中国科学技术大学

学士学位论文

O'Stolz 公式在有理函数作为通项的级数求和问题中的应用

作者姓名： 吴天

学科专业： 数学与应用数学

导师姓名： 李平 教授

完成时间： 二〇一九年五月十九日

University of Science and Technology of China

A dissertation for bachelor's degree

The application of O'Stolz formula in calculating a series with general terms of rational functions

Author: Tian Wu

Speciality: Mathematics and Applied Mathematics

Supervisor: Prof. Ping Li

Finished time: May 19, 2019

致　谢

我虽然是学基础数学专业的学生，但是一直以来对于计算数学都很感兴趣。大二期间，李平教授曾经辅导过我的大学生数学竞赛分析部分，并且，我还在大三期间当过一整年李老师数学分析 (B) 课程的助教。在与李老师接触期间，我无论是对于知识，还是为人，均收获颇多。在确定下做关于计算数学方向的毕业论文的时候，我就想起了李老师本科期间学习计算数学专业，于是我将我的问题与想法和李老师交流了一番，李老师也表示出了极大的兴趣. 在做毕业论文的半年多来，我更是通过一次次的向李老师请教，产生了许多算法的灵感，令我对于数学有了进一步的认识，也更令我为本科学习画上了圆满的句号。在此，由衷地感谢李老师对于我的所有帮助！

此外，在刚刚确定课题之时，我还获得过胡森教授的指点与帮助，胡老师与我交流了很长时间，为我的课题研究指明了几条方向，为我最终确定课题的思路奠定了良好的基础。也十分感谢胡老师对我的指点！

“红专并进一甲子，科教报国六十年。”感谢我的母校——中国科学技术大学，感谢所有帮助过我的老师，四年的培养与教育，令我终身受益匪浅. 祝母校教育越办越好，老师们身体健康、工作顺利！

最后，谨以此论文，献给我的父母，献给我的所有家人，献给我的家乡——齐齐哈尔，感谢养育之恩。

吴天　2019 年 5 月 19 日

于中国科学技术大学

目　录

中文内容摘要

O’Stolz 公式给出了计算 $\frac{*}{\infty}$ 型与 $\frac{0}{0}$ 型极限的有效方法，在分析学的研究中广为应用。本文就这一公式，结合递推公式法与逼近思想，给出了一种计算以有理函数作为通项的级数的近似和的更快速、更高精度的一种算法。

本文先对 Riemann-ζ 函数的情况进行研究，对于 $p > 1$ 时的 Riemann-ζ 函数的数值计算，以及 $\forall p > 0$ 时的 Riemann-ζ 函数的部分和数值计算问题给出了详细的算法与证明。之后，对于一般的有理函数为通项的级数，利用类似的方法给出了理论上的推导。

由于 Riemann-ζ 函数的渐近展开式中的系数规律极其复杂，因此本文算法的稳定性依旧有待研究。但是通过大量实验观测得知，只要所取测试求和项数 N 足够大的情况下，算法的稳定性能够得以保证。除此之外，本文还对于其能否在 Riemann-ζ 函数的研究领域，以及更一般的级数数值计算领域中进一步发展给出相应的推测与展望。

关键词：O’Stolz 公式；级数；数值计算；算法；逼近论

Abstract

O'Stolz Formula provides a powerful method of dealing with $\frac{*}{\infty}$ and $\frac{0}{0}$ limits, which is widely applied in Analysis research. With the help of this formula, this article carries out a quicker and more accurate algorithm to calculate a series with general terms of rational functions by the method of recurrence formula and approximation.

In this article, the Riemann-ζ function is studied firstly. The algorithm and proof of the numerical calculation of Riemann-ζ function ($p > 1$) and the partial summation of Riemann-ζ function ($\forall p > 0$) are given in detail. Besides that, theoretical deduction for the series of general terms of general rational functions are given.

Because it's too complicated to find the law of coefficients in the asymptotic expansion of Riemann-ζ function, the stability of this algorithm still needs to be studied. However, a large number of experimental observations show that the stability of the algorithm can be guaranteed as long as the number of test summation terms N is large enough. In addition, this paper also discusses whether it can be used in Riemann-ζ function, and gives corresponding speculation and prospects for its further development in the field of numerical calculation of more general series.

