \mathbb{Z}\boldsymbol{a}_i = \left\{ \sum_{i=1}^{n} z_i \boldsymbol{a}_i : z_i \in \mathbb{Z} \right\} \subseteq \mathbb{R}^m. \tag{2.7}$$

给定如式 (2.7) 的有限生成加法子群 $\mathcal{S}$, 称矩阵$A \in \mathbb{R}^{n\times m} = (\boldsymbol{a}_1^T, \cdots, \boldsymbol{a}_n^T)^T$ 为 $\mathcal{S}$ 的*生成矩阵* (generating matrix). 矩阵 A 的秩被称为 $\mathcal{S}$ 的*秩*. 若 $U \in \mathrm{GL}_n(\mathbb{Z})$, 则 $U \cdot A$ 也是 $\mathcal{S}$ 的生成矩阵.

如果向量 $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n$ 线性无关, 那么 $\mathcal{S}$ 是一个格, 并且 $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n$ 形成该格的一组基. 如果 $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n$ 线性相关, 但是存在线性无关的向量 $\boldsymbol{b}_i$ 使得 $\mathcal{S} = \sum_{i=1}^{d} \mathbb{Z}\boldsymbol{b}_i$, 那么 $\mathcal{S}$ 仍然是一个格. 在这种情形下, $\boldsymbol{a}_i$ 便不再是格 $\mathcal{S}$ 的一组基, 并且 $\dim(\mathcal{S}) < n$.

本章也将讨论第三种情形, 即 $\mathcal{S}$ 不是一个格. 对应这种情形的一个简单实例便是 $\mathcal{S} = \mathbb{Z} + \alpha\mathbb{Z}$, 其中 $\alpha \in \mathbb{R} \setminus \mathbb{Q}$: $\mathcal{S}$ 中包含了一些非零元素但却无限接近于 0, 从而便不再是格. 更一般地, 任意的 $\mathbb{R}^m$ 中的有限生成加法子群都可以看做是有限个欧几里得格的和.

同时, $\mathbb{R}^m$ 中的有限生成加法子群也可以看作欧几里得格到向量空间的正交投影. 设 $\Lambda = \sum_i \boldsymbol{b}_i \subseteq \mathbb{R}^m$ 为一个格, $E \subseteq \mathbb{R}^m$ 为一向量空间. 从 Λ 到 E 的*正交投影* (orthogonal projection) 定义为集合 $\pi(\Lambda, E) = \{\boldsymbol{v}_1 \in E : \exists \boldsymbol{v}_2 \in E^\perp, \boldsymbol{v}_1 + \boldsymbol{v}_2 \in \Lambda\}$. 则 $\pi(\Lambda, E)$ 是 $\mathbb{R}^m$ 中的一个由 $\boldsymbol{b}_i$ 到 E 的正交投影生成的有限生成加法子群. 反之, 给定一个 $\mathbb{R}^m$ 的有限生成加法子群 $\mathcal{S}$, 设其生成矩阵为 $A \in \mathbb{R}^{n\times m}$. 考虑以 $(I_n|A)$ 为基矩阵的格 $\Lambda \subseteq \mathbb{R}^{n+m}$ 和向量空间 $E = \mathrm{span}(I_n|0)^\perp \subseteq \mathbb{R}^{n+m}$. 则 $\mathcal{S} = \pi(\Lambda, E)$.

2.2.2　Decomp 问题和 Intersect 问题

考虑有限生成加法子群 $\mathcal{S} \subseteq \mathbb{R}^m$ 的*拓扑闭包* (topological closure) $\overline{\mathcal{S}}$, 即由 $\mathcal{S}$ 中所有收敛序列的极限构成的集合. 因此 $\overline{\mathcal{S}}$ 是一个 $\mathbb{R}^m$ 中的闭子群.

引理 2.1 ([31], Chapter VII, Theorem 2)**.** *设 G 为 $\mathbb{R}^n$ 的闭子群. 则存在一个包含在 G 中的最大的向量子空间 V 使得: 若 W 是 V 的补空间, 则 $W \cap G$ 是离散的且 G 是 V 和 $W \cap G$ 的直和.*

由此引理知,存在唯一的格 $\Lambda \subseteq \mathbb{R}^m$ 和唯一的向量空间 $E \subseteq \mathbb{R}^m$ 使得它们的直和为 $\overline{\mathcal{S}}$, 并且由格 Λ 张成的向量空间 $\mathrm{span}(\Lambda)$ 与 E 正交, 记该直和分解为 $\overline{\mathcal{S}} = \Lambda \oplus E$. 具体地讲, 若 $\mathrm{rank}(\mathcal{S}) = \dim(\mathrm{span}(\mathcal{S})) = r \leq m$, 则存在 $0 \leq d \leq r$, $(\boldsymbol{b}_i)_{i \leq d}$, $(\boldsymbol{e}_i)_{i \leq r-d}$ 满足:

1. $\overline{\mathcal{S}} = \sum_{i \leq d} \mathbb{Z}\boldsymbol{b}_i + \sum_{i \leq r-d} \mathbb{R}\boldsymbol{e}_i$;
2. $\boldsymbol{b}_i$ $(i \leq d)$ 和 $\boldsymbol{e}_i$ $(i \leq r-d)$ 这 r 个向量线性无关;
3. 对任意的 $i \leq d$, $j \leq r-d$, $\langle \boldsymbol{b}_i, \boldsymbol{e}_j \rangle = 0$ 成立.

于是 $(\boldsymbol{b}_i)_{i \leq d}$ 和 $(\boldsymbol{e}_i)_{i \leq r-d}$ 分别为格 Λ 和向量空间 E 的基. 称 Λ 为 $\mathcal{S}$ 的**格分支** (lattice component), 称 E 为 $\mathcal{S}$ 的**向量空间分支** (vector space components). 同时, 定义 $\mathcal{S}$ 的 ΛE **分解** (ΛE decomposition) 为 (Λ, E).

例 2.2. 设 $\boldsymbol{a}_1 = (1, 0)$, $\boldsymbol{a}_2 = (\sqrt{2}, 0)$, $\boldsymbol{a}_3 = (1, 1)$. 则 $\mathcal{S} = \sum_{i=1}^{3} \mathbb{Z}\boldsymbol{a}_i$ 为一个 $\mathbb{R}^2$ 中的有限生成加法子群. 如图 2.2 所示, $\overline{\mathcal{S}} = \mathbb{Z} \cdot (0, 1) \oplus \mathbb{R} \cdot (1, 0)$.

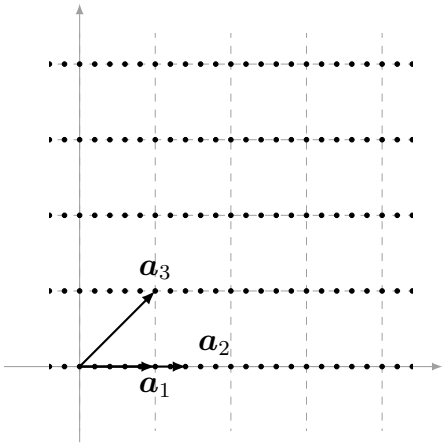

图 2.2: $\mathbb{R}^2$ 中一个有限生成加法子群

定义 2.2. *Decomp* 问题定义如下: 给定有限生成加法子群 $\mathcal{S}$ 的一组生成元, 目标是计算 $\mathcal{S}$ 的 ΛE 分解, 即计算其格分支 Λ 和向量空间分支 E 各自的一组基.

下面的引理对第 2.4 节中的算法分析起着至关重要的作用. 它将 Decomp 问题转化为计算一组有限生成加法子群的生成元使得这组生成元中包含尽可能多的线性无关的向量.

引理 2.2. *设 $(\boldsymbol{a}_i)_{i\le n}$ 是一个有限生成加法子群 $\mathcal{S}$ 的一组生成元, 其 ΛE 分解为 $\overline{\mathcal{S}} = \Lambda \textcircled{+} E$. 对 $k < n$, 定义 $\boldsymbol{a}'_j$ 为 $\boldsymbol{a}_j$ 到向量空间 $\mathrm{span}(\boldsymbol{a}_i)_{i\le n-k}$ 的正交投影 $(n-k+1 \le j \le n)$. 若 $\boldsymbol{a}'_j$ 线性无关, 则 $\boldsymbol{a}'_{n-k+1}, \ldots, \boldsymbol{a}'_n$ 为 Λ 到某个向量空间的投影的基, 且 $E \subseteq \mathrm{span}(\boldsymbol{a}_i)_{i\le n-k}$. 进一步地, 若 $k = \dim \Lambda$, 则 $\Lambda = \sum_{n-k+1\le i\le n} \mathbb{Z}\boldsymbol{a}'_i$ 且 $E = \mathrm{span}(\boldsymbol{a}_i)_{i\le n-k}$.*

该引理的证明可由 ΛE 分解的定义直接得到. 事实上, 由引理 2.1 知, 向量空间 E 是 $\mathrm{span}(\mathcal{S})$ 的包含在 $\overline{\mathcal{S}}$ 中的最大的子空间. 这意味着 $E \subseteq \mathrm{span}(\boldsymbol{a}_i)_{i\le n-k}$. 这些 $\boldsymbol{a}'_j$ 线性无关, 从而可以构成一个与 E 正交的离散子群. 再由 ΛE 分解的唯一性, 当 $k = \dim \Lambda$ 时, 向量 $\boldsymbol{a}'_{n-k+1}, \ldots, \boldsymbol{a}'_n$ 形成了格分支 Λ 的一组基.

现在, 引入另外一个计算问题 Intersect. 它可以看作是整数关系探测问题的推广.

定义 2.3. 定义 *Intersect* 问题如下: 给定格 Λ 和向量空间 E 的一组基, 目标是计算格 $\Lambda \cap E$ 的一组基.

整数关系探测问题对应于给定 $\Lambda = \mathbb{Z}^n$, $E = \mathrm{span}(\boldsymbol{x})^\perp$, 计算 $\Lambda \cap E$ 中的一个非零向量. 在此情形下, Intersect 问题旨在找到所有的 $\boldsymbol{x}$ 的整数关系. 当 E 任意, 但 Λ 仍为 $\mathbb{Z}^n$ 时, Intersect 问题对应于同步整数关系探测. 此种特殊情形可以参考文献 [3, 66], 或详见本文第三章.

在第 2.4 小节中将给出一个解决 Decomp 问题的算法. 由于不能够决定格分支的维数 (类似地, 不能够决定整数关系是否存在), 在算法中将假设格分支的维数已知.

2.2.3　Decomp 问题和 Intersect 问题之间的联系

本小节将揭示 Decomp 问题和 Intersect 问题之间的关系. 为了阐释这一关系, 需要引入如下概念.

定义 2.4. 欧几里得空间 $\mathbb{R}^m$ 中的有限生成加法子群 $\mathcal{S}$ 的对偶格 (dual lattice) 定义为

$$\widehat{\mathcal{S}} = \{\boldsymbol{x} \in \mathrm{span}(\mathcal{S}) : \forall \boldsymbol{b} \in \mathcal{S}, \langle \boldsymbol{x}, \boldsymbol{b}\rangle \in \mathbb{Z}\}.$$

等价地, $\mathcal{S}$ 的对偶也可采用如下定义:

$$\widehat{\mathcal{S}} = \{\boldsymbol{x} \in \mathrm{span}(\mathcal{S}) : \forall \boldsymbol{b} \in \overline{\mathcal{S}}, \langle \boldsymbol{x}, \boldsymbol{b} \rangle \in \mathbb{Z}\}.$$

实际上, 对任意的 $\boldsymbol{b} \in \overline{\mathcal{S}}$, 存在一个 $\mathcal{S}$ 中的收敛序列 $\{\boldsymbol{b}_i\}$ 使得 $\boldsymbol{b}_i \to \boldsymbol{b}$ $(i \to \infty)$. 因此, 对任意的 $\boldsymbol{x} \in \widehat{\mathcal{S}}$, 有 $\langle \boldsymbol{x}, \boldsymbol{b} \rangle = \langle \boldsymbol{x}, \lim_i \boldsymbol{b}_i \rangle = \lim_i \langle \boldsymbol{x}, \boldsymbol{b}_i \rangle \in \mathbb{Z}$ 成立. 在下文中, 将根据实际情况选择这两个等价的描述作为 $\widehat{\mathcal{S}}$ 的定义.

特别地, 当 $\mathcal{S}$ 为格时, $\widehat{\mathcal{S}}$ 就是 $\mathcal{S}$ 的对偶格. 更加有趣的是, 欧几里得空间中的任意有限生成加法子群的对偶都是一个欧几里得格.

引理 2.3. *设 $\mathcal{S}$ 为一个欧几里得空间中的有限生成加法子群, Λ 是它的格分支. 则 $\widehat{\Lambda} = \widehat{\mathcal{S}}$.*

证明. 记 $\mathcal{S}$ 的 ΛE 分解为 $\overline{\mathcal{S}} = \Lambda \oplus E$. 对任意的 $\boldsymbol{x} \in \overline{\mathcal{S}}$, 存在唯一的 $\boldsymbol{x}_\Lambda \in \Lambda$ 和 $\boldsymbol{x}_E \in E$ 使得 $\boldsymbol{x} = \boldsymbol{x}_\Lambda + \boldsymbol{x}_E$ 且 $\langle \boldsymbol{x}_\Lambda, \boldsymbol{x}_E \rangle = 0$.

首先证明 $\widehat{\Lambda} \subseteq \widehat{\mathcal{S}}$. 对任意的 $\widehat{\boldsymbol{x}} \in \widehat{\Lambda}$ 和任意的 $\boldsymbol{x} \in \overline{\mathcal{S}}$, 有

$$\langle \widehat{\boldsymbol{x}}, \boldsymbol{x} \rangle = \langle \widehat{\boldsymbol{x}}, \boldsymbol{x}_\Lambda \rangle + \langle \widehat{\boldsymbol{x}}, \boldsymbol{x}_E \rangle = \langle \widehat{\boldsymbol{x}}, \boldsymbol{x}_\Lambda \rangle \in \mathbb{Z},$$

其中第二个等号成立是因为 E 和 $\mathrm{span}(\Lambda)$ 正交, $\langle \widehat{\boldsymbol{x}}, \boldsymbol{x}_\Lambda \rangle \in \mathbb{Z}$ 则可由 $\widehat{\Lambda}$ 的定义直接得到.

进一步地, 对任意的 $\widehat{\boldsymbol{x}} \in \widehat{\mathcal{S}}$ 和任意的 $\boldsymbol{x} \in \Lambda \subseteq \overline{\mathcal{S}}$, 由 $\widehat{\mathcal{S}}$ 的等价定义知 $\langle \widehat{\boldsymbol{x}}, \boldsymbol{x} \rangle \in \mathbb{Z}$, 于是 $\widehat{\boldsymbol{x}} \in \widehat{\Lambda}$. 证毕. □

由引理 2.3 便可以得到欧几里得空间中有限生成加法子群的格分支的如下描述.

引理 2.4. *设 $\mathcal{S}$ 为欧几里得空间中的一个有限生成加法子群, Λ 为其格分支. 则 $\Lambda = \widehat{\widehat{\mathcal{S}}}$.*

设 $\Lambda \subseteq \mathbb{R}^m$ 为一个格, $E \subseteq \mathbb{R}^m$ 为一个向量空间. 若 $\pi(\widehat{\Lambda}, E)$ 为一个格, 则 $\pi(\widehat{\Lambda}, E) = \widehat{\Lambda \cap E}$ (见 [109, Proposition 1.3.4]). 然而一般而言, $\pi(\widehat{\Lambda}, E)$ 不一定是格, 而仅是一个有限生成加法子群. 由定义 2.4, 可以证明以下结果. 在此基础上, 便可建立 Decomp 问题和 Intersect 问题之间的联系.

定理 2.5. *对任意的格 $\Lambda \subseteq \mathbb{R}^m$ 和向量空间 $E \subseteq \mathbb{R}^m$, 如下关系成立:*

$$\Lambda \cap E = \widehat{\pi(\widehat{\Lambda}, E)}.$$

证明. 设 $\boldsymbol{b} \in \Lambda \cap E$, $\boldsymbol{y} \in \pi(\widehat{\Lambda}, E)$. 则存在 $\widehat{\boldsymbol{b}} \in \widehat{\Lambda}$, $\boldsymbol{y}' \in E^\perp$ 使得 $\widehat{\boldsymbol{b}} = \boldsymbol{y} + \boldsymbol{y}'$. 于是

$$\langle \boldsymbol{b}, \boldsymbol{y} \rangle = \langle \boldsymbol{b}, \widehat{\boldsymbol{b}} \rangle - \langle \boldsymbol{b}, \boldsymbol{y}' \rangle = \langle \boldsymbol{b}, \widehat{\boldsymbol{b}} \rangle \in \mathbb{Z}.$$

所以 $\Lambda \cap E \subseteq \widehat{\pi(\widehat{\Lambda}, E)}$.

现设 $\boldsymbol{b} \in \widehat{\pi(\widehat{\Lambda}, E)}$. 则

$$\boldsymbol{b} \in \operatorname{span}(\pi(\widehat{\Lambda}, E)) \subseteq E.$$

进一步地, 对任意的 $\widehat{\boldsymbol{b}} \in \widehat{\Lambda}$, 由 $\widehat{\boldsymbol{b}} = \pi(\widehat{\boldsymbol{b}}, E) + \pi(\widehat{\boldsymbol{b}}, E^\perp)$ 得

$$\langle \boldsymbol{b}, \widehat{\boldsymbol{b}} \rangle = \langle \boldsymbol{b}, \pi(\widehat{\boldsymbol{b}}, E) \rangle + \langle \boldsymbol{b}, \pi(\widehat{\boldsymbol{b}}, E^\perp) \rangle = \langle \boldsymbol{b}, \pi(\widehat{\boldsymbol{b}}, E) \rangle \in \mathbb{Z}.$$

因此, $\boldsymbol{b} \in \widehat{\widehat{\Lambda}} = \Lambda$. 所以 $\boldsymbol{b} \in \Lambda \cap E$. 证毕. □

将 Decomp 问题转化为 Intersect 问题. 给定 $\mathbb{R}^m$ 中有限生成加法子群 $\mathcal{S}$ 的一组生成元 $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n$, 此处的目标是通过调用一个解决 Intersect 问题的预言机去计算 $\mathcal{S}$ 的 ΛE 分解. 欲完成此任务, 只要找到其格分支 Λ 的一组基就足够了. 由引理 2.4, 有 $\Lambda = \widehat{\widehat{\mathcal{S}}}$.

类似于第 2.2.1 小节末尾处所介绍的, 对于 $\mathcal{S}$, 可以构造格 Λ' 和向量空间 E 使得

$$\mathcal{S} = \pi(\Lambda', E).$$

由定理 2.5, $\mathcal{S}$ 的格分支 Λ 是格 $\widehat{\Lambda'} \cap E$ 的对偶格. 于是, 若以 $\widehat{\Lambda'}$ 和 E 为输入调用 Intersect 问题的预言机, 然后计算该预言机返回的格的对偶基, 则计算结果便为所求.

将 Intersect 问题转化为 Decomp 问题. 给定格 $\Lambda \subseteq \mathbb{R}^m$ 的一组基 $(\boldsymbol{b}_i)_i$ 和向量空间 $E \subseteq \mathbb{R}^m$ 的一组基 $(\boldsymbol{e}_i)_i$, 此处的目标是通过调用一个解决 Decomp 问题的预言机来计算 $\Lambda \cap E$ 的一组基.

首先计算 $(\boldsymbol{b}_i)_i$ 的对偶基 $(\widehat{\boldsymbol{b}}_i)_i$, 然后对任意的 i 计算 $\widehat{\boldsymbol{b}}'_i = \pi(\widehat{\boldsymbol{b}}_i, E)$. 设 $\mathcal{S}$ 为 $\{\widehat{\boldsymbol{b}}'_i\}$ 生成的有限生成加法子群. 现以 $(\widehat{\boldsymbol{b}}'_i)_i$ 为输入调用 Decomp 的预言机计算出 $\mathcal{S}$ 的格分支 Λ'. 于是由定理 2.5, 该格分支的对偶格便是 $\Lambda \cap E$.

由此, Intersect 问题和 Decomp 问题等价. 并且, 图 2.3 交换, 其中 Λ' 和 E' 分别表示有限生成加法子群 $\pi(\widehat{\Lambda}, E)$ 的格分支和向量空间分支, 且 $\widehat{\Lambda'} = \Lambda \cap E$.

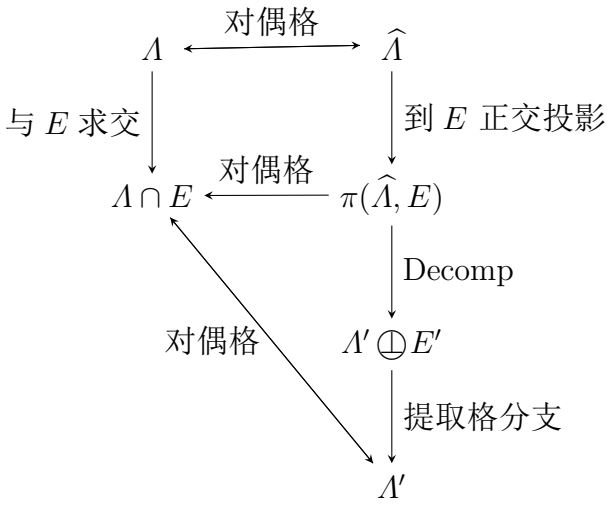

图 2.3: Intersect 问题和 Decomp 问题的等价性

2.3 HJLS-PSLQ 算法的一个新理解

本节将运用第 2.2 节的结果来阐述第 2.1 节中所描述的 HJLS-PSLQ 算法. 从一个较高的层面来理解, HJLS-PSLQ 算法是沿着第 2.2 节末尾所介绍的"将 Intersect 问题转化为 Decomp 问题" 的方向来解决整数关系探测问题. 整数关系一旦被发现, 第 2.1 节中所描述的算法立即终止. 所以 HJLS-PSLQ 仅仅是利用了一个能部分解决 Decomp 问题的方法部分解决了 Intersect 问题.

步骤 1 : 从 $\mathbb{Z}^n$ 到 $\mathrm{span}(\boldsymbol{x})^\perp$ 的投影. "将 Intersect 问题转化为 Decomp 问题" 始于计算 $\widehat{\Lambda}$ 到 E 的正交投影. 对于整数关系探测而言, $\Lambda = \widehat{\Lambda} = \mathbb{Z}^n$, $E = \mathrm{span}(\boldsymbol{x})^\perp$. 因此, 该转化的起始步骤与 HJLS-PSLQ 中步骤 1 一致. 在步骤 1 中, 计算矩阵 $(\boldsymbol{x}^T|I_n)^T$ 的 Q-因子 $Q_x := (\boldsymbol{x}^T|L_x)^T$ (因为 $\boldsymbol{x}$ 已被单位化). 由 $Q_x^T \cdot Q_x = I_n$ 可知 L_x 满足以下方程

$$\begin{pmatrix} \boldsymbol{x} \\ I_n \end{pmatrix} = \begin{pmatrix} 1 & \\ \boldsymbol{x}^T & L_x \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{x} \\ L_x^T \end{pmatrix}.$$

因此, 矩阵 L_x 是下梯形阵. 事实上, 因为矩阵 Q_x 的第 i 行正交于前 $i-1$ 行张成的子空间, 且该子空间包含了前 $i-2$ 个单位向量, 所以 Q_x 的第 i 行的前 $i-2$ 个分量为 0. 因此上述方程给出了矩阵 $\left(\boldsymbol{x}^T|I_n\right)^T$ 的 LQ 分解. 值得注意的是, L-因子和 Q-因子中都涉及矩阵 L_x 是比较特殊的. 同时, 由上述方程可知矩

阵 $\pi_x = I_n - \boldsymbol{x}^T\boldsymbol{x} = L_x L_x^T$ 是从 $\mathbb{R}^n$ 到 $\mathrm{span}(\boldsymbol{x})^\perp$ 的正交投影的变换矩阵. 因此, L_x 的行向量是 π_x 的行向量在 $\mathrm{span}(\boldsymbol{x})^\perp$ 的一组以 L_x^T 的行构成的标准正交基下的坐标向量. 综上, $(\mathbf{0}|L_x)\cdot Q_x$ 是有限生成加法子群 $\mathcal{S}_x = \pi(\mathbb{Z}^n, \mathrm{span}(\boldsymbol{x})^\perp)$ 的生成矩阵.

步骤 2: Decomp 问题的部分解答. 因为 $(\mathbf{0}|L_x)\cdot Q_x$ 为有限生成加法子群 $\mathcal{S}_x$ 的生成矩阵, 所以 HJLS-PSLQ 算法的步骤 2 仅仅是对 $\mathcal{S}_x$ 进行操作 (实际上, HJLS-PSLQ 算法仅仅对矩阵 L_x 进行操作, 因为计算出幺模矩阵 U 就足够了). 第 2.4 节将给出一个该步骤的推广以使其能够完整地解决 Decomp 问题. 由定理 2.5, 计算 $\mathcal{S}_x$ 的格分支只要计算出 $\boldsymbol{x}$ 所有的整数关系便足够了. 然而, 当 HJLS-PSLQ 运行结束时, 通常并没有计算出 $\boldsymbol{x}$ 所有的整数关系, 因此也不一定能够计算出 $\mathcal{S}_x$ 的格分支 Λ'. 若 HJLS-PSLQ 因为 $l_{n-1,n-1} = 0$ 而结束, 则步骤 2 计算出了 $\mathcal{S}_x$ 的格分支 Λ' 中一个向量到另一个 1-维向量空间的投影. 在此意义下, 步骤 2 部分地解决了输入为 $\mathcal{S}_x$ 的 Decomp 问题. 仅当 $\dim(\mathbb{Z}^n \cap \mathrm{span}(\boldsymbol{x})^\perp) = \dim(\Lambda') = 1$ 时, 步骤 2 能给出完全解 (见例 2.3).

步骤 3: 通过对偶求整数关系. 假设 HJLS-PSLQ 退出 while 循环是因为 $l_{n-1,n-1} = 0$. 由引理 2.2 知算法已经找到一个 Λ' 的 1-维投影:

$$\boldsymbol{b} := (\mathbf{0}|l_{n-1,n-1})\cdot \mathrm{diag}(1, Q)\cdot Q_x.$$

将该向量看作由其自身生成格的一组基, 则其对偶为

$$\widehat{\boldsymbol{b}} = \boldsymbol{b}/\|\boldsymbol{b}\|^2 = (\mathbf{0}|l_{n-1,n-1}^{-1})\cdot \mathrm{diag}(1, Q)\cdot Q_x.$$

由于 $\widehat{\boldsymbol{b}}$ 是 Λ' 的一个非零的基向量, 所以 $\widehat{\boldsymbol{b}} \in \widehat{\Lambda'} = \mathbb{Z}^n \cap \mathrm{span}(\boldsymbol{x})^\perp$. 又因为 Q_x 特殊的性质, 有

$$\begin{aligned}\widehat{\boldsymbol{b}} &= (\mathbf{0}|l_{n-1,n-1}^{-1})\cdot \left(\begin{array}{c|c} 1 & \\ \hline & Q \end{array}\right)\cdot \left(\begin{array}{c} \boldsymbol{x} \\ \hline L_x^T \end{array}\right) \\ &= (\mathbf{0}|l_{n-1,n-1}^{-1})\cdot \left(\begin{array}{c} \boldsymbol{x} \\ \hline QL_x^T \end{array}\right).\end{aligned}$$

现结合 $UL_x = LQ$, 便得到 $\widehat{\boldsymbol{b}} = (\mathbf{0}|l_{n-1,n-1}^{-1})\cdot(\boldsymbol{x}^T|U^{-1}L)^T = (\mathbf{0}|1)U^{-T}$. 这便解释了为什么算法所返回的整数关系是变换矩阵的逆矩阵的最后一列. 这在一定程度上出乎意料, 而这正是由于矩阵 L_x 和 Q_x 之间的相似性.

类似于上节的讨论, 图 2.4 交换.

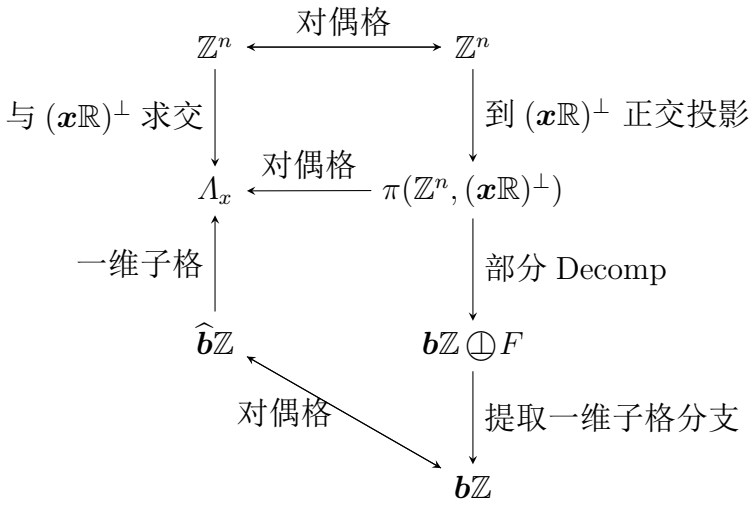

图 2.4: HJLS-PSLQ 算法的新理解

例 2.3. 运用 HJLS-PSLQ 算法寻找 $(1, \sqrt{2}, 2)$ 的一个整数关系. 单位化之后, $\boldsymbol{x} = (\frac{1}{\sqrt{7}}, \frac{\sqrt{2}}{\sqrt{7}}, \frac{2}{\sqrt{7}})$. 首先, 计算

$$L_x = \begin{pmatrix} \frac{6}{\sqrt{42}} & 0 \\ -\frac{\sqrt{2}}{\sqrt{42}} & \frac{\sqrt{2}}{\sqrt{3}} \\ -\frac{2}{\sqrt{42}} & -\frac{1}{\sqrt{3}} \end{pmatrix}, \quad Q_x = \begin{pmatrix} \frac{1}{\sqrt{7}} & \frac{\sqrt{2}}{\sqrt{7}} & \frac{2}{\sqrt{7}} \\ \frac{\sqrt{6}}{\sqrt{42}} & -\frac{\sqrt{2}}{\sqrt{42}} & -\frac{2}{\sqrt{42}} \\ 0 & \frac{\sqrt{2}}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \end{pmatrix}.$$

则矩阵 $(\mathbf{0}|L_x) \cdot Q_x$ 为有限生成加法子群 $\mathcal{S}_x = \pi(\mathbb{Z}^n, \mathrm{span}(\boldsymbol{x})^\perp)$ 的一个生成矩阵. 经过 5 次迭代之后, HJLS-PSLQ 终止. 此时,

$$L = \begin{pmatrix} \frac{15-10\sqrt{2}}{\sqrt{35}} & 0 \\ -\frac{5\left(-41+29\sqrt{2}\right)}{\sqrt{35}(-3+2\sqrt{2})} & 0 \\ \frac{41\sqrt{2}-58}{\sqrt{35}(-3+2\sqrt{2})} & \frac{1}{\sqrt{5}} \end{pmatrix},$$

$$U = \begin{pmatrix} -2 & -3 & -4 \\ 5 & 7 & 10 \\ -1 & -2 & -3 \end{pmatrix}, \; Q = \begin{pmatrix} \frac{-4+3\sqrt{2}}{\sqrt{30}(3-2\sqrt{2})} & -\frac{\sqrt{14}}{\sqrt{15}} \\ \frac{\sqrt{14}}{\sqrt{15}} & \frac{-24+17\sqrt{2}}{\sqrt{30}(17-12\sqrt{2})} \end{pmatrix}.$$

在此例中, HJLS-PSLQ 计算出了整个格分支, ΛE 分解 $\overline{\mathcal{S}}_x = \Lambda \oplus E$ 能够从 $(\mathbf{0}|L)$ 得到. 又由定理 2.5 知 $\Lambda = \widehat{\Lambda}_x$, 因此 $\dim(\Lambda) = \dim(\widehat{\Lambda}_x) = \dim(\Lambda_x) = 1$

(因为 $\boldsymbol{x}$ 包含一个无理数). 由上面的矩阵分解得

$$\Lambda = \mathbb{Z} \cdot \left(0, 0, \frac{1}{\sqrt{5}}\right) \cdot \mathrm{diag}\,(1, Q) \cdot Q_x = \mathbb{Z} \cdot \left(\frac{2}{5}, 0, -\frac{1}{5}\right)$$

且 $E = (0,1,0) \cdot \mathrm{diag}(1, Q) \cdot Q_x$. 再由定理 2.5, 得 $\mathbb{Z}^3 \cap \mathrm{span}(\boldsymbol{x})^{\perp} = \widehat{\Lambda} = \mathbb{Z} \cdot (2, 0, -1)$. 然而 HJLS-PSLQ 算法通过计算 U^{-1} 得到这组整数关系.

2.4　通过 HJLS 算法解 Decomp 问题

设矩阵 $A \in \mathbb{R}^{n\times m}$ 为有限生成加法子群 $\mathcal{S}$ 的一组生成元, $\overline{\mathcal{S}} = \Lambda \textcircled{+} E$ 为 $\mathcal{S}$ 的 ΛE 分解, 且 $\dim(\Lambda) = d$. 本节中将讨论运用 HJLS 算法解决 Decomp 问题的 Decomp_HJLS 算法.

2.4.1　Decomp_HJLS 算法

算法 2.2 (Decomp_HJLS) 是一个 Decomp 问题的完整的解决方案. 该算法得益于第 2.3 节的新理解, 同时类似于 [66, § 5] 中的同步整数关系探测算法. 这里, 仍然保持 PSLQ 的描述风格, 并参考 [112, § 2.5] 中的算法对 Decomp_HJLS 算法进行描述. 值得注意的是, 相对于 [66, § 5] 中的描述, Decomp_HJLS 算法在交换策略上略有不同: Decomp_HJLS 算法中的交换位置 κ' 可能不是 HJLS 中所选择的 $\kappa + 1$. 然而, 就像 HJLS 与 PSLQ 的区别一样 (见第 2.1 节), 这个改变不会影响到算法的渐进迭代次数.

算法 2.2 (Decomp_HJLS). 输入: 有限生成加法子群 $\mathcal{S}$ 的一个生成矩阵 $A = (\boldsymbol{a}_1^T, \cdots, \boldsymbol{a}_n^T)^T \in \mathbb{R}^{n\times m}$ ($\max_{i\le m} \|\boldsymbol{a}_i\|_2 \le X$); $\mathcal{S}$ 的格分支 Λ 的维数 d; 参数 $\gamma > 2/\sqrt{3}$.

输出: 格 Λ 的一个基矩阵.

1. (a) 计算 $r = \mathrm{rank}(A)$. 若 $d = r$, 则输出 $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_r$. 否则, 利用初等行变换使得矩阵 A 的前 r 行线性无关.
   (b) 计算矩阵 A 的 LQ 分解 $A = L_0 \cdot Q_0$.
   (c) 令 $L := L_0$ 并量度约化 L_0; 令 $Q := Q_0$, $\ell := 0$.
2. 当 $l_{r-d+1,r-d+1} \neq 0$ 时执行以下步骤:
   (a) 选择使得 $\gamma^{\kappa} \cdot l_{\kappa,\kappa} = \max_{k\le r-\ell} \gamma^k \cdot l_{k,k}$ 成立的 κ.
   (b) 若 $\kappa < r - \ell$, 则交换 L 的第 κ 行和第 $\kappa + 1$ 行; 计算 L 的 LQ 分解; 令 L 为新的 L-因子并更新矩阵 Q.

(c) 否则交换 L 的第 κ 行和第 κ' 行, 其中 $\kappa' \geq \kappa+1$ 是使得 $|l_{\kappa',\kappa}| = \max_{\kappa+1\leq k\leq n-\ell}|l_{k,\kappa}|$ 成立的最大下标. 若 $l_{\kappa,\kappa}=0$, 则令 $\ell := \ell+1$.

(d) 量度约化 L.

3. 输出 $\left(\mathbf{0}_{d\times(r-d)}|(l_{i,j})_{i\in[n-d+1,n],j\in[r-d+1,r]}\right)\cdot Q$.

尽管在算法方面, Decomp_HJLS 与 HJLS-PSLQ 的循环步骤颇为相似, 但是在 Decomp 问题的意义下来分析 Decomp_HJLS 算法并不是一件容易的事. 为了方便描述 Decomp_HJLS 算法不同阶段矩阵 L 的状态, 引进如下定义.

定义 2.5. 设 $0\leq\ell\leq r$. 若下梯形矩阵 $L\in\mathbb{R}^{n\times r}$ 能有如下的分块矩阵形式

$$L=\begin{pmatrix} M & \\ F & \\ G & N\end{pmatrix},$$

其中 $F\in\mathbb{R}^{(n-r)\times(r-\ell)}$, $G\in\mathbb{R}^{\ell\times(r-\ell)}$, 且 $M\in\mathbb{R}^{(r-\ell)\times(r-\ell)}$ 和 $N\in\mathbb{R}^{\ell\times\ell}$ 均为对角线元素全为正的下三角矩阵, 则称 L 具有 *Trap(ℓ)* 型.

Decomp_HJLS 以有限生成加法子群的生成矩阵和其格分支的维数 (见第 2.2.2 节末尾处的解释) 为输入. 不失一般性, 可以假设初始的 L-因子 $L^{(0)}$ 具有 Trap(0) 型 (步骤 1a 便能保证). Decomp_HJLS 算法的主体是对当前的生成矩阵 $L\cdot Q$ 应用幺模变换 (包括量度约化和交换) 以使得最终能得到一个 L-因子具有 Trap(d) 型, 这里 d 为格分支的维数. 这些幺模变换通过步骤 2 迭代使得 L-因子从 Trap(0) 型变化为 Trap(1) 型, ..., 最终具有 Trap(d) 型. 当后者发生时, 算法退出 while 循环, 执行步骤 3: 格分支能够通过取 L-因子的最后 d 行并令其前 $r-d$ 个分量为 0 计算得到, 其中 r 为输入的有限生成加法子群的秩.