Key Words: O'Stolz formula; series; numerical calculation; algorithm; approximation

第一章　引言

第一节　问题背景

级数，是可列个数相加的一个极限过程，我们一般定义为

$$\sum_{n=1}^{\infty} a_n = \lim_{N\to\infty} \sum_{n=1}^{N} a_n.$$

如果这个极限是收敛 (发散) 的，我们就称对应的级数是收敛 (发散) 的. 有些收敛的级数是能够从理论上去化简并计算出来的，然而对于大多数的收敛级数，我们无法给出它精确的解析化简结果. 通常来说，数值逼近是研究这类级数的重要方法之一. 此外，对于一些发散级数，当 N 比较大的时候，我们想要研究它的前 N 项和的数值近似，这也是一大类需要处理的问题. 以上两类问题的解决在积分的数值逼近上具有极其重要的作用，不仅如此，许多实际应用问题也需要研究级数的数值计算问题. 因此，本文就有理函数为通项的级数数值计算问题进行讨论，给出相应的一种算法，并给出其思想对于其他几类级数数值求和问题的应用.

在不加说明的情况下，本文中出现的所有函数均为单变量实函数，所有级数均从第 1 项开始求和.

第二节　预备定义与结论

本节列出了处理目标问题需要用到的定理，它们都是数学分析中比较基本的结论，此处略去证明. 它们的详细证明可以参考[1].

定理 1.1 ($\frac{*}{\infty}$ 型 O'Stolz 公式)　设 $\{a_n\}$ 是单调递增且趋向于 $+\infty$ 的数列，如果 $\lim\limits_{n\to\infty} \dfrac{b_{n+1}-b_n}{a_{n+1}-a_n}$ 存在或为 ∞，则数列 $\{\dfrac{b_n}{a_n}\}$ 收敛，且

$$\lim_{n\to\infty} \frac{b_n}{a_n} = \lim_{n\to\infty} \frac{b_{n+1}-b_n}{a_{n+1}-a_n}.$$

定理 1.2 ($\frac{0}{0}$ 型 O'Stolz 公式)　设 $\{a_n\}$ 是单调递减且趋向于 0 的数列，如果 $\lim\limits_{n\to\infty} \dfrac{b_{n+1}-b_n}{a_{n+1}-a_n}$ 存在或为 ∞，则数列 $\{\dfrac{b_n}{a_n}\}$ 收敛，且

$$\lim_{n\to\infty} \frac{b_n}{a_n} = \lim_{n\to\infty} \frac{b_{n+1}-b_n}{a_{n+1}-a_n}.$$

针对函数，我们有类似于 O'Stolz 公式的结论：

定理 1.3 ($\frac{*}{\infty}$ 型 L'Hospital 法则)　设 f, g 在 (a, b) 上可导，且 $g(x) \neq 0$，$g(a^+) = \infty$. 如果 $\lim\limits_{x\to a^+} \dfrac{f'(x)}{g'(x)}$ 存在或为 ∞，则 $\lim\limits_{x\to a^+} \dfrac{f(x)}{g(x)}$ 存在，且

$$\lim_{x\to a^+} \frac{f(x)}{g(x)} = \lim_{x\to a^+} \frac{f'(x)}{g'(x)}.$$

定理 1.4 ($\frac{0}{0}$ 型 L'Hospital 法则)　设 f, g 在 (a, b) 上可导，且 $g(x) \neq 0$，$f(a^+) = g(a^+) = 0$. 如果 $\lim\limits_{x\to a^+} \dfrac{f'(x)}{g'(x)}$ 存在或为 ∞，则 $\lim\limits_{x\to a^+} \dfrac{f(x)}{g(x)}$ 存在，且

$$\lim_{x\to a^+} \frac{f(x)}{g(x)} = \lim_{x\to a^+} \frac{f'(x)}{g'(x)}.$$

定理 1.5 (单调有界定理)　单调有界数列一定收敛.

定理 1.6　级数 $\sum\limits_{n=1}^{\infty} \dfrac{1}{n^p}$ 在 $p > 1$ 时收敛，$p \leqslant 1$ 时发散.

定义 1.1 (Riemann-ζ 函数)　$\zeta(p) := \begin{cases} \sum\limits_{n=1}^{\infty} \dfrac{1}{n^p}, & p > 1, \\ \lim\limits_{N\to\infty} \left(\sum\limits_{n=1}^{N} \dfrac{1}{n^p} - \dfrac{N^{1-p}}{1-p} \right), & 0 < p < 1. \end{cases}$

我们会经常使用无穷小和同阶量的符号，它们的定义如下：

定义 1.2　称 $\alpha(x)$ 是 $\beta(x)$ 在 $x \to x_0$ 时的高阶无穷小，如果 $\lim\limits_{x\to x_0} \dfrac{\alpha(x)}{\beta(x)} = 0$，并记作 $\alpha(x) = \mathrm{o}(\beta(x))\ (x \to x_0)$. 称 $\alpha(x)$ 是 $\beta(x)$ 在 $x \to x_0$ 时的同阶量，如果 $\lim\limits_{x\to x_0} \dfrac{\alpha(x)}{\beta(x)} = c \in \mathbf{R}\backslash\{0\}$，并记作 $\alpha(x) = \mathrm{O}(\beta(x))\ (x \to x_0)$.

定理 1.7 (带 Peano 余项的 Taylor 定理)　设函数 f 在 x_0 处 n 阶可导，则

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k + \mathrm{o}((x - x_0)^n)\ (x \to x_0).$$

定理 1.8　$\sum\limits_{n=1}^{N} \dfrac{1}{n} = \log N + \gamma + \mathrm{o}(1)\ (N \to \infty)$，其中 γ 为 Euler 常数.

再给出一个关于函数的次数的定义，它能够与已有的多项式次数的概念吻合.

定义 1.3　称 $\deg_g f = \inf\{\alpha : f(x) = \mathrm{o}(g(x)^\alpha)\ (x \to \infty)\}$ 为 f 关于 g 的次数. 如果取 $g(x) = x$，则称 $\deg_x f$ 为 f 的次数，并记作 $\deg f$.

第三节　算法实现方式

Wolfram Mathematica(以下简称 Mathematica) 是一款科学计算软件，它十分擅长于数值计算，而本文的算法正是需要先对所有结果在程序内部求得解析表

达式之后，再进行数值近似，因此 Mathematica 是很好的选择. 论文中的算法所使用的计算软件版本为 Wolfram Mathematica 11.3.0.0.

WolframAlpha 是与 Mathematica 出自同一家公司的搜索引擎，它具有联网计算的功能，在进行搜索、计算时，它使用服务器云端大量的闲置计算机进行高速运算，因此其速度与精度远超普通个人计算机. 为了证明论文给出的算法的优越性，我们需要将其与 Mathematica 已有函数的效率进行对比，这需要控制变量，保持在同一台计算机的相同状态下分别运行. 衡量算法效率指标主要为在控制住同等有效精度情况下的计算速度，在求有效精度的时候，就需要“精确值”(换句话说，是相对更精确的值)，我们使用 WolframAlpha 作为检验平台，获得精确值.

运行平台如下：

- 运行计算机型号：华硕 FL5600L.
- 操作系统：Windows 10 x64 教育版 (Build 1809).
- CPU：Intel 酷睿 i7-5500U @ 2.40GHz 2.39GHz 双核四线程.
- 内存 (RAM)：4GB.

第二章　算法的理论证明

在无特殊说明的情况下，用 $\dfrac{u(n)}{v(n)}$ 表示有理函数为通项的级数 (以下简称有理级数)，其中 u, v 均为多项式，且 $\deg u < \deg v$，v 不以任何一个正整数作为零点. 本章给出算法的思路，并作出详细的理论推导. 算法思路分为以下两步：

(1) Riemann-ζ 函数，$p > 0$，考察其有限项求和问题；

(2) 通过使用 Riemann-ζ 函数部分和问题的思想，对一般通项分母不以正整数为零点的有理收敛级数类似进行逼近，考察其数值计算问题.

由上述思路可以看出，我们的核心问题是做到逼近 Riemann-ζ 函数的数值结果，这也是理论证明的主要组成部分.

第一节　算法的引入

本节通过举出一个数学分析中的竞赛题目，作为算法的引入. 它表现出了算法来源的灵感.

例题　设 $\{a_n\}$ 满足，$a_0 \in (0,1)$，$a_{n+1} = \sin a_n$，计算 $\lim\limits_{n\to\infty} \sqrt{n}a_n$.