在本节余下的部分, 将算法 Decomp_HJLS 在第 t 次迭代开始时的矩阵 L 记为 $L^{(t)}=(l_{i,j}^{(t)})$, 并用 $\ell(t)$ 和 $\kappa(t)$ 分别表示在第 t 次迭代中步骤 2a 执行后的 ℓ 和 κ 的值. 令 τ 表示总的迭代次数, 用 $L^{(\tau+1)}$ 和 $Q^{(\tau+1)}$ 分别表示步骤 3中的矩阵 L 和 Q.

2.4.2 Decomp_HJLS 的正确性

注意到若 $\kappa(t)=r-\ell(t)$ 且 $l_{\kappa'(t),r-\ell(t)}^{(t)}=0$, 则 $l_{r-\ell,r-\ell}^{(t+1)}=0$. 这是矩阵 L 从 Trap(ℓ) 型变化为 Trap($\ell+1$) 型的唯一途径. 事实上, LQ 分解、量度约化、当 $\kappa<r-\ell$ 时交换 L 连续的两行均不会改变 L 在定义 2.5 意义下的型.

接下来的两个引理将深入地分析 Decomp_HJLS 算法的执行过程. 特别地, 它们将被用来证明算法的终止性, 并被用来界定算法所需要的迭代次数. 一方面, 当前 L-因子中 M-部分 (定义 2.5) 的对角元的最大值不会随着迭代次数的增加而增加 (引理 2.6). 另一方面, 由于格分支的存在性, 在 M-部分的维数不变的情形下, 其对角元的最大值不会任意地减小. 因为, 一旦格分支被计算出来, 该部分对角元的最大值至少为格分支的第一个 Minkowski 极小值 (引理 2.7).

引理 2.6. *对任意的 $t \in [1, \tau]$, 有 $\max_i l_{i,i}^{(t+1)} \le \max_i l_{i,i}^{(t)}$ 成立, 其中不等号两边的下标 i 分别从 $[1, r-\ell(t+1)]$ 和 $[1, r-\ell(t)]$ 中取值.*

证明. 考虑 $L^{(t)}$ 和 $L^{(t+1)}$. 若 $\kappa = r - \ell$, 则 $l_{r-\ell,r-\ell}^{(t+1)} \le \frac{1}{2} l_{r-\ell,r-\ell}^{(t)}$. 这是因为 $L^{(t)}$ 是量度约化的, 且 $l_{j,j}^{(t+1)} = l_{j,j}^{(t)}$ 对 $1 \le j < r - \ell$ 成立. 若 $\kappa < r - \ell$, 则只需证明如下不等式成立即可:

$$\max\{l_{\kappa,\kappa}^{(t+1)}, l_{\kappa+1,\kappa+1}^{(t+1)}\} \le \max\{l_{\kappa,\kappa}^{(t)}, l_{\kappa+1,\kappa+1}^{(t)}\}. \tag{2.8}$$

因为其它的对角元素在这种情形下保持不变. 由 $\gamma > 2/\sqrt{3}$ 知在步骤 2a 中的交换条件意味着 $\left|l_{\kappa+1,\kappa+1}^{(t)}\right| \le \gamma \cdot l_{\kappa+1,\kappa+1}^{(t)} \le l_{\kappa,\kappa}^{(t)}$. 由矩阵 LQ 分解的性质可知

$$\begin{aligned} l_{\kappa,\kappa}^{(t+1)} &= \sqrt{(l_{\kappa+1,\kappa}^{(t)})^2 + (l_{\kappa+1,\kappa+1}^{(t)})^2}, \\ l_{\kappa+1,\kappa+1}^{(t+1)} &= l_{\kappa,\kappa}^{(t)} \cdot l_{\kappa+1,\kappa+1}^{(t)} / l_{\kappa,\kappa}^{(t+1)}. \end{aligned} \tag{2.9}$$

因此,

$$s \triangleq \frac{l_{\kappa,\kappa}^{(t+1)}}{l_{\kappa,\kappa}^{(t)}} \le \sqrt{\frac{1}{4} + \frac{1}{\gamma^2}} < 1, \tag{2.10}$$

并且

$$\frac{l_{\kappa+1,\kappa+1}^{(t+1)}}{l_{\kappa,\kappa}^{(t)}} = \frac{l_{\kappa+1,\kappa+1}^{(t)}}{l_{\kappa,\kappa}^{(t+1)}} = \frac{l_{\kappa+1,\kappa+1}^{(t)}}{\sqrt{(l_{\kappa+1,\kappa}^{(t)})^2 + (l_{\kappa+1,\kappa+1}^{(t)})^2}} \le 1.$$

这便证明了 (2.8). 证毕. □

引理 2.7. *设 Λ 为输入的有限生成加法子群的格分支, 其维数 $d = \dim(\Lambda) \ge 1$. 则对任意的 $t \in [1, \tau]$,*

$$\lambda_1(\Lambda) \le \max_{i \le r-\ell(t)} l_{i,i}^{(t)}.$$

证明. 矩阵 $L^{(\tau+1)}$ 具有形状 Trap(d), 且

$$\left(\mathbf{0}^{r-d}, l^{(\tau+1)}_{n-d+1,r-d+1}, \mathbf{0}^{d-1}\right) \cdot Q^{(\tau+1)}$$

属于 Λ (因为引理 2.2). 由矩阵 $Q^{(\tau+1)}$ 是正交矩阵知上述向量的 2-范数为 $l^{(\tau+1)}_{n-d+1,r-d+1}$. 因此, $\lambda_1(\Lambda) \le l^{(\tau+1)}_{n-d+1,r-d+1}$. 又因为 τ 为最后一次迭代, 步骤 2c 已经被执行, 且最后一次交换一定发生在矩阵 $L^{(\tau)}$ 的第 $\kappa(\tau) = r-d+1$ 行和第 $n-d+1$ 行之间, 所以

$$\begin{aligned}\lambda_1(\Lambda) &\le l^{(\tau+1)}_{n-d+1,r-d+1} = l^{(\tau)}_{r-d+1,r-d+1} \\ &\le \max_{i\le r-d+1} l^{(\tau)}_{i,i} \le \max_{i\le r-\ell(t)} l^{(t)}_{i,i}.\end{aligned}$$

其中的最后一个不等号可由引理 2.6 得到. 证毕. □

现在着手证明 Decomp_HJLS 算法的正确性, 即该算法输出的确为输入的有限生成加法子群的格分支的基矩阵. 同时, 也将证明这组基是弱约化的 (其定义见第 2.1 节).

定理 2.8. *若 Decomp_HJLS 算法终止 (算法的终止性见定理 2.10), 则它是正确的: 给定一个秩为 r 的有限生成加法子群 $\mathcal{S}$ 的一个生成矩阵和 $\mathcal{S}$ 的格分支的维数 d, Decomp_HJLS 算法输出 $\mathcal{S}$ 的格分支的一组以参数 γ 和 $C = \gamma^{r-d}$ 弱约化的基.*

证明. 记 $\mathcal{S}$ 的格分支为 Λ. 当算法的 while 循环终止时, L-因子 $L^{(\tau+1)}$ 具有 Trap(d) 型, 其中 $d = \dim(\Lambda)$. 由于算法中仅使用幺模矩阵进行操作, 有限生成加法子群 $\mathbb{Z}^n \cdot L^{(\tau+1)} \cdot Q^{(\tau+1)}$与输入 $\mathbb{Z}^n \cdot A$ 一致. 令 $\Lambda' = \mathbb{Z}^d \cdot \left(\mathbf{0}_{d\times(r-d)} | (l^{(\tau+1)}_{i,j})_{i\in[n-d+1,n], j\in[r-d+1,r]}\right) \cdot Q^{(\tau+1)}$ 表示算法 Decomp_HJLS 的输出. 由引理 2.2 知 $\Lambda' = \Lambda$.

将 L 右下方的 $d\times d$ 子矩阵记为 $L' \in \mathbb{R}^{d\times d}$. 现证明 L' 是量度约化的且满足 Schönhage 条件. 由于步骤 1c 和步骤 2d 中的量度约化, 整个矩阵 $L^{(\tau+1)}$ 是量度约化的. 因此只需证明 $l'_{j,j} \le \gamma^{r-d+i} \cdot l'_{i,i}$ 对 $1 \le j < i \le d$ 成立. 为此, 考查算法执行过程中的两个时刻 $t_i < t_j$: 第 t_i (t_j) 次迭代使得 $L^{(t)}$ 第一次具有 Trap($d-i+1$) (Trap($d-j+1$)) 型. 因此,

$$\begin{aligned}l'_{i,i} &= l^{(\tau+1)}_{n-d+i,r-d+i} = l^{(t_i)}_{n-d+i,r-d+i}, \\ l'_{j,j} &= l^{(\tau+1)}_{n-d+j,r-d+j} = l^{(t_j)}_{n-d+j,r-d+j}.\end{aligned}$$

由于 t_i 和 t_j 是最小的 (第一次成型), 也同时考虑步骤 2c 在第 t_i-1 和第 t_j-1 次迭代的状态. 于是有 $\kappa(t_i-1)=r-d+i$ 且 $\kappa(t_j-1)=r-d+j$. 由步骤 2a 中 κ 的选择, 并结合 $\ell(t_i-1)=d-i$ 和 $\ell(t_2-1)=d-j$ 知

$$\begin{aligned} l'_{i,i} &= l^{(t_i)}_{n-d+i,r-d+i} = \gamma^{-(r-d+i)} \cdot \max_{k\le r-d+i} \gamma^k \cdot l^{(t_i-1)}_{k,k}, \\ l'_{j,j} &= l^{(t_j)}_{n-d+i,r-d+i} = \gamma^{-(r-d+j)} \cdot \max_{k\le r-d+j} \gamma^k \cdot l^{(t_j-1)}_{k,k}. \end{aligned}$$

再由引理 2.6 和假设条件 $t_i<t_j$ 知

$$\begin{aligned} l'_{j,j} &\le \max_{k\le r-d+j} l^{(t_j-1)}_{k,k} \le \max_{k\le r-d+i} l^{(t_i-1)}_{k,k} \\ &\le \max_{k\le r-d+i} \gamma^k \cdot l^{(t_i-1)}_{k,k} = \gamma^{r-d+i} \cdot l'_{i,i}. \end{aligned}$$

证毕. □

一般地, 整数关系探测算法对某些实例可能没有那么多的先验信息可以提供. 因此, HJLS-PSLQ 算法只能排除小的整数关系存在的可能, 并不能判断这个实例的整数关系是否存在. 并且, 在实数计算模型下, 一组实数是否存在整数关系不可判定 [19]. 类似地, Decomp_HJLS 也仅仅可以用来排除输入的有限生成加法子群的格分支一些大的不变量的存在性, 而不能直接得到格分支的维数. 如果格分支的维数已经提前知晓, 那么 Decomp_HJLS 算法便可以计算出格分支的一组基. 然而如果输入的整数 d 小于格分支 Λ 的维数 d', 则 Decomp_HJLS 算法输出一个 d-维格, 该格是 Λ 的某子格到一个向量空间的正交投影, 并且可以证明

$$\lambda_{d'-d}(\Lambda) \le \sqrt{2r}\gamma^{2r} \cdot \max_{k\le r-d} l^{(\tau+1)}_{k,k}, \tag{2.11}$$

其中 τ 为算法以 d 为输入的迭代次数.

式 (2.11) 的证明. 当算法 Decomp_HJLS 以 $d<d'$ 为输入终止时, 假设经过 τ 次迭代, 矩阵 $L^{(\tau+1)}$ 具有 Trap(d) 型. 若继续运行 Decomp_HJLS 直至 d'-维的格分支 Λ 经过 $\tau'(>\tau)$ 次迭代被完全地计算出来, 则矩阵 $L^{(\tau'+1)}$ 的右下方的 $d\times d$ 子矩阵与 $L^{(\tau+1)}$ 相应位置的子矩阵相同. 如前所述, 该子矩阵形成的格正是 Λ 的某子格的投影.

假设 Decomp_HJLS 最终输出格分支 Λ 的一组基 $\boldsymbol{b}_1,\cdots,\boldsymbol{b}_{d'}$, 且这组基的 L-因子记为 $L'=(l'_{i,j})$. 则 L' 对应于 $L^{(\tau'+1)}$ 的右下方的 $d'\times d'$ 子矩阵. 由定理

2.8, 这组基是弱约化的. 于是由式 (2.3) 知

$$\begin{aligned}\lambda_{d'-d}(\Lambda) \leq & \max\{\|\boldsymbol{b}_1\|, \cdots, \|\boldsymbol{b}_{d'-d}\|\} \\ \leq & \max_{k\leq d'-d} \sqrt{d} C\gamma^i \cdot l'_{k,k} \\ \leq & \sqrt{r} C \cdot \gamma^r \cdot \max_{k\leq d'-d} l'_{k,k}.\end{aligned}$$

再由 Decomp_HJLS 算法的步骤 2 知, 对任意的 $k \leq d'-d$, 存在 $\tau < \tau_k \leq \tau'$ 使得 $l'_{k,k} = l^{(\tau_k)}_{r-d'+k,r-d'+k}$. 最后由引理 2.6 得

$$\lambda_{d'-d}(\Lambda) \leq \sqrt{r} C\gamma^r \cdot \max_{k\leq r-d} l^{(\tau)}_{k,k},$$

其中 $C = \gamma^{r-d'}$. 证毕. □

2.4.3 Decomp_HJLS 算法的终止性

这一小节将改进 [66] 中的复杂度分析以分析 Decomp_HJLS 所需的迭代次数. 对每一次迭代 $t \geq 1$, 定义

$$\pi_j^{(t)} := \begin{cases} l^{(t)}_{j,j} & \text{若 } l^{(t)}_{j,j} \neq 0, \\ l^{(t)}_{n-r+j,j} & \text{若 } l^{(t)}_{j,j} = 0 \end{cases}$$

和

$$\Pi(t) := \prod_{i=1}^{r-1}\prod_{j=1}^{i} \max\left(\pi_j^{(t)}, \gamma^{-r-1}\cdot\lambda_1(\Lambda)\right).$$

下面的引理用于量化算法中每一次迭代所取得的进展: 函数 $\Pi(t)$ 随 t 的增加而严格递减.

引理 2.9. 令 $\beta = 1/\sqrt{1/\gamma^2 + 1/4} > 1$. 则对任意的 $t \in [1, \tau]$, $\Pi(t) \geq \beta \cdot \Pi(t+1)$. 进一步地, $\Pi(1) \leq X^{\frac{r(r-1)}{2}}$ 且 $\Pi(\tau+1) \geq (\gamma^{r+1})^{-\frac{r(r-1)}{2}} \cdot \lambda_1(\Lambda)^{\frac{r(r-1)}{2}}$, 其中 $X = \max_{i\leq n}\|\boldsymbol{a}_i\|$.

证明. 该引理的第一部分的证明可通过调整 [66, Theorem 3.2] 和 [50, Lemma 9] 的证明过程获得.

当交换位置 $\kappa=\kappa(t)$ 不是最后一个非零的对角元时,

$$\frac{\Pi(t)}{\Pi(t+1)}=\left(\frac{\max\left(l_{\kappa,\kappa}^{(t)},\,\gamma^{-r-1}\lambda_1(\Lambda)\right)}{\max\left(l_{\kappa,\kappa}^{(t+1)},\,\gamma^{-r-1}\lambda_1(\Lambda)\right)}\right)^{r-\kappa}\cdot\left(\frac{\max\left(l_{\kappa+1,\kappa+1}^{(t)},\,\gamma^{-r-1}\lambda_1(\Lambda)\right)}{\max\left(l_{\kappa+1,\kappa+1}^{(t+1)},\,\gamma^{-r-1}\lambda_1(\Lambda)\right)}\right)^{r-\kappa-1}.$$

令

$$x=\frac{\gamma^{-r-1}\cdot\lambda_1(\Lambda)}{l_{\kappa,\kappa}^{(t)}\cdot s},\qquad y=\frac{\gamma^{-r-1}\cdot\lambda_1(\Lambda)}{l_{\kappa+1,\kappa+1}^{(t)}}\,.$$

由式 (2.10) 知 $0<s<1$. 则

$$\frac{\Pi(t)}{\Pi(t+1)}=\frac{\max\{\frac{1}{s},x\}}{\max\{1,x\}}\cdot\left(\frac{\max\{\frac{1}{s},x\}}{\max\{1,x\}}\cdot\frac{\max\{1,y\}}{\max\{\frac{1}{s},y\}}\right)^{r-\kappa-1}.$$

由式 (2.10) 亦知 $x\le y$. 又由引理 2.7 知

$$\lambda_1(\Lambda)\le\max_{i\le r-\ell(t)}l_{i,i}^{(t)}\le\max_{i\le r-\ell(t)}\gamma^i\cdot l_{i,i}^{(t)}\le\gamma^r\cdot l_{\kappa,\kappa}^{(t)},$$

于是 $x\le 1/(s\gamma)$. 因此 $\beta\le\gamma\le 1/(xs)$, 其中 $\beta=1/\sqrt{1/\gamma^2+1/4}$. 令 $f(x)=\frac{\max\{1/s,x\}}{\max\{1,x\}}$. 则 $0<s<1$, $0<x<1/s$ 且 $x\le y$, 所以 $f(x)\ge f(y)>0$. 于是,

$$\begin{aligned}\frac{\Pi(t)}{\Pi(t+1)}&=f(x)\cdot\left(\frac{f(x)}{f(y)}\right)^{r-\kappa-1}\\&\ge f(x)=\frac{\max\{1/s,x\}}{\max\{1,x\}}\\&=\begin{cases}\frac{1}{s}\ge\beta & \text{若}\quad x\le 1,\\ \frac{1}{xs}\ge\gamma\ge\beta & \text{若}\quad x>1.\end{cases}\end{aligned}$$

当交换位置 κ 为最后一个非零对角元所在行时, 唯一的改变是

$$l_{\kappa,\kappa}^{(t+1)}=\left|l_{\kappa',\kappa}^{(t)}\right|\le\frac{1}{2}l_{\kappa,\kappa}^{(t)},$$

所以

$$\frac{\Pi(t)}{\Pi(t+1)}\ge\frac{\max\{1/s,x\}}{\max\{1/(2s),x\}}=\begin{cases}2\ge\beta & \text{若}\quad 2xs\le 1,\\ \frac{1}{xs}\ge\beta & \text{若}\quad 2xs>1.\end{cases}$$

综上, $\Pi(t) \geq \beta \cdot \Pi(t+1)$.

而 $\Pi(1)$ 的上界可由

$$\gamma^{-r-1} \cdot \lambda_1(\Lambda) \leq \lambda_1(\Lambda) \leq \max_{i \leq r} l_{i,i}^{(1)} \leq \max_i \|\boldsymbol{a}_i\| = X$$

得到, 其中第二个不等号利用了引理 2.7 对 $t = 1$ 的情形. 而 $\Pi(\tau+1)$ 的下界是因为 $\max(\pi_j^{(\tau+1)}, \gamma^{-r-1} \cdot \lambda_1(\Lambda)) \geq \gamma^{-r-1} \cdot \lambda_1(\Lambda)$. 证毕. □

由引理 2.9 易得如下结论.

定理 2.10. *Decomp_HJLS 算法所需的迭代次数不超过* $\mathcal{O}\left(r^3 + r^2 \log \frac{X}{\lambda_1(\Lambda)}\right)$.

2.5 通过格约化解 Decomp 问题

本节将给出一个采用格约化算法, 如 LLL 算法 [102], 来求解 Decomp 问题的方法, 并给出相应的分析.

2.5.1 LLL 算法

著名的 LLL 算法以一个格的任意一组基为输入, 返回该格的 LLL-约化基. 文献 [66, §2] 中的 Lovász 算法和 Pohst 的 MLLL 算法 [125], 能够从一个格的任意一组生成元 (不一定线性无关) 计算出格的一组约化基. 为了方便, 给出下面的 LLL 算法的抽象描述.

算法 2.3 (LLL)**.** 输入: 格 Λ 的一组基 $(\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n)$.
输出: 格 Λ 的一组 *LLL*-约化基.

1. $k := 2$.
2. 当 $k \leq n$ 时, 执行以下步骤:
   (a) 量度约化 $\boldsymbol{b}_k$ (即确保 $|l_{k,i}| \leq l_{i,i}/2$ 对 $i < k$ 成立).
   (b) 若 *Lovász* 条件对 k 成立, 则 $k := k + 1$.
   (c) (LLL-交换) 否则, 交换 $\boldsymbol{b}_{k-1}$ 和 $\boldsymbol{b}_k$; 令 $k := \max\{k-1, 2\}$.
3. 输出 $(\boldsymbol{b}_1, \cdots, \boldsymbol{b}_d)$.

本节中将应用如下的 LLL-算法的一个结果. 其证明可在 LLL 的原始论文中找到 [102, 式 (1.26) 的证明], 亦见 [69, Lemma 1].

引理 2.11. *一次发生在 $\boldsymbol{b}_{k-1}$ 和 $\boldsymbol{b}_k$ 之间的 LLL-交换既不增加 $\max(l_{k-1,k-1}, l_{k,k})$, 也不减小 $\min(l_{k-1,k-1}, l_{k,k})$, 同时对任意的 $j \notin \{k-1, k\}$, $l_{j,j}$ 不变.*

在本节中, 将采用文献 [122] 中的处理方式, 只考虑算法的交换复杂度 (swap complexity), 即算法中所需的 LLL-交换次数. 比如, 上述 LLL 算法的交换复杂度为 $\mathcal{O}(n^2 \log X)$, 其中 $X = \max_i \|\boldsymbol{b}_i\|$.

想要计算一个欧几里得空间中的有限生成加法子群的 ΛE 分解, 直接调用 LLL 算法是行不通的, 因为这可能使 LLL 算法陷入死循环而不能计算出格分支. 考虑以如下矩阵

$$A = \begin{pmatrix} 1 & 0 \\ \sqrt{2} & 0 \\ x & 2 \end{pmatrix},$$

为生成矩阵的有限生成加法子群, 其中 $x \in \mathbb{R}$ 为任意实数. 算法 LLL 将始终交换第一行和第二行 (并量度约化) 而并不能找到该有限生成加法子群的格分支 $\mathbb{Z} \cdot (0, 2)$. 其原因是 LLL 算法中的交换策略仅仅是局部的, 而不是全局的 (亦见 [66, § 1-2]).

2.5.2 Decomp_LLL 算法

在算法 Decomp_LLL 中, 欲计算有限生成加法子群的 ΛE 分解, 首先引入参数 $c > 0$, 通过输入的生成矩阵 $A \in \mathbb{R}^{n\times m}$ 构造 $A_c := (c^{-1} I_n | A) \in \mathbb{R}^{n\times(m+n)}$. 则 A_c 将成为某一个欧几里得格的基矩阵. 然后调用 LLL 算法来约化这组基. 这将生成幺模矩阵 U 使得 $U \cdot A_c$ 是 LLL-约化的. 此时 $U \cdot A_c$ 形如 $(c^{-1}U | U \cdot A)$, 因此其右半部分 $U \cdot A$ 为输入的有限生成加法子群的新的生成矩阵. 该算法的基本思想是选取足够大的参数 c 使得在矩阵 A_c 中, 那些对应于格分支的向量与对应于向量空间分支的向量之间有一条明显的界限. 最理想的情况是, 经过 LLL 约化后, 矩阵 $U \cdot A$ 的前面的行非常小. 同时, 对应于格分支中的那些向量将不可能任意的小, 所以对于充分大的参数 c, 它们便使得 A_c 的一些向量非常大, 从而与向量空间分支中对应的向量分离. 综上, Decomp_LLL 算法的关键之处就在于参数 c 的选取.

算法 2.4 (Decomp_LLL)**.** *输入: 有限生成加法子群 $\mathcal{S}$ 的一个生成矩阵 $A = (\boldsymbol{a}_1^T, \cdots, \boldsymbol{a}_n^T)^T \in \mathbb{R}^{n\times m}$; $\mathcal{S}$ 的格分支 Λ 的维数 d; 参数 $c > 0$.*

输出: Λ 的一个基矩阵.

1. 构造 $A_c := (c^{-1} \cdot I_n | A)$.

2. 对 A_c 的行调用 *LLL* 算法, 记其输出为 A'_c.

3. 设 $\pi_m(A'_c)$ 表示 A'_c 的最后 m 列所形成的矩阵. 计算其 LQ 分解: $\pi_m(A'_c) = L \cdot Q$; 令 $L' := \left(\mathbf{0}_{d\times(r-d)} | (l_{i,j})_{i\in[n-d+1,n],j\in[r-d+1,r]}\right)$, 其中 $r = \mathrm{rank}(A)$; 输出 $L' \cdot Q$.

定理 2.12. 设 $c > 0$, 并令 Λ_c 表示以矩阵 A_c 为基矩阵的格. 若

$$2^{\frac{n-1}{2}} \cdot \lambda_{n-d}(\Lambda_c) < \lambda_1(\Lambda), \tag{2.12}$$

则算法 *2.4* 正确地输出格分支 Λ 的一组基. 进一步地, 对任意的生成矩阵 A, 总存在 $c_0 > 0$ 使得 $2^{\frac{n-1}{2}} \cdot \lambda_{n-d}(\Lambda_c) < \lambda_1(\Lambda)$ 对所有的 $c > c_0$ 成立. 最后, 算法 *2.4* 至多要求 $\mathcal{O}(n^2 \log(cX))$ 次 *LLL*-交换, 其中 $X = \max_i \|\boldsymbol{a}_i\|$.

证明. 因为 $\dim(\Lambda) = d$, 所以存在幺模矩阵 U 使得 UA 的 L-因子具有 $\mathrm{Trap}(d)$ 型(见定义 2.5) 且 UA 的前 $n-d$ 个向量的 2-范数 $\le 2^{-n}\lambda_1(\Lambda)$. 比如, 可以采用算法 Decomp_HJLS 中的迭代步骤来生成这样一个幺模矩阵使得 L-因子的 M-部分充分小. 于是 $c > 2^n \cdot \frac{\max_{i\le n-d}\|\boldsymbol{u}_i\|}{\lambda_1(\Lambda)}$ 便蕴含着 $\lambda_{n-d}(\Lambda_c) < 2^{\frac{1-n}{2}} \cdot \lambda_1(\Lambda)$, 这里 $\boldsymbol{u}_i$ 表示矩阵 U 的第 i 行.

令 $A'_c = (\boldsymbol{a}'^T_1, \cdots, \boldsymbol{a}'^T_n)^T$. 因为 $\boldsymbol{a}'_1, \cdots, \boldsymbol{a}'_n$ 是 LLL-约化的, 所以

$$\forall i \le n-d: \quad \|\boldsymbol{a}'_i\| \le 2^{(n-1)/2} \cdot \lambda_i(\Lambda_c) \le 2^{(n-1)/2} \cdot \lambda_{n-d}(\Lambda_c).$$

因此, 定理中选择参数 c 的条件意味着 $\|\boldsymbol{a}'_i\| < \lambda_1(\Lambda)$ 对 $1 \le i \le n-d$ 成立. 将 $\boldsymbol{a}'_i$ 中最后 m 个分量形成的 $\mathbb{R}^m$ 中的向量记为 $\pi_m(\boldsymbol{a}'_i)$. 则对任意的 $i \le n-d$, 有 $\pi_m(\boldsymbol{a}'_i) \in \mathcal{S}$ 且 $\|\pi_m(\boldsymbol{a}'_i)\| < \lambda_1(\Lambda)$. 因此 $\pi_m(\boldsymbol{a}'_i) \in E$, 其中 E 是 $\mathcal{S}$ 的向量空间分支. 又因为 $\dim(\Lambda) = d$, 结合引理 2.2 知 $E = \mathrm{span}_{i\le n-d}(\pi_m(\boldsymbol{a}'_i))$, 且算法输出确为 Λ 的一组基.

在经典的 LLL 复杂度分析中, LLL 算法所需要的 LLL-交换次数不超过 $\mathcal{O}(n^2 \log K)$, 其中 K 为输入向量的最大的 2-范数. 对于算法 2.4, 可以看作是 LLL 算法约化矩阵 $c \cdot A_c$, 则输入向量最大的 2-范数不超过 cX. 证毕. □

在实际应用中, 参数 c 可能要求任意大. 考虑如下矩阵

$$A = \begin{pmatrix} 0 & 1 \\ 1/c_0 & 1 \\ 3 & 0 \end{pmatrix}$$

生成的有限生成加法子群, 其中 c_0 为一个大无理数. 其格分支为 $\mathbb{Z}\cdot(0,1)$. 若在算法 2.4 中选取 $2 \le c \le c_0$, 则在执行 LLL 约化后, 矩阵 $(c^{-1}U|UA)$ 的子矩阵 UA 的前两行将是 $(1/c_0, 0)$ 和 $(0,1)$. 此时, Decomp_LLL 并不能够找到格分支, 这便意味着需要选取 $c > c_0$. 然而, 当 c_0 趋于无穷时, 参数 c 将选取得任意大, 即使在限定输入向量的 2-范数的情况下也不例外.

2.5.3　整数情形

如上所述, 在一般情形下, 很难给出一个参数 c 的下界. 然而, 对于 $\mathbb{Z}^m$ 中的有限生成加法子群, 能够给出一个 Decomp_LLL 算法中选取参数 c 的充分条件, 并且该条件给出的下界几乎是最优的. 更为有趣的是, 在这种情形下, 能够进一步证明, Decomp_LLL 算法所需要的 LLL-交换次数不依赖于参数 c 的选取.

参数 c. 对秩为 r 的生成矩阵 $A \in \mathbb{Z}^{n\times m}$, 记其生成的有限生成加法子群为 $\mathcal{S}$, $\mathcal{S}$ 的 ΛE 分解记为 $\mathcal{S} = \Lambda \textcircled{+} E$. 则 $\mathcal{S} = \Lambda$ 且 $d = \dim(\Lambda) = r$. 矩阵 A 的核与 $\mathbb{Z}^n$ 的交集

$$\ker_{\mathbb{Z}}(A) = \{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{x}A = \mathbf{0}\}$$

是一个格, 称为 A 的**核格** (kernel lattice). 若 $\dim(\Lambda) = d$, 则 $\dim(\ker_{\mathbb{Z}}(A)) = n - d$. 根据 [120, Proposition 2.9] 知

$$\det(\ker_{\mathbb{Z}}(A)) \le \det(\Lambda_{A^T}),$$

其中 Λ_{A^T} 为 A 的列生成的格.

对任意的向量 $\boldsymbol{v} \in \ker_{\mathbb{Z}}(A)$, $(n+m)$-维向量 $(\frac{1}{c}\boldsymbol{v}, \mathbf{0})$ 为 Λ_c 中的元素, 这便意味着

$$\lambda_{n-d}(\Lambda_c) \le c^{-1}\lambda_{n-d}(\Lambda_{ker}(A)).$$

由此,

$$c > 2^{\frac{n-1}{2}} \cdot \lambda_{n-d}(\Lambda_{ker}(A))/\lambda_1(\Lambda) \tag{2.13}$$

便蕴含了式 (2.12). 所以, 对于整数情形参数 c 的选取, 可以用式 (2.13) 中的条件代替式 (2.12) 中的条件.

实际上, 式 (2.13) 中关于参数 c 的条件几乎是最优的. 例如, 考虑生成矩阵 $A = (1,1)^T$. 则 $\ker_{\mathbb{Z}}(A) = (1,-1)\cdot\mathbb{Z}$ 且 $\Lambda = 1\cdot\mathbb{Z}$. 由式 (2.13) 知 $c > 2^{\frac{2-1}{2}}\sqrt{2} = 2$. 若取 $c = 2$, Decomp_LLL 算法能够正确地计算出结果, 这也进一步地说明式

(2.13) 中的条件仅仅是充分的. 然而, 算法在当 $c=1$ 时不能准确地计算出结果则说明, 式 (2.13) 中的条件已经没有多大改进的余地了.

LLL-交换次数. 现在开始分析算法 2.4 所需要的 LLL-交换次数与参数 c 之间的关系. 为此, 考查矩阵 $B=c\cdot A_c=(I_n|c\cdot A)$ 的行所生成的格 Λ'_c. 该格本质上与 Λ_c 一致, 因为 $\Lambda'_c=c\cdot\Lambda_c$. 设 $\boldsymbol{b}_i$ 是矩阵 B 的第 i 行, $L=(l_{i,j})$ 为矩阵 B 的 LQ 分解的 L-因子. 经过 t 次 LLL-交换后, 令 $L^{(t)}=(l_{i,j}^{(t)})$ 为 $B^{(t)}$ 的 L-因子. 因为 $B^{(t)}$ 为行满秩的, 所以 $L^{(t)}$ 的所有对角元是正数. 重新排列 $l_{i,i}^{(t)}$ 使得 $l_{i_1,i_1}^{(t)}\le l_{i_2,i_2}^{(t)}\le\cdots\le l_{i_n,i_n}^{(t)}$ 满足 $i_{n-d}<i_{n-d+1}$. 现引进如下两个下标集合, 它们将在下面的分析中起重要作用:

$$S^{(t)}=\{i_j:j=1,\cdots,n-d\},$$
$$G^{(t)}=\{i_j:j=n-d+1,\cdots,n\}.$$

可以验证 $S^{(k)}$ 和 $G^{(k)}$ 的定义良好.

为了统计 LLL-交换的次数, 首先来考查矩阵 B 在算法开始时和终止时的状态, 分别记作 $B^{(0)}$ 和 $B^{(e)}$. 不失一般性, 假设矩阵 A 的前 d 行线性无关. 又知单位矩阵为 $B^{(0)}$ 的子矩阵, 所以 $l_{1,1}^{(0)},\cdots,l_{d,d}^{(0)}=\mathcal{O}(c)$ 且 $1<l_{d+1,d+1}^{(0)},\cdots,l_{n,n}^{(0)}=\mathcal{O}(1)$. 因此, $\max\{l_{d+1,d+1}^{(0)},\cdots,l_{n,n}^{(0)}\}\ll\min\{l_{1,1}^{(0)},\cdots,l_{d,d}^{(0)}\}$, 所以 $S^{(0)}=\{d+1,\cdots,n\}$, $G^{(0)}=\{1,\cdots,d\}$.

当步骤 2 终止时, 矩阵 $B^{(e)}$ 的前 $n-d$ 行具有 $(\boldsymbol{v}_j,0)$ 的形式, 其中 $\boldsymbol{v}_j\in\ker_{\mathbb{Z}}(A)\subseteq\mathbb{Z}^n$. 事实上, 这些向量 $\boldsymbol{v}_j$ 形成了 $\ker_{\mathbb{Z}}(A)$ 的一组基. 同时, $B^{(e)}$ 右下方的 $d\times d$ 子矩阵形成了格 $c\cdot\Lambda$ 的一组基. 因此, 对充分大的参数 c (如满足式 (2.13) 中的条件) $S^{(e)}=\{1,\cdots,n-d\}$ 且 $G^{(e)}=\{n-d+1,\cdots,n\}$.

假设第 t 次 LLL-交换发生在 $\boldsymbol{b}_i^{(t)}$ 和 $\boldsymbol{b}_{i+1}^{(t)}$ 之间. 第 $t+1$ 次 LLL-交换可能有如下三种情形.

情形 1. 当下一次交换发生在 $S^{(t)}$ 内部时, 即 $\{i,i+1\}\subseteq S^{(t)}$, 由引理 2.11 有 $S^{(t+1)}=S^{(t)}$ 且 $G^{(t+1)}=G^{(t)}$.

情形 2. 当下一次交换发生在 $G^{(t)}$ 内部时, 同上.

情形 3. 当下一次交换发生在 $S^{(t)}$ 与 $G^{(t)}$ 之间时, 即 $i\in G^{(t)}$ 且 $i+1\in S^{(t)}$, 要么与前两种情形雷同: $S^{(t+1)}=S^{(t)}$, $G^{(t+1)}=G^{(t)}$, 要么 $S^{(t+1)}=S^{(t)}\cup\{i\}\setminus\{i+1\}$, $G^{(t+1)}=G^{(t)}\cup\{i+1\}\setminus\{i\}$.

由此, 发生在 Decomp_LLL 算法中的 LLL-交换便被划分成了互不相交的三种类型. 因此, 总的 LLL-交换次数等于这三种类型的交换次数的总和. 下面, 对每一种类型分别进行统计.

引理 2.13. *发生在 $S^{(k)}$ 内部的 LLL-交换次数不超过 $\mathcal{O}((n-d)\log(\det(\ker_{\mathbb{Z}}(A))))$.*

证明. 令

$$P_1(t)=\sum_{i_j\in S^{(t)}} g(i_j)\log l^{(t)}_{i_j,i_j}=\sum_{j=1}^{n-d} j\log l^{(t)}_{i_j,i_j},$$

其中 $g:S^{(t)}\to\{1,\cdots,n-d\}$ 为映射 $g(i_j)=j$. 则发生在交换下标为$\{i,i+1\}\in S^{(t)}$ 的 LLL-交换使得 P_1 至少增加 $\log\frac{2}{\sqrt{3}}$. 事实上, $g(i+1)-g(i)=1$ 且

$$\begin{aligned}P_1(t+1)-P_1(t)=&g(i)\log\frac{l^{(t+1)}_{i,i}}{l^{(t)}_{i,i}}+g(i+1)\log\frac{l^{(t+1)}_{i+1,i+1}}{l^{(t)}_{i+1,i+1}}\\=&\log\frac{l^{(t+1)}_{i+1,i+1}}{l^{(t)}_{i+1,i+1}}>\log\frac{2}{\sqrt{3}},\end{aligned}$$

其中应用了 $l^{(t)}_{i,i}\cdot l^{(t)}_{i+1,i+1}/(l^{(t+1)}_{i,i}\cdot l^{(t+1)}_{i+1,i+1})=1$ 和

$$\frac{l^{(t)}_{i,i}}{l^{(t+1)}_{i,i}}>\frac{2}{\sqrt{3}}. \tag{2.14}$$

在开始时,

$$P_1^{(b)}=P_1(0)=\sum_{j=1}^{n-d} j\log l^{(0)}_{i_j,i_j}\geq 0,$$

因为 $l^{(0)}_{i_j,i_j}>1$ 对 $i_j\in S^{(0)}$ 成立. 在终止时, $B^{(e)}$ 的前 $n-d$ 行具有 $(\boldsymbol{v}_j,0)$ 的形式, 其中 $\boldsymbol{v}_j\in\ker_{\mathbb{Z}}(A)\subseteq\mathbb{Z}^n$; 这些向量 $\boldsymbol{v}_j$ 便形成了 $\ker_{\mathbb{Z}}(A)$ 的一组基; $B^{(e)}$ 右下方的 $d\times d$ 矩阵形成了格 $c\cdot\Lambda$ 的一个基矩阵. 于是,

$$P_1^{(e)}=\sum_{j=1}^{n-d} j\log l^{(e)}_{i_j,i_j}\leq(n-d)\log(\det(\ker_{\mathbb{Z}}(A))).$$

因此, 发生在 $S^{(k)}$ 内部的 LLL-交换至多为

$$\begin{aligned}P_1^{(e)}-P_1^{(b)}\leq&(n-d)\log(\det(\ker_{\mathbb{Z}}(A)))/\log(2/\sqrt{3}\,)\\=&\mathcal{O}((n-d)\log(\det(\ker_{\mathbb{Z}}(A)))).\end{aligned}$$

证毕. □

引理 2.14. *发生在 $G^{(t)}$ 内部的交换次数不超过 $\mathcal{O}(d^3 + d^2 \log \frac{X}{\lambda_1(\Lambda)})$.*

证明. 考虑

$$P_2(t) = \sum_{j=1}^{d} (d - j + 1) \log l^{(t)}_{i_{n-d+j}, i_{n-d+j}}.$$

类似地, 能够证明发生在交换下标为 $\{i, j\} \in G^{(t)}$ 的 LLL-交换使得 P_2 至少减小 $\log \frac{2}{\sqrt{3}}$.

在开始时,

$$\begin{aligned} P_2^{(b)} = P_2(0) &= \sum_{j=1}^{d} (d - j + 1) \log l^{(0)}_{i_{n-d+j}, i_{n-d+j}} \\ &\leq \sum_{j=1}^{d} (d - j + 1) \log(cX) = \frac{d^2 + d}{2} \log(cX), \end{aligned}$$

因为 $l^{(0)}_{i_j, i_j} \leq \|\boldsymbol{b}^{(0)}_{i_j}\| \leq cX$ for $i_j \in G^{(0)}$. 在终止时, $B^{(e)}$ 的右下方的 $d \times d$ 子矩阵为格 $c \cdot \Lambda$ 的基矩阵. 因此, $\log l^{(e)}_{i_{n-d+1}, i_{n-d+1}} \geq \log(c \cdot \lambda_1(\Lambda))$. 进一步地, 由于 $B^{(e)}$ 是 LLL-约化的, 所以 $\log l^{(e)}_{i_{n-d+j}, i_{n-d+j}} \geq \log(2^{\frac{1-j}{2}} l^{(e)}_{i_{n-d+1}, i_{n-d+1}}) \geq \frac{1-j}{2} + \log(c \cdot \lambda_1(\Lambda)) \geq \frac{1-d}{2} + \log(c \cdot \lambda_1(\Lambda))$, $j = 1, \cdots, d$. 因此,

$$\begin{aligned} P_2^{(e)} &= \sum_{j=1}^{d} (d - j + 1) \log l^{(e)}_{i_j, i_j} \\ &\geq \sum_{j=1}^{d} (d - j + 1)((1 - d)/2 + \log(c \cdot \lambda_1(\Lambda))) \\ &= \frac{d^2 + d}{2} \log(c \cdot \lambda_1(\Lambda)) - \mathcal{O}(d^3). \end{aligned}$$

所以, 发生在 $G^{(k)}$ 内部的交换次数至多为 $(P_2^{(b)} - P_2^{(e)}) / \log \frac{2}{\sqrt{3}} = \mathcal{O}(d^3 + d^2 \log \frac{X}{\lambda_1(\Lambda)})$. 证毕. □

引理 2.15. *发生在 $S^{(t)}$ 和 $G^{(t)}$ 之间的 LLL-交换次数不超过 $\mathcal{O}(\log(\det(\ker_{\mathbb{Z}}(A))) + (n^2 - d^2))$.*

证明. 考虑

$$P_3(k) = \sum_{i \in S^{(k)}} (\log l^{(t)}_{i,i} + (n - i)).$$

一次发生在 $\boldsymbol{b}_i^{(t)}$ 和 $\boldsymbol{b}_{i+1}^{(t)}$ 之间的 LLL-交换至少使得 P_3 增加 $\log\frac{2}{\sqrt{3}}$, 其中 $i+1\in S^{(t)}$, $i\in G^{(t)}$. 如前所述, 此时有两种情况. 当 $S^{(t)}$ 和 $G^{(t)}$ 保持不变时, 由式 (2.14), 有 P_3 至少增加 $\log\frac{2}{\sqrt{3}}$. 当 $S^{(t+1)}=S^{(t)}\cup\{i\}\setminus\{i+1\}$, $G^{(t+1)}=G^{(t)}\cup\{i+1\}\setminus\{i\}$ 时, 由引理 2.11 知 $|l_{i+1,i+1}^{(t)}|\le|l_{i,i}^{(t+1)}|$, 所以 $P_3(t+1)-P_3(t)\ge 1>\log\frac{2}{\sqrt{3}}$.

在开始时, 有 $P_3^{(b)}=P_3(0)\ge 0$ 成立. 在终止时,

$$\begin{aligned}P_3^{(e)}&=\sum_{i\in S^{(e)}}(\log l_{i,i}^{(t)}+(n-i))\\&\le\log(\det(\ker_{\mathbb{Z}}(A)))+\sum_{i\in S^{(e)}}(n-i)\\&=\log(\det(\ker_{\mathbb{Z}}(A)))+\sum_{i=1}^{n-d}(n-i)\\&=\mathcal{O}(\log(\det(\ker_{\mathbb{Z}}(A)))+(n^2-d^2)).\end{aligned}$$

证毕. □

将引理 2.13, 引理 2.14 和引理 2.15 中的三类 LLL-交换次数进行加总, 得到如下定理:

定理 2.16. *对秩为 r 的矩阵 $A\in\mathbb{Z}^{n\times m}$, Decomp_LLL 算法需要的 LLL-交换次数不超过 $\mathcal{O}(r^3+n^2+r^2\log\frac{X}{\lambda_1(\Lambda)}+(n-r)\log(\det(\ker_{\mathbb{Z}}(A))))$.*

可见, 该交换复杂度与参数 c 的选取无关. 如前所述, 参数 c 对某些实例可以取得很大. 因此, 该结果较之已有文献中的经典的 LLL 算法的分析结果 (如定理 2.12) 有着明显的优势.

2.6　本章小结

给定一组非零实数, HJLS 算法和 PSLQ 算法是计算这组实数非平凡的整数关系的标准算法. 本章给出了理解这两个著名算法的一个新视角: 将其看作是欧几里得格和向量空间的求交问题的特殊情形. 进一步地, 从这两个算法中, 提取出计算欧几里得空间中有限生成加法子群 (包括格到向量空间的投影、格的有限和等) 结构的算法 (Decomp_HJLS). 通过改进 HJLS 和 PSLQ 的分析方法得到了该算法的正确性和收敛性. 另外, 针对 $\mathbb{R}^n$ 中有限生成加法子群的结构, 也考查了将输入嵌入到更高维的向量空间然后调用 LLL 格约化算法的方法. 对该方法整数情形的分析提供了一个有趣的、非经典的结果.

第三章　同步整数关系探测

对一组实数向量 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t \in \mathbb{R}^n$ $(2 \le t < n)$, 若一个非零的整数向量 $\boldsymbol{m}$ 满足 $\boldsymbol{x}_i \boldsymbol{m}^T = 0$ 对所有的 $i = 1, \cdots, t$ 成立, 则称 $\boldsymbol{m}$ 为 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$ 的**同步整数关系** (simultaneous integer relation).

这一章中, 将通过推广 PSLQ 算法中所采用的完全的量度约化策略 (Hermite 约化) 得到 SIRD 算法. 将复向量 $\boldsymbol{x} \in \mathbb{C}^n$ 分为实部 $\mathrm{Re}(\boldsymbol{x})$ 和虚部 $\mathrm{Im}(\boldsymbol{x})$ 两个实向量, 利用本章中的 SIRD 算法可以得到一个非零的整数向量 $\boldsymbol{m} \in \mathbb{Z}^n$ 使得 $\boldsymbol{x}\boldsymbol{m}^T = 0$. 比如, 对曾在第 1.2 节中提及的复数向量 $\boldsymbol{x} = (1+I, 1+2I, 2+I)$, SIRD 算法将输出一个整数向量 $\boldsymbol{m} = (-3, 1, 1)$ 使得 $\boldsymbol{x}\boldsymbol{m}^T = 0$ 成立.

同时, 本章也将讨论 SIRD 算法在计算机代数系统 Maple [108] 中的实现. 本文作者在 Maple 中实现了 HJLS 同步整数关系探测算法 [66, § 5] 和 SIRD 算法, 实例和实验数据说明 SIRD 算法具有更高的效率. 同时, 为了进一步地提高效率, 通过借鉴文 [23] 中的 "multi-level" 技术, 应用 Maple 中的硬件精度 (hfloat) 和软件精度 (sfloat) 两种数据结构实现了 SIRD 算法. 高效的算法实现为进一步的应用打下了坚实的基础.

3.1　同步整数关系探测算法

在本章中, 总是假设 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t \in \mathbb{R}^n$ 线性无关, 其中 $\boldsymbol{x}_i = (x_{i,1}, \cdots, x_{i,n})$, $t < n$. [1] 显然每一个 $\boldsymbol{x}_i$ 都是非零向量. 记矩阵 $(\boldsymbol{x}_1^T, \cdots, \boldsymbol{x}_t^T) \in \mathbb{R}^{n \times t}$ 为 X, 并假设 X 满足

$$\begin{vmatrix} x_{1,n-t+1} & x_{2,n-t+1} & \cdots & x_{t,n-t+1} \\ x_{1,n-t+2} & x_{2,n-t+2} & \cdots & x_{t,n-t+2} \\ \vdots & \vdots & & \vdots \\ x_{1,n} & x_{2,n} & \cdots & x_{t,n} \end{vmatrix} \neq 0. \tag{3.1}$$

若上式不成立, 则总是能够通过交换 X 的行得到 $X' = DX$ 使得 X' 满足式

[1] 假设 $t < n$ 是因为: 任意的一个 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t \in \mathbb{R}^n$ 的同步整数关系 $\boldsymbol{m}$ 都是 $\mathrm{span}(\mathbf{x}_1, \cdots, \mathbf{x}_t)$ 的正交补空间的元素. 由 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$ 线性无关知 $t \le n$. 若 $t = n$, 则 $\dim(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t) = n$, 此时 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$ 便不存在同步整数关系了.

(3.1), 其中 $D \in GL(n,\mathbb{Z})$ 为变换矩阵. 然后计算 X' 的列向量的同步整数关系. 若 $\boldsymbol{m}$ 为 X' 的列向量的同步整数关系, 则 $D^T\boldsymbol{m}$ 为 $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_t$ 的同步整数关系.

定义 3.1. 设 $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_t \in \mathbb{R}^n$ $(2 \le t < n)$ 满足式 (3.1). 将矩阵 $H \in \mathbb{R}^{n\times(n-t)}$ 称作 $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_t$ 的超平面矩阵 (hyperplane matrix), 如果 H 的各列构成向量空间 $X^\perp = \{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{x}_i\boldsymbol{y}^T = 0,\ i = 1,\cdots,t\}$ 的一组基.

设 $\boldsymbol{b}_1,\cdots,\boldsymbol{b}_n$ 是 $\mathbb{R}^n$ 的标准基, 即 $\boldsymbol{b}_i$ 的第 i 个分量为 1, 其他分量为 0. 对 $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_t,\boldsymbol{b}_1,\cdots,\boldsymbol{b}_n$ 实施标准的(单位化的) Gram-Schmidt 正交化过程后得到的新向量记为 $\boldsymbol{x}_1^*,\cdots,\boldsymbol{x}_t^*,\boldsymbol{b}_1^*,\cdots,\boldsymbol{b}_n^*$. 因为 $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_t$ 满足式 (3.1), 所以 $\boldsymbol{b}_{n-t+1}^* = \cdots = \boldsymbol{b}_n^* = \boldsymbol{0}$. 记矩阵 $(\boldsymbol{b}_1^{*T},\cdots,\boldsymbol{b}_n^{*T})$ 为 H_X, H_X 的 Frobenius 范数定义为 $\|H_X\|_F = \sqrt{\mathrm{Tr}(H_x^T H_X)}$, 其中 Tr 表示矩阵的迹.

引理 3.1. 上述的 $H_X \in \mathbb{R}^{n\times(n-t)}$ 具有如下性质:

1. $H_X^T H_X = I_{n-t}$.
2. $\|H_X\|_F = \sqrt{n-t}$.
3. $\left(\boldsymbol{x}_1^{*T},\cdots,\boldsymbol{x}_t^{*T},H_X\right)$ 是一正交阵.
4. $X^T H_X = \boldsymbol{0}$, 即 H_X 为 $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_t$ 的超平面矩阵.
5. H_X 为一个下梯形阵, 且对角元素非零.

证明. 因为 H_X 的列两两正交, 所以第 1 项成立, 从而第 2 项也成立. 记 $X^* = (\boldsymbol{x}_1^{*T},\cdots,\boldsymbol{x}_t^{*T})$. 由标准的 Gram-Schmidt 正交化过程知 $(X^*|H_X)$ 是正交阵, 从而 $X^{*T}H_X = \boldsymbol{0}$. 并且存在可逆矩阵 $A \in \mathbb{R}^{t\times t}$ 使得 $X = X^*A$. 所以 $X^T H_X = A^T X^{*T} H_X = \boldsymbol{0}$, 即第 4 项成立. 为证第 5 项成立, 记 $\boldsymbol{b}_i^*$ 的第 k 个元素为 $b_{i,k}^*$. 则对 $i = 1,\cdots,n-t$, H_X 的对角元为 $b_{i,i}^*$. 在对 $\boldsymbol{b}_i^*$ 进行单位化以前,

$$b_{i,i}^* = 1 - \sum_{k=1}^{t} x_{k,i}^{*2} - \sum_{j=1}^{i-1} b_{j,i}^{*2},$$

同时,

$$0 \ne \|\boldsymbol{b}_i^*\|^2 = \langle \boldsymbol{b}_i^*, \boldsymbol{b}_i^* \rangle = 1 - \sum_{k=1}^{t} x_{k,i}^{*2} - \sum_{j=1}^{i-1} b_{j,i}^{*2}.$$

所以, H_X 的所有对角元非零. 又对任意的 $i > k$, 由标准的 Gram-Schmidt 正交化过程知 $b_{i,k}^* = \langle \boldsymbol{b}_i^*, \boldsymbol{b}_k^* \rangle = 0$. 证毕. □

基于上述分析, 可以将文献 [50, Theorem 1] 推广到同步整数关系探测的情形.

定理 3.2. *若对任意的 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$ 的同步整数关系 $\boldsymbol{m}$ 和任意的矩阵 $A \in \mathrm{GL}_n(\mathbb{Z})$ 存在一个正交矩阵 $Q \in \mathbb{R}^{(n-t)\times(n-t)}$ 使得 $H = (h_{i,j}) = AH_XQ$ 是一个下梯形矩阵且每个对角元 $h_{j,j} \neq 0$, 则*

$$\frac{1}{\max_{1\leq j\leq n-t}|h_{j,j}|} = \min_{1\leq j\leq n-t}\frac{1}{|h_{j,j}|} \leq \|\boldsymbol{m}\|. \tag{3.2}$$

将 [50, Theorem 1] 的证明作相应调整便能得到定理 3.2 的证明, 在此不再赘述. 定理 3.2 的意义在于给出了一个探测同步整数关系的基本思想: 若能够以左乘幺模矩阵, 右乘正交矩阵的形式对超平面矩阵对角元的模进行约化, 则不等式 (3.2) 给出了 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$ 的任意一个同步整数关系 2-范数的不断增加的下界. 另一方面, 如果存在同步整数关系, 那么这个下界便不会无限制地增加. 无论是 PSLQ 算法还是 HJLS 算法都依赖于这一来自于文献 [51] 中的基本思想. 同时, 定理 3.2 对任意的 $A \in \mathrm{GL}_n(\mathbb{Z})$ 都成立, 保证了约化方法的多样性. PSLQ 算法采用的是如下的 Hermite 约化, 其主要目的是让矩阵 H 是量度约化的 (定义 2.1).

定义 3.2. 设 $H = (h_{i,j}) \in \mathbb{R}^{n\times(n-1)}$ 为对角元非零的下梯形矩阵. 令 $D := I_n$. 对 i 从 2 到 n, j 从 $i-1$ 到 1 (步长为 -1), 令 $q := \lfloor h_{i,j}/h_{j,j} \rceil$; 对 k 从 1 到 n, 令 $d_{i,k} := d_{i,k} - qd_{j,k}$. 更新 $H = D \cdot H$. 称 H 是原矩阵 H 的 *Hermite 约化* (Hermite reduction), 称 D 为 *Hermite 约化矩阵* (Hermite reducing matrix).

若 $a \in \mathbb{R}$, 则上述定义中的记号 $\lfloor a \rceil$ 表示离 a 最近的整数. 若 $a \in \mathbb{C}$, 则 $\lfloor a \rceil$ 表示离 a 最近的 Gauss 整数.

易知, 若一下梯形阵是 Hermite 约化的, 则其也是量度约化的. 若矩阵 H 的元素是复数, 则 $q := \lfloor h_{i,j}/h_{j,j} \rceil$ 便可能不再是整数, 而是 Gauss 整数. 由此将 Gauss 整数引入了约化矩阵 D 中. 因此, 对于复数向量, PSLQ 算法不再适合探测整数关系, 而只能得到一组 Gauss 整数关系. 为了顺利地将 PSLQ 中算法的优点保留下来以探测同步整数关系, 可将 Hermite 约化进行如下推广.

定义 3.3. 设 $H = (h_{i,j}) \in \mathbb{R}^{n\times(n-t)}$ 满足 $h_{j,j} \neq 0$, 且对 $j > i$ 有 $h_{i,j} = 0$. 初始化 $D = (d_{i,j})$ 为 n 阶单位阵 I_n. 对 i 从 2 到 n, j 从 $\min\{i-1, n-t\}$ 到 1 (步长为

$-1)$, 令 $q := \lfloor h_{i,j}/h_{j,j} \rceil$; 对 k 从 1 到 n, 令 $d_{i,k} := d_{i,k} - qd_{j,k}$. 更新 $H = D \cdot H$. 若存在两个整数 $s_1, s_2 \in \{n-t+1, \cdots, n\}$ 满足 $s_1 < s_2$, $h_{s_1,n-t} = 0$ 且 $h_{s_1,n-t} \neq 0$, 则交换 D 和 H 的第 s_1 行和第 s_2 行. 称 H 是原矩阵 H 的广义 *Hermite* 约化 (generalized Hermite reduction), 称 D 为广义 *Hermite* 约化矩阵 (generalized Hermite reducing matrix).

广义的 Hermite 约化保留了 Hermite 约化的两个性质: 约化矩阵 $D \in \mathrm{GL}_n(\mathbb{Z})$ 为幺模矩阵; 约化后得到的矩阵均为量度约化的. 它们的主要区别在于: 广义 Hermite 约化能够约化 $H \in \mathbb{R}^{n\times(n-t)}$ 的最后 $t-1$ 行, 而 Hermite 约化未对此进行处理. 同时, "若存在 $s_1, s_2 \in \{n-t+1, \cdots, n\}$ 满足 $s_1 < s_2$, $h_{s_1,n-t} = 0$ 且 $h_{s_1,n-t} \neq 0$, 则交换 D 和 H 的第 s_1 行和第 s_2 行" 也是原来的 Hermite 约化中没有的. 这意味着下面的引理成立.

引理 3.3. *在经过广义的 Hermite 约化后, 若 $h_{n-t+1,n-t} = 0$, 则 $h_{n-t+2,n-t} = \cdots = h_{n,n-t} = 0$.*

至此, 基于定理 3.2 给出的基本思想和 HJLS-PSLQ 算法, 利用广义的 Hermite 约化便可以得到一个同步整数关系探测算法 SIRD.

算法 3.1 (SIRD)**.** 输入: $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t \in \mathbb{R}^n$ 满足式 (3.1); $M > 0$; $\gamma > 2/\sqrt{3}$.
输出: 要么输出 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$ 的一个整数关系, 要么断言 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$ 不存在 2-范数小于 M 的同步整数关系.

1. 计算超平面矩阵 H_X. 初始化 $H := H_X$, $A := I_n$, $B := I_n$.
2. 计算 H 广义的 *Hermite* 约化矩阵 D. 令 $X^T := X^T D^{-1}$, $H := DH$, $A := DA$, $B := BD^{-1}$.
3. 循环执行以下步骤:
   (a) 选择 r 使得 $\gamma^r |h_{r,r}| \geq \gamma^i |h_{i,i}|$ $(1 \leq i \leq n-t)$. 交换 H 的第 r 行和第 $r+1$ 行, 交换 X^T 和 B 的第 r 列和第 $r+1$ 列.
   (b) 若上一步选择的 $r < n-t$, 则 H 便不再是下梯形的, 此时对 H 做 LQ 分解, 更新 H 为 L-因子.
   (c) 计算 H 广义的 *Hermite* 约化矩阵 D. 令 $X^T := X^T D^{-1}$, $H := DH$, $A := DA$, $B := BD^{-1}$.
   (d) 计算 $G := 1/\max_{1\leq j\leq n-t} |h_{j,j}|$. 若 $G > M$, 则断言 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$ 不存在 2-范数小于 M 的同步整数关系, 算法终止.

(e) 若 X^T 的第 j 列全为 0, 则输出 B 的第 j 列; 若 $h_{n-t,n-t}=0$, 则输出 B 的第 $n-t$ 列.

定理 3.4. 若 $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_t\in\mathbb{R}^n$ 存在同步整数关系, 记所有的同步整数关系形成的格为 Λ_X. 则 *SIRD* 算法一定能够找到 $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_t\in\mathbb{R}^n$ 的一个同步整数关系 $\boldsymbol{m}$, 且

$$\|\boldsymbol{m}\|\leq\gamma^{n-t}\lambda_1(\Lambda_X). \tag{3.3}$$

证明. 如前所述, SIRD 算法是通过改进 HJLS-PSLQ 算法中的交换步骤、推广 Hermite 约化而得到的, 其正确性 (定理 3.4) 的证明可以通过改进 [50, Theorem 2 & 3] 的证明而得到. 不同的地方在于广义 Hermite 约化的性质 (引理 3.3) 在证明过程中起了至关重要的作用. 现简要描述如下.

设 $H(k)$ 为 SIRD 算法中经过 k 次迭代之后的矩阵 H. 如果存在 $1\leq j\leq n-t$ 使得 $h_{j,j}(k)=0$ 且 k 是满足上述条件的最小正整数, 那么 $j=n-t$. 这是因为广义的 Hermite 约化不会将对角元化为 0. 此时矩阵 B 的第 $n-t$ 列必为 $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_t$ 的同步整数关系. 因为在 SIRD 算法中, 有

$$\mathbf{0}=X^TH_X=X^TBAH_X=X^TBAH_XQ=X^TBH(k-1),$$

其中 Q 为某一适当的 $n-t$ 阶正交矩阵. 记 $X^TB=(\boldsymbol{z}_1^T,\cdots,\boldsymbol{z}_t^T)$, 其中 $\boldsymbol{z}_i=(z_{i,1},\cdots,z_{i,n})$. 则

$$\begin{aligned}
&\begin{pmatrix}0&\cdots&0\\ \vdots&\ddots&\vdots\\ 0&\cdots&0\end{pmatrix}_{t\times(n-t)}=X^TBH(k-1)=\begin{pmatrix}\boldsymbol{z}_1\\ \vdots\\ \boldsymbol{z}_t\end{pmatrix}H(k-1)\\
=&\begin{pmatrix}\cdots,&\sum_{k=n-t}^{n}z_{1,k}h_{k,n-t}(k-1)\\ \cdots,&\cdots\\ \cdots,&\sum_{k=n-t}^{n}z_{t,k}h_{k,n-t}(k-1)\end{pmatrix}\\
=&\begin{pmatrix}\cdots,&z_{1,n-t}h_{n-t,n-t}(k-1)\\ \cdots,&\cdots\\ \cdots,&z_{t,n-t}h_{n-t,n-t}(k-1)\end{pmatrix}.
\end{aligned}$$

由 $h_{n-t,n-t}(k)=0$ 知 $h_{n-t+1,n-t}(k-1)=0$ 且 $h_{n-t,n-t}(k-1)\neq0$. 所以由引理 3.3 知上式中的最后一个等号成立, 从而 $z_{1,n-t}=\cdots=z_{t,n-t}=0$, 即矩阵 B 的第 $n-t$ 列是 $\boldsymbol{x}_1,\cdots,\boldsymbol{x}_t$ 的同步整数关系.

下证式 (3.3) 成立. 设 $\boldsymbol{m}$ 是经过 k 次迭代输出的同步整数关系. 可以证明 $\|\boldsymbol{m}\| = 1/h_{n-t,n-t}(k-1)$; 按照 r 的选取又知: 当 $r = n-t$ 时, $\max|h_{j,j}(k-1)| \leq \gamma^{n-t}|h_{n-t,n-t}(k-1)|$. 从而由定理 3.2 得 $\lambda_1(\Lambda_X) \geq 1/\max|h_{j,j}(k)| \geq \gamma^{t-n}/|h_{n-t,n-t}(k-1)| = \gamma^{t-n}\|\boldsymbol{m}\|$. 证毕. □

采用与 HJLS-PSLQ 类似的分析方法 (类似于本文第 2.4.3 节中的分析), 可以证明如下定理.

定理 3.5. *设 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t \in \mathbb{R}^n$ 满足式 (3.1), 且这组向量具有同步整数关系. 则 SIRD 算法将在不超过*

$$\left(\begin{pmatrix} n \\ 2 \end{pmatrix} - \begin{pmatrix} t \\ 2 \end{pmatrix}\right) \frac{\log(\gamma^{n-t}\lambda_1(\Lambda_X))}{\frac{1}{2}\log\left(\frac{4\gamma^2}{\gamma^2+4}\right)}$$

次迭代步骤内输出 $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$ 的一个同步整数关系.

至此, SIRD 算法对给定的一组实数向量, 要么找到这组向量的同步整数关系, 要么证明 2-范数小于 $1/\max_{1\leq j\leq n-t}|h_{j,j}|$ 的同步整数关系不存在.

另外, 本章上面的结论也可以直接对复数向量应用. 此时只需参数 $\gamma > \sqrt{2}$ 即可. 但这种直接的调用方式的输出也将是 Gauss 同步整数关系.

3.2 算法的实现

本文给出的同步整数关系探测算法 SIRD 并不是第一个可用于探测一组向量同步整数关系的算法. 在 [66, § 5] 中就已经给出了一个同步整数关系探测算法. 其基本思想与 SIRD 是一致的, 都是基于 [51] 中的多维 Euclid 算法. 该算法也可以分为求超平面矩阵、初始约化超平面矩阵、迭代这三个步骤. 该算法与 SIRD 的区别在于所采用的量度约化方式不同: HJLS 同步整数关系探测算法采用了如 LLL-算法中所采用的部分量度约化方式, 而 SIRD 采用了广义的 Hermite 约化. 正如第 2.1 节末尾所描述的, 在实数计算模型下, 这并不会影响算法的迭代次数. 然而在实际的计算中, 差别明显. 比如, 对 $\boldsymbol{x}_1 = (11, 27, 31)$ 和 $\boldsymbol{x}_2 = (1, 2, 3)$, HJSL 同步整数关系探测算法将在执行 5 次迭代之后输出一个 $\boldsymbol{x}_1$ 和 $\boldsymbol{x}_2$ 的同步整数关系 $\boldsymbol{m} = (19, -2, -5)$. 而 SIRD 算法仅需要 3 次迭代便会输出相同的同步整数关系 $\boldsymbol{m}$.

由于这两个算法在计算机实现时都需要调用高精度的软件包 ARPREC [20], GNU 高精度算法库 [61] 等. 通用的商用计算机代数系统 Maple [108] 和 Mathematica [110] 都支持高精度的运算. 本文作者在 Maple 平台上实现了 HJLS 同步整数关系探测算法和 SIRD 算法. 为了对两个算法进行进一步地比较, 在实现时, 所有的数据结构均进行相同设置, 并且对相同功能的模块均采用相同的技术, 比如对算法中涉及的 Gram-Schmidt 正交化过程均采用数值稳定的方法 (如 Householder 变换 [71]). 下表的所有的实验数据都是在 AMD Athlon 7750 处理器 (2.70 GHz) 2GB 内存的个人计算机中获得的.

表 3.1: HJLS 同步整数关系探测算法与 SIRD 算法的比较

No.	n	itr_{HJLS}	itr_{SIRD}	t_{HJLS}	t_{SIRD}
1	4	15	8	0.063	0.
2	4	13	6	0.062	0.
3	4	21	11	0.094	0.015
4	5	25	12	0.109	0.016
5	5	27	7	0.141	0.
6	5	21	10	0.094	0.
7	30	51	7	0.922	0.125
8	54	34	9	2.203	0.453
9	79	34	5	4.860	0.625
10	97	37	5	7.438	1.047
11	128	45	5	13.765	1.687
12	149	29	2	19.016	1.610
13	173	26	3	26.812	2.421
14	192	29	5	34.218	3.563
15	278	28	5	85.797	8.860
16	290	35	4	95.656	8.328
17	293	23	4	98.062	8.750
18	305	22	3	109.187	8.063
19	316	19	3	120.187	8.766
20	325	18	2	129.031	6.953

表 3.1 给出了 HJLS 同步整数关系探测算法和 SIRD 算法对 Maple 中的 `rand` 命令产生的两个长度为 n 的有理向量探测其同步整数关系的实验数据, 其中 itr_{HJLS} 和 itr_{SIRD} 分别表示而这实际的迭代次数, t_{HJLS} 和 t_{SIRD} 表示二者的运行时间 (单位: 秒). 前 6 个例子是维数较小时的比较, 后面的例子维数逐渐升高. 从表中可以发现: 随着维数的不断增加, HJLS 同步整数关系探测算法和 SIRD 的性能差别也越来越显著. 这不仅体现在迭代次数 HJLS 比 SIRD 多, 更直接的体现在运算时间上. 以表 3.1 中的数据来看, SIRD 算法探测两个实数向量的同步整数关系所需要的时间平均约为 HJLS 同步整数关系探测算法所需时间的 1/10.

另外, SIRD 算法中的参数 γ 也会对算法的效率产生影响. 比如对 $\boldsymbol{x}_1 = (86, 6, 8, 673)$ 和 $\boldsymbol{x}_2 = (83, 5, 87, 91)$, 如果在运行时令 $\gamma = 2$, 则 SIRD 算法在完成 10 次迭代后输出整数向量 $(-32, -747, 63, 10)$, 但是如果增大参数 γ 的值, 令 $\gamma = 93$, 则 SIRD 算法将在 7 次迭代后输出 $(-35, -2624, 157, 26)$, 可以验证这两个输出的整数向量都是 $\boldsymbol{x}_1$ 和 $\boldsymbol{x}_2$ 的同步整数关系. 如果继续增加参数 γ 的值, 会发现迭代次数总是 7 次, 不再减少. 对更多的实验数据观察, 可以发现参数 γ 的值越大, 得到正确整数关系所需要的精度就越高, 这会影响计算的效率, 但同时, 对更大的 γ 算法所需的迭代次数更少. 基于这样的观察, 本文所呈现的实验数据都是在 $\gamma = 1000$ 时得到的. 因此, 如何选取参数 γ 的值使算法所需精度和迭代次数之间达到平衡, 仍然需要进一步的研究.

3.2.1 SIRD 算法的双层实现

尽管 SIRD 算法比已有的 HJLS 同步整数关系探测算法的效率更高, 但是在现实应用中, 针对规模稍大的问题, 仍然不能够快速的得到结果. 尤其是在高精度的数据结构下, 每一次算术操作都很耗时, 导致整体运行时间的增加. 在实现 PSLQ 算法时, 可以采用“two-level” 技术 [23, §5] 以提高计算效率. 该技术对 SIRD 算法也适用. 本节将描述如何在 Maple 中实现 SIRD 算法的 “双层” 技术. 其基本原理是将能够在硬件精度下实施的计算就在硬件精度下实现, 避免不了的高精度运算步骤才采用高精度的数据结构.

Maple 中的 “hfloat” 数据类型. 计算机代数系统 Maple 中提供了适用于硬件浮点算术的数据结构 “hfloat”. 通常, “hfloat” 数据结构下的代码执行要比采用软件浮点数据结构编写的代码快得多. 但缺点在于, 不同的计算机上采用的硬

件算术可能不一样.

Maple 中的 “sfloat” 数据类型. Maple 能够表示精确的数量, 如 π, e, $3/4$ 等, 也能够表示这些精确数量的近似值. 其中 “sfloat” 数据结构就是软件精度的数据结构, 支持任意高精度的运算, 并且可以通过 `Digits` (十进制位数) 来设定计算的精度. 以此为基础的软件浮点数系是工业界的浮点运算标准 (如 IEEE 754-2008 [118]) 的自然扩展. 欲了解 Maple 中浮点运算的更多细节, 可以参考 [116, Chap. 4].

SIRD 的 “双层” 实现. 首先, 采用 “sfloat” 数据结构执行 SIRD 算法中的步骤 1 和步骤 2. 然后采用 “hfloat” 数据结构重新初始化这些结果: 令 $\bar{A} := I_n$, $\bar{B} := I_n$, 令 $\bar{X}$ 和 $\bar{H}$ 分别为 X 的硬件精度近似. 对一些极大的实例, 可能需要采用单位化处理以避免数据的上溢出或下溢出. 然后对 $\bar{H}$ 进行 LQ 分解, 更新 $\bar{H}$ 为 L-因子.

接下来, 运用 “hfloat” 数据结构实施 SIRD 算法的迭代步骤. 尽管 $\bar{A}$ 和 $\bar{B}$ 是浮点类型的数据结构, 但是其中的元素总是整数. 由于 $\bar{A}$ 和 $\bar{B}$ 的元素会随着迭代次数的增加而不断增加, 有可能超过 “hfloat” 所能表示的最大数, 所以需要自行设定一个上界 (PSLQ 算法中设为 10^{13}); 同时 $\bar{X}$ 中的元素也可能变得非常接近于 0, 所以也可以设定一个下界 (比如 10^{-14}). 如果在 “hfloat” 数据结构下的运算超出了上述的两个界中的任意一个, 则需要将 “sfloat” 中的数据做如下更新:

$$\begin{aligned} X^T &:= X^T \cdot \bar{B}, \\ B &:= B \cdot \bar{B}, \\ A &:= \bar{A} \cdot A, \\ H &:= \bar{A} \cdot H. \end{aligned}$$

在执行更新之后, SIRD 算法中的步骤 3d 和 3e 被执行, 以检查是否找到一个同步整数关系. 否则, 便找到一个同步整数关系的 2-范数的下界, 并且再次重新初始化, 重复第执行以上步骤.

这种 “双层”的实施只是将那些在 “hfloat” 数据结构下处理不了的运算采用 “sfloat” 数据结构进行运算. 实验显示, 这种技术大幅度地提高了算法的效率. 直接实施的 SIRD 算法和 “双层” 实施的 SIRD 算法的实验比较将结合 SIRD 算法在近似重构代数数极小多项式的应用, 在下一章中给出 (表 4.2).

3.3 本章小结

本章通过给出广义 Hermite 约化用来约化超平面矩阵, 基于 HJLS-PSLQ 算法的基本框架得到了一个可以探测 t 个实数向量的同步整数关系的算法 SIRD, 克服了 PSLQ 算法只能对复数向量输出 Gauss 整数关系的不足. 并分别采用高精度的浮点运算和部分高精度浮点运算这两种策略给出了该算法在计算机代数系统 Maple 中的两个实现, 与已有的 HJLS 同步整数关系探测算法相比, 本章的算法在运行效率上更具优势.

第四章　代数数极小多项式的近似重构

求解单变元多项式方程是数学领域中一个古老, 基本而内容丰富的问题. 著名的代数学基本定理告诉我们任意一个 n 次复系数单变元多项式有一个复根, 而 Galois 理论则指出次数大于等于 5 的代数方程没有根式解. 因此在实际应用中, 求解单变元多项式方程主要利用数值方法 (如 Newton 迭代法) 求得近似解. 本章主要研究这一问题的逆问题, 即给定一个近似值, 找出一个准确的一元多项式, 使得给定的近似值恰好对应到该多项式的某个准确根.

称 α 为一个*代数数* (algebraic number), 如果存在 $P(x) \in \mathbb{Z}[x]$ 使得 $P(\alpha) = 0$. 称 $P(x)$ 为代数数 α 的*极小多项式* (minimal polynomial), 如果 $P(x)$ 是 $\mathbb{Z}[x]$ 中满足 $P(\alpha) = 0$ 的次数最低的本原多项式. 代数数 α 的*次数* (degree) 定义作其极小多项式的次数. 代数数 α 的*高* (height), 记作 height(α), 定义为其极小多项式 $P(x)$ 的*高度* height(P), 即 $P(x)$ 系数绝对值的最大值. 现将本文所讨论的问题作如下精确描述:

问题 4.1. 设一未知代数数 α 的次数不超过 n, 高度不超过 H. 能否从某一精度的近似值 $\bar{\alpha}$ 推断出 α 准确的极小多项式?

该问题最早是由著名的理论计算机科学家, 1995 年 Turing 奖获得者 Manuel Blum 于上世纪 80 年代在研究伪随机序列时提出的 (见 [95]).

当 $n = 1$ 时, 即有理数的近似重构, 可以通过 Euclid 算法 [147] 或者连分数算法 [150] 予以解决.

当 $n > 1$ 时, Kannan, Lenstra 和 Lovász 于 1984 年利用著名的 LLL 算法 给出了第一个肯定回答 [95]. 随着 LLL 算法在计算机科学中的广泛应用, 其改进和推广也不断涌现, 如 [80, 117, 121, 141] 等. 相应的, 基于 LLL 的极小多项式重构算法也得到改进, 较新的进展可以参考文献 [69].

事实上, 对于实代数数, 通过整数关系探测也可以解决该问题 [127]. 为了得到 n 次代数数 α 的极小多项式, [127] 中通过整数关系探测算法 PSLQ 寻找 $(1, \alpha, \cdots, \alpha^n)$ 的一个整数关系. 然而如第 1.2 节所述, 对于复代数数的情形, PSLQ 算法便无能为力了. 因为对一个复数向量, PSLQ 仅仅能找到一个 Gauss 整数关系. 上一章所给出的同步整数关系探测算法 SIRD 能有效地弥补

这一缺陷, 找到一个非零整数向量使其同时成为 t $(1 \le t \le n-1)$ 个已知实数向量的整数关系. 对一个 n 次复代数数 α, 记 $\boldsymbol{x}_1 = (1, \mathrm{Re}(\alpha), \cdots, \mathrm{Re}(\alpha^n))^T$ 和 $\boldsymbol{x}_2 = (0, \mathrm{Im}(\alpha), \cdots, \mathrm{Im}(\alpha^n))^T$. 对 $\boldsymbol{x}_1$ 和 $\boldsymbol{x}_2$ 应用同步整数关系探测算法 SIRD 可以得到一个整数向量 $\boldsymbol{p} \in \mathbb{Z}^{n+1}$ 使得 $\langle \boldsymbol{p}, \boldsymbol{x}_1 \rangle = \langle \boldsymbol{p}, \boldsymbol{x}_2 \rangle = 0$, 从而得到 α 的极小多项式.

本章将采用上述策略, 得到一个解决问题 4.1 的完备方法. 此方法不同于已有基于 LLL 格约化的算法 [69, 95]. 然而, 要从 α 的近似值推断出其准确的极小多项式, 必须对该近似值的误差进行控制, 才能使得到的结果准确可信.

设 α 为一个高度不超过 H 的 n 次复代数数, 其近似值 $\bar{\alpha}$ 满足

$$\max_{1 \le i \le n} |\alpha^i - \bar{\alpha}^i| \le \epsilon. \tag{4.1}$$

为了从近似值 $\bar{\alpha}$ 推断出 α 的精确的极小多项式, 可以利用 SIRD 算法寻找 $\boldsymbol{x}_1 = (1, \mathrm{Re}(\bar{\alpha}), \cdots, \mathrm{Re}(\bar{\alpha}^n))$ 和 $\boldsymbol{x}_2 = (0, \mathrm{Im}(\bar{\alpha}), \cdots, \mathrm{Im}(\bar{\alpha}^n))$ 的一个同步整数关系 $\boldsymbol{p} = (p_0, p_1, \cdots, p_n) \in \mathbb{Z}^{n+1}$ 使得 $\langle \boldsymbol{p}, \boldsymbol{x}_1 \rangle = \langle \boldsymbol{p}, \boldsymbol{x}_2 \rangle = 0$. 令 $p(x) = \sum_{i=0}^{n} p_i x^i$. 则 $p(\bar{\alpha}) = 0$. 尽管由于计算机不能精确地实现实数操作, 得到的整数关系不一定满足 $p(\bar{\alpha}) = 0$, 但总可以保证 $|p(\bar{\alpha})|$ 很小. 本章中, 将通过误差分析证明如下定理, 从而得到上述近似值的误差控制条件.

定理 4.1. *记号同上. 设多项式 $p(x) = \sum_{i=0}^{n} p_i x^i \in \mathbb{Z}[x]$ 的高度 $height(p) \le H$. 若*

$$\epsilon < \frac{1}{2}(n+1)^{-\frac{3}{2}n} H^{-2n}, \tag{4.2}$$

则

$$p(\alpha) = 0 \;\Leftrightarrow\; |p(\bar{\alpha})| < \frac{1}{2}(n+1)^{-\frac{3}{2}n+1} H^{-2n+1}. \tag{4.3}$$

定理 4.1 指出在满足一定的误差控制条件下(见式 (4.5)), 可以通过近似值获得代数数准确的极小多项式. 这在某种意义上也意味着在此误差控制下, 从代数数的近似值可以得出其准确值, 从而将张景中和冯勇在 [150] 中提出的"采用近似计算获得准确值"这一思想的适用范围从有理数扩展到所有的代数数. 而且本文的误差控制向下兼容, 即该误差控制不仅适合于虚代数数, 也适用于实代数数. 特别地, 当 α 为有理数时, $n = 1$, 相应的误差控制为 $|\alpha - \bar{\alpha}| < 1/(\sqrt{2}H^2)$, 优于 [150] 中的误差控制 $|\alpha - \bar{\alpha}| < 1/(2H^2)$.

根据式 (4.5) 的误差控制, 基于同步整数关系探测算法 (SIRD), 本章将给出一个代数数极小多项式近似重构的新算法, 该算法是完备的, 确定的; 并通过调

用上一章中所描述的 SIRD 算法的两种在计算机代数系统 Maple 中的实现来实现了该算法; 通过理论分析、实例研究和与已有算法的比较, 可以发现本章的结果具有以下几个方面的特点:

1. 相对于 [127] 中方法而言, 本文的算法是完备的, 完整地解决了问题 4.1.
2. 误差控制条件优于已有相应的误差控制.
3. 在满足该误差控制的条件下, 本文的算法能确保输出是该代数数准确值的极小多项式.
4. 针对较大规模的问题, 使用本文的算法有较好的效果.

4.1　基于 LLL 的重构算法

本节简要介绍基于 LLL 算法的代数数极小多项式的近似重构算法, 详见 [95].

假设 $\alpha \in \mathbb{C}$ 为一个次数不超过 n_0, 高度不超过 H 的代数数, 且 $|\alpha| \leq 1$. 令 $\bar{\alpha}_i \in 2^{-s} \cdot \mathbb{Z}[I]$ 是 α^i 的近似值, 且满足 $|\alpha^i - \bar{\alpha}_i| \leq 2^{-s}$, 其中 s 满足

$$2^s \geq 2^{n_0^2/2} \cdot (n_0+1)^{(3n_0+4)/2} \cdot H^{2n_0}.$$

对正整数 n $(1 \leq n \leq n_0)$, 考虑由矩阵

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & 2^s \cdot \mathrm{Re}(\bar{\alpha}_0) & 2^s \cdot \mathrm{Im}(\bar{\alpha}_0) \\ 0 & 1 & \cdots & 0 & 2^s \cdot \mathrm{Re}(\bar{\alpha}_1) & 2^s \cdot \mathrm{Im}(\bar{\alpha}_1) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 2^s \cdot \mathrm{Re}(\bar{\alpha}_n) & 2^s \cdot \mathrm{Im}(\bar{\alpha}_n) \end{pmatrix}$$

生成的格 Λ_α. 记上述矩阵的第 i 行为 $\boldsymbol{b}_i$. 对上述矩阵的行调用 LLL-算法, 返回的第一个向量记为 $\bar{\boldsymbol{v}} = \sum_{i=0}^{n} v_i \boldsymbol{b}_i$. 则下面两条叙述等价 [95, Theorem 1.15]:

1. α 满足 $v(x) = \sum_{i=0}^{n} v_i x^i$, 即 $v(\alpha) = 0$.
2. α 的次数不超过 n.

并且, 若 α 的次数为 n, 则 α 的极小多项式为 $p_\alpha(x) = \pm v(x)$. 同时, 若调用经典的 LLL 算法, 则其复杂度为对长度为 $\mathcal{O}(n^2(n + \log H))$ 的整数进行 $\mathcal{O}(n^5(n + \log H))$ 算术操作.

4.2 基于同步整数关系探测的重构算法

SIRD 算法构造的超平面矩阵为一个实矩阵, 算法中的矩阵 B 和 D 的元素均不会有虚数出现, 从而克服了 PSLQ 对复数向量仅能输出 Gauss 整数关系的不足. 至此, 复数向量 $\boldsymbol{v} = (v_1, \cdots, v_n)$ 的整数关系便可以由 SIRD 算法探测 $\boldsymbol{v}$ 的实部向量 $(\mathrm{Re}(v_1), \cdots, \mathrm{Re}(v_n))$ 和虚部向量 $(\mathrm{Im}(v_1), \cdots, \mathrm{Im}(v_n))$ 的同步整数关系得到.

例 4.1. 设 $\alpha = 2 + \sqrt{3}\,I$. 显然 α 在 $\mathbb{Z}[x]$ 中的极小多项式是 $x^2 - 4x + 7$. 取 α 的四位有效数字的近似值 $\bar{\alpha} = 2.000 + 1.732\,I$. 于是得到 $\boldsymbol{v}_1 = (1., 2., 1.)$, $\mathbf{v}_2 = (0., 1.732, 6.928)$. 对 $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ 运行 SIRD 算法, 仅需要两次迭代就可以得到一组 $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ 的同步整数关系. 迭代过程中的矩阵 B 如下

$$\begin{pmatrix} 2 & 1 & 0 \\ -1 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 7 & 0 & 2 \\ -4 & 0 & -1 \\ 1 & 1 & 0 \end{pmatrix},$$

显然后一个矩阵的第一列即为所求. 若只取三位有效数字, 则 SIRD 算法通过 3 次迭代后输出 $(1213, -693, 173)$, 这组输出尽管是 $(1., 2., 1.)$ 和 $(0., 1.73, 6.93)$ 的同步整数关系, 但不再是 $(1, \alpha, \alpha^2)$ 的整数关系. 因此为了近似重构代数数的极小多项式, 必须对误差进行控制.

4.2.1 误差控制

本文近似重构 n 次代数数极小多项式的基本思想是: 探测 $(1, \bar{\alpha}, \cdots, \bar{\alpha}^n)$ 的一组整数关系 $(p_0, \cdots, p_n) \in \mathbb{Z}^{n+1}$ (当 α 为实代数数时用 PSLQ 算法 [127]; 当 α 为虚代数数时用上一章介绍的 SIRD 算法), 使得 $p(x) = \sum_{i=0}^{n} p_i x^i$ 满足 $|p(\bar{\alpha})| = 0$. 利用定理 4.1 便可以保证该多项式就是 α 的极小多项式. 为证定理 4.1, 需做如下准备.

引理 4.2. *设 f 是一个 n 次单变元多项式. 若 $\max_{1\le i\le n} |\alpha^i - \bar{\alpha}^i| \le \epsilon$, 则*

$$|f(\alpha) - f(\bar{\alpha})| \le \epsilon \cdot n \cdot \mathit{height}(f).$$

证明. 设 $f = \sum_{i=0}^{n} f_i x^i$. 则 $|f(\alpha) - f(\bar{\alpha})| = |\sum_{i=1}^{n} f_i(\alpha^i - \bar{\alpha}^i)| \le \epsilon \cdot n \cdot \mathrm{height}(f)$. 证毕. □

定义 4.1. 设 m 次多项式 $g=\sum_{i=0}^{m} g_i x^i \in \mathbb{Z}[x]$ 的所有复根为 $z_1, z_2, \ldots, z_m$. 定义 g 的 *Mahler* 度量 (Mahler measure) 为

$$M(g)=|g_m|\prod_{j=1}^{m}\max\{1,|z_j|\}.$$

代数数 α 的 *Mahler* 度量定义为其极小多项式的 Mahler 度量, 记为 $M(\alpha)$.

引理 4.3 ([114], Lemma 3)**.** *设 $\alpha_1,\cdots,\alpha_q$ 为次数分别为 $d_1,\cdots,d_q$ 的代数数. 令 $D=[\mathbb{Q}(\alpha_1,\cdots,\alpha_q):\mathbb{Q}]$. 设 $P\in\mathbb{Z}[x_1,\cdots,x_q]$ 关于 x_h 的次数不超过 N_h $(1\le h\le q)$. 若 $P(\alpha_1,\cdots,\alpha_q)\neq 0$, 则*

$$|P(\alpha_1,\ldots,\alpha_q)|\ge \|P\|_1^{1-D}\prod_{h=1}^{q}M(\alpha_h)^{-DN_h/d_h},$$

其中 $\|P\|_1$ 表示 P 的 1-范数, 即所有系数绝对值之和.

证明. 见 [47, Lemma 2] 的证明. □

对任意的多元多项式 $P\in\mathbb{Z}[x_1,\cdots,x_q]$, 若 $P(\alpha_1,\cdots,\alpha_q)\neq 0$, 则上述引理给出了一个关于 $|P(\alpha_1,\ldots,\alpha_q)|$ 的下界. 若将其应用到 $\mathbb{Z}[x]$ 中的多项式, 则得到

推论 4.4. *设 α 为一个 n_0 次的代数数, $g(x)=\sum_{i=0}^{m} g_i x^i\in\mathbb{Z}[x]$, 且 α 和 $g(x)$ 的高度均不超过 H. 若 $g(\alpha)\neq 0$, 则 $|g(\alpha)|\ge (m+1)^{-(n_0-1)}\cdot(n_0+1)^{-\frac{m}{2}}\cdot H^{-(m+n_0-1)}$.*

证明. 由引理 4.3 得

$$\begin{aligned}|g(\alpha)| &\ge \|g\|_1^{1-n_0}M(\alpha)^{-m}\\ &\ge ((m+1)\mathrm{height}(g))^{1-n_0}M(\alpha)^{-m}\\ &\ge ((m+1)\mathrm{height}(g))^{1-n_0}((n_0+1)^{1/2}\mathrm{height}(\alpha))^{-m},\end{aligned}\tag{4.4}$$

再由 α 和 $g(x)$ 的高度均不超过 H, 结论得证. 式 (4.4) 中的第二个不等号是因为 $\mathrm{height}(g)\le\|g\|_1\le(m+1)\mathrm{height}(g)$, 第三个不等号是由 Landau 不等式 [58, Theorem 6.31] $M(\alpha)\le\|p\|_2$ 及 $\mathrm{height}(p)\le\|p\|_2\le\sqrt{n+1}\mathrm{height}(p)$ 得到的, 其中 $p=\sum_{i=0}^{n_0}p_i x^i\in\mathbb{Z}[x]$ 为 α 的极小多项式, $\|p\|_2$ 表示其 2-范数, 即 $\|p\|_2=(\sum_{i=0}^{n_0}p_i^2)^{1/2}$. □

定理 4.1 的证明. 必要性由 $|p(\bar{\alpha})| = |p(\bar{\alpha}) - p(\alpha)| \leq \epsilon \cdot n \cdot H$ 易得; 由 $|p(\alpha)| - |p(\bar{\alpha})| \leq |p(\alpha) - p(\bar{\alpha})|$ 知 $|p(\alpha)| \leq |p(\bar{\alpha})| + \epsilon \cdot n \cdot H < (n+1)^{1-\frac{3}{2}n} H^{1-2n}$, 从而由推论 4.4 得 $p(\alpha) = 0$, 充分性得证. 证毕. □

定理 4.1 保证了能够通过近似值 $\bar{\alpha}$ 得到其对应准确值的极小多项式. 结合式 (4.2) 和式 (3.3), 并令 $\gamma = 2$ 便得到基于整数关系探测算法近似重构代数数极小多项式的误差控制条件.

推论 4.5. *设 α 和 $\bar{\alpha}$ 意义同上, 且满足*

$$\max_{1 \leq i \leq n} |\alpha^i - \bar{\alpha}^i| < \max\{2^{-2n^2+4n}(n+1)^{-\frac{5}{2}n},\, 2^{-n^2+2n}(n+1)^{-2n}\} \cdot H^{-2n}. \quad (4.5)$$

则对 $i = 1, \cdots, n$, 一个 $\bar{\boldsymbol{v}} = (1, \bar{\alpha}, \cdots, \bar{\alpha}^i)^T$ 的高度不超过 $2^{n-2}\sqrt{n+1}H$ 的整数关系一定同时是 $\boldsymbol{v} = (1, \alpha, \cdots, \alpha^i)^T$ 的整数关系.

证明. 取最大值的两个部分是分别将 (4.2) 中的 H, 定理 4.1 中的 $p(x)$ 和推论 4.4 中 $g(x)$ 的高度的上界 H 替换为 $2^{n-2}\sqrt{n+1}H$ 得到的. 证毕. □

然而, 推论 4.5 并未保证 $\boldsymbol{v}$ 的一个高度不超过 $2^{n-2}\sqrt{n+1}H$ 的整数关系一定同时是 $\bar{\boldsymbol{v}}$ 的整数关系. 即通过上述误差控制, 利用算法 3.1 并不能保证得到 $\bar{\boldsymbol{v}}$ 的高度不超过 $2^{n-2}\sqrt{n+1}H$ 同步整数关系. 但是由定理 4.1, 对某一个 $i \in \{1, \cdots, n\}$ 一定存在高度不超过 $2^{n-2}\sqrt{n+1}H$ 的一组向量 $\boldsymbol{p}$, 使得

$$|\langle \bar{\boldsymbol{v}}, \boldsymbol{p} \rangle| < \delta, \quad (4.6)$$

其中 $\delta = 2^{-2n^2+5n-3}(n+1)^{-\frac{5}{2}n+\frac{3}{2}} H^{-2n+1}$. 比如, α 极小多项式的系数向量便满足上式. 为了找出这样的整数向量, 需要对算法 3.1 的步骤 3e 作如下调整, 得到算法 SIRD′:

3(e') 记 X^T 的第 i 列为 $\begin{pmatrix} x_{i,1} \\ x_{i,2} \end{pmatrix}$. 若 $|x_{j,1}|$ 和 $|x_{j,2}|$ 同时小于 $\frac{\delta}{\sqrt{2}}$, 则输出 B 的第 j 列; 若 $h_{n-2,n-2} = 0$, 则输出 B 的第 $n-2$ 列.[1]

[1]因为对于复数代数数极小多项式的恢复, 仅需 SIRD 算法的 $t = 2$ 的情形即可, 所以 X^T 的行数为 2.

4.2.2　重构算法及分析

基于上述分析, 在 (4.5) 的误差控制下, 对实代数数用改进的 PSLQ 算法, 虚代数数用算法 SIRD′ 便得到一个解决问题 4.1 的完备算法.

算法 4.1 (MiniPoly)**.** 输入: 代数数 α 的满足 *(4.5)* 的近似值 $\bar{\alpha}$; α 次数的上界 n 和高度的上界 H.

输出: α 的极小多项式.

1. 对 i 从 1 到 n 执行如下循环
   (a) 构造向量 $\boldsymbol{v} := (1, \bar{\alpha}, \cdots, \bar{\alpha}^i)^T$.
   (b) 令 $M = 2^{n-2}\sqrt{n+1}H$. 以 $\boldsymbol{v}$, M 为输入调用整数关系探测算法 *PSLQ* 或算法 *SIRD′*.
      i. 若整数关系探测算法输出向量 $(p_0, p_1, \cdots, p_i)^T$, 则输出多项式 $\sum_{k=0}^{i} p_k x^k$ 的本原部分.

定理 4.6. *设一未知代数数 α 的次数不超过 n, 高度不超过 H. 若 α 的近似值 $\bar{\alpha}$ 满足 (4.5), 则算法 4.1 将正确输出 α 准确的极小多项式.*

证明. 设 α 的精确次数为 $n_0 (\leq n)$. 当算法 4.1 中的 $i < n_0$ 时, 不可能有输出. 假设此时有输出. 这意味着整数关系探测算法返回了一个高度不超过 $M = 2^{n-2}\sqrt{n+1}H$ 的的整数向量, 并且满足式 (4.6). 而由这两点, 结合误差控制条件 (4.5) 便可以得到该输出就是准确的极小多项式, 与 $i < n_0$ 矛盾. 当 $i = n_0$ 时, 算法 4.1 一定能返回正确的极小多项式. 这是因为定理 3.4 作相应调整后对算法 SIRD′ 仍然成立: 若满足式 (4.6) 的 $\boldsymbol{p}$ 存在, 则算法 SIRD′ 一定能找到. 所以对 $\bar{\boldsymbol{v}} = (1, \bar{\alpha}, \cdots, \bar{\alpha}^{n_0})$, 算法 SIRD′ 一定会输出一个整数向量. 若 $h_{n-2,n-2} = 0$, 则输出是 $\bar{\boldsymbol{v}}$ 的整数关系; 否则输出的整数向量满足式 (4.6). 尽管这个整数向量可能不是 $\bar{\boldsymbol{v}}$ 的整数关系, 但在 (4.5) 的误差控制下定理 4.1 保证了它必定是 $(1, \alpha, \cdots, \alpha^{n_0})$ 的整数关系. 所以, 从该整数向量必能得到 α 准确的极小多项式. 证毕. □

于是, 由误差控制条件式 (4.5) 和定理 3.5 可得如下推论.

推论 4.7. *设 α 为次数不超过 n, 高度不超过 H 的代数数. 假设 α 的近似值 $\bar{\alpha}$ 满足式 (4.5). 则 α 的极小多项式能够在对不超过 $\mathcal{O}(n(n + \log H))$ 位的整数进行 $\mathcal{O}(n^4(n + \log H)$ 次算术操作后得到.*

证明. 由本节的误差分析和对算法 3.1 的分析可知, 算法 4.1 即满足此推论. □

表 4.1: 不同的极小多项式恢复算法的复杂度比较

	精度	复杂度
KLL [95]	$\mathcal{O}(n^2+n\log H)$	$\mathcal{O}(n^5+n^4\log H)$
Just [80]	$\mathcal{O}(n^2+n^2\log H)$	$\mathcal{O}(n^8\log n+n^8\log H)$
QFCZ [127]	$\mathcal{O}(n^2+n\log H)$	——
MiniPoly	$\mathcal{O}(n^2+n\log H)$	$\mathcal{O}(n^5+n^4\log H)$

表 4.1 给出了四种不同的极小多项式近似重构的策略在最坏情况下所需的精度和复杂度比较. 由于 [127] 只能够对实代数数求出极小多项式, 因此不纳入比较. 同时, 需要提及的一点是, 对于基于 LLL 的重构算法 (见第 4.1 节), 若采用较新的浮点的 LLL 算法, 如 L^2 [121], H-LLL 算法 [117], Gradual-LLL 算法 [69], $\tilde{\mathrm{L}}^1$ 算法 [123] 等, 则可以得到更低的复杂度. 尽管如此, 算法 4.1 仍然为问题 4.1 提供了一个异于基于 LLL 算法的解决方案. 并且, 如果能够像 LLL 算法一样, 建立起一系列高效的 (同步) 整数关系探测算法的浮点版本, 则算法 4.1 的效率可能也会进一步提高.

4.3 实例分析

本文作者在计算机代数系统 Maple 13 中实现了 SIRD 算法和算法 4.1, 开发了 sird 程序包. 在算法 4.1 的实现中, 涉及到实代数数的处理时, 直接调用 Maple 中自带的 `PSLQ` 命令; 由于采用高精度运算, 只能保证对 $\boldsymbol{v}=(1,\bar{\alpha},\cdots,\bar{\alpha}^i)$ 探测近似的(同步)整数关系, 即可以求出一组整数向量 $\boldsymbol{p}=(p_0,p_1,\cdots,p_i)$, 使得 $\langle\boldsymbol{v},\boldsymbol{p}\rangle$ 充分接近于 0. 为了进一步保证程序输出的正确性, 在得到整数关系 $\boldsymbol{p}_i$ 后, 程序中还利用式 (4.3) 中对应的条件 (H 替换为 $2^{n-2}\sqrt{n+1}H$) 作为判断其是否确为 $\bar{\alpha}$ 所对应的准确极小多项式系数的依据.

本节所有的数值实验都是使用 2GB 内存, AMD Athlon™ 7750 处理器 (2.70 GHz) 的机器在 Windows XP 下用 Maple 13 计算出来的.

例 4.2 ([150], 例 1)**.** 设 α 为一未知的正有理数, 即次数为 1 的代数数, 其分母绝对值的上界为 $H=170$. 此例中分母的上界恰好是该有理数高度的上界. 首先 (4.5) 给出的误差控制为 $\epsilon\le 1/(\sqrt{2}H^2)=\sqrt{2}/57800$. 在此误差控制下通过某种

数值方法得到该有理数的一个近似值 $\bar{\alpha} = .8106335868$, 在计算过程中将精度设置为 $-\lfloor \log_{10}(\sqrt{2}/57800) \rfloor = 5$ 位, 执行下面的命令

```
[> read sird; Digits := -floor(evalf(log[10](sqrt(2)/57800)));
[> f := ExactMinimalPolynomial(.8106335868, 1, 170);
```

输出 $f := -137 + 169 * \_X$, 即 α 的极小多项式. (在 sird 包中对应于算法 4.1 的命令是 `ExactMinimalPolynomial`.) 由此得出 α 的准确值为 $\frac{137}{169}$. 运行

```
[> f := ExactMinimalPolynomial(.81063, 1, 170);
```

同样输出 $f := -137 + 169 * \_X$, 而运用 [150] 中的算法对近似值 $\bar{\alpha} = .81063$ 仅仅能够返回 $\frac{107}{132}$. 这再次说明了式 (4.5) 误差控制条件的优势.

另外, 当 α 是有理数时, [150] 中的算法只需知道分母绝对值的上界, 而本文中的算法需要知道该有理数高度的上界, 即分子分母绝对值的最大值的上界. 在实际应用时, 若只知道分母绝对值的上界, 则将得到的近似值分成整数部分和小数部分, 然后对小数部分应用算法 4.1 将其恢复成准确值即可. 由于整数部分本身就是精确的, 因此结果仍是精确的.

同时, 在此例中若将 `Digits` 设成一个小于 5 的正整数, 则不能返回准确结果. 这说明式 (4.5) 的误差控制在转化成精确十进制表示时是比较精确的. 另外, 若将 `Digits` 设成一个大于 5 的整数, 则对应的近似值也需提高到相应精度才能保证结果的准确性.

在引言部分已经提到, 利用 LLL 算法也可以解决问题 4.1. 计算机代数系统 Maple 13 已经集成了该功能, 即 `PolynomialTools` 包中的 `MinimalPolynomial` 命令. 但是其实现方法并不能保证得到该代数数的极小多项式, 而仅仅能够得到一个以该近似代数数为根的整系数多项式. 为了将本文算法和基于 LLL 的算法进行比较, sird 程序包中也包含了一个基于 SIRD 的与 `MinimalPolynomial` 命令具有相同功能的命令 `MiniPoly`.

例 4.3. 设 $\alpha = \sqrt[5]{23} + \sqrt[6]{7}$. 易知其次数为 30. 若想求得 α 的极小多项式, 可以采用算法 2 通过其近似值重构获得. 经试验, 当 `Digits:=300` 时, 算法 4.1 便能保证输出正确结果. 通过某种近似计算方法得到一个 α 的具有 300 位准确十进制表示的近似值 $a = 3.2552587848233\cdots$, 并运行如下命令

```
[> Digits := 300:  p_SIRD := MiniPoly(a, 30);
```

经过 9.718 秒的运算, 最终输出的多项式 p_SIRD 是一个 30 次的不可约多项式, 经验证确为 α 的极小多项式. 而运行 Maple 13 中自带的命令

```
[> PolynomialTools:-MinimalPolynomial(a, 30);
```

经过 9.218 秒的运算, 输出一个错误的多项式. 这并不是 Maple 13 的失误, 而是由其算法本身需要的高精度导致的. 若将 `Digits` 设为 398, 则 Maple 13 自带的命令经 13.719 秒获得正确的结果.

经多次试验发现, 当 `Digits` 不大时, `MiniPoly` 的运行时间略长于 `MinimalPolynomial`. 但当 `Digits` 较大时, 如上例所示, 算法 4.1 运行时间更优. 因此算法 4.1 对较大规模问题有较好的效果. 例如利用 `MiniPoly` 命令可以成功地从代数数 $3^{1/6} - I * 2^{1/7}$ 的近似值重构出其准确的极小多项式, 其次数为 84, 高度为 3029254676588448:

$$
\begin{aligned}
& 5067001 + 783962907\,x^{36} + 21027764272536\,x^{40} - 7504504\,x^{42} \\
+\ & 83639618394696\,x^{34} - 36683081862336\,x^{38} + 770305668258672\,x^{32} \\
+\ & 142394998636968\,x^{28} + 1254656434122\,x^{30} + 1370000831472\,x^{10} \\
+\ & 34207465357611\,x^{12} - 284692059376032\,x^{14} + 24758141678424\,x^{16} \\
+\ & 190959510258972\,x^{18} + 2306173886216928\,x^{20} + 99120704967648\,x^{22} \\
+\ & 64111149001809\,x^{24} - 3029254676588448\,x^{26} + 250312437648\,x^{44} \\
+\ & 2189187\,x^{48} - 112615776\,x^{50} - 486486\,x^{54} + 81081\,x^{60} - 378550368\,x^{2} \\
+\ & 11935794528\,x^{4} + 190431110646\,x^{6} + 3293025660288\,x^{8} + x^{84} \\
+\ & 88074554904\,x^{52} + 240\,x^{56} + 1041237288\,x^{58} + 1952496\,x^{64} - 9828\,x^{66} \\
+\ & 24\,x^{70} + 819\,x^{72} - 42\,x^{78} + 2212809521832\,x^{46}.
\end{aligned}
$$

现在, 回到上一章中对 SIRD 算法的两种实施上来.

表 4.2 中, r 和 s 定义了一个代数数 $\alpha = 3^{1/r} - 2^{1/s}I$, 易知 α 的次数为 $2rs$. 为测试 SIRD 算法的两种实现在恢复近似代数数极小多项式中的效率, 在 `Digits` 位十进制精度下, 探测两个 $n = 2rs + 1$ 维实向量的同步整数关系. 表 4.2 中 itr 和 t 分别表示迭代次数和运行时间 (单位：秒), 下标为 SIRD 的对应 SIRD 算法的直接实现 (全部采用高精度运算), 下标为 TLSIRD 的对应于 SIRD 算法的 “双层” 实现 (部分采用高精度运算). 表中最后两个例子, 对 SIRD 算法的直接实现没有完成计算. 由于 hfloat 数据类型采用硬件算术操作, 效率远远高于采用高精度的 sfloat 数据类型的操作, 所以 TLSIRD 程序的效率应明显优于 SIRD 程序, 这与表 4.2 中所给出的数据相符. SIRD 算法的高效的实现也为更多 SIRD 算法和算法 4.1 的应用提供了可能.

表 4.2: 用 SIRD 算法的两种实现来恢复极小多项式的效率比较

r	s	Dim. n	Digits	itr_{SIRD}	t_{SIRD}	itr_{TLSIRD}	t_{TLSIRD}
4	3	25	100	5685	75.937	5611	8.125
3	5	31	150	11792	356.890	11792	28.157
6	3	37	300	18927	556.031	18993	73.109
4	5	41	350	26600	942.516	26192	134.360
5	5	51	400	50084	2432.672	49738	440.110
6	5	61	550	84677	6422.079	81758	1267.985
4	9	73	1000			159326	4889.922
6	7	85	1300			234422	10658.735

比如算法 4.1 可以应用到有理数域上的一元多项式因式分解. 其基本思想是先求出待分解多项式的一个适当精度的近似根, 然后利用算法 4.1 可以得到该根准确的极小多项式, 而该极小多项式就是原多项式的一个因子. 对原多项式除以该因子后得到的多项式重复以上过程, 直到得到原多项式完全的不可约分解. 尽管该方法并不新颖, 效率也不比现有的分解算法高, 但该方法的中间步骤都采用近似计算, 输出结果却是精确的. 因此, 仍不失为一个符号-数值混合计算的成功案例.

例 4.4. 采用上述策略分解一元整系数多项式

$$\begin{aligned} g = \; & 3650 + 6745\,\_X^6 - 1188\,\_X^5 - 5973\,\_X^3 + 10094\,\_X^2 - 7916\,\_X \\ & + 4601\,\_X^4 + 4200\,\_X^{10} + 560\,\_X^9 - 6263\,\_X^7 + 4650\,\_X^8. \end{aligned}$$

该多项式无实根, 因此 [127] 中的分解方法不再奏效. 首先运行如下命令

```
[> Digits := 51:  a := fsolve(g, _X, complex)[1];
```

得到 g 的一个复近似根, 然后调用 sird 包中的 `ExactMinimalPolynomial` 命令

```
[> f1 := ExactMinimalPolynomial(a, 10, 256*90106990^ (1/2));
```

便得到该近似根对应的准确极小多项式 $50 - 42\,\_X + 40\,\_X^2 + 7\,\_X^3 + 10\,\_X^5 + 75\,\_X^6$, 它自然是 g 的一个不可约因子, 其中 $256\sqrt{90106990}$ 为该近似根对应的准确代数数高度的上界, 是利用 Mignotte 界 [58] 得到的. 再通过多项式除法得到 $g/f1 = 56\,\_X^4 + 62\,\_X^2 - 97\,\_X + 73 \equiv \_X^4 + 2\,\_X^2 + 3\,\_X + 3 \mod 5$ 不可约, 从而完成在有理数域上对 g 的因式分解.

4.4 本章小结

对任意的复代数数, 本文在其近似值和准确的极小多项式之间建立了一座桥梁. 在已知某代数数次数和高度上界的情形下, 只要预先得到该代数数的一个满足 (4.5) 的近似值, 算法 4.1 便能有效地给出该近似代数数的准确值的极小多项式. 这便有力地推动了“采用近似计算获得准确值”这一问题的研究, 使其适用范围扩大到任意的代数数. 同时, 这一算法也蕴含了一个代数数的整数表示方案 (如 [9, 95] 等). 并且本文的算法可以应用于因式分解等诸多实际问题.

在下一章中, 将给出一个采用恢复代数数的极小多项式来分解有理系数二元多项式的算法.

第五章　有理数域上二元多项式因式分解

多项式的因式分解是属于计算机代数中的核心部分, 它在诸如代数关系式的化简、符号积分、多项式求解、编码理论及序列密码等研究领域中都有着非常重要的应用. 然而, 目前的有理数域上的多项式准确的因式分解主要采用符号方法. 符号计算的主要特点是计算结果准确和计算过程稳定, 但它的劣势在于计算复杂度高. 因此, 在实际应用中效率不高, 往往只能解决中小规模的问题. 目前的计算机代数系统如 Maple [108], Mathematica [110], Magma [107] 等都将因式分解作为一个基本的功能给予实现. 数值近似计算具有速度快、计算规模大的特点, 但是它只能给出近似结果. 正是由于人们注意到符号计算和数值近似计算各自的优势, 近三十年来, 陆续有学者研究将数值方法应用到多项式的因式分解中, 发展出了基于符号-数值混合计算的因式分解新方向, 如 [36, 39, 55, 72, 73, 84, 93, 131, 135, 137]. 本章中将给出有理数域上的二元多项式的两个分解算法. 它们都是符号-数值混合计算的算法. 算法的中间过程是数值的, 而计算结果是精确的.

给定一个待分解的多项式 $f(x,y) = \sum_i \sum_j f_{i,j} x^i y^j \in \mathbb{Q}[x,y]$, 用 d_x 和 d_y 表示 f 关于 x 和 y 的次数, $||f||_1 = \sum_i \sum_j |f_{i,j}|$ 表示其 1-范数, $\|f\| = (\sum_i \sum_j |f_{i,j}|^2)^{1/2}$ 为其 2-范数, $\mathrm{height}(f) = \max_{i,j} |f_{i,j}|$ 为其高. 由 Gauss 引理知, 只需讨论本原多项式 (primitive polynomial, 系数的最大公因子为 1) $f(x,y) = \sum_i \sum_j f_{i,j} x^i y^j \in \mathbb{Z}[x,y]$ 在整数环上的因式分解.

1985 年, Hulst 和 Lenstra 给出了一个基于超越数的分解有理系数二元多项式的算法 [73]. 其基本思想是: 将待分解的二元多项式 $f(x,y) \in \mathbb{Z}[x,y]$ 中的一个变量, 比如 y, 用一个超越数 λ 来替换, 然后计算 $f(x,\lambda)$ 在 $\mathbb{Q}(\lambda)[x]$ 中的因式分解. 由于 λ 是超越数, 所以 $\mathbb{Q}(\lambda)$ 同构于 $\mathbb{Q}(y)$, 由此 $f(x,\lambda)$ 在 $\mathbb{Q}(\lambda)[x]$ 中的分解本质上就给出了 $f(x,y)$ 在 $\mathbb{Q}$ 上的分解, 由 Gauss 引理, 这个分解就是 $f(x,y)$ 在 $\mathbb{Z}$ 上的分解. 为了得到 $f(x,\lambda)$ 在 $\mathbb{Q}(\lambda)[x]$ 中的分解, 超越测度 ([73, Definition (1.1)]) 的计算不可避免. 然而, 这一数量有两个弊端: 第一, 对于不同的超越数, 超越测度的表达式不一样; 第二, 每一个超越测度的表达式中都含有一个常数 $c > 0$, 该常数不易确定. 在第 5.1 节中, 将利用 [73] 中的算法思想, 但是采用高次代数数来替代超越数的方法来分解有理系数二元多项式的算法, 该算法不仅

克服了超越测度的弊端, 而且实验结果显示, 对于小规模的多项式进行分解, 该算法的效率甚至高于 Maple 自带的 `factor()` 命令.

在第 5.2 节中, 将给出一个基于多项式支撑集的 *Newton* 多胞体 (Newton polytope) 的有理二元多项式的分解算法 *BiFactor*. 该算法对于稀疏的二元有理多项式具有很好的分解效果. 特别地, 给定一个 $\mathbb{Q}$ 上满足假设 *(H)* (见第 5.2 节) 的二元多项式 f, 该算法能够在 $\tilde{\mathcal{O}}(Td_xd_y + s^\omega)$ 次算术操作步骤内将 f 的有理不可约因子的计算任务约化为一个次数不超过 d_x 的有理单变元多项式的因式分解, 其中 T 为 f 中的非零项数, $s \le d_x$ 为某整数, ω 为线性代数指数 (两个 $n \times n$ 的矩阵乘法能够在 $\mathcal{O}(n^\omega)$ 次算术操作内完成). 据本文作者所知, 约化二元有理多项式的有理不可约分解到单变元有理分解的最好的复杂度是 Lecerf 得到的 [101]: 确定性算法 $\tilde{\mathcal{O}}((d_xd_y)^{(\omega+1)/2})$; 概率算法 $\tilde{\mathcal{O}}((d_xd_y)^{1.5})$. Lecerf 的算法针的是稠密的二元多项式. 本章第 5.2 节中的算法对稠密的二元多项式也有效, 只是此时 $T = \mathcal{O}(d_xd_y)$. 所以, 针对稀疏的情形, 比如非零项数满足 $T < (d_xd_y)^{1/2}$, 则 $\tilde{\mathcal{O}}(Td_xd_y + s^\omega) \subseteq \tilde{\mathcal{O}}((d_xd_y)^{1.5})$, 即本文中的算法在这种情形下优于 [101] 中的方法. 在 [148] 中, 基于 Newton 多胞体, Weimann 给出了 Lecerf 算法的一个稀疏版本, 其约化的复杂度为 $\mathcal{O}(\mathtt{Vol}(N_f)^\omega)$, 其中 $\mathtt{Vol}(N_f)$ 为 f 的 Newton 多胞体的体积. 然而, 即使是稀疏情形, 最坏情况下的 $\mathtt{Vol}(N_f)$ 也接近于 d_xd_y. 并且 [148, Hyp. (H1)] 过于强烈, 影响了该算法的适用范围.

本文作者在 Maple 中实现了 BiFactor 算法, 实验显示, 该算法具有非常好的效果 (见第 5.2 节).

5.1 基于近似代数数的分解算法

本节中, 将用 $p_\lambda(x)$ 表示代数数 λ 在 $\mathbb{Q}$ 上的极小多项式. 记代数数 λ 的次数为 M, 即 $M = [\mathbb{Q}(\lambda) : \mathbb{Q}] = \deg(p_\lambda)$. 设 $f(x, y) \in \mathbb{Z}[x, y]$ 为本原多项式, $d_x = n$, $d_y = m$. 为了计算 $f(x, y)$ 在 $\mathbb{Z}$ 上的不可约分解, 首先用一个次数 $M > 2m(n + 1)$ 代数数 λ 代替变量 y, 然后计算 $f(x, \lambda)$ 的一个近似根 $\bar{\alpha}$, 记其对应的准确根为 α. 至此, 可以计算 α 在 $\mathbb{Q}(\lambda)$ 上的极小多项式, 记为 $h(x, \lambda) \in \mathbb{Z}[\lambda][x]$. 由如下引理可知, $h(x, y)$ 就是 f 在 $\mathbb{Z}[x, y]$ 中的一个因子. 而通过引理 5.5 知该因子不可约. 此时, 更新 $f := f/h$, 重复以上过程直至分解结束.

引理 5.1. 设 $f(x, y) \in \mathbb{Z}[x, y]$, $d_x = n$, $d_y = m$, λ 为一次数 $M > (n + 2)m$ 的代

数数. 若 $h(x,y)\in\mathbb{Z}[x,y]$ 满足$\deg_x(h)\le n$, $\deg_y(h)\le m$, 且 $h(x,\lambda)$ 为 $f(x,\lambda)$ 在 $\mathbb{Z}[\lambda][x]$ 中的因子, 则 $h(x,y)$ 为 $f(x,y)$ 在 $\mathbb{Z}[x,y]$ 中的一个因子.

证明. 因为 $h(x,\lambda)|f(x,\lambda)$, 所以

$$f(x,\lambda)=h(x,\lambda)g(x,\lambda). \tag{5.1}$$

对 $f(x,y)$ 和 $h(x,y)$ 关于变量 x 做除法得

$$I(y)^t f(x,y)=q(x,y)h(x,y)+r(x,y), \tag{5.2}$$

其中$I(y)=\mathrm{lc}_x(h(x,y))$, $q,h,r\in\mathbb{Z}[x,y]$, $t\in\mathbb{Z}^+$, $\deg_x(r)<\deg_x(h)$. 因此, $I(\lambda)^t f(x,\lambda)=q(x,\lambda)h(x,\lambda)+r(x,\lambda)$. 再由 (5.1) 知

$$h(x,\lambda)(I(\lambda)^t g(x,\lambda)-q(x,\lambda))=r(x,\lambda).$$

比较上式两端关于 x 的次数知 $r(x,\lambda)=0$. 令 $r(x,y)=\sum_{i=0}^{\deg_x(r)}\sum_{j=0}^{\deg_y(r)}r_{i,j}x^iy^j$. 则

$$\sum_{i=0}^{\deg_x(r)}\left(\sum_{j=0}^{\deg_y(r)}r_{i,j}\lambda^j\right)x^i=0. \tag{5.3}$$

从式 (5.2) 知

$$\begin{aligned}\deg_y(I(y))\le&\deg_y(h),\\ \deg_y(r)\le\deg_y(I(y)^t)+&\deg_y(f)\le t\cdot\deg_y(h)+\deg_y(f),\\ t\le\deg_x(f)-&\deg_x(h)+1\le n+1,\end{aligned}$$

因此, $\deg_y(r)\le(n+2)m<M$. 所以由式 (5.3) 知 $r_{i,j}=0$, 即 $r(x,y)=0$. 同时, $q(x,y)=I(y)^t g(x,y)$, 故 $h(x,y)$ 为 $f(x,y)$ 的一个因子. □

5.1.1　误差控制

为了避免中间过程膨胀 (intermediate expression swell), 在算法中不直接对代数数 λ 进行操作, 而是采用其近似值 $\bar{\lambda}$. 记 $\bar{\lambda}_j$ ($0\le j\le m$) 为 λ^j 的近似值, 其中 $\bar{\lambda}_0=1$. 现引入如下记号表示 $f(x,y)$ 在 (x,λ) 处的近似值: $f_{\bar{\lambda}}=\sum_i\sum_j f_{i,j}x^i\bar{\lambda}_j$. 在算法中, 用 $f_{\bar{\lambda}}$ 代替 $f(x,\lambda)$. 不失一般性, 可以假设 $f_{\bar{\lambda}}$ 有一个绝对值不超过 1 的根, 否则可以采用多项式 $x^nf(\frac{1}{x},y)$ 替代 $f(x,y)$.

接下来, 将首先讨论如何确定 λ 的近似值使得 $f_{\bar{\lambda}}$ 的的根能够近似 $f(x,\lambda)$ 的根.

引理 5.2 ([73], Lemma 1.3)**.** 设 $f=\sum_{i=0}^{n} f_i x^i$, $\bar{f}=\sum_{i=0}^{n} \bar{f}_i x^i \in \mathbb{C}[x]$ 为次数为 $n>0$ 的两个多项式, 令 $\Delta=\max_{0\le i\le n}|f_i-\bar{f}_i|$. 假设 $\bar{f}$ 有复根 $\beta\in\mathbb{C}$ 满足 $|\beta|\le 1$. 则存在 f 的零点 $\alpha\in\mathbb{C}$ 满足

$$|\beta-\alpha|\le\left(\frac{(n+1)\Delta}{|f_n|}\right)^{1/n}.$$

证明. 因为 $f(x)-\bar{f}(x)=\sum_{i=0}^{n}(f_i-\bar{f}_i)x^i$, 所以 $|f(\beta)|\le\Delta\sum_{i=0}^{n}|\beta|^i$. 又因为 $|f(\beta)|=|f_n|\prod_{i=1}^{n}|\beta-\alpha_i|$, 其中 $\alpha_1,\alpha_2,\ldots,\alpha_n$ 为 f 的零点, 所以结论成立. □

引理 5.3. 设 $f(x,y)=\sum_{i=0}^{n}\sum_{j=0}^{m} f_{i,j}x^i y^j$, λ 为一代数数. 若 $f(x,\lambda)=\sum_{i=0}^{n} f_i x^i$ 满足 $f_n\neq 0$, 则

$$|f_n|\ge((1+m)\mathrm{height}(f))^{1-M}\|p_\lambda\|^{-m},$$

其中p_λ 为 λ 的极小多项式, $\deg(p_\lambda)=M>m$.

证明. 因为 $0\neq f_n=\sum_{j=0}^{m} f_{n,j}\lambda^j$, 所以对多项式 $f_n(y)=\sum_{j=0}^{m} f_{n,j}y^j\in\mathbb{Z}[y]$ 有 $f_n=f_n(\lambda)\neq 0$. 因此, 由引理 4.3 知

$$|f_n|=|f_n(\lambda)|\ge\| f_n(y)\|_1^{1-M}\ M(\lambda)^{-m}. \tag{5.4}$$

因为 $||f_n(y)||_1\le(1+m)\mathrm{height}(f_n(y))$, $M(\lambda)\le\|p_\lambda\|$ (见 [58, Theorem 6.31]), $\mathrm{height}(f_n(y))\le\mathrm{height}(f)$, 结合式 (5.4) 知结论成立. □

称 b 为 a 的一个 2^{-s}-近似 (2^{-s}-approximation), 如果 $|a-b|<2^{-s}$.

引理 5.4. 设 β 为 $f_{\bar{\lambda}}$ 的一个绝对值不超过 1 的根的 2^{-s-1}-近似. 对任意的 k, 若

$$|\lambda^k-\bar{\lambda}_k|<(2^{sn+n}(n+1)((1+m)\mathrm{height}(f))^M\ \|p_\lambda\|^m)^{-1}, \tag{5.5}$$

则 β 也是 $f(x,\lambda)$ 的一个根的 2^{-s}-近似.

证明. 设 $f(x,\lambda)=\sum_{i=0}^{n} f_i x^i$ 且 $f_{\bar{\lambda}}(x)=\sum_{i=0}^{n}\bar{f}_i x^i$. 由引理 5.2, 存在 $f(x,\lambda)$ 的一个根 α 满足

$$|\beta-\alpha|\le\left(\frac{(n+1)\max_i|f_i-\bar{f}_i|}{|f_n|}\right)^{1/n}.$$

由引理 5.3, $\max_i|f_i-\bar{f}_i|\le\mathrm{height}(f)\sum_{j=1}^{m}|\lambda^j-\bar{\lambda}_j|$, 以及式 (5.5) 知结论成立. □

假设现已计算出 $f_{\bar{\lambda}}$ 的一个绝对值不超过 1 的某个根的一个 2^{-s-1}- 近似 $\bar{\alpha} \in \mathbb{Q}(I)$. 由引理 5.4, $\bar{\alpha}$ 也是 $f(x,\lambda)$ 的根 $\alpha \in \mathbb{C}$ 的一个 2^{-s}-近似. 因为 $|\alpha| - |\bar{\alpha}| \le |\alpha - \bar{\alpha}| \le 2^{-s}$, 所以

$$|\alpha| \le 1 + 2^{-s}. \tag{5.6}$$

用 $\beta_{ij} \in \mathbb{Q}(I)$ 表示 $\alpha^i\lambda^j$ 的近似值, 其中 $0 \le i \le n$, $0 \le j \le m$, $\beta_{00} = 1$. 对正整数 s, 定义欧几里得格 $\Lambda_s \subseteq \mathbb{R}^{(n+1)(m+1)+2}$ 以如下 $(n+1)(m+1) \times ((n+1)(m+1)+2)$ 阶矩阵

$$B_s = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 2^s\mathrm{Re}(\beta_{00}) & 2^s\mathrm{Im}(\beta_{00}) \\ 0 & 1 & 0 & \dots & 0 & 2^s\mathrm{Re}(\beta_{01}) & 2^s\mathrm{Im}(\beta_{01}) \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 2^s\mathrm{Re}(\beta_{nm}) & 2^s\mathrm{Im}(\beta_{nm}) \end{pmatrix}$$

为生成矩阵, 记其行为 $\boldsymbol{b}_{00}, \boldsymbol{b}_{01}, \dots, \boldsymbol{b}_{0,m}, \boldsymbol{b}_{10}, \dots, \boldsymbol{b}_{nm}$. 考虑映射 $\mathbb{Z}[x,y] \to \Lambda_s$. 对于满足 $\deg_x(g) \le n$ 和 $\deg_y(g) \le m$ 的多项式 $g(x,y) = \sum_i \sum_j g_{i,j}x^iy^j \in \mathbb{Z}[x,y]$, 该映射下的象定义为 $\bar{\boldsymbol{g}} = \sum_i \sum_j g_{i,j}\boldsymbol{b}_{ij} \in \Lambda_s$. 记 $g_\beta = \sum_{i=0}^{n} \sum_{j=0}^{m} g_{i,j}\beta_{ij}$, 其中若 $\deg_x(g) < i \le n$ 或 $\deg_y(g) < j \le m$, 则 $g_{i,j} = 0$. 于是,

$$\|\bar{\boldsymbol{g}}\|^2 = \|g\|^2 + 2^{2s}|g_\beta|^2. \tag{5.7}$$

对上述格调用 LLL 算法得到 Λ_s 的一组 LLL-约化基. 设 $\bar{\boldsymbol{v}}$ 为这组 LLL-约化基的第一个向量. 则由 [102, Proposition (1.11)] 知

$$\|\bar{\boldsymbol{v}}\|^2 \le 2^{(n+1)(m+1)-1}\|\bar{\boldsymbol{h}}\|^2, \tag{5.8}$$

其中 $\bar{\boldsymbol{h}}$ 是 α 在 $\mathbb{Q}(\lambda)$ 上的极小多项式 $h(x,\lambda) \in \mathbb{Z}[\lambda][x]$ 所对应的二元多项式 $h(x,y)$ 在上述映射下的象.

若 $g \in \mathbb{Z}[x,y]$ 满足 $\deg_x(g) \le n$, $\deg_y(g) \le m$ 且 $g(\alpha,\lambda) \ne 0$, 则可以选择适当的 s (引理 5.8) 使得

$$\|\bar{\boldsymbol{g}}\|^2 > 2^{(n+1)(m+1)-1}\|\bar{\boldsymbol{h}}\|^2. \tag{5.9}$$

记向量 $\bar{\boldsymbol{v}}$ 的原象为二元多项式 $v(x,y)$. 由式 (5.8) 和式 (5.9) 知 $v(\alpha,\lambda) = 0$, 因此 $h(x,\lambda)|v(x,\lambda)$. 事实上, 因为 $\bar{\boldsymbol{v}}$ 是格 Λ_s 的基向量, 而 $\bar{\boldsymbol{h}}$ 为格 Λ_s 的一个元素, 所以 $h(x,\lambda)$ 必为 $\pm v(x,\lambda)$.

现在, 已经计算出 $f(x,\lambda)$ 的一个绝对值小于 1 的根 α 在 $\mathbb{Q}(\lambda)$ 上的极小多项式 $h(x,\lambda)$. 将其中的 λ 用 y 代替得到 $h(x,y)\in\mathbb{Z}[x,y]$. 由引理 5.1 知 $h(x,y)$ 是 $f(x,y)$ 的一个因子. 而下面的引理保证了 $h(x,y)$ 在 $\mathbb{Z}[x,y]$ 中的不可约性.

引理 5.5. *设 $h(x,\lambda)\in\mathbb{Z}[\lambda][x]$ 为代数数 λ 在 $\mathbb{Q}(\lambda)$ 上的极小多项式, 且 $[\mathbb{Q}(\lambda):\mathbb{Q}]=M>\deg_y(h(x,y))$, 其中 $h(x,y)$ 是将 $h(x,\lambda)$ 中的 λ 由 y 替换得到的. 则 $h(x,y)$ 在 $\mathbb{Z}[x,y]$ 中不可约.*

证明. 假设 $h(x,y)=h_1(x,y)h_2(x,y)$, 其中 $h_1,h_2\in\mathbb{Z}[x,y]$. 则

$$h(x,\lambda)=h_1(x,\lambda)h_2(x,\lambda).$$

因为 $h(x,\lambda)$ 是 α 在 $\mathbb{Q}(\lambda)[x]$ 中的极小多项式, 所以 $h_1(x,\lambda)=1$ 或者 $h_2(x,\lambda)=1$. 不妨设 $h_2(x,\lambda)\neq 1$. 由 $M>\deg_y(h(x,y))$, 再由证明引理 5.1 时所采用的伪除方法知 $h_1(x,y)=1$. 因此 $h(x,y)$ 在 $\mathbb{Z}[x,y]$ 中不可约. □

为了选取参数 s 使得式 (5.9) 成立, 需要如下一些引理.

引理 5.6. *若*

$$|\alpha^i\lambda^j-\beta_{ij}|\le 2^{-s+1} \tag{5.10}$$

对 $0\le i\le n$ 和 $0\le j\le m$ 成立, 则

$$|g(\alpha,\lambda)-g_\beta|\le 2^{-s+1}(mn+m+n)\mathrm{height}(g).$$

如下引理给出了一个当 $g(\alpha,\lambda)\neq 0$ 时, $|g(\alpha,\lambda)|$ 的下界.

引理 5.7. *设 $f(x,y)\in\mathbb{Z}[x,y]$ 满足 $\deg_x(f)=n$, $\deg_y(f)=m$, $g(x,y)\in\mathbb{Z}[x,y]$ 满足 $\deg_x(g)\le n$, $\deg_y(g)\le m$. 设 λ 为次数为 $M\ge 2mn$ 且满足 $|\lambda|\le 1/2$ 的代数数, α 为 $f(x,\lambda)$ 的根, $h(x,\lambda)$ 为 α 在 $\mathbb{Z}[\lambda][x]$ 中的极小多项式. 若 $g(\alpha,\lambda)\neq 0$, 则*

$$|g(\alpha,\lambda)|\ge\frac{((2mn+1)^{\frac{1}{2}}B)^{1-M}\|p_\lambda\|^{-2mn}}{4nB}, \tag{5.11}$$

其中 $B=(2^{m+n}\mathrm{height}(f)\mathrm{height}(g)(n+1)^{\frac{3}{2}}(m+1)^{\frac{5}{2}})^n$.

证明. 若 $\deg_x(g)=0$, 则 $g(\alpha,y)=g(y)\in\mathbb{Z}[y]$. 由 $M\geq 2mn>m\geq\deg_y(g)$ 知 $g(\lambda)\neq 0$. 由引理 4.3, 有

$$|g(\lambda)|\geq\| g\|_1^{1-M}\,\|p_\lambda\|^{-m}. \tag{5.12}$$

因此, 由式 (5.12) 知式(5.11) 成立.

现设 $\deg_x(g)>0$. 因为 $h(x,\lambda)$ 为 α 在 $\mathbb{Q}(\lambda)[x]$ 中的极小多项式, 又因为 $g(\alpha,\lambda)\neq 0$, 所以若将 h 和 g 看成是 $\mathbb{Z}[x,y]$ 的多项式, 则 $\gcd(h,g)=1$, 因此存在 $a,b\in\mathbb{Z}[x,y]$ 使得

$$a\cdot h+b\cdot g=R, \tag{5.13}$$

其中$R=\mathrm{Res}_x(g,h)\in\mathbb{Z}[y]$. 由于 $\deg_x(h)\leq n$ 和 $\deg_y(h)\leq m$, 有 $\deg_y(R)\leq m\deg_x(g)+n\deg_y(g)\leq 2mn$, $\deg_x(a)\leq\deg_x(g)-1$, $\deg_y(a)\leq m(\deg_x(g)-1)+n\deg_y(g)$, $\deg_x(b)\leq n-1$, 且 $\deg_y(b)\leq m\deg_x(g)+(n-1)\deg_y(g)$.

将 $x=\alpha$ 和 $y=\lambda$ 代入式 (5.13) 得

$$b(\alpha,\lambda)g(\alpha,\lambda)=R(\lambda),$$

因此,

$$|g(\alpha,\lambda)|=\frac{|R(\lambda)|}{|b(\alpha,\lambda)|}. \tag{5.14}$$

因为 $R(\lambda)\neq 0$, 所以

$$\begin{aligned}|R(\lambda)|&\geq\| R\|_1^{1-M}\,|p_\lambda|^{-deg_y(R)}\\&\geq((1+2mn)^{1/2}\|R\|)^{1-M}\|p_\lambda\|^{-2mn}\\&\geq((1+2mn)^{1/2}B)^{1-M}\|p_\lambda\|^{-2mn}.\end{aligned} \tag{5.15}$$

上式第一部分成立是因为引理 4.3, 第二部分成立是因为 $\deg_y(R)\leq 2mn$ 以及 $\|R\|_1\leq(1+\deg_y(R))^{1/2}\|R\|$.

因为 $h(x,y)|f(x,y)$, 由 [73] 知

$$\begin{aligned}\mathrm{height}(h)\leq\|h\|&\leq 2^{m+n}\|f\|\\&\leq 2^{m+n}(1+n)^{\frac{1}{2}}(1+m)^{\frac{1}{2}}\mathrm{height}(f),\end{aligned}$$

再由多项式行列式的 Hadamard 界 [62] 知

$$\mathrm{height}(R)\leq\|R\|\leq B. \tag{5.16}$$

从而, 式 (5.15) 的第三部分成立.

由式 (5.6) 知 $|\alpha|^i \le (1+2^{-s})^n \le 2$, 再结合 $|\lambda| \le 1/2$ 得

$$\begin{aligned} |b(\alpha,\lambda)| &\le \mathrm{height}(b)\sum_{i=0}^{n-1}\sum_{j=0}^{2mn-1}\left|\alpha^i\right|\cdot\left|\lambda^j\right| \\ &\le 4n\cdot\mathrm{height}(b) \\ &\le 4nB. \end{aligned} \tag{5.17}$$

式 (5.17) 成立还因为式 (5.16) 对将 R 替换成 b 后仍然成立. 因此, 由式 (5.14), 式 (5.15) 和式 (5.17) 可得式 (5.11) 成立. 证毕. □

结合引理 5.1 和引理 5.7 中的条件, 在本节的算法中, 将选择代数数 λ 满足 $|\lambda| \le 1/2$ 且其次数为 $M \ge 2m(n+1)$.

现在, 已做好了如何选取 s 的所有准备.

引理 5.8. *若*

$$2^s \ge \frac{2^{\frac{mn+3(m+n)+10}{2}}\|f\|(m+1)^2(n+1)^2}{(1+2mn)^{\frac{1-M}{2}}\|p_\lambda\|^{-2mn}}\cdot A^M, \tag{5.18}$$

其中 $A = (2^{\frac{mn+5(m+n)+2}{2}}\|f\|^2(n+1)^{\frac{5}{2}}(m+1)^{\frac{7}{2}})^n$, *则如下不等式成立:*

$$\|\bar{\boldsymbol{h}}\| < 2^{m+n+1}\|f\|(n+1)(m+1), \tag{5.19}$$

$$\|\bar{\boldsymbol{g}}\| > 2^{(mn+3(m+n)+2)/2}\|f\|(n+1)(m+1). \tag{5.20}$$

证明. 由 h 整除 f 知 $\deg_x(h) \le n$ 且 $\deg_y(h) \le m$, 所以 $\bar{\boldsymbol{h}} \in \Lambda_s$. 由 [73] 知 $\mathrm{height}(h) \le \|h\| \le 2^{m+n}\|f\|$. 结合 $h(\alpha,\lambda)=0$ 和引理 5.6, 有

$$\begin{aligned} \|\bar{\boldsymbol{h}}\|^2 &= \|h\|^2 + 2^{2s}|h_\beta|^2 \\ &\le (2^{m+n}\|f\|)^2 + 2^{2s}(2^{-s+1}2^{m+n}\|f\|(mn+n+m))^2 \\ &= (2^{m+n}\|f\|)^2(1+4(mn+n+m)^2) \\ &< (2^{m+n+1}\|f\|(n+1)(m+1))^2. \end{aligned}$$

现证式 (5.20). 因为 $\|\bar{\boldsymbol{g}}\|^2 = \|g\|^2 + 2^{2s}|g_\beta|^2$, 所以考虑两种情形: 若 $\|g\| > 2^{(mn+3(m+n)+2)/2}\|f\|(n+1)(m+1)$, 则 $|\bar{\boldsymbol{g}}| > 2^{(mn+3(m+n)+2)/2}\|f\|(n+1)(m+1)$. 若 $\|g\| < 2^{(mn+3(m+n)+2)/2}\|f\|(n+1)(m+1)$, 则证明

$$2^s\,|g_\beta| > 2^{(mn+3(m+n)+2)/2}\|f\|(n+1)(m+1). \tag{5.21}$$

由引理 5.6 和引理 5.7 知

$$\begin{aligned}|g_\beta| &\geq |g(\alpha,\lambda)| - 2^{-s+1}\mathrm{height}(g)(mn+m+n)\\ &\geq \frac{(1+2mn)^{\frac{1-M}{2}}\|p_\lambda\|^{-2mn}}{4nA^M}\\ &\quad - 2^{-s+1}2^{(mn+3(m+n)+2)/2}\|f\|(m+1)^2(n+1)^2\\ &\geq \frac{(1+2mn)^{\frac{1-M}{2}}\|p_\lambda\|^{-2mn}}{8nA^M}.\end{aligned}$$

所以 s 的选取条件式 (5.18) 蕴含着式 (5.21) 成立. □

引理 5.9. *设 f, h, n, m, λ, s, Λ_s 意义同上, 且式 (5.18) 成立. 若 $\bar{\boldsymbol{h}} \in \Lambda_s$, 则 $h = \pm v$. 特别地,*

$$\|\bar{\boldsymbol{v}}\| < 2^{(mn+3(m+n)+2)/2}\|f\|(n+1)(m+1). \tag{5.22}$$

证明. 若 $\bar{\boldsymbol{h}} \in \Lambda_s$, 则由式 (5.8) 知 $\|\bar{\boldsymbol{v}}\| \leq 2^{((m+1)(n+1)-1)/2}\left|\bar{\boldsymbol{h}}\right|$. 因此, 由式 (5.19), 式 (5.20) 和引理 5.8 知 $\|\bar{\boldsymbol{v}}\| < 2^{(mn+3(m+n)+2)/2}\|f\|(n+1)(m+1)$. 由此 h 整除 v. 又 $\bar{\boldsymbol{h}} \in \Lambda_s$, 且 $\bar{\boldsymbol{v}}$ 为 Λ_s 的一组基中的元素, 所以 $h = \pm v$. □

在算法中, 对每个 $m_0 = 0, 1, \ldots, m$, 取 $n_0 = 1, 2, \ldots, n$ 进行探测, 记格 Λ_s 的维数为 $N = n_0(m+1) + m_0 + 1$. 如果 N 是使得 $\bar{h} \in L_s$ 成立的最小的 L_s 的秩, 则式 (5.22) 成立, 于是得到一个 $f(x,y)$ 的因子. 若 $\bar{\boldsymbol{h}} \notin L_s$, 则需要一个更大的 N. 如果对所有的 N, 式 (5.22) 都不成立, 那么 $h = f$.

5.1.2　算法、分析和实验

由以上分析可以得到一个分解二元有理多项式的符号-数值混合计算的算法. 本节将描述和分析该算法, 并给出一些该算法的具体实例.

算法 5.1 (BiFact). *输入: 一个二元本原多项式 $f(x,y) \in \mathbb{Z}[x,y]$, $\deg_x(f) = n$, $\deg_y(f) = m$; 一个次数为 $M \geq 2m(n+1)$ 的代数数 λ 且 $|\lambda| \leq 1/2$.*
输出: $f(x,y)$ 在 $\mathbb{Z}[x,y]$ 中的所有的不可约因子.

1. 去除 f 的单项式因子, 并更新 f 为无单项因子的多项式.
2. 重复执行以下步骤直到 $f = 1$:
(a) 对 $0 \leq j \leq m$, 计算使得式 (5.5) 成立的 $\bar{\lambda}_j \in \mathbb{Q}$.

(b) 令 $f_{\bar{\lambda}} := \sum_{i=0}^{n}\sum_{j=0}^{m} f_{i,j}x^i\bar{\lambda}_j$. 计算 $f_{\bar{\lambda}}$ 的一个近似根 $\bar{\alpha}$. 对 $0 \le i \le n$, $0 \le j \le m$, 计算 β_{ij} 使得 *(5.10)* 成立.

(c) 对 n_0 从 1 到 n, 对 m_0 从 0 到 m 执行以下步骤:

i. 计算使式 *(5.18)* 成立的最小的 $s \in \mathbb{Z}$, 并调用 *LLL* 算法计算维数为 N 的格 Λ_s 的一组 *LLL*-约化基.

ii. 若式 *(5.22)* 成立则执行以下步骤:

A. 输出 $h = v$.

B. 令 $f := f/h$.

C. 若 f 是一个单变元多项式, 则调用 *Factor(f)*, 输出因子, 终止算法.

(d) 输出 $h = f$.

(e) 令 $f := f/h$.

算法 5.1 的第一步处理了一种特殊情形, 这保证了 $f_\lambda \neq 0$, 于是 β_{ij} 也不为 0. 从而 LLL-算法能够正确的计算出 Λ_s 的一组约化基.

在运行算法 5.1 的步骤 2(c)iiB 后, f 可能变为单变元的多项式, 此时算法 5.1 调用任意的一个一元多项式的因式分解算法 `Factor`, 然后直接输出结果.

算法 5.1 的分析. 首先, 由第 5.1.1 节的分析易知算法 5.1 能够正确的给出一个二元的本原多项式的因式分解.

现在, 来考虑算法 5.1 的时间复杂度. 由式 (5.18) 知

$$s = \mathcal{O}(n^3m^2 + n^2m\log\|f\|). \tag{5.23}$$

由 [73] 知计算 $f_{\bar{\lambda}}$ 的一个绝对值 < 1 的根的 2^{-s-1}-近似 $\bar{\alpha}$ 需要

$$\mathcal{O}(n^2(\max\{s, n\log\|f_{\bar{\lambda}}\|\})^{1+\epsilon})$$

次位操作. 又由式 (5.5) 和式 (5.23) 知 $\log\|f_{\bar{\lambda}}\| = \mathcal{O}(mn^2(n^3m^2 + n^2m\log\|f\|))$, 所以计算 $\bar{\alpha}$ 需要

$$\mathcal{O}(mn^5(n^3m^2 + n^2m\log\|f\|)^{1+\epsilon})$$

次位操作.

另外, 计算 h 需要 $\mathcal{O}(\bar{n}n^2m^3(n^3m^2 + n^2m\log\|f\|)^{2+\epsilon})$ 次位操作, 其中 $\bar{n} = \deg_x(h)$. 又 $\|f/h\| \le 2^{m+n}\|f\|$, 所以完全分解 f 为不可约因子的乘积需要 $\mathcal{O}(n^3m^3(n^3m^2 + n^2m\log\|f\|)^{2+\epsilon})$ 次位操作. 于是有

定理 5.10. *算法 5.1 能够在多项式时间内正确计算 $f(x,y)\in\mathbb{Q}[x,y]$ 的因式分解, 至多需要位操作的次数为 $\mathcal{O}(n^3m^3(n^3m^2+n^2m\log\|f\|)^{2+\epsilon})$.*

这个结果相对于 [73] 中的算法, 在时间复杂度上减少了一个 $\log^{4+\epsilon}(mn)$ 因子. 而以下的实验也显示, 该算法对于小规模的二元多项式, 有较高的效率.

实例研究. 在计算机代数系统 Maple 中, 算法 5.1 可以得到成功的实现. 这里列举几个简单的例子, 使得算法 5.1 能更加清晰明了. 本小节余下部分的实验结果都是在 2.53 GHz PIV CPU, 内存 384Mb 的 PC 中 Windows XP 操作系统下利用 Maple 11 计算得到的.

在下面的例子中, 均设 $\lambda=3^{-1/(2m(n+1))}$. 于是 $p_\lambda(x)=3x^{2m(n+1)}-1$, $\|p_\lambda\|=\sqrt{10}$. 运行 LLL-算法后得到向量 $v=(v_{00},v_{01},\ldots,v_{0m},v_{11},\ldots,v_{nm})$, 其对应的多项式为 $\sum_{i=0}^{n}\sum_{j=0}^{m}v_{ij}x^iy^j$.

例 5.1. 设 $f:=2\,x^2+3\,xy+2\,x+y+y^2$. 对上述 f 有 $n=2, m=2, \mathrm{height}(f)=3$, $\|f\|=\sqrt{19}$. 首先根据式 (5.18) 计算出 s 的取值. 在这个算例中, $s=32$, 而若采用文献 [73] 中的方法, 则 $s=148$. 然后计算出 f_λ 的根

$$\begin{aligned}[&-.44254407603573012077377941295814,\\ &-1.88508815207146024154755882591 63\,]\end{aligned}$$

选取绝对值 ≤ 1 的

$$\bar{\alpha}=-.44254407603573012077377941295814.$$

通过运行算法 5.1 可以得到一个向量 (0., 1., 0., 2., 0., 0.,0., 0., 0., 0., 0.). 该向量正好对应 f 的一个因子 $2y+x$. 接着用 $f/(2y+x)=1+y+x$ 更新 f, 得到 f 的不可约分解为 $(2y+x)(1+y+x)$.

表 5.1 是对一些随机生成的多项式, 将算法 5.1、文献 [73] 中的算法和 Maple 11 中 `factor` 进行比较得到的, 其中 n,m 分别表示输入多项式 f 关于 x, y 的次数, H 表示 f 的高度, s_{HL} 表示文献 [73] 算法需要的 s 的取值, s_{al} 是算法 5.1 要求的 s 的取值, t_{al} 和 t_{fa} 分别表示本文的算法所需时间和直接利用 `factor` 命令所需的时间.

从表 5.1 可知: 对表中实例, 算法 5.1 的运行时间略优或接近于 `factor` 的运行时间. 然而由定理 5.10 知, 算法 5.1 的运行时间由 s 和 $\|f\|$ 共同决定, 而 s

表 5.1: 算法 5.1 的性能

i	(n,m)	H	s_{HL}	s_{al}	s_l	$t_{al}(s)$	$t_{fa}(s)$
1	(2,2)	3	148	32	10	0.	0.
2	(3,2)	9	507	76	16	0.031	0.125
3	(4,4)	3	8000	1024	30	0.047	0.032
4	(4,5)	6	10000	2000	38	0.030	0.047
5	(2,2)	79247	4100	1024	71	0.125	0.187
6	(3,2)	782	500	72	63	0.014	0.
7	(4,4)	25382	8000	1024	108	0.047	0.063
8	(3,6)	3542	31104	1944	77	0.046	0.032

具有式 (5.23) 中的大小. 所以当 n, m 或 $\|f\|$ 增加时, s 也会随之增加, 这在很大程度上影响着算法 5.1 的效率. 通过更多的实例研究发现, 对算法 5.1 可以采用更小的 s. 在表 5.1 中, s_l 就是表示在算法 5.1 正确运行的前提下, s 可取的最小值. 实际上, 在上表中 t_{al} 的数据都是用 s_l 得到的. 此时算法的效率会相应地提高. 对任意的输入, 如果能够找出在算法 5.1 正确运行的前提下, 最小的 s 的取值, 则算法 5.1 的复杂度将进一步降低.

尽管如此, 该算法对于次数很高的有理二元多项式, 其运行效率远不如 Maple 中的 `factor` 命令. 一方面是由于构造格的时候, 已带入了很多关于多项式本身的信息, 所以复杂度必不会很低. 另一方面, 当次数很高的时候, 由于 s 选取得也很大, 从而在恢复极小多项式的时候所需要的精度也就越高. 这也降低了算法的效率. 然而, 算法 5.1 仍不失为一个基于符号-数值混合计算的有理二元多项式因式分解的解决方案, 尤其是对规模不大, 次数不高的情形具有不错的效果.

另外, 尽管本节中讨论的极小多项式的恢复技术采用的是基于 LLL 算法的重构算法 (第 4.1 节), 但是基于 SIRD 算法的代数数近似重构方案 (第 4.2 节) 仅需作微小调整便可应用于算法 5.1 中. 若如此, 其复杂度结果并不会改变, 因此不再赘述.

下一节中将给出一个非常高效的因式分解算法, 其基本思想是由待分解多项式 f 的部分信息出发, 通过推广的 Hensel 提升, 使得 f 的各个因子逐项显现.

5.2　基于 Newton 多胞体的稀疏分解算法

对于一个满足假设 *(H)* (见第 5.2.1 节) 的有理二元多项式 $f(x,y)$, 本节将给出一个高效的因式分解算法.

5.2.1　假设 (H)

在众多的二元多项式因式分解方法中, 最流行的技术就是 "提升和重组". 欲分解 $f(x,y)\in\mathbb{Z}[x,y]$, 首先在 $\mathbb{Q}$ 上分解 $f(x,0)$, 然后在一个幂级数代数 (power series algebra) 上提升 $f(x,0)$ 的因子, 最后通过因子重组得到 $f(x,y)$ 在 $\mathbb{Q}$ 上的因式分解. 其中的提升技术是经典的 *Hensel* 提升 (见 [60, Chap. 6]), 并称 $f(x,0)$ 的因子为经典的 Hensel 提升的初始因子 (initial factors).

本节将要给出一个推广的 *Hensel* 提升 (generalized Hensel lifting) 技术, 其初始因子所起的作用与经典的 Hensel 提升类似, 但其定义不同于经典的 Hensel 提升. 为了解释它们之间的不同之处, 需要引进如下一些概念.

对于 $V\subseteq\mathbb{R}^n$, 定义其凸包 (convex hull) 为如下集合

$$\mathrm{conv}(V)=\left\{\sum_{i=1}^{k}\lambda_i\boldsymbol{v}_i:\boldsymbol{v}_i\in V,\ \lambda_i\in\mathbb{R}_{\geq 0},\ 且\ \sum_{i=1}^{k}\lambda_i=1\right\},$$

其中 $\mathbb{R}_{\geq 0}$ 表示非负实数集合. 若 V 是有限集, 则称 $\mathrm{conv}(V)$ 为凸多面体 (convex polytope). 称凸多面体中的一个点为顶点 (vertex), 若其不在多面体中任意两点形成的线段之间. 凸多面体的边 (edge) 是两个顶点所形成的线段. 易知, 凸多面体总是其所有顶点所形成的集合的凸多面体. 对于二元多项式 $f=\sum f_{i,j}x^iy^j$, 其 *Newton* 多胞体 (Newton polytope) 定义为所有满足 $f_{ij}\neq 0$ 的点 (i,j) 所形成的集合的凸多面体, 记作 N_f.

定义 5.1. 定义 f 的 *Newton* 线 (Newton line) 为 N_f 的一条边, 其中一个顶点为 N_f 最右下方的一个顶点, 另一个顶点使得 N_f 中没有在 Newton 线之下的顶点. 称 f 中那些在 Newton 线上的项所形成的多项式为 *Newton* 多项式 (Newton polynomial), 记作 $f^{(0)}(x,y)$.

由此定义决定的 Newton 线和 Newton 多项式均由多项式 f 唯一确定. 令 $\hat{\delta}$ 和 $\hat{d}>0$ 为两个整数使得 $\gcd(\hat{\delta},\hat{d})=1$ 且 $\hat{\delta}/\hat{d}$ 为 f 的 Newton 线的斜率. 若斜率为 0, 则令 $\hat{\delta}=0$, $\hat{d}=1$.

定义 5.2. 对整数 $k \geq 0$ 和整数 $n > 0$, 定义

$$I_k(n) := \left\{ x^j y^{-(n-j)\hat{\delta}/\hat{d}} \cdot y^{k/\hat{d}} : j = 0, \cdots, n \right\}.$$

定义 5.3. 设 f 和 g 为 $\mathbb{Z}[x, y]$ 中的两个多项式. 对整数 $k \geq 0$ 和整数 $n > 0$, 定义 $f - g \equiv 0 \mod I_k(n)$ (或记为$f \equiv g \mod I_k(n)$) 使得 $f - g$ 中 x^j 的稀疏关于 y 的最低次数不低于 $(k - \hat{\delta}(n-j))/\hat{d}$, 其中 $j = 0, \cdots, n$.

通常, $f \equiv g \mod I$ 通常表示$f - g$ 属于理想 I, 而此处的记号 mod 不同于通常意义.

定义 5.4. 设 f 为一个整系数二元多项式, $G_1, \cdots, G_r$ 为其在 $\mathbb{Q}$ 上的所有的不可约因子, d_i 为 G_i 关于 x 的次数. 对 $i = 1, \cdots, r$, 定义 f 的*初始因子* (initial factors) 为使得如下条件成立的 $G_i^{(0)}$:

$$\begin{cases} G_i^{(0)} \equiv 0 \mod I_0(d_i), \\ G_i^{(0)} \equiv G_i \mod I_1(d_i). \end{cases}$$

例 5.2. 设 $f = x^8 - 3\,x^4y^2 + 5\,x^4y^5 - 4\,y^4 + 5\,y^7 + 2\,y^3x^4 - 8\,y^5 + 10\,y^8$. 在 $\mathbb{Q}$ 上, f 有两个不可约因子 $G_1 = x^4 + y^2 + 2\,y^3$ 和 $G_2 = x^4 - 4\,y^2 + 5\,y^5$.

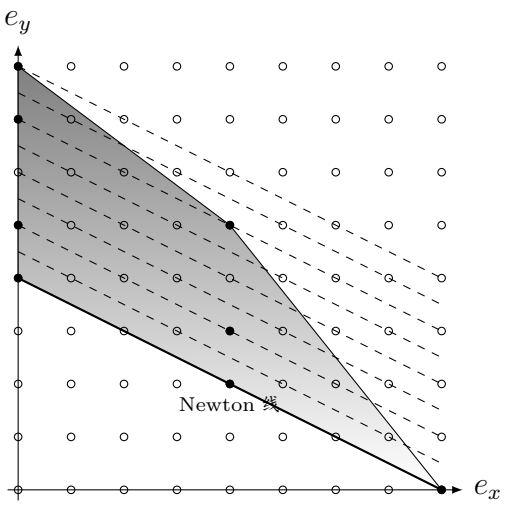

图 5.1: 多项式 f 的 Newton 多胞体

图 5.1 描述了 f 的 Newton 多胞体, 其中 e_x 和 e_y 分别表示单项式 $x^{e_x}y^{e_y}$ 关于 x 和 y 的次数. 因此, f 的 Newton 多项式为

$$f^{(0)}(x, y) = x^8 - 3\,x^4y^2 - 4\,y^4.$$

该多项式的每一项都对应于其 Newton 线上的一个整点 (见图 5.1). 对此例, $\hat{\delta}=-1$, $\hat{d}=2$. 因此,

$$I_0(4)=\left\{y^2, xy^{\frac{3}{2}}, x^2y, x^3y^{\frac{1}{2}}, x^4\right\}.$$

如图 5.2 所示, 该集合对应了 G_2 的 Newton 线上 $e_x=0,1\ldots,\deg_x(G_2)$ 的点. 同时,

$$I_1(4)=\left\{y^{\frac{5}{2}}, xy^2, x^2y^{\frac{3}{2}}, x^3y, x^4y^{\frac{1}{2}}\right\}.$$

于是, 由定义 5.4 知 $G_2^{(0)}=x^4-4\,y^2$ 是 f 的一个初始因子. 值得注意的是, $G_2^{(0)}$ 在 $\mathbb{Q}$ 上可约.

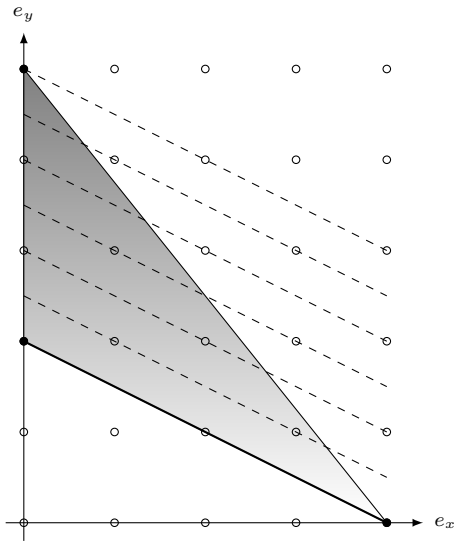

图 5.2: 多项式 G_2 的 Newton 多胞体

在本章余下的部分, 设 $f(x,y)\in\mathbb{Z}[x,y]$ 和其如定义 5.4中所述的初始因子满足如下假设:

$$\textbf{(H)}\begin{cases}(a) & f \text{ 无平方且无单项式因子;}\\ (b) & f \text{ 关于 } x \text{ 的首项系数为 1;}\\ (c) & f \text{ 的所有的初始因子的最大公因子为 1.}\end{cases}$$

实际上, (Ha) 和 (Hb) 的限制并不是必须的, 此处只是为了描述的方便: 若 f 有单项式因子, 则能够通过计算 $f/\gcd(f,\frac{\partial f}{\partial x})$ 或 $f/\gcd(f,\frac{\partial f}{\partial y})$ 消除; 首项系数

问题也能够由文献 [86, 144, 145] 中的技术来解决. 但是, (Hc) 是必须的. 同时, (Hc) 规定了本节所讨论方法的适用范围.

计算模型. 如文献 [30, 98, 101] 所采用的计算模型一样, 此处也采用计算树模型 (computational tree model) (见 [32, Chap. 4]). 粗略地说, 这意味着基域上的所有算术操作 $(+, -, \times, \div)$ 和比较都能在单位时间内完成. 用 $\mathsf{M}(n)$ 表示两个次数不超过 n 的多项式乘法的复杂度. 如文献 [58, Chap. 8] 中所述, 假设 M 满足 "超加法性质": 对正整数 m 和 n, $\mathsf{M}(m+n) \geq \mathsf{M}(m) + \mathsf{M}(n)$. 常数 ω 表示线性代数指数, 即两个 $n \times n$ 矩阵乘法能够在 $\mathcal{O}(n^\omega)$ 次算术操作内完成 $(2 < \omega \leq 3)$, 同时, $n \times n$ 矩阵求逆也具有此复杂度 (见 [126, Chap. 2]). 另外, 记号 $\tilde{\mathcal{O}}$ 表示忽略了其中的对数因子. 如 [58, Chap. 25], 若存在两个正整数 c 和 N 使得 $f(n) \leq g(n)(\log(3 + g(n)))^c$ 对所有的 $n \geq N$ 成立, 则记为 $f \in \tilde{\mathcal{O}}(g)$.

算法概述. 在假设 (H) 成立的情况下, 考虑如下的分解算法:

1. 在 $\mathbb{Z}[x, \hat{y}]$ 上分解 Newton 多项式 $f^{(0)}(x, y)$, 记为 $f^{(0)} = g_1 \cdots g_s$, 其中 $\hat{y} = y^{-\hat{\delta}/\hat{d}}$.
2. 通过组合 $g_1, \cdots, g_s$ 得到初始因子 $G_1^{(0)}, \cdots, G_r^{(0)}$.
3. 提升 f 的初始因子得到在 $\mathbb{Z}$ 上的不可约因子.

显然, 这个算法不同于传统的提升重组算法, 因为这里的算法是先重组再提升. 这个算法的第一个阶段本质上是分解单变量多项式 (见命题 5.18). 因此, 将着重考察后面的两个步骤.

首先, 算法所采用的提升为推广的 Hensel 提升. 本文的一些概念是受到 Sasaki 等提出*扩展的 Hensel 构造* (extended Hensel construction) 的启发 [74, 75, 78, 79, 132–134, 136] 而得到的. 此处推广的 Hensel 提升与扩展的 Hensel 构造的主要区别体现在如下两个方面. 第一,在定义 $I_k(n)$ 时引入了参数 n, 这可以看作是扩展的 Hensel 构造中相应定义的推广, 但是这一推广纠正了文献 [134] 中的式 (3.11), 而这在提升过程中起着至关重要的作用. 另外, 此处的算法重新定义了 f 的初始因子. 这样做的好处是提升开始于初始因子, 从而保证了在提升过程中所有的提升因子均在 $\mathbb{Z}[x, y]$ 内. 而扩展的 Hensel 构造中, 只能保证提升因子在某幂级数代数中. 另外, 在提升过程中, 此处的算法保持了稀疏性, 即在提升过程中, 因子的乘积不会产生不在 f 中的项, 从而克服了传统的 Hensel 提升可能遇到的中间过程膨胀问题.

为了得到正确的初始因子, 将采用一种基于数值线性代数的新颖的方法.

其基本思想是首先采用如文献 [24, 30, 68, 96, 101] 的对数方法, 将因子重组问题转化为线性代数问题. 因为 $f^{(0)} = G_1^{(0)} \cdots G_r^{(0)}$ (见引理 5.14), 所以必存在 $\boldsymbol{\mu}_i = (\mu_{j,i}) \in \{0,1\}^s$ 使得 $G_i^{(0)} = \prod_{j=1}^s g_j^{\mu_{j,i}}$ 对 $i = 1, \cdots, r$ 成立. 在两边取对数后得到

$$\operatorname{Ln} G_i^{(0)} = \sum_{j=1}^{s} \mu_{j,i} \operatorname{Ln} g_j.$$

其中, 对 $z \in \mathbb{C}$, $\operatorname{Ln} z$ 表示复的对数函数. 与传统方法不同的是, 此处将通过赋值来建立方程. 因此 $\boldsymbol{\mu}_i$ 便能够通过数值计算得到.

5.2.2　推广的 Hensel 提升

设多项式 $f \in \mathbb{Z}[x,y]$ 满足假设 (H) 且 $G_1, \cdots, G_r$ 为其在 $\mathbb{Q}$ 上的不可约因子. 本节将假设初始因子 $G_1^{(0)}, \cdots, G_r^{(0)}$ 已经得到. 在这些假设下, 将给出推广的 Hensel 提升算法. 该提升能够保持稀疏性, 进一步地, 提升过程中的所有结果都在 $\mathbb{Z}[x,y]$ 中.

Moses-Yun 多项式. *Moses-Yun* 多项式在提升过程中将起着至关重要的作用, 其概念由如下引理给出.

引理 5.11. *设 $h_i(x,y)\,(i = 1, \cdots, r)$ 为关于 x 和 y 的齐次多项式, $r \geq 2$, $\deg_x(h_i) = d_i \geq 1$, 并且满足*

$$\gcd(h_i, h_j) = 1 \quad i \neq j. \tag{5.24}$$

则对每一个 $l \in \{0, \cdots, d_x - 1\}$, 存在唯一的多项式集合 $\left\{W_i^{(l)}(x,y)\right\}$ 使得

$$\begin{aligned} &W_1^{(l)}[h_1 \cdots h_r/h_1] + \cdots + W_r^{(l)}[h_1 \cdots h_r/h_r] = x^l y^{d_x - l}, \\ &\deg_x(W_i^{(l)}) < \deg_x(h_i), \quad i = 1, \cdots, r, \end{aligned} \tag{5.25}$$

其中 $d_x = \sum_{i=1}^r d_i$. 进一步地, $W_i^{(l)}$ 为总次数为 d_i 的关于 x 和 y 的齐次多项式.

称 $W_i^{(l)}$ $(i = 1, \cdots, r,\ l = 0, \cdots, d_x - 1)$ 为 *Moses-Yun (插值)* 多项式. 其意义在于可以将 Newton 线上的项由初始因子表出. 该引理及其证明可以在文献 [134] 中找到. 另外, 该引理的单变元情形可在文献 [136, 137] 中找到. 由该引理的证明过程, 可以得到如下计算 Moses-Yun 多项式的算法.

算法 5.2 (Moses-Yun). 输入: $h_i(x,y)\,(i=1,\cdots,r)$ 满足引理 *5.11* 中的条件. 输出: *Moses-Yun* 多项式 $W_i^{(l)}$.

1. 对 i 从 1 到 r, 令 $G:=h_i(x,1)$, $H:=\prod_{j=1}^{r}h_j(x,1)/G$. 对 V 和 W 调用扩展的 *Euclid* 算法使得 $VG+WH=1$, $\deg(V)<\deg(H)$ 且 $\deg(W)<\deg(G)$; 对 l 从 1 到 d_x-1, 令 $W_i^{(l)}(x,1):=\operatorname{rem}\left(xW_i^{(l-1)},G,x\right)$, 其中 $W_i^{(0)}(x,1)=W$.
2. 对每一个 $W_i^{(l)}(x,1)$ 齐次化使得 $W_i^{(l)}(x,y)$ 为关于 x 和 y 的总次数为 d_i 的齐次多项式. 输出 $W_i^{(l)}(x,y)$.

若 $G,H\in\mathbb{Z}[x]$, 则由 Sylvester 结式的性质 ([41, §3.5, Proposition 9]) 知步骤 1 中的 W 满足

$$\operatorname{height}(W)\le\max\{\operatorname{height}(G),\operatorname{height}(H)\}^{\deg G+\deg H-1}.$$

因此,

$$\operatorname{height}(W_i^{(l)})\le\max\{\operatorname{height}(G),\operatorname{height}(H)\}^{\deg G+\deg H}. \tag{5.26}$$

命题 5.12. *算法 5.2 能够在 $\mathcal{O}\left(r\log d_x\,\mathsf{M}(d_x)+d_x^2\right)$ 次算术操作内正确地计算 Moses-Yun 多项式.*

证明. 算法 5.2 的正确性可由引理 5.11 及其证明保证. 步骤 1 中, 为了计算 H, 首先计算 $\prod h_i(x,1)$. 计算 r 个次数之和为 d_x 的单变元多项式乘积最多花费 $\mathcal{O}(\mathsf{M}(d_x)\log r)$ 次算术操作 [58, Chap. 10]. 之后对 i 从 1 到 r 计算 H 至多需要 $r\mathsf{M}(d_x)$ 次算术操作. 由 [58, Chap. 11], 扩展的 Euclid 算法需要 $\mathcal{O}\left(\mathsf{M}(d_x)\log d_x\right)$ 次操作. 此处需要调用扩展的 Euclid 算法 r 次. 故总共需要 $\mathcal{O}\left(r\mathsf{M}(d_x)\log d_x\right)$ 次操作. 进一步地, 通过带余除法计算所有的 $W_i^{(l)}(x,1)$ 需要 $\mathcal{O}(r\mathsf{M}(d_x))$ 次操作. 由引理 5.11, $W_i^{(l)}(x,y)$ 是一个总次数为 d_i 的关于 x 和 y 的齐次多项式. 因此, 对每一个 l, 至多花费 $d_1+\cdots+d_r=d_x$ 次操作来齐次化 $W_i^{(l)}(x,y)$, 因此, 步骤 2 最多需要 $\mathcal{O}(d_x^2)$ 次操作. 综上, 算法5.2 至多需要 $\mathcal{O}\left(r\log d_x\,\mathsf{M}(d_x)+d_x^2\right)$ 次算术操作. □

推广的 Hensel 提升. 回顾如定义 5.3 中的 mod 操作以及

$$I_k(n):=\left\{x^jy^{-(n-j)\hat{\delta}/\hat{d}}\cdot y^{k/\hat{d}}:j=0,\cdots,n\right\}, \tag{5.27}$$

其中 $\hat{\delta}$ 和 $\hat{d}>0$ 是两个整数使得 $\gcd(\hat{\delta},\hat{d})=1$ 且 $\hat{\delta}/\hat{d}$ 为 f 的 Newton 线的斜率; 若斜率为 0, 则 $\hat{\delta}=0$, $\hat{d}=1$. 令 $\hat{y}=y^{-\hat{\delta}/\hat{d}}$. 则集合 $I_k(n)$ 中的每个元素都可以看成是 $y^{k/\hat{d}}$ 和一个 n 次齐次项 $x^j\hat{y}^{n-j}$ 的乘积. 若 $n=d_x$, 则将 $I_k(d_x)$ 简记为 I_k, 其中 d_x 是 f 关于 x 的次数. 由定义 5.2 知 f 的每一个单项都包含在某一个 I_k 中. 比如, $f^{(0)}(x,y)$ 的每一个单项都在 I_0 中, 即

$$f^{(0)}\equiv 0 \mod I_0 \quad \text{and} \quad f^{(0)}\equiv f \mod I_1. \tag{5.28}$$

从几何的角度来讲, 每一个 I_k 对应于一条平行于 Newton 线的直线, 而 $f\equiv g \mod I_k$ 意味着 $f-g$ 的所有的单项都在对应于 I_{k-1} 的直线之上(如例 5.2 所示).

引理 5.13. *设 $\hat{G}_i\,(i=1,\cdots,r)$ 为关于 x 和 $\hat{y}$ 的整系数多项式, $r\geq 2$ 且 $\deg_x(\hat{G}_i)=d_i\geq 1$. 对固定的整数 $2\leq m\leq r$, 若对 $i=1,\cdots,m$, $\hat{G}_i$ 的每一个单项在集合 $I_k(d_i)$ 中且 $\hat{G}_j\equiv G_j^{(0)} \mod I_1(d_j)$ 对 $j=m+1,\cdots,r$ 成立, 则*

$$\prod_{i=1}^{m}\hat{G}_i\prod_{j=m+1}^{r}\hat{G}_j\equiv 0 \mod I_{k+1}\left(d_x\right), \tag{5.29}$$

其中 $d_x=\deg_x(\hat{G}_1\cdots\hat{G}_r)$, $k\geq 1$.

证明. 记式 (5.29) 中的乘积为 $P=P_1\cdot P_2$. 现考查 P 关于 x^j 的系数关于 y 的最低次数, 记为 $\operatorname{ldeg}(\operatorname{coeff}(P,x^j),y)$. 则

$$\begin{aligned}
\operatorname{ldeg}\left(\operatorname{coeff}\left(P,x^j\right),y\right)&=\operatorname{ldeg}\left(\sum_{\ell_1+\ell_2=j}\operatorname{coeff}\left(P_1,x^{\ell_1}\right)\cdot\operatorname{coeff}\left(P_2,x^{\ell_2}\right),y\right)\\
&\geq\frac{(d_1+\cdots+d_m-\ell_1)\cdot(-\hat{\delta})}{\hat{d}}+\frac{mk}{\hat{d}}+\frac{(d_{m+1}+\cdots+d_r-\ell_2)\cdot(-\hat{\delta})}{\hat{d}}\\
&\geq\frac{(2k-\hat{\delta}(d_x-j))}{\hat{d}}\geq\frac{(k+1-\hat{\delta}(d_x-j))}{\hat{d}},
\end{aligned}$$

其中第一个不等号是因为 $\hat{G}_i$ 的每一个单项在集合 $I_k(d_i)$ 中且 $\hat{G}_j\equiv G_j^{(0)} \mod I_1(d_j)$ ($j=m+1,\cdots,r$), 第二个不等号是因为 $m\geq 2$, 最后一个不等号是因为 $k\geq 1$. 于是由定义 5.3 知结论成立. □

该引理在提升过程中扮演着重要的角色. 尽管引理 5.13 的证明对 $k=0$ 不成立, 但此时仍有下面的引理成立.

引理 5.14. 设 f 为 $\mathbb{Z}[x,y]$ 中的二元多项式, $f^{(0)}$ 为其 *Newton* 多项式, G_i 为其在 $\mathbb{Q}$ 上的不可约因子, $G_i^{(0)}$ 为其初始因子 $(i=1,\cdots,r)$. 则

$$f^{(0)} = G_1^{(0)} \cdots G_r^{(0)}.$$

证明. 由定义 5.4 知

$$\begin{aligned} G_i^{(0)} \equiv & 0 \mod I_0(d_i), \\ G_i^{(0)} \equiv & G_i \mod I_1(d_i). \end{aligned}$$

于是, 由定义 5.3 容易验证

$$\begin{aligned} G_1^{(0)} \cdots G_r^{(0)} \equiv & 0 \mod I_0, \\ G_1^{(0)} \cdots G_r^{(0)} \equiv & G_1 \cdots G_r \mod I_1. \end{aligned}$$

再由式 (5.28) 知结论成立. 证毕. □

下面将给出推广的 Hensel 提升, 其中以 $I_k\,(k=1,2,\cdots)$ 为模. 几何上来看, 每一次推广的 Hensel 提升都将所有的因子沿着 e_y 的正方向提升 $y^{1/\hat{d}}$ (如图 5.2 所示).

定理 5.15 (推广的 Hensel 提升)**.** 设 $f \in \mathbb{Z}[x,y]$ 满足假设 *(H)*, $G_1,\cdots,G_r$ 为其有理不可约因子. 令 $f^{(0)}$ 为 f 的 *Newton* 多项式, $G_1^{(0)},\cdots,G_r^{(0)}$ $(r\geq 2)$ 为其初始因子, $\deg_x G_i^{(0)} = d_i \geq 1$. 并令 $I_k(n)\,(k=0,1,2,\cdots)$ 如式 *(5.27)*. 则对任意的非负整数 k, 存在唯一的 $\Delta G_i^{(k)} \in \mathbb{Z}[x,y]$ 使得其每个单项都在 $I_k(d_i)$ 中, 并且

$$G_i \equiv G_i^{(0)} + \sum_{j=0}^{k} \Delta G_i^{(j)} \mod I_{k+1}(d_i), \tag{5.30}$$

$$f \equiv \prod_{i=1}^{r} \left(G_i^{(0)} + \sum_{j=0}^{k} \Delta G_i^{(j)} \right) \mod I_{k+1}.$$

证明. 对 k 使用数学归纳法. 当 $k=0$ 时, 对 $i=1,\cdots,r$, 令 $\Delta G_i^{(0)} = 0$. 由引理 5.14 知该定理成立. 假设定理对 $k-1\,(k\geq 1)$ 为真. 令

$$G_i^{(k-1)} = G_i^{(0)} + \sum_{j=0}^{k-1} \Delta G_i^{(j)}.$$

由归纳假设,

$$f(x,y) \equiv \prod_{i=1}^{r} G_i^{(k-1)}(x,y) \mod I_k.$$

令

$$\Delta f^{(k)}(x,y) = f(x,y) - \prod_{i=1}^{r} G_i^{(k-1)}(x,y) \mod I_{k+1}. \tag{5.31}$$

则 $\Delta f^{(k)}(x,y) \in \mathbb{Z}[x,y]$ 可以表示为

$$\begin{aligned} &\Delta f^{(k)}(x,y) = c_{d_x-1}^{(k)} \cdot x^{d_x-1}\hat{y} + \cdots + c_0^{(k)} \cdot \hat{y}^{d_x}, \\ &c_l^{(k)} = a_l^{(k)} y^{k/\hat{d}}, \quad a_l^{(k)} \in \mathbb{Z}, \quad l = 0, \cdots, d_x - 1. \end{aligned} \tag{5.32}$$

由引理 5.11, 对 $G_i^{(0)}$ 计算 $W_i^{(l)}$, 其中 $i = 1, \cdots, r$, $l = 0, \cdots, d_x - 1$. 构造 $\Delta G_i^{(k)}$ 为

$$\Delta G_i^{(k)}(x,y) = \sum_{l=0}^{d_x-1} W_i^{(l)}(x,\hat{y}) c_l^{(k)}. \tag{5.33}$$

易知其每一个单项都在集合 $I_k(d_i)$ 中. 令 $G_i^{(k)}(x,y) = G_i^{(k-1)}(x,y) + \Delta G_i^{(k)}(x,y)$. 则

$$\begin{aligned} f - \prod_{i=1}^{r} G_i^{(k)} &= f - \prod_{i=1}^{r} \left(G_i^{(k-1)} + \Delta G_i^{(k)} \right) \mod I_{k+1} \\ &\equiv f - \left(\prod_{i=1}^{r} G_i^{(k-1)} + \sum_{i=1}^{r} \frac{\Delta G_i^{(k)}}{G_i^{(0)}} \prod_{j'=1}^{r} G_{j'}^{(0)} \right. \\ &\qquad \left. + \sum_{i=1}^{r} \sum_{j=1}^{r} \frac{\Delta G_i^{(k)} \Delta G_j^{(k)}}{G_i^{(0)} G_j^{(0)}} \prod_{j'=1}^{r} G_{j'}^{(0)} + \cdots \right) \mod I_{k+1}. \end{aligned}$$

因为 $\Delta G_i^{(k)}(x,y)$ 的每一个单项在 $I_k(d_i)$ 中, 由引理 5.13 知

$$\sum_{i=1}^{r} \sum_{j=1}^{r} \frac{\Delta G_i^{(k)} \Delta G_j^{(k)}}{G_i^{(0)} G_j^{(0)}} \prod_{j'=1}^{r} G_{j'}^{(0)} + \cdots \equiv 0 \mod I_{k+1}.$$

因此,

$$f - \prod_{i=1}^{r} G_i^{(k)} \equiv \Delta f^{(k)} - \sum_{i=1}^{r} \frac{\Delta G_i^{(k)}}{G_i^{(0)}} \prod_{j'=1}^{r} G_{j'}^{(0)} \mod I_{k+1},$$

然后将 $\Delta G_i^{(k)}$ 替换为式 (5.33) 中的表达式得

$$\begin{aligned} f-\prod_{i=1}^{r} G_i^{(k)} &\equiv \Delta f^{(k)} - \sum_{i=1}^{r} \frac{\sum_{l=0}^{d_x-1} c_l^{(k)} W_i^{(l)}}{G_i^{(0)}} \prod_{j'=1}^{r} G_{j'}^{(0)} \mod I_{k+1} \\ &\equiv \Delta f^{(k)} - \sum_{l=0}^{d_x-1} c_l^{(k)} \sum_{i=1}^{r} W_i^{(l)} \frac{\prod_{j'=1}^{r} G_{j'}^{(0)}}{G_i^{(0)}} \mod I_{k+1} \\ &\equiv \Delta f^{(k)} - \sum_{l=0}^{d_x-1} c_l^{(k)} x^l \hat{y}^{d_x-l} \mod I_{k+1}, \end{aligned}$$

其中最后一个等号可由引理 5.11 中 Moses-Yun 多项式的性质得到. 由式 (5.32) 知

$$f(x,y) \equiv \prod_{i=1}^{r} G_i^{(k)}(x,y) \mod I_{k+1}.$$

而 $\Delta G_i^{(k)}$ 之所以唯一是因为 Moses-Yun 多项式唯一. 同时, $\Delta G_i^{(k)}$ 的唯一性也蕴含着式 (5.30) 成立, 因此 $\Delta G_i^{(k)} \in \mathbb{Z}[x,y]$. 证毕. □

定理 5.15 的基本思想与 Sasaki 等的扩展的 Hensel 构造 [134, Theorem 1] 的思想有相似之处. 它们的区别主要体现为: 第一, f 的初始因子被重新定义为 $G_1^{(0)}, \cdots, G_r^{(0)}$. 在 [134] 中, 初始因子为 $f^{(0)}$ 在 $\mathbb{Z}[x,\hat{y}]$ 中的不可约因子. 第二, [134, 式 (3.11)] 断言 $G_i^{(k)} \equiv G_i^{(0)} \mod I_1$, 这是不正确的. 由定理 5.15, 仅能够得到 $G_i^{(k)} \equiv G_i^{(0)} \mod I_1(d_i)$ 对 $i = 1, \cdots, r$ 成立. 第三, 定理 5.15 意味着 $G_i^{(k)} = G_i^{(0)} + \Delta G_i^{(1)} + \cdots + \Delta G_i^{(k)}$ 总是在 $\mathbb{Z}[x,y]$ 中的, 然而在 Sasaki 等的工作中, $G_i^{(k)}$ 属于 $\mathbb{C}\{y^{1/\hat{d}}\}[x]$. 根本原因在于初始因子的定义不同. 综合以上这些分析, 本节所给出的推广的 Hensel 提升技术既可以看成是对 Sasaki 等的工作的改进, 也可以看成是对他们工作的改正.

同时, 由式 (5.30) 知 $G_i \equiv G_i^{(k)} \mod I_{k+1}(d_i)$, 即 $G_i^{(k)}$ 总是 G_i 的一部分, 并且随着 k 的增加, $G_i^{(k)}$ 趋向于 G_i. 这一性质保证了该提升技术不会改变原多项式的稀疏性, 即提升过程中产生的各个因子中不会出现不在真因子中的项, 从而他们的乘积也不会产生 f 中没有的项. 于是, 随着提升次数的增加, $\Delta f^{(k)}$ 会越来越小, 最后收敛到 0, 提升终止. 这一性质有效地避免了经典的 Hensel 提升和扩展的 Hensel 构造中可能出现的中间过程膨胀问题.

由上述定理的证明可知, Moses-Yun 多项式的唯一性是非常重要的一环, 因为它保证着提升的唯一性. 而 Moses-Yun 多项式的唯一性是由假设 (Hc) 保证的. 也就是说, 假设 (Hc) 所起的作用就是保证推广的 Hensel 提升的唯一性.

由定理 5.15 可以证明如下推论. 该推论将被应用于第 5.2.3 节.

推论 5.16. *设 $f \in \mathbb{Q}[x,y]$ 满足假设 (H), $G_1,\cdots,G_r$ 为其全部的有理不可约因子. 则 G_i 的非零项数不超过 $(T+1)(d_i+1)$, 其中 T 为 f 的非零项数, d_i 为 G_i 关于 x 的次数.*

证明. 由推广的 Hensel 提升知存在 k 使得 $G_i = G_i^{(0)} + \Delta G_i^{(1)} + \cdots + \Delta G_i^{(k)}$. 这里有 $k \le T$, 因为若按 f 的 Newton 线的平行线族沿 e_y 的正方向将 f 和 G_i 分层, 则 G_i 的 Newton 多胞体 N_{G_i} 的层数不会超过 N_f 的层数, 而 N_f 的层数不会超过 f 的非零项数 T. 又由于 $G_i^{(0)}$ 是关于 x 和 y 的总次数为 d_i 的齐次多项式, 于是其非零项数不超过 d_i+1. 由式 (5.33) 知每一个 $\Delta G_i^{(j)}$ 至多有 d_i+1 个非零项. 证毕. □

基于定理 5.15, 可得如下的提升算法, 该算法将每一个因子提升 $y^{1/\hat{d}}$.

算法 5.3 (Lifting). *输入: 一个满足假设 (H) 的二元多项式 $f \in \mathbb{Z}[x,y]$; f 的初始因子的 Moses-Yun 多项式 $W_i^{(l)}(x,\hat{y})$ $(i=1,\cdots,r,\ l=0,\cdots,d_x-1)$, $G_1^{(k-1)},\cdots,G_r^{(k-1)}$ 使得 $f \equiv G_1^{(k-1)}\cdots G_r^{(k-1)} \mod I_k$.*

输出: $G_1^{(k)},\cdots,G_r^{(k)}$ 使得 $f \equiv G_1^{(k)}\cdots G_r^{(k)} \mod I_{k+1}$.

1. 计算 $\Delta f^{(k)} := f - G_1^{(k-1)}\cdots G_r^{(k-1)} \mod I_{k+1}$.

2. 将 $\Delta f^{(k)}$ 表示为

$$\begin{aligned}\Delta f^{(k)}(x,y) &= c_{d_x-1}^{(k)}\cdot x^{d_x-1}\hat{y} + \cdots + c_0^{(k)}\cdot \hat{y}^{d_x},\\ c_l^{(k)} &= a_l^{(k)} y^{k/\hat{d}},\quad a_l^{(k)} \in \mathbb{Z},\quad l=0,\cdots,d_x-1.\end{aligned}$$

对 i 从 1 到 r, 令

$$\Delta G_i^{(k)}(x,y) := \sum_{l=0}^{d_x-1} W_i^{(l)} c_l^{(k)},$$

$G_i^{(k)} := G_i^{(k-1)} + \Delta G_i^{(k)}$. 输出 $G_i^{(k)}$.

命题 5.17. *算法 5.3 是正确的, 并能在 $\mathcal{O}(\mathsf{M}(d_x d_y) + r d_x)$ 次算术操作内终止.*

证明. 算法的正确性是由定理 5.15 保证的. 若采用 Kronecker 替换, $\mathbb{Z}[x,y]$ 中的两个关于 x 的次数不超过 n, 关于 y 的次数不超过 d 的多项式乘法能够在 $\mathcal{O}(\mathsf{M}(nd))$ 次操作内完成. 由于 $G_1^{(k-1)}\cdots G_r^{(k-1)}$ 关于 x 和 y 的次数分别不超过 d_x 和 d_y, 所以步骤 1 能够在 $\mathcal{O}(\mathsf{M}(d_xd_y))$ 次操作内完成 [58, Chap. 10]. 再由 $W_i^{(l)}$ 是关于 x 和 $\hat{y}$ 的总次数为 d_i 的齐次多项式知其最多有 d_i+1 个非零项. 进一步地, $c_l^{(k)}$ 为单项式, 所以计算所有的 $G_i^{(k)}$ 能够在 $\mathcal{O}(rd_x)$ 次操作内完成. 综上, 算法 5.3 至多需要 $\mathcal{O}(\mathsf{M}(d_xd_y)+rd_x)$ 次算法操作. □

截至目前, 已有了一个将初始因子提升为有理不可约的有效的提升算法, 并且该算法具有保持稀疏性的特性. 下一节中将讨论如何获得正确的初始因子.

5.2.3 重组初始因子

回顾 $\hat{\delta}$ 和 $\hat{d}>0$ 是两个整数使得 $\gcd(\hat{\delta},\hat{d})=1$ 且 $\hat{\delta}/\hat{d}$ 为 f 的 Newton 线的斜率; 若斜率为 0, 则 $\hat{\delta}=0$, $\hat{d}=1$. 若 $\hat{\delta}=0$, 则 $f^{(0)}$ 为关于 x 的单变元多项式 (由于假设 (Ha), f 没有单变元因子, 包括 y 的方幂), 否则 $f^{(0)}$ 是一个关于 x 和 $\hat{y}=y^{-\hat{\delta}/\hat{d}}$ 的齐次多项式. 因此, 分解 Newton 多项式相当于分解单变元多项式 $f^{(0)}(x,1)$, 记为

$$f^{(0)}(x,y)=f^{(0)}(x,\hat{y})=g_1(x,\hat{y})\cdots g_s(x,\hat{y}),$$

其中 $g_i(x,\hat{y})$ ($i=1,\cdots,s$) 是 $\mathbb{Z}[x,\hat{y}]$ 中的元素. 于是以下结论成立:

命题 5.18. *在 $\mathbb{Z}[x,\hat{y}]$ 中分解 Newton 多项式 $f^{(0)}(x,y)$ 等价于在 $\mathbb{Q}$ 上对一个次数为 d_x 的单变元多项式进行有理不可约分解.*

但是, $g_1,\cdots,g_s$ 并不总是初始因子.

例 5.3. 对于例 5.2 中的多项式, $\hat{y}=y^{1/2}$ 且

$$f^{(0)}(x,\hat{y})=(x^4+\hat{y}^4)(x^2+\hat{y}^2)(x^2-2\hat{y}^2)\triangleq g_1g_2g_3,$$

但其初始因子为 $G_1^{(0)}=x^4+y^2=g_1$, $G_2^{(0)}=x^4-4y^2=g_2g_3$.

此时, $f^{(0)}(x,y)$ 的因子个数比 $f(x,y)$ 的因子个数要多, 即 $r\le s$, 因子组合便不可避免.

重组算法. 为了叙述上的方便, 仅考虑 f 的 Newton 线的斜率为 -1 的情形, 即 $\hat{\delta}=-1$, $\hat{d}=1$ 且 $\hat{y}=y$. 其余情形仅需做微小调整. 此时, Newton 多项式有以下分解:

$$f^{(0)}(x,y)=g_1(x,y)\cdots g_s(x,y). \tag{5.34}$$

现在, 来描述重组初始因子的基本思想. 由引理 5.14, 所有的初始因子的乘积即是 Newton 多项式. 因此, 对每一个 $i=1,\cdots,r$, 必存在唯一的向量 $\boldsymbol{\mu}_i=(\mu_{j,i})\in\{0,1\}^s$ 使得 $G_i^{(0)}=\prod_{j=1}^s g_j^{\mu_{j,i}}$. 两端取自然对数得到

$$\mathrm{Ln}\, G_i^{(0)}=\sum_{j=1}^{s}\mu_{j,i}\mathrm{Ln}\, g_j. \tag{5.35}$$

因此, 若能够计算出式 (5.35) 两边在点 (x_i,y_i) 的值, 则能构建一个线性系统, 该系统的解正是那些 0-1 向量 $\boldsymbol{\mu}_i$. 但是, 事先并不知道 $G_i^{(0)}$. 幸运的是, 能够近似地计算它在某点的值.

假设 f 有 r 个有理不可约因子 $G_1,\cdots,G_r$. 由定理 5.15 知存在整数 k_i 使得 $G_i=G_i^{(0)}+\sum_{k=1}^{k_i}\Delta G_i^{(k)}$ 对 $i=1,\cdots,r$ 成立, 其中 $\Delta G_i^{(k)}$ 的每一个单项都在 $I_k(d_i)$ 中, $d_i=\deg_x G_i^{(0)}$. 由 Newton 线的定义和假设 (Ha), 有 $\deg_x G_i=d_i$. 令 $w=y/x$. 则 G_i/x^{d_i} 可以表示为

$$\frac{G_i}{x^{d_i}}=\frac{G_i^{(0)}}{x^{d_i}}+\sum_{k=1}^{k_i}\frac{\Delta G_i^{(k)}}{x^{d_i}}=G_i^{(0)}(w)+g_i^{(1)}(w)\cdot y+\cdots+g_i^{(k_i)}(w)\cdot y^{k_i}, \tag{5.36}$$

其中 $g_i^{(k)}\in\mathbb{Z}[w]$ 满足 $x^{d_i}y^k\cdot g_i^{(k)}=\Delta G_i^{(k)}$ $(k\geq 1)$. 因此, 若选择 (x,y) 使得 $w\sim\mathcal{O}(1)$ 且 $y\sim o(1)$, 则

$$\left|\frac{G_i}{x^{d_i}}-\frac{G_i^{(0)}}{x^{d_i}}\right|\sim o(1), \tag{5.37}$$

其中 $y\sim o(1)$ 表示 y 是一个充分小的量.

算法 5.4 (Combine). *输入: 一个满足假设 (H) 的二元多项式 $f\in\mathbb{Z}[x,y]$; $f^{(0)}$ 的因子 $g_1,\cdots,g_s$; 任意的 $M>0$.*

输出: f 的初始因子 $G_1^{(0)},\cdots,G_r^{(0)}$.

1. *令 $z=y/x$. 则 $g_j(x,y)/x^{\deg_x g_j}=g_j(1,z)$. 随机地选择 s 个绝对值 $\leq M$ 的数 $z_1,\cdots,z_s\in\mathbb{C}$, 令 $A:=(a_{i,j})_{s\times s}$, 其中 $a_{i,j}:=Re\,(Ln\,(g_j(1,z_i)))$; 并令*

$\tilde{A}=(\tilde{a}_{i,j})=A+\delta A$, 其中 $\tilde{a}_{i,j}\in\mathbb{Q}$ 且

$$\|\delta A\|_\infty \le \frac{1}{10\|A^{-1}\|_\infty}. \tag{5.38}$$

计算 $\tilde{A}^{-1}$ 和 $\tilde{A}^{-1}$ 每一行元素绝对值之和的最大值, 记为 $\|\tilde{A}^{-1}\|_\infty$.

2. 令 T 为 f 的非零项数. 令 $\delta_i=\prod_{j=1}^{s}\min\{|a_{i,j}|,1\}$. 令 y_0 为一个随机的有理数满足

$$|y_0|<\min\left\{\min_{1\le i\le s}\{1/\delta_i\}\left(2\|A^{-1}\|_\infty(T+1)(d_x+1)M^{d_x}N\right)^{-1},1\right\}, \tag{5.39}$$

其中 $N=2^{d_x^2}(d_x+1)^{d_x/2}\|f\|_\infty^{d_x+1}$. 调用一个单变元的因式分解算法在 $\mathbb{Q}[x]$ 中分解 $f(x,y_0)$, 记其因子为 $\tilde{G}_1(x),\cdots,\tilde{G}_r(x)$. 若 $r=s$ 则 $g_1,\cdots,g_s$. 对 $i=1,\cdots,s$, 令 $x_i:=y_0/z_i$, $B:=(b_{i,j})_{s\times r}$, 其中 $b_{i,j}:=\mathrm{Re}\left(\mathrm{Ln}\left(\tilde{G}_j(x_i)/x_i^{d_j}\right)\right)$.

3. 计算 $U:=A^{-1}B=(u_{i,j})_{s\times r}$, 令 $\mu_{i,j}:=\lfloor u_{i,j}+0.5\rfloor$, $G_i^{(0)}:=\prod_{j=1}^{s}g_j^{\mu_{j,i}}$. 输出 $G_1^{(0)},\cdots,G_r^{(0)}$.

重组分析. 下面将给出算法 5.4 的分析.

引理 5.19. *算法 5.4 中步骤 1 中的矩阵 A 以概率 1 可逆.*

证明. 对 $j=1,\cdots,s$, 记 $g_j(z)=g_j(1,z)$. 由复分析 (如 [12]) 知 $\mathrm{Ln}z=\ln|z|+I\cdot\mathrm{Arg}\,z$, 其中 $I=\sqrt{-1}$, $\mathrm{Arg}\,z$ 为 $z\in\mathbb{C}$ 的辐角. 于是在步骤 1 中, $a_{i,j}=\ln|g_j(z_i)|$. 假设矩阵

$$A=\begin{pmatrix}\ln|g_1(z_1)| & \ln|g_2(z_1)| & \cdots & \ln|g_s(z_1)|\\ \ln|g_1(z_2)| & \ln|g_2(z_2)| & \cdots & \ln|g_s(z_2)|\\ \cdots & \cdots & \cdots & \cdots\\ \ln|g_1(z_s)| & \ln|g_2(z_s)| & \cdots & \ln|g_s(z_s)|\end{pmatrix}$$

的秩为 $k<s$. 不妨设其 k 阶主子式非零. 现考虑关于 $\lambda_j\in\mathbb{R}$ 的如下系统:

$$\lambda_1\ln|g_1(z_i)|+\cdots+\lambda_{s-1}\ln|g_{s-1}(z_i)|=\ln|g_s(z_i)|, \tag{5.40}$$

其中 $i=1,\cdots,s-1$, $j=1,\cdots,s-1$. 由线性代数 (如 [143]) 知式 (5.40) 至少有一个解. 令 $\lambda_1,\cdots,\lambda_{s-1}$ 为该系统的任意一组解. 则对 $z_1,\cdots,z_{s-1}$, 有

$$\left|g_1(z_i)^{\lambda_1}\cdots g_{s-1}(z_i)^{\lambda_{s-1}}\right|=|g_s(z_i)|.$$

对 $z = x + I\,y \in \mathbb{C}$, $x, y \in \mathbb{R}$, 令 $C(z) = |c(z)| - |g_s(z)|$, 其中 $c(z) = g_1(z)^{\lambda_1} \cdots g_{s-1}(z)^{\lambda_{s-1}}$. 令 $z_s = x_s + I\,y_s$ 为骤 1 中选择的第 s 个复数. 因为 A 的秩 $< s$, 所以有 $C(z_s) = 0$.

但是, $C(z)$ 可以被看成是关于 z 的实部和虚部的二元实函数, 记为 $C(x, y)$. 可以断言 $C(x, y)$ 不是零函数. 事实上, 若 $z_0 = x_0 + I\,y_0$ 为 $c(z)$ 的一个零点, 则 $g_s(z_0)$ 非零, 因为所有的初始因子的最大公因子为 1, 从而 $C(x_0, y_0) \neq 0$. 又因为 z_s 的选取独立于 $z_1, \cdots, z_{s-1}$, 并且 x_s 的选取独立于 y_s, 所以对随机产生的 $x_s \in \mathbb{R}$, $C(x_s, y)$ 也不会是零函数. 因此, 对随机的 $y_s \in \mathbb{R}$, $C(x_s, y_s) \neq 0$ 成立的概率为 1. 这与 $C(z_s) = 0$ 矛盾. □

从如上分析可知, 正数 M 是为了保证矩阵 A 可逆的概率为 1, 因此 $M > 0$ 可选为任意的正数.

命题 5.20. *若随机数 y_0 使得 $f(x, y_0) = \tilde{G}_1(x) \cdots \tilde{G}_r(x)$, 则算法 5.4 能够正确地重组初始因子, 其中 $G_i(x, y_0) = \tilde{G}_i(x)$, $i = 1, \cdots, r$.*

证明. 由引理 5.19, 矩阵 A 可逆的概率为 1, 因此 $\|A^{-1}\|_\infty$ 是有限数. 由本命题的条件知 $G_i(x, y_0) = \tilde{G}_i(x)$, $i = 1, \cdots, r$.

而另一方面, 必须对误差进行控制, 以使得通过舍入线性系统 $AU = B$ 的数值解得到正确的 $\mu_{i,j}$.

下面以 $G_1^{(0)}$ 为例说明误差控制方法. 记 $A\boldsymbol{u} = \boldsymbol{b}$ 为精确的情形, $\tilde{A} \cdot (u + \delta u) = b + \delta b$ 为带误差的情形, 其中 $\tilde{A} = A + \delta A$, $\boldsymbol{u}$ 和 $\boldsymbol{b}$ 为分别为矩阵 U 和 B 的第一列, δA, $\delta\boldsymbol{u}$ 和 $\delta\boldsymbol{b}$ 分别为扰动误差.

式 (5.38) 意味着 $\|A^{-1} \cdot \delta A\| < 1$, 因此由 [63, Theorem 2.3.4] 可知 $\tilde{A}$ 也可逆, 于是 $\|\tilde{A}^{-1}\|_\infty$ 也是一个有限数.

首先分析

$$\begin{aligned}\|\delta\boldsymbol{b}\|_\infty &= \max_{1\le j\le s}\left\{\left|\mathrm{Re}\left(\mathrm{Ln}\left(G_1(x_j, y_0)/x_j^{d_1}\right)\right) - \mathrm{Re}\left(\mathrm{Ln}\left(G_1^{(0)}(x_j, y_0)/x_j^{d_1}\right)\right)\right|\right\}\\ &= \max_{1\le j\le s}\left\{\left|\ln\left|G_1(x_j, y_0)/x_j^{d_1}\right| - \ln\left|G_1^{(0)}(x_j, y_0)/x_j^{d_1}\right|\right|\right\}.\end{aligned}$$

由式 (5.36),

$$\frac{G_1}{x^{d_1}} = \frac{G_1^{(0)}}{x^{d_1}} + \sum_{k=1}^{k_1}\frac{\Delta G_1^{(k)}}{x^{d_1}} = G_1^{(0)}(w) + g_1^{(1)}(w)\cdot y + \cdots + g_1^{(k_1)}(w)\cdot y^{k_1}.$$

记

$$G_1(x_j, y_0)/x_j^{d_1} = G_1^{(0)}(z_j) + \bar{G}_1(z_j, y_0) \cdot y_0,$$

其中 $\bar{G}_1(z_j, y_0) = \sum_{k=1}^{k_1} g_1^{(k)}(z_j) \cdot y_0^{k-1}$. 因此,

$$\begin{aligned}
\|\delta \boldsymbol{b}\|_\infty &= \max_{1\le j\le s}\left\{\left|\ln\left|1+\frac{\bar{G}_1(z_j,y_0)\cdot y_0}{G_1^{(0)}(z_j)}\right|\right|\right\} \le \max_{1\le j\le s}\left\{\ln\left(1+\left|\frac{\bar{G}_1(z_j,y_0)\cdot y_0}{G_1^{(0)}(z_j)}\right|\right)\right\} \\
&\le \max_{1\le j\le s}\left\{\left|\frac{\bar{G}_1(z_j,y_0)\cdot y_0}{G_1^{(0)}(z_j)}\right|\right\} \le \max_{1\le j\le s}\left\{\left|\bar{G}_1(z_j,y_0)\cdot y_0\right|\cdot \min_{1\le i\le s}\{\delta_i\}\right\} \\
&\le (T+1)\cdot(d_x+1)\cdot M^{d_x}\cdot N\cdot \delta_1\cdot |y_0|.
\end{aligned}$$

其中, 第一个不等号成立是因为对 $z \in \mathbb{C}$, 有 $|1+z| \le 1+|z|$; 第二个不等号成立是因为 $\ln(1+x) \le x$ 对 $x > -1$ 成立; 第三个不等号是因为 $|G_1^{(0)}(z_j)| \ge \delta_1 \ge \min_{1\le i\le s}\{\delta_i\}$ 和 $G_1^{(0)} = \prod_{j=1}^{s} g_j^{\mu_{j,1}}$. 另外, 因为 $|y_0| < 1$ 和 $|z_j| \le M$, 所以对 $\bar{G}_1(z_j, y_0)$ 的每一个单项式 $z_j^a y_0^b$, 有 $|z_j^a y_0^b| \le |z_j^{d_1}| \le M^{d_x}$. 由式 (5.26), 式 (5.33), $\|G_1\|_\infty \le N$ 和推论 5.16 知最后一个不等号成立.

因此, 若 y_0 满足式 (5.39), 则

$$\|\tilde{A}^{-1}\|_\infty \cdot \|\delta \boldsymbol{b}\|_\infty < \frac{1}{4}. \tag{5.41}$$

进一步地, 由 [63, Theorem 2.3.4] 和式 (5.38) 知

$$\begin{aligned}
\|\tilde{A}^{-1}\|_\infty \cdot \|\delta A\|_\infty &\le \left(\|A^{-1}\|_\infty + \|\delta A\|_\infty \frac{\|A^{-1}\|_\infty^2}{1-\|A^{-1}\delta A\|_\infty}\right)\cdot \|\delta A\|_\infty \\
&\le 2\left(\|A^{-1}\|_\infty \|\delta A\|_\infty\right)^2 + \left(\|A^{-1}\|_\infty \|\delta A\|_\infty\right) < \frac{1}{4}.
\end{aligned} \tag{5.42}$$

于是,

$$\|\delta \boldsymbol{u}\|_\infty = \|\tilde{A}^{-1}(\delta \boldsymbol{b} + \delta A \cdot \boldsymbol{u})\|_\infty \le \|\tilde{A}^{-1}\|_\infty (\|\delta \boldsymbol{b}\|_\infty + \|\delta A\|_\infty) < 1/2,$$

其中第一个不等号成立是因为 $\|\boldsymbol{u}\|_\infty = 1$, 第二个成立是因为式 (5.41) 和式 (5.42). 证毕. □

命题 5.21. *算法 5.4 至多需要 $\mathcal{O}(s^\omega + \mathsf{M}(d_x)\log s)$ 次操作便将重组问题约化为在有理数域上分解次数不超过 d_x 的单变元多项式的问题.*

证明. 对 $i \le s$, 首先通过快速多点赋值计算 $g_j(1, z_i)$. 这需要 $\mathcal{O}(\mathsf{M}(d_j)\log s)$ 次算术操作, 因此由 M 的 "超加法性质", 构造矩阵 A 至多需要 $\mathcal{O}(\mathsf{M}(d_1)\log s + \cdots + \mathsf{M}(d_s)\log s) \subseteq \mathcal{O}(\mathsf{M}(d_x)\log s)$ 次算术操作. 类似地, 构造矩阵 B 需要相同的操作次数. 计算矩阵 A^{-1} 和矩阵 A^{-1} 与 B 的乘积至多需要 $\mathcal{O}(s^\omega)$ 次操作. 又由假设 (Hb), f 关于 x 首一, 所以 $f(x, y_0)$ 关于 x 的次数为 d_x. 证毕. □

重组算法 5.4 是基于线性化的思想, 即把问题转化为线性代数方程组的求解. 这一思想已经广泛应用于多项式的因式分解中, 比如迹重组方法 [68, 135–137], 对数微分重组方法 [24, 30, 96, 101]. 但这些方法都是符号计算方法. 而算法 5.4 在取对数后, 通过不同点的赋值来建立线性代数系统. 尽管所建立的系统只是近似的, 但是通过上述的误差控制, 可以保证能够从该系统的近似解中获得准确的 0-1 解. 这不同于文献中已有的方法.

最后, 不得不提及上述的重组算法不是一个确定的算法, 因为其成功与否取决于 Hilbert 不可约定理有效的算法版本. 在有理数域上, 该问题仍然是一个著名的公开问题 (见 [58, pp. 469–472]). 这便意味着在理论上, 随机选取的 y_0 不一定满足 $f(x, y_0) = \tilde{G}_1(x)\cdots\tilde{G}_r(x)$, 其中 $G_i(x, y_0) = \tilde{G}_i(x)$. 但是, 在实际的计算中, 这种现象鲜有发生.

5.2.4　算法和分析

在本小节中, 将描述算法 BiFactor, 并给出一个实例来描述算法中的关键步骤, 同时分析该算法.

算法 5.5 (BiFactor)**.** 输入: 满足假设 (H) 的二元多项式 $f \in \mathbb{Z}[x, y]$.
输出: f 在 $\mathbb{Z}$ 上的不可约分解.

1. 计算 f 的*Newton* 多项式 $f^{(0)}(x, y)$ 和*Newton* 线的两个端点 $L = (l_1, l_2)$ 和 $R = (r_1, r_2)$. 设 $d_x := \deg_x f$, $d_y := \deg_y f$.
2. 若 $l_2 = r_2 = 0$ 则 $f^{(0)}$ 为单变元多项式, 否则 $f^{(0)}$ 为总次数为 $\frac{r_2 l_1 - r_1 l_2}{r_2 - l_2}$ 的关于 $x\ \hat{y}$ 的齐次多项式, 其中 $\hat{y} = y^{-\frac{l_2 - r_2}{l_1 - r_1}}$. 令 $\hat{d}$ 和 $\hat{\delta}$ 为非负整数使得 $\frac{\hat{\delta}}{\hat{d}} = \left|\frac{l_2 - r_2}{l_1 - r_1}\right|$ 为最简分数. 若 $\left|\frac{l_2 - r_2}{l_1 - r_1}\right| = 0$, 则令 $\hat{\delta} := 0$, $\hat{d} := 1$. 令 $I_k := \left\{x^j \hat{y}^{d_x - j} \cdot y^{k/\hat{d}} : j = 0, \cdots, d_x\right\}$. 调用单变元因式分解算法在 $\mathbb{Z}[x, \hat{y}]$ 中分解 $f^{(0)}(x, 1)$ 得到
$$f^{(0)}(x, y) = g_1(x, \hat{y})\cdots g_s(x, \hat{y}).$$

3. 以 $g_1,\cdots,g_s$ 和 f 为输入调用算法 *5.4*. 记返回的因子为 $G_1(x,y),\cdots,G_r(x,y)$. 令 $\Delta f := f - f^{(0)}$. 若 $\Delta f = 0$, 则输出 $G_1,\cdots,G_r$.

4. 以上一步得到的初始因子为输入, 调用算法 *5.2* 计算 *Moses-Yun* 多项式 $W_i^{(l)}(x,\hat{y})$.

5. 设 k 为使得 $\Delta f \equiv 0 \mod I_k$ 成立的最大值, 用 I_{k+1} 作为推广的 *Hensel* 提升的模.

6. 计算 $\Delta f^{(k)} := \Delta f \mod I_{k+1}$. 对 i 从 1 到 r, 令

$$\Delta G_i(x,y) := \sum_{l=0}^{d_x-1} W_i^{(l)} c_l^{(k)},$$

更新 $G_i := G_i + \Delta G_i$.

7. 令 $\Delta f := f - G_1\cdots G_r$. 若 $\Delta f = 0$, 则输出 $G_1,\cdots,G_r$, 否则执行步骤 *5.*

现用一个例子来说明算法 5.5 的主要步骤.

例 5.4. 考虑例 5.2 中的多项式: $f = x^8 - 3\,x^4y^2 + 5\,x^4y^5 - 4\,y^4 + 5\,y^7 + 2\,y^3x^4 - 8\,y^5 + 10\,y^8$. 则 f 的 Newton 多项式为

$$f^{(0)}(x,y) = x^8 - 3\,x^4y^2 - 4\,y^4 = (x^2+\hat{y}^2)(x^2+2\,\hat{y}^2)(x^2-2\,\hat{y}^2) \triangleq g_1g_2g_3,$$

其中 $\hat{\delta} = -1$, $\hat{d} = 2$, 因此 $\hat{y} = y^{\frac{1}{2}}$. 所以

$$I_0 = \left\{y^4, xy^{\frac{7}{2}}, x^2y^3, x^3y^{\frac{5}{2}}, x^4y^2, x^5y^{\frac{3}{2}}, x^6y, x^7y^{\frac{1}{2}}, x^8\right\}.$$

在因子组合阶段, 首先令 $z_1 = \frac{2}{5}e^{\frac{\pi}{11}I}$, $z_2 = \frac{7}{5}e^{\frac{27\pi}{55}I}$, $z_3 = e^{\frac{49\pi}{55}I}$, $y_0 = 1/223$, 然后便能够建立如下的线性系统:

$$\begin{pmatrix} \frac{200466}{6353} & \frac{355429}{21663} & \frac{120385}{7398} \\ \frac{88198}{3365} & \frac{47978}{4353} & \frac{185008}{12743} \\ \frac{120647}{4175} & \frac{79463}{5347} & \frac{100672}{7121} \end{pmatrix} \begin{pmatrix} \mu_{1,1} & \mu_{1,2} \\ \mu_{2,1} & \mu_{2,2} \\ \mu_{3,1} & \mu_{3,2} \end{pmatrix} = \begin{pmatrix} \frac{272848}{8645} & \frac{117484}{3595} \\ \frac{68722}{2621} & \frac{296190}{11597} \\ \frac{615298}{21291} & \frac{60868}{2099} \end{pmatrix}.$$

通过解所得到的线性方程组并进行取整运算得到两个 0-1 向量 $(1,0,0)$ 和 $(0,1,1)$, 这便意味着初始因子为 $G_1^{(0)} = g_1 = x^4 + y^2$, $G_2^{(0)} = g_2g_3 = x^4 - 4\,y^2$. 此时, $\Delta f = 5\,x^4y^5 + 5\,y^7 + 2\,y^3x^4 - 8\,y^5 + 10\,y^8$. 因为仅有 x^0 和 x^4 在 Δf 中出现, 所以仅需对 $l = 0, 4$ 计算 $W_i^{(l)}$:

$$W_1^{(0)} = \tfrac{1}{5}\,y^2, \quad W_2^{(0)} = -\tfrac{1}{5}\,y^2,$$
$$W_1^{(4)} = \tfrac{4}{5}\,y^2, \quad W_2^{(4)} = \tfrac{1}{5}\,y^2.$$

由于 $\Delta f = x^4y^2\cdot(5y^{6/\hat{d}}) + y^4\cdot(5y^{6/\hat{d}}) + x^4y^2\cdot(2y^{2/\hat{d}}) + y^4\cdot(-8y^{2/\hat{d}}) + y^4\cdot(10y^{8/\hat{d}})$, 所以 $k=2$ 为满足 $\Delta f \equiv 0 \mod I_k$ 的最大正数. 所以, 计算 $\Delta f^{(2)} \equiv \Delta f \mod I_3 \equiv x^4y^2\cdot(2y^{2/\hat{d}}) + y^4\cdot(-8y^{2/\hat{d}})$, 于是 $c_7^{(2)} = c_6^{(2)} = c_5^{(2)} = 0$, $c_4^{(2)} = 2y$, $c_3^{(2)} = c_2^{(2)} = c_1^{(2)} = 0$, $c_0^{(14)} = -8y$. 因此,

$$G_1 := G_1 + \Delta G_1 = G_1 + \left(W_1^{(0)}c_0^{(2)} + W_1^{(4)}c_4^{(2)}\right) = x^4 + 2\,x^2y + y^2 + 2\,y^3,$$
$$G_2 := G_2 + \Delta G_2 = G_2 + \left(W_2^{(0)}c_0^{(2)} + W_2^{(4)}c_4^{(2)}\right) = x^4 - 4\,y^2.$$

更新 $\Delta f = 5\,x^4y^5 + 10\,x^2y^6 + 5\,y^7 + 10\,y^8 \neq 0$. 再次执行步骤 5. 再进行一次提升, 得到 f 在 $\mathbb{Z}[x,y]$ 的不可约因子为: $G_1 = x^4+2\,x^2y+y^2+2\,y^3$, $G_2 = x^4-4\,y^2+5\,y^5$.

式 (5.39) 中选取的 y_0 仅仅是使得 $\|\delta\boldsymbol{u}\| < \frac{1}{2}$ 的充分条件, 而不是必要的. 因此在算法实施的时候, 可以先从较小规模的 y_0 进行尝试. 如有必要, 再朝着式 (5.39) 的方向选择 y_0. 比如在上例中, $y_0 = 1/223$ 便可以了.

另外, 此例直观地描述了 BiFactor 算法是如何利用稀疏性的. 一方面, 对于一般的稀疏多项式, 关于变量 x 从 0 到 d_x 的所有次数不一定都出现. 因此, 不必计算所有的 $W_i^{(l)}$, 仅需计算 Δf 中出现的那些 x^l 所对应的 $W_i^{(l)}$. 若输入的多项式的非零项数 $T < d_x$, 则步骤 4 至多需要 $\mathcal{O}\left(r\log d_x\,\mathsf{M}(d_x) + Td_x\right)$ 次算法操作, 而不需要命题 5.12 给出的 $\mathcal{O}\left(r\log d_x\,\mathsf{M}(d_x) + d_x^2\right)$ 次操作. 另一方面, 当输入稠密多项式时, 每次提升仅能提升 $y^{1/\hat{d}}$. 然而, 对一般的稀疏多项式, 对某些正数 k, I_k 中可能不含有 f 中的项. 对于这些 k, 有 $\Delta f^{(k)} = 0$, 这便意味着 $\Delta G_i^{(k)} = 0$ 对 $i = 1,\cdots,r$ 成立. 因此, 可以直接处理使得 $\Delta f^{(k)} \neq 0$ 的最小的 k, 即步骤 5 中选取的 k.

定理 5.22. *给定 $\mathbb{Q}$ 上满足假设 (H) 的二元多项式 f, 算法 5.5 将计算 f 在 $\mathbb{Q}$ 上的不可约因子的计算任务转化为次数为 d_x 的 $\mathbb{Q}$ 上的单变元多项式因式分解, 加上 $\mathcal{O}(Trd_x + T\log r\mathsf{M}(d_xd_y) + d_x^2 + s^\omega + r\log d_x\mathsf{M}(d_x))$ 或 $\tilde{\mathcal{O}}(Trd_x + Td_xd_y + d_x^2 + s^\omega)$ 次 $\mathbb{Q}$ 中的算术操作, 其中 T 是 f 的非零项数, d_x 和 d_y 分别是 f 关于 x 和 y 的次数, r 为 f 的不可约有理因子的个数, s 是 f 的 Newton 多项式在 $\mathbb{Z}[x,\hat{y}]$ 中的因子个数.*

证明. 假设 (Ha) 意味着 $d_i = \deg_x G_i^{(0)} \geq 1$. 则算法 5.5 的正确性可由命题 5.17, 命题 5.18 和命题 5.20 得到. 下面, 分析算法的复杂度.

由 Newton 多项式的定义知, 确定 f 的 Newton 线至多需要 $\mathcal{O}(T)$ 次操作. 由引理 5.18 和命题 5.21 知, 步骤 2 和步骤 3 至多需要 $\mathcal{O}(s^\omega + \mathsf{M}(d_x)\log s)$ 次操作加上分解一个次数不超过 d_x 的单变元多项式. 由命题 5.12, 计算 Moses-Yun 多项式至多消耗 $\mathcal{O}\left(r\log d_x\,\mathsf{M}(d_x) + d_x^2\right)$ 次操作, 其中 r 为 f 在 $\mathbb{Q}$ 上的不可约因子个数. 由命题 5.17 知, 步骤 6 计算 ΔG_i 至多需要 $\mathcal{O}(rd_x)$ 次操作. 由命题 5.17 知, 步骤 7 至多需要 $\mathcal{O}(\mathsf{M}(d_xd_y))$ 次操作. 进一步地, 提升的次数不会超过 T, 因为每次提升 Δf 的非零项数至少降低一项. 最后, 步骤 5 至多需要 $\mathcal{O}(T)$ 次操作.

因此, 算法 5.5 的复杂度为次数不超过 d_x 的一元多项式分解的复杂度加上 $\mathcal{O}(Trd_x + T\mathsf{M}(d_xd_y) + d_x^2 + s^\omega + r\log d_x\mathsf{M}(d_x))$ 或 $\tilde{\mathcal{O}}(Trd_x + Td_xd_y + d_x^2 + s^\omega)$ 次算术操作. □

5.2.5 数值试验

本文作者在 Maple 中实现了 BiFactor 算法, 本节将介绍一些算法实现的细节, 并给出一些实验结果.

在诸多的应用中, 均可能遇到中间过程膨胀的问题, 比如计算多项式的最大公因子问题. 在实现 BiFactor 算法的时候, 为了彻底避免中间过程的膨胀, 因此在计算 Mose-Yun 多项式时, 采用了模方法, 在有限域中进行计算.

由前面的分析可以知道, BiFactor 算法将问题转化为单变元多项式的因式分解问题. 对于稀疏情形, 存在一些单变元因式分解的稀疏算法, 如 [43, 103, 104]. 在实现 BiFactor 算法时, 单变元的因式分解采用 Maple 自带的 `factor` 命令.

另外, 对于部分不满足假设 (Hc) 的多项式 f, 可以采用如下的仿射变换对多项式 f 进行预处理:

$$\begin{pmatrix} x \\ y \end{pmatrix} := \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix},$$

其中 $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ 是由 $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$ 和 $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ 生成的群中的元素. 变换之后, 若 $F(X,Y)$ 满足假设 (Hc), 则计算 $F(X,Y) = X^mY^nf(X,Y)$ 的分解, 其中 m 和 n 是使得 $F(X,Y)$ 满足假设 (Ha, Hb) 的两个整数. 则 $f(x,y)$ 的因式分解

能够从 $F(X,Y)$ 的因子得到. 尽管这样的预处理能够扩大 BiFactor 算法的适用范围, 但是该算法仍然不完备. 因为, 存在多项式, 如 $f=(xy+x^3+x^2y^5+y^4)(xy+x^4+x^2y^6+y^3)$, 在上述所有的仿射变换下均不满足假设 (Hc).

以下的实验数据都是在 AMD Athlon™ 7750 处理器 (2.70 GHz) 2GB 内存的个人电脑上完成的. 在下面的表格中, $time_{Maple}$ 表示 Maple 的命令 `factor` 分解所用的时间, $time_{BiFactor}$ 表示 BiFactor 算法所用的时间. 时间单位为秒. 所有的单变元多项式因式分解都是直接调用 Maple 自带的命令 `factor`.

表 5.2: 当因子组合发生时算法 BiFactor 的运行时间

no.	d_x	d_y	T	s	$time_{Maple}$	$time_{BiFactor}$
1	16	27	56	5	0.064	0.113
2	32	55	88	6	0.266	0.160
3	64	113	135	7	3.273	0.417
4	128	230	194	8	36.071	0.663
5	256	478	253	9	685.706	2.433

在表 5.2 中, 考查算法 BiFactor 在因子组合发生时 ($s>r=2$) 的效率. 每一个测试的多项式都只有两个有理的不可约因子, 其 Newton 多项式 $x^{d_x}-y^{d_x}$ 能够在 $\mathbb{Z}[x,\hat{y}]$ 中分解为 s 个因子. 因此, 不得不进行因子组合. 此处, Newton 多项式的因式分解很容易计算, 但是, 在因子组合的时候, 分解次数为 d_x 的一元多项式不可避免. 尽管如此, 但是从表 5.2 中的数据可知算法 BiFactor 在这种情形下具有很高的效率.

表 5.3 的目的是针对随机产生的多项式, 比较 Maple 自带的 `factor` 命令和 BiFactor 算法的效率. 表 5.3 中测试的多项式的总次数为 d, 并且是由两个总次数为 $d/2$ 的随机多项式的乘积, 而每一个都由如下 Maple 代码生成:

```
[> a := rand(0..d/2)();

[> rand(1..100)()*x^a*y^(d/2-a)+rand(1..100)() +randpoly([x, y],terms = t, degree = d/2) ;
```

前面的两项是为了保证输入多项式的总次数为 d, 且没有单项式因子. 前 5 个多项式是由 `t` := 10 生成的, 即每个因子有 $10\sim12$ 个非零项. 后面 5 个多项式的其中一个因子由 `t` := 4 得到, 而另一个由 `t` := 150 得到. 从表中的数据可知, 该算法比 Maple 中自带的 *factor* 命令要效率高得多.

表 5.3: Maple 自带的 `factor` 与 BiFactor 算法的比较

no.	d	d_x	d_y	T	$time_{Maple}$	$time_{BiFactor}$
1	50	46	29	135	0.281	0.157
2	100	95	89	143	2.125	0.109
3	200	156	152	144	7.578	0.235
4	400	335	305	144	87.297	0.218
5	800	580	595	144	943.094	0.422
6	50	39	50	705	0.532	0.437
7	100	100	65	827	4.282	1.468
8	200	193	179	890	50.75	2.156
9	400	306	302	746	208.515	2.110
10	800	784	613	912	10301.844	4.578

最后, 值得一提的是, 对于如下总次数为 20000 的多项式

$$\begin{aligned} f =& x^{4120} + x^{4118}y^2 + x^{3708}y^{400} + x^{3706}y^{402} + x^{2781}y^{1300} + x^{2779}y^{1302} + x^{1339}y^{2700} \\ &+ x^{927}y^{3100} + y^{4000} + x^{7172}y^{4167} + x^{8349}y^{4432} + x^{8347}y^{4434} + x^{6760}y^{4567} \\ &+ x^{5833}y^{5467} + x^{5568}y^{7132} + x^{11401}y^{8599}, \end{aligned}$$

文献 [15] 中在 Intel Pentium IV 的处理器上 (3.0 GHz, 1 G 内存) 运行了 1055 分 4 秒得到了该多项式的不可约分解. 但如果采用 BiFactor 算法来计算, 则仅需不到 2 秒的时间. 而且, [15] 中的实验结果是针对有限域上的分解, 而算法 BiFactor 直接计算出多项式 f 在有理数域 $\mathbb{Q}$ 上的不可约分解.

5.3 本章小结

针对有理系数二元多项式因式分解问题, 本章给出了两个符号-数值混合计算的算法. 首先利用代数数极小多项式的恢复算法给出了一个有理数域上二元多项式的分解算法. 该算法将 [73] 中算法的复杂度降低了一个 $\log^{4+\epsilon}(mn)$ 因子, 并且实验数据显示, 对于小规模的问题, 该算法仍然有较好的效果. 针对规模较大的二元有理系数多项式, 基于 Newton 多胞体, 给出了一个非常高效的分解算法. 该算法无论是在理论的复杂度方面, 还是在实际的效率方面都有非常优异的表现.

第六章　总结与展望

本文以符号-数值混合计算中的一个热点问题——“采用近似计算获得准确值” ——为中心, 深入地研究了欧几里得格和整数关系探测算法之间的关系, 揭示了整数关系探测问题与欧几里得空间中有限加法子群的分解问题之间的对偶关系; 由此得到同步整数关系探测算法 SIRD, 给出了代数数极小多项式近似重构的一个完整的新方案; 同时, 基于代数数极小多项式的近似重构, 研究了符号计算的另一个热点问题——多项式的因式分解; 最后, 基于 Newton 多胞体, 给出了针对二元有理系数多项式因式分解的高效算法.

6.1　总结

本文的主要贡献有如下几个方面.

第一, 揭示了欧几里得格和线性空间求交的 Intersect 问题 (定义 2.3) 和有限生成加法子群分解的 Decomp 问题 (定义 2.2) 之间的对偶关系 (定理 2.5). 以此为基础, 给出了 HJLS-PSLQ 算法 (二十世纪十大著名算法之一) 的一个新的观察角度, 并由此提出第一个针对有限生成加法子群分解问题的算法 (算法 2.2).

第二, 优化了现有文献中对 LLL 算法的分析方法. 在讨论运用 LLL 格约化算法解决有限生成加法子群的分解问题的方法时, 针对整数情形, 通过将 LLL-交换分类, 得到了一个全新的结果 (定理 2.16).

第三, 提出一个可以探测多个实数向量的同步整数关系的算法 SIRD (算法 3.1), 并且在计算机代数系统 Maple 中有效地实现了该算法, 为进一步地应用该算法打下了基础.

第四, 提出解决代数数极小多项式近似重构问题的一个新的完整的解决方案 (算法 4.1). 无论是在理论上, 还是在实际计算中, 该方案都显示了在数值精度和算法效率上的优越性. 从而将张景中和冯勇提出的 “采用近似计算获得准确值” [150] 的设想进一步拓展到代数数域.

第五, 提出了两个计算有理数域上的二元多项式因式分解的符号-数值混合计算算法. 算法 5.1 基于代数数域上代数数极小多项式的近似重构, 该方法继承

了 [73] 中算法的优势, 并改进了相应算法的复杂度, 针对小规模的问题有较好的效果. 理论分析和数值试验均说明, 算法 5.5 是一个基于 Newton 多胞体的高效算法, 尤其是针对稀疏的有理系数二元多项式, 效果甚佳.

6.2 正在开展的工作及未来展望

尽管本文在欧几里得格、整数关系探测、代数数极小多项式的近似重构和多项式的因式分解方面均取得了一些进展, 但也存在一些不够完善的地方, 以待在将来的工作中深入地研究.

整数关系探测的浮点算法及分析. 整数关系探测算法 HJLS-PSLQ 和同步整数关系探测算法 SIRD 理论上的计算模型均假设精确的实数计算, 这与现实计算机中的计算模型 (如 IEEE 浮点运算标准) 不一致. 因此, 如何在浮点算术模型下得到这些算法的有效版本将是一个非常有意义的课题. 这方面的工作正在进行之中. 可以预见, 这些算法的浮点算术版本将显著地改善这些算法在诸多应用中的效率.

Decomp_LLL 整数情形分析的一般化. 本文对 Decomp_LLL 算法整数情形分析是独立于本文初衷的一个有趣结果. 如何将这一分析方法应用到更为一般的情形中去也是正在进行的工作之一.

改进的 HJLS-PSLQ 算法. 如第二章中的分析, LLL 算法和 HJLS-PSLQ 算法的最大区别之一就是在约化过程中所采取的交换策略不一样. HJLS-PSLQ 算法采用的全局的交换策略有效地避免了局部交换策略可能进入死循环的可能. 近期, 本文作者所在团队发现, 针对某些特殊的应用, 这些交换策略还可以进一步地改进, 使得算法效率进一步提高.

直接求解 Intersect 问题的算法. 按照第二章中提出的观点, HJLS-PSLQ 等整数关系探测算法实际上是通过调用解决 Decomp 问题的预言机来获得 Intersect 问题的解. 找到一个直接求解 Intersect 问题的算法将是一个非常有意义的工作.

高效的符号-数值混合计算分解有理数域上的多项式. 如何得到更为高效, 适用范围更广的多项式因式分解算法不仅是符号计算中经久不衰的热门课题, 也是本文作者未来的研究方向之一.

实数的整数化表示. 在代数数极小多项式近似重构算法的支撑下, 已可以实现代数数的整数化表示 [9, 95, 127, 128]. 接下来的任务之一就是如何将超越数整

数化. 对于一些特殊情形, 已有学者展开了研究 [22, 23, 27]. 但是, 本文作者认为, 得到一个一般性的方法具有很大的难度. 事实上, 代数数之所以能够整数化表示是因为一个代数数是一个有理系数多项式方程的根, 而这个多项式本身可以用有限个整数来表示. 如果也从这个角度来尝试超越数的整数化表示, 可知信息量将会 “无穷大”.

参考文献

[1] 陈国良. *并行算法的设计与分析*. 高等教育出版社, 北京, 第三版, 2009.

[2] 陈经纬, 冯勇, 秦小林, 张景中. 代数数极小多项式的近似重构. *系统科学与数学*, 31(8):903–912, 2011.

[3] 陈经纬, 冯勇, 秦小林, 张景中. SIRD: 一个同步整数关系探测算法. *四川大学学报 (工程科学版)*, 43(6):127–132, 2011.

[4] 郭兵, 沈艳, 邵子立. 绿色计算的重定义与若干探讨. *计算机学报*, 32(12):2311–231, 2009.

[5] 国家自然科学基金委员会. *2008* 项目指南. `http://www.nsfc.gov.cn/nsfc/cen/xmzn/2008xmzn/01ms/06xx/001.htm`.

[6] 洪加威. 近似计算有效位数的增长不超过几何级数. *中国科学: 数学*, 29(3):225–233, 1986.

[7] 刘克, 单志广, 王戟, 何积丰, 张兆田, 秦玉文. "可信软件基础研究" 重大研究计划综述. *中国科学基金*, 22(3):145–151, 2008.

[8] 秦小林. *误差可控的混合计算理论及其算法研究*. 博士学位论文, 中国科学院研究生院, 北京, 2011.

[9] 秦小林, 冯勇, 陈经纬, 李骏. 采用近似方法的实代数数准确表示及其应用. *四川大学学报 (工程科学版)*, 42(2):126–131, 2010.

[10] 吴文俊. 混合计算. 李喜先编, *21* 世纪 *100* 个交叉学科难题, 656–657. 科学出版社, 北京, 2005.

[11] 支丽红. 符号和数值混合计算. *系统科学与数学*, 28(8):1040–1052, 2008.

[12] 钟玉泉. *复变函数论*. 高等教育出版社, 北京, 第三版, 2004.

[13] K. Aardal and F. Eisenbrand. The LLL algorithm and integer programming. In P. Nguyen and B. Vallée, editors, *The LLL Algorithm*, pages 293–314. Springer, Berlin, 2010.

[14] F. Abu Salem. *Factorisation Algorithms for Univariate and Bivariate Polynomials over Finite Fields*. PhD thesis, Oxford University Computing Laboratory, 2004.

[15] F. Abu Salem. An efficient sparse adaptation of the polytope method over $\mathbb{F}_p$ and a record-high binary bivariate factorisation. *Journal of Symbolic Computation*, 43(5):311–341, 2008.

[16] F. Abu Salem, S. Gao, and A. Lauder. Factoring polynomials via polytopes. In *Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, pages 4–11, Santander, Spain, 2004.

[17] M. Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract). In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 10–19, Dallas, Texas, USA, 1998.

[18] M. Avendaño, T. Krick, and M. Sombra. Factoring bivariate sparse (lacunary) polynomials. *Journal of Complexity*, 23(2):193–216, 2007.

[19] L. Babai, B. Just, and F. Meyer auf der Heide. On the limits of computations with the floor function. *Information and Computation*, 78(2):99 – 107, 1988.

[20] D. Bailey. A Fortran-90 based multiprecision system. *ACM Transactions on Mathematical Software*, 21(4):379–387, 1995.

[21] D. Bailey, J. Borwein, N. Calkin, R. Girgensohn, D. Luke, and V. Moll. *Experimental Mathematics in Action*. A K Peters, 2007.

[22] D. Bailey, P. Borwein, and S. Plouffe. On the rapid computation of various polylogarithmic constants. *Mathematics of Computation*, 66(218):903–914, 1997.

[23] D. Bailey and D. Broadhurst. Parallel integer relation detection: Techniques and applications. *Mathematics of Computation*, 70(236):1719–1736, 2000.

[24] K. Belabas, M. van Hoeij, J. Klüners, and A. Steel. Factoring polynomials over global fields. *Journal de Théorie des Nombres de Bordeaux*, 21(1):15–39, 2009.

[25] L. Bernardin. On bivariate Hensel and its parallelization. In *Proceedings of the 1998 international symposium on Symbolic and algebraic computation*, pages 96–100, New York, USA, 1998.

[26] J. Berthomieu and G. Lecerf. Reduction of bivariate polynomials from convex-dense to dense, with application to factorizations. *Mathematics of Computation*, 81(279):1799–1821, 2012.

[27] D. Borwein, J. Borwein, and R. Girgensohn. Explicit evaluation of Euler sums. *Proceedings of the Edinburgh Mathematical Society*, 38(2):277–294, 1995.

[28] J. Borwein and D. Bailey. *Mathematics by Experiment: Plausible Reasoning in the 21st Century*. A K Peters, 2004.

[29] P. Borwein. *Computational Excursions in Analysis and Number Theory*. Springer, New York, 2002.

[30] A. Bostan, G. Lecerf, B. Salvy, E. Schost, and B. Wiebelt. Complexity issues in bivariate polynomial factorization. In *Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, pages 42–49, Santander, Spain, 2004.

[31] N. Bourbaki. *Elements of Mathematics: General Topology, Part II*. Addison-Wesley, Massachusetts, 1967. A translation of *Éléments de Mathématique : Topologie Générale*, Hermann, Paris, 1966.

[32] P. Bürgisser, M. Clausen, and M. Shokrollahi. *Algebraic Complexity Theory*. Springer, 1997.

[33] A. Chattopadhyay, B. Grenet, P. Koiran, N. Portier, and Y. Strozecki. Factoring bivariate lacunary polynomials without heights. In *Proceedings of the 2013 international symposium on Symbolic and algebraic computation*, Boston, USA, 2013. To appear.

[34] J. Chen, Y. Feng, X. Qin, and J. Zhang. Exact polynomial factorization by approximate high degree algebraic numbers. In *Proceedings of the 2009 conference on Symbolic numeric computation*, pages 21–28, Kyoto, Japan, 2009.

[35] J. Chen, D. Stehlé, and G. Villard. A new view on HJLS and PSLQ: Sums and projections of lattices. In *Proceedings of the 2013 international symposium on Symbolic and algebraic computation*, Boston, USA, 2013. To appear.

[36] G. Chéze and A. Galligo. From an approximate to an exact absolute polynomial factorization. *Journal of Symbolic Computation*, 41(6):682–696, 2006.

[37] G. Chéze and G. Lecerf. Lifting and recombination techniques for absolute factorization. *Journal of Complexity*, 23(3):380–420, 2007.

[38] B. Cipra. The best of the 20th century: Editors name top 10 algorithms. *SIAM News*, 33(4):1–2, 2000.

[39] R. Corless, M. Giesbrecht, M. van Hoeij, I. Kotsireas, and S. Watt. Towards factoring bivariate approximate polynomials. In *Proceedings of the 2001 international symposium on Symbolic and algebraic computation*, pages 85–92, London, Ontario, Canada, 2001.

[40] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*, volume 85 of *Graduate Texts in Mathematics*. Springer, New York, 2005.

[41] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, New York, 3rd edition, 2007.

[42] F. Cucker, P. Koiran, and S. Smale. A polynomial time algorithm for diophantine equations in one variable. *Journal of Symbolic Computation*, 27(1):21–29, 1999.

[43] J. Davenport. Factorisation of sparse polynomials. In J. van Hulzen, editor, *Computer Algebra*, volume 162 of *Lecture Notes in Computer Science*, pages 214–224. 1983.

[44] J. Dongarra and F. Sullivan. Guest editors' introduction: The top 10 algorithms. *Computing in Science & Engineering*, 2(1):22–23, 2000.

[45] J. Dora, A. Maignan, M. Mirica-Ruse, and S. Yovine. Hybrid computation. In *Proceedings of the 2001 international symposium on Symbolic and algebraic computation*, pages 101–108, London, Ontario, Canada, 2001.

[46] P. van Emde Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical Report Report 81-04, Department of Mathematics, University of Amsterdam, April 1981.

[47] N. Fel'dman. Estimate for a linear form of logarithms of algebraic numbers. *Sbornik: Mathematics*, 5(2):291–307, 1968.

[48] Y. Feng, X. Qin, J. Zhang, and X. Yuan. Obtaining exact interpolation multivariate polynomial by approximation. *Journal of Systems Science and Complexity*, 24(4):803–815, 2011.

[49] H. Ferguson and D. Bailey. A polynomial time, numerically stable integer relation algorithm. Technical Report RNR-91-032, SRC-TR-92-066, NAS Applied Research Branch, NASA Ames Research Center, July 1992.

[50] H. Ferguson, D. Bailey, and S. Arno. Analysis of PSLQ, an integer relation finding algorithm. *Mathematics of Computation*, 68(225):351–369, 1999.

[51] H. Ferguson and R. Forcade. Generalization of the Euclidean algorithm for real numbers to all dimensions higher than two. *Bulletin of the American Mathematical Society*, 1(6):912–914, 1979.

[52] H. Ferguson and R. Forcade. Multidimensional Euclidean algorithms. *(Crelle's) Journal für die reine und angewandte Mathematik*, 334:171–181, 1982.

[53] S. Gao. Absolute irreducibility of polynomials via Newton polytopes. *Journal of Algebra*, 237(2):501–520, 2001.

[54] S. Gao. Factoring multivariate polynomials via partial differential equations. *Mathematics of Computation*, 72(242):801–822, 2003.

[55] S. Gao, E. Kaltofen, J. May, Z. Yang, and L. Zhi. Approximate factorization of multivariate polynomials via differential equations. In *Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, pages 167–174, Santander, Spain, 2004.

[56] J. von zur Gathen. Factoring sparse multivariate polynomials. In *24th Annual Symposium on Foundations of Computer Science*, pages 172–179, Tucson, USA, 1983.

[57] J. von zur Gathen. Who was who in polynomial factorization. In *Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation*, pages 1–2, Genova, Italy, 2006.

[58] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, London, 1999.

[59] J. von zur Gathen and E. Kaltofen. Factoring sparse multivariate polynomials. *Journal of Computer and System Sciences*, 31(2):265–287, 1985.

[60] K. Geddes, S. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Kluwer Academic Publishers, Boston, 1992.

[61] GNU. *The GNU Mulitiple Precision Arithmetic Library*. `http://gmplib.org`.

[62] A. Goldstein and R. Graham. A Hadamard-type bound on the coefficients of a determinant of polynomials. *SIAM Review*, 15(3):657–658, 1973.

[63] G. Golub and C. van Loan. *Matrix Computations*. The John Hopkins University Press, London, 3rd edition, 1996.

[64] G. Hanrot. LLL: A tool for effective diophantine approximation. In P. Nguyen and B. Vallé, editors, *The LLL Algorithm*, pages 215–263. Springer, Berlin, 2010.

[65] B. Hassibi and H. Vikalo. On the sphere-decoding algorithm I. Expected complexity. *IEEE Transaction on Signal Processing*, 53(8):2806–2818, 2005.

[66] J. Håstad, B. Just, J. Lagarias, and C. Schnorr. Polynomial time algorithms for finding integer relations among real numbers. *SIAM Journal on Computing*, 18(5):859–881, 1989. Preliminary version: Proceedings of STACS'86, pp. 105–118, 1986.

[67] N. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, 1st edition, 1996.

[68] M. van Hoeij. Factoring polynomials and the knapsack problem. *Journal of Number Theory*, 95(2):167–189, 2002.

[69] M. van Hoeij and A. Novocin. Gradual sub-lattice reduction and a new complexity for factoring polynomials. *Algorithmica*, 63(3):616–633, 2012. Preliminary version: Proceedings of LATIN '10, pp. 539–553, 2010.

[70] C. Hoppen, V. Rodrigues, and V. Trevisan. A note on Gao's algorithm for polynomial factorization. *Theoretical Computer Science*, 412(16):1508–1522, 2011.

[71] A. Householder. Unitary triangularization of a nonsymmetric matrix. *Journal of ACM*, 5(4):339–342, 1958.

[72] Y. Huang, W. Wu, H. Stetter, and L. Zhi. Pseudofactors of multivariate polynomials. In *Proceedings of the 2000 international symposium on Symbolic and algebraic computation*, pages 161–168, St. Andrews, Scotland, 2000.

[73] M. van der Hulst and A. Lenstra. Factorization of polynomials by transcendental evaluation. In B. Caviness, editor, *EUROCAL '85*, volume 204 of *Lecture Notes in Computer Science*, pages 138–145. 1985.

[74] D. Inaba. Factorization of multivariate polynomials by extended Hensel construction. *ACM SIGSAM Bulletin*, 39(1):2–14, 2005.

[75] D. Inaba and T. Sasaki. A numerical study of extended Hensel series. In *Proceedings of the 2007 international workshop on Symbolic-numeric computation*, pages 103–109, London, Canada, 2007. ACM.

[76] INRIA. *SYmbolic Numeric APplicationS*. `http://www-sop.inria.fr/galaad/software/synaps/`.

[77] ISSAC. *The International Symposium on Symbolic and Algebraic Computation*. `http://www.issac-conference.org/`.

[78] M. Iwami. Analytic factorization of the multivariate polynomial. In *Proceedings of the 6th International Workshop on Computer Algebra in Scientific Computing, CASC'2003*, pages 213–226, Passau, Germany, 2003.

[79] M. Iwami. Extension of expansion base algorithm to multivariate analytic factorization. In *Proceedings of the 7th International Workshop on Computer Algebra in Scientific Computing, CASC'2004*, pages 269–281, St. Petersburg, Russia, 2004.

[80] B. Just. Integer relations among algebraic numbers. In A. Kreczmar and G. Mirkowska, editors, *Mathematical Foundations of Computer Science 1989*, volume 379 of *Lecture Notes in Computer Science*, pages 314–320. 1989.

[81] E. Kaltofen. A polynomial reduction from multivariate to bivariate integral polynomial factorization. In *Proceedings of the 14th annual ACM symposium on Theory of computing*, pages 261–266. New York, USA, 1982.

[82] E. Kaltofen. A polynomial-time reduction from bivariate to univariate integral polynomial factorization. In *23rd Annual Symposium on Foundations of Computer Science*, pages 57–64, Chicago, USA, 1982.

[83] E. Kaltofen. On the complexity of finding short vectors in integer lattices computer algebra. volume 162 of *Lecture Notes in Computer Science*, pages 236–244. 1983.

[84] E. Kaltofen. Effective Hilbert irreducibility. *Information and Control*, 66(3):123–137, 1985.

[85] E. Kaltofen. Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization. *SIAM Journal on Computing*, 14(2):469–489, 1985.

[86] E. Kaltofen. Sparse Hensel lifting. In B. Caviness, editor, *EUROCAL '85*, volume 204 of *Lecture Notes in Computer Science*, pages 4–17. 1985.

[87] E. Kaltofen. Polynomial factorization 1982–1986. In *Computers and Mathematics*, volume 125 of *Lecture Notes in Pure and Applied Mathematics*, pages 285–309. 1990.

[88] E. Kaltofen. Polynomial factorization 1987–1991. In I. Simon, editor, *LATIN '92*, volume 583 of *Lecture Notes in Computer Science*, pages 294–313. 1992.

[89] E. Kaltofen. Polynomial factorization: A success story. In *Proceedings of the 2003 international symposium on Symbolic and algebraic computation*, pages 3–4, Philadelphia, USA, 2003.

[90] E. Kaltofen and P. Koiran. On the complexity of factoring bivariate supersparse (lacunary) polynomials. In *Proceedings of the 2005 international symposium on Symbolic and algebraic computation*, pages 208–215, Beijing, China, 2005.

[91] E. Kaltofen and P. Koiran. Finding small degree factors of multivariate supersparse (lacunary) polynomials over algebraic number fields. In *Proceedings of the 2006 international symposium on Symbolic and algebraic computation*, pages 162–168, Genoa, Italy, 2006.

[92] E. Kaltofen and G. Lecerf. Factorization of multivairate polynomials. In *Handbook of Finite Fields*. CRC Press, 2013. to appear.

[93] E. Kaltofen, J. May, Z. Yang, and L. Zhi. Approximate factorization of multivariate polynomials using singular value decomposition. *Journal of Symbolic Computation*, 43(5):359–376, 2008.

[94] E. Kaltofen and L. Zhi. Hybrid symbolic-numeric computation. In *Proceedings of the 2006 international symposium on Symbolic and algebraic computation*, pages 7–7, 2006.

[95] R. Kannan, A. Lenstra, and L. Lovász. Polynomial factorization and nonrandomness of bits of algebraic and some transcendental numbers. *Mathematics of Computation*, 50(181):235–250, 1988. Preliminary version: STOC '84, pp. 191–200, 1984.

[96] J. Klüners. The van Hoeij algorithm for factoring polynomials. In P. Q. Nguyen and B. Vallée, editors, *The LLL Algorithm: Survey and Applications*, pages 283–291. 2010.

[97] H. Koy and C. Schnorr. Segment LLL-reduction of lattice bases. In J. Silverman, editor, *Cryptography and Lattices*, volume 2146 of *Lecture Notes in Computer Science*, pages 67–80. 2001.

[98] G. Lecerf. Sharp precision in Hensel lifting for bivariate polynomial factorization. *Mathematics of Computation*, 75(254):921–934, 2006.

[99] G. Lecerf. Improved dense multivariate polynomial factorization algorithms. *Journal of Symbolic Computation*, 42(4):477–494, 2007.

[100] G. Lecerf. Fast separable factorization and applications. *Applicable Algebra in Engineering, Communication and Computing*, 19(2):135–160, 2008.

[101] G. Lecerf. New recombination algorithms for bivariate polynomial factorization based on Hensel lifting. *Applicable Algebra in Engineering, Communication and Computing*, 21(2):151–176, 2010.

[102] A. Lenstra, H. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[103] H. Lenstra. Finding small degree factors of lacunary polynomials. *Number Theory in Progress*, 1:267–276, 1999.

[104] H. Lenstra. On the factorization of lacunary polynomials. *Number Theory in Progress*, 1:277–291, 1999.

[105] S. Liu, C. Ling, and D. Stehlé. Randomized lattice decoding: Bridging the gap between lattice reduction and sphere decoding. In *Proceedings of the 2010 IEEE International Symposium on Information Theory*, pages 2263–2267, Austin, Texas, USA, 2010.

[106] L. Lovász. *An Algorithmic Theory of Numbers, Graphs and Convexity*. Society for Industrial and Applied Mathematics, Philadephia, 1986.

[107] Magma. *Computational Algebra Group, University of Sydney*. Sydney, Australia. `http://magma.maths.usyd.edu.au/magma/`.

[108] Maple. *Waterloo Maple Inc.* Waterloo, Ontario, Canada. `http://www.maplesoft.com`.

[109] J. Martinet. *Perfect Lattices in Euclidean Spaces*, volume 327 of *Comprehensive Studies in Mathematics*. Springer, Berlin, 2003.

[110] Mathematica. *Wolfram Research, Inc.* Champaign, IL, USA. `http://www.wolfram.com`.

[111] A. Meichsner. Integer Relation Algorithms and the Recognition of Numerical Constants. Master's thesis, Simon Fraser University, 2001.

[112] A. Meichsner. *The Integer Chebyshev Problem: Computational Explorations*. PhD thesis, Simon Fraser University, 2009.

[113] D. Micciancio and O. Regev. Lattice-based cryptography. In D. Bernstein, J. Buchmann, and E. Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer Berlin, 2009.

[114] M. Mignotte and M. Waldschmidt. Linear forms in two logarithms and sehneider's method. *Mathematische Annalen*, 231:241–267, 1978.

[115] H. Minkowski. *Geometrie der Zahlen*. Teubner-Verlag, Leipzig, 1896.

[116] M. Monagan, K. Geddes, K. Heal, G. Labahn, S. Vorkoetter, J. McCarron, and P. DeMarco. *Maple Advanced Programming Guide*. Maplesoft, Waterloo Maple Inc., 2009.

[117] I. Morel, D. Stehlé, and G. Villard. H-LLL: Using householder inside LLL. In *Proceedings of the 2009 international symposium on Symbolic and algebraic computation*, pages 271–278, Seoul, Republic of Korea, 2009.

[118] J.-M. Muller, N. Brisebarre, F. Dinechin, C.-P. Jeannerod, V. Lefèvre, G. Melquiond, N. Revol, D. Stehlé, and S. Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser, Boston, 2009.

[119] D. Musser. Multivariate polynomial factorization. *Journal of ACM*, 22(2):291–308, 1975.

[120] P. Nguyen. *La Géométrie des Nombres en Cryptologie*. PhD thesis, Université Paris 7, 1999.

[121] P. Nguyen and D. Stehlé. An LLL algorithm with quadratic complexity. *SIAM Journal on Computing*, 39(3):874–903, 2009. Preliminary version: EUROCRYPT '05, LNCS 3494, pp. 215–233, 2005.

[122] A. Novocin. *Factoring Univariate Polynomials over the Rationals*. PhD thesis, Florida State University, 2008.

[123] A. Novocin, D. Stehlé, and G. Villard. An LLL-reduction algorithm with quasi-linear time complexity: extended abstract. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 403–412, San Jose, USA, 2011.

[124] A. Oil. *APplied COmputations in COmutative Algebra*. `http://www.apcocoa.org`.

[125] M. Pohst. A modification of the LLL reduction algorithm. *Journal of Symbolic Computation*, 4(1):123–127, 1987.

[126] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, 3rd edition, 2007.

[127] X. Qin, Y. Feng, J. Chen, and J. Zhang. Finding exact minimal polynomial by approximations. In *Proceedings of the 2009 conference on Symbolic numeric computation*, pages 125–131, Kyoto, Japan, 2009.

[128] X. Qin, Y. Feng, J. Chen, and J. Zhang. A complete algorithm to find exact minimal polynomial by approximations. *International Journal of Computer Mathematics*, 89(17):2333–2344, 2012.

[129] O. Regev. Lattices in Computer Science, 2004. Lecture notes of a course taught at Tel Aviv University. Available at `http://www.cims.nyu.edu/~regev/`.

[130] C. Rössner and C. Schnorr. Diophantine approximation of a plane, 1997. Available at `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.47.6330`.

[131] T. Sasaki. Approximate multivariate polynomial factorization based on zero-sum relations. In *Proceedings of the 2001 international symposium on Symbolic and algebraic computation*, pages 284–291, London, Ontario, Canada, 2001.

[132] T. Sasaki and D. Inaba. Hensel construction of $f(x, u_1, \cdots, u_l)$, $l \geq 2$, at a singular point and its applications. *ACM SIGSAM Bulletin*, 34(1):9–17, 2000.

[133] T. Sasaki and D. Inaba. Convergence and many-valuedness of Hensel series near the expansion point. In *Proceedings of the 2009 conference on Symbolic numeric computation*, pages 159–168, Kyoto, Japan, 2009. ACM.

[134] T. Sasaki and F. Kako. Solving multivariate algebraic equation by Hensel construction. *Japan Journal of Industrial and Applied Mathematics*, 16(2):257–285, 1999.

[135] T. Sasaki, T. Saito, and T. Hilano. Analysis of approximate factorization algorithm I. *Japan Journal of Industrial and Applied Mathematics*, 9(3):351–368, 1992.

[136] T. Sasaki and M. Sasaki. A unified method for multivariate polynomial factorizations. *Japan Journal of Industrial and Applied Mathematics*, 10(1):21–39, 1993.

[137] T. Sasaki, M. Suzuki, M. Kolář, and M. Sasaki. Approximate factorization of multivariate polynomials and absolute irreducibility testing. *Japan Journal of Industrial and Applied Mathematics*, 8(3):357–375, 1991.

[138] C. Schnorr. A more efficient algorithm for lattice basis reduction. *Journal of Algorithms*, 9(1):47–62, 1988.

[139] C. Schnorr. Fast LLL-type lattice reduction. *Information and Computation*, 204(1):1–25, 2006.

[140] C. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66(1–3):181–199, 1994.

[141] A. Schönhage. Factorization of univariate integer polynomials by Diophantine approximation and an improved basis reduction algorithm. In *Proceedings of ICALP*, volume 172 of *Lecture Notes in Computer Science*, pages 436–447. Springer, 1984.

[142] A. Storjohann and G. Labahn. Asymptotically fast computation of Hermite normal forms of integer matrices. In *Proceedings of the 1996 international symposium on Symbolic and algebraic computation*, pages 259–266, Zurich, Switzerland, 1996.

[143] G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, Wellesley MA, 4th edition, 2009.

[144] P. Wang. Preserving sparseness in multivariate polynominal factorization. In *Proc. 1977 MACSYMA Users' Conference (NASA)*, pages 55–64, 1977.

[145] P. Wang. An improved multivariate polynomial factoring algorithm. *Mathematics of Computation*, 32(144):1215–1231, 1978.

[146] P. Wang and L. Rothschild. Factoring multivariate polynomials over the integers. *Mathematics of Computation*, 29(131):935–950, 1975.

[147] X. Wang and V. Pan. Acceleration of euclidean algorithm and rational number reconstruction. *SIAM Journal on Computing*, 32(2):548–556, 2003.

[148] M. Weimann. A lifting and recombination algorithm for rational factorization of sparse polynomials. *Journal of Complexity*, 26(6):608–628, 2010.

[149] W. Wu, J. Chen, and Y. Feng. An efficient algorithm to factorize sparse bivariate polynomials over the rationals. *ACM Communications in Computer Algebra*, 43(3):125–126, 2012.

[150] J. Zhang and Y. Feng. Obtaining exact value by approximate computations. *Science in China Series A: Mathematics*, 50(9):1361–1368, 2007.

[151] L. Zhi. Numerical optimization in hybrid symbolic-numeric computation. In *Proceedings of the 2007 international workshop on Symbolic-numeric computation*, pages 33–35, London, Ontario, Canada, 2007.

[152] R. Zippel. Probabilistic algorithms for sparse polynomials. In E. Ng, editor, *Symbolic and Algebraic Computation*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. 1979.

[153] R. Zippel. Newton's iteration and the sparse Hensel algorithm (extended abstract). In *Proceedings of the 4th ACM symposium on Symbolic and algebraic computation*, pages 68–72, Snowbird, United States, 1981.

索引

在学期间发表文章目录

[1] 陈经纬. 矩阵特征值的分布. 西南大学学报 (自然科学版), 29(11):45–47, 2007. (中文核心)

[2] 陈经纬, 冯勇, 秦小林, 张景中. 代数数极小多项式的近似重构. 系统科学与数学, 31(8):903–912, 2011. (中文核心)

[3] 陈经纬, 冯勇, 秦小林, 张景中. SIRD: 一个同步整数关系探测算法. 四川大学学报 (工程科学版), 43(6):127–132, 2011. (EI)

[4] 秦小林, 冯勇, 陈经纬, 李骏. 采用近似方法的实代数数准确表示及其应用. 四川大学学报 (工程科学版), 42(2):126–131, 2010. (EI)

[5] J. Chen, Y. Feng, X. Qin, and J. Zhang. Exact polynomial factorization by approximate high degree algebraic numbers. In *Proceedings of the 2009 conference on Symbolic numeric computation*, pages 21–28, Kyoto, Japan, 2009. (EI)

[6] J. Chen, D. Stehlé, and G. Villard. A new view on HJLS and PSLQ: Sums and projections of lattices. In *Proceedings of the 2013 international symposium on Symbolic and algebraic computation*. Accepted. (EI)

[7] Y. Feng, W. Wu, J. Zhang and J. Chen. Exact bivariate polynoimial factoriziation over $\mathbb{Q}$ by approximation of roots. *Journal of Systems Science and Complexity*. Accepted. (SCI)

[8] X. Qin, Y. Feng, J. Chen, and J. Zhang. Finding exact minimal polynomial by approximations. In *Proceedings of the 2009 conference on Symbolic numeric computation*, pages 125–131, Kyoto, Japan, 2009. (EI)

[9] X. Qin, Y. Feng, J. Chen, and J. Zhang. A complete algorithm to find exact minimal polynomial by approximations. *International Journal of Computer Mathematics*, 89(17):2333–2344, 2012. (SCI)

[10] W. Wu, J. Chen, and Y. Feng. An efficient algorithm to factorize sparse bivariate polynomials over the rationals. *ACM Communications in Computer Algebra*, 43(3):125–126, 2012. (ISSAC'12 poster)

主要学术活动

2013 年 4 月, 在可信计算研讨会上做题为 *A New View on HJLS and PSLQ: Sums and Projections of Lattices* 的学术报告 (中国, 北京).

2012 年 12 月, 在 LIP AriC Tuesday Work Sessions 上作题为 *New Results on the PSLQ Algorithm* 的学术报告 (法国, 里昂). `http://www.ens-lyon.fr/LIP/AriC/Tuesday.html`

2012 年 10 月, 参加 ECRYPT II Summer School on Lattices (葡萄牙, 波尔图). `http://latticeschool.di.uminho.pt/`

2012 年 7 月, 参加第 37 届符号与代数计算国际会议 ISSAC (法国, 格勒诺布尔), 作题为 *An Efficient Algorithm to Factorize Sparse Bivariate Polynomials over the Rationals* 的 Poster 展示. `http://www.issac-conference.org/2012/posters.html`

2012 年 1 月至 2012 年 12 月, 访问法国里昂高等师范学院 (ENS de Lyon) 并行计算实验室 (LIP). `http://www.ens-lyon.fr/LIP/web/`

2010 年 10 月, 参加全国第 3 届计算机数学会议 CM 2010 (中国, 上海), 作题为《代数数极小多项式近似重构》的学术报告. `http://www.mmrc.iss.ac.cn/cscm/cscm2010/index.htm`

2009 年 8 月, 参加符号计算暑期讲习班 SSSC (中国, 成都). `http://sssc.uestc.edu.cn/`

简　　历

基本情况

陈经纬, 男, 四川巴中人, 1984 年 11 月出生, 中国科学院成都计算机应用研究所博士研究生.

教育状况

2007 年 9 月至今, 中国科学院成都计算机应用研究所自动推理实验室硕博连读研究生, 计算机软件与理论专业, 导师: 张景中研究员和冯勇研究员.

2012 年 1 月至 2012 年 12 月, 中国科学院-法国国家科研中心 (CNRS) 联合培养博士研究生, 里昂高等师范学院 (ENS de Lyon) 并行计算实验室 (LIP), 理论计算机科学专业, 导师: Damien Stehlé 教授和 Gilles Villard 研究员.

2003 年 9 月至 2007 年 7 月, 西南大学数学与统计学院, 本科, 应用数学专业.

研究兴趣

符号-数值混合计算、计算代数几何、数值线性代数

项目情况

获得中国科学院-法国国家科研中心 (CNRS) 联合培养博士项目资助 (2012.1–2012.12)

主持四川省科技创新苗子工程项目 (2012.1–2012.12): 基于符号数值混合计算的高性能多项式因式分解软件的设计与实现 (项目负责人: 陈经纬)

参与国家重点基础研究发展计划 (973) 项目 (2011.1–2015.8): 数学机械化方法及其在数字化设计制造中的应用, 基于混合计算的误差可控算法课题 (项目编号: 2011CB302402; 课题负责人: 支丽红研究员)

参与国家自然科学基金面上项目 (2012.1–2015.12): 基于数值方法的有理数域上的准确多元多项式因式分解 (项目编号: 11171053; 项目负责人: 冯勇研究员)

参与国家自然科学基金面上项目 (2008.1–2010.12): 从近似值获取准确值的理论, 方法及其应用 (项目编号: 10771205; 项目负责人: 冯勇研究员)

联系方式

电子邮件: velen.chan@163.com

个人主页: https://sites.google.com/site/jingweichen84/

致　　谢

能够有机会从事自己热爱的工作是上天的眷顾, 我心存感激. 尽管如此, 在攻读博士学位期间, 仍然遇到过不少困难, 诸多老师和同学对本文工作都给予过支持和帮助. 没有他们, 便难成此文!

感谢我的两位中国导师: 张景中研究员和冯勇研究员. 本文的大部分内容都是在两位导师的指导下完成的. 正是他们在 “采用近似计算获得准确值” 方面的开创性工作促使我选择本文的研究内容当做博士课题. 在攻读博士学位期间, 他们始终给予我关心、帮助和支持. 张老师深厚的学术造诣、敏锐的洞察力、平易近人的作风将使我受益终生. 冯老师渊博的知识、严谨的治学态度、对学生无微不至的关怀也是我永远的榜样.

感谢我的两位法国导师: Damien Stehlé 教授和 Gilles Villard 研究员. 在法期间, 他们无私地帮助我融入当地生活, 工作上给予我耐心细致的指导, 并资助我参加学术交流. 本文的部分内容是在他们的指导下完成的.

感谢中国科学院重庆绿色智能技术研究院吴文渊副研究员对我的指导, 与他合作是一件非常愉快的事情. 感谢我的师兄秦小林博士自我入学以来至今, 对我学习和生活上的指导和帮助. 感谢杨路研究员. 他多次精彩的学术报告不仅介绍了当前研究的最新研究进展, 而且加深了我对所从事的方向和学科的理解. 感谢中国科学院数学与系统科学研究院支丽红研究员对我的支持与鼓励, 与她的讨论使我受益匪浅.

感谢王晓京研究员、付忠良研究员、符红光研究员、陈斌研究员、钟勇研究员、崔喆研究员、姚勇老师等. 从与他们的交流和他们的报告中, 我更加懂得如何追求至善尽美. 感谢汤琳老师、苏茂老师、莫邦辉老师给予的关心和帮助.

感谢曾与我一起共事的师兄、同学和师弟们: 李轶博士、李骏博士、杨世翰博士、刘栋博士、靳泰戈博士、袁勋博士、余伟博士、代翔博士、方佳嘉博士、牟琳博士、杨治安博士、李玲娜博士、于传帅博士、蔡红亮、季振义、谭志英、冷拓、蒋海波、肖宜龙、许振兴、杨建书、肖劲飞、胡康达、周瑾、邱炜、徐靖、陆见光、张国华、唐卷、李光远、何易德等. 和他们一起, 度过了一段难忘的愉快时光.

感谢在法期间给予过我帮助的其他老师和同学: Claude-Pierre Jeannerod 研究员、Nathalie Revol 研究员、Nicolas Brisebarre 研究员、Guillaume Hanrot 教授、Florent de Dinechin 副教授、San Ling 教授、Bruno Salvy 研究员、Fabien Laguillaumie 教授、Nicolas Louvet 老师、Jean-Michel Muller 研究员、Vincent Lefèvre 研究员、唐兵博士、侯正雄博士、Jingyan Jourdan-Lu 博士、Rishiraj Bhattacharyya 博士、Philippe Théveny, Adeline Langlois 和 Matei Istoan. 谢谢他们曾给予我生活上的帮助和工作上的指导.

衷心感谢审阅本论文的各位专家. 他们所给出的宝贵意见和建议使得本文更加完善.

最后, 诚挚地感谢我的父亲、母亲和弟弟, 他们多年以来对我的宽容理解与坚定支持是我在前进道路上源源不断的动力. 特别感谢我的爱人刘洋女士对我的支持与鼓励, 她的悉心照料和无私付出给了我战胜一切困难的勇气和决心.

二零一三年五月于成都