解　由单调有界定理容易得知：$\lim\limits_{n\to 0} a_n = 0$. 下面考虑 O'Stolz 公式：

$$\lim_{n\to\infty} na_n^2 = \lim_{n\to\infty} \frac{n}{\frac{1}{a_n^2}} = \lim_{n\to\infty} \frac{1}{\frac{1}{a_{n+1}^2} - \frac{1}{a_n^2}} = \lim_{n\to\infty} \frac{a_n^2 \sin^2 a_n}{a_n^2 - \sin^2 a_n}.$$

由 O'Stolz 公式的条件，只要右侧的极限存在，即可解决问题. 由于 $\lim\limits_{n\to 0} a_n = 0$，不妨考察以下极限：

$$\lim_{t\to 0} \frac{t^2 \sin^2 t}{t^2 - \sin^2 t} = \lim_{t\to 0} \frac{t^4}{2t(t - \sin t)} = \lim_{t\to 0} \frac{t^3}{2(\frac{1}{6}t^3 + \mathrm{o}(t^3))} = 3$$

因此 $\lim\limits_{n\to\infty} \sqrt{n}a_n = \sqrt{3}$.

这道例题告诉我们，我们对于一些已有的小量，通常可以继续乘以一个比较简单的无穷大量 (例如上面的幂函数 $\sqrt{n}$) 来继续研究它更精细的大小，而这里的核心方法便是 O'Stolz 公式. 而只要能够每一次都能猜得出这个无穷大量具体取多大，我们就能够重复进行这个过程，把未知的小量研究的更加透彻.

第二节　Riemann-ζ 函数的数值计算

本节总是假设 $p>0$. 考虑对于充分大的 N，$\sum\limits_{n=1}^{N}\dfrac{1}{n^p}$ 的数值计算问题. 下面以定理的形式给出本节算法的核心结论：

定理 2.1 (Riemann-ζ 函数部分和逼近定理)

(1) 设 $p=1$，则存在 $\{c_m^{(1)}\}_{m=1}^{\infty}$，使得 $\forall s\geqslant 2$且$s\in\mathbf{N}^*$，

$$\sum_{n=1}^{N}\frac{1}{n}=\log N+\gamma+\frac{1}{2}N^{-1}+\sum_{m=1}^{s-1}c_m^{(1)}N^{-2m}+\mathrm{O}(N^{-2s})\ (N\to\infty).$$

更进一步，$c_1^{(1)}=-\dfrac{1}{12}$，$\forall s\geqslant 2$且$s\in\mathbf{N}^*$，

$$c_s^{(1)}=\frac{1}{2s}\left(\frac{1}{1+2s}-\frac{1}{2}-\sum_{m=1}^{s-1}\binom{2s}{2m-1}c_m^{(1)}\right).$$

(2) 设 $p\neq 1$，则存在 $\{c_m^{(p)}\}_{m=1}^{\infty}$，使得 $\forall s\geqslant 2$且$s\in\mathbf{N}^*$，

$$\sum_{n=1}^{N}\frac{1}{n^p}=\frac{N^{1-p}}{1-p}+\zeta(p)+\frac{1}{2}N^{-p}+\sum_{m=1}^{s-1}c_m^{(p)}N^{-p-2m+1}+\mathrm{O}(N^{-p-2s+1})\ (N\to\infty).$$

更进一步，$c_1^{(p)}=-\dfrac{p}{12}$，$\forall s\geqslant 2$且$s\in\mathbf{N}^*$，

$$c_s^{(p)}=\frac{1}{-p-2s+1}\left(-\frac{1}{1-p}\binom{1-p}{2s+1}+\frac{1}{2}\binom{-p}{2s}-\sum_{m=1}^{s-1}c_m^{(p)}\binom{-p-2m+1}{2s-2m+1}\right).$$

一、确定最高阶项——面积原理

在定义 Riemann 积分的时候，我们是用分割的级数求和定义的. 而面积原理就是用积分估计和式.

定理 2.2 (面积原理[1])　设 $x\geqslant m\in\mathbf{N}^*$ 时 f 是非负递减函数，则

$$\lim_{n\to\infty}\left(\sum_{k=m}^{n}f(k)-\int_{m}^{n}f(x)\mathrm{d}x\right)=\alpha\in[0,f(m)].$$

证明　利用 f 的单调递减性：

$$\sum_{k=m}^{n}f(k)-\int_{m}^{n}f(x)\mathrm{d}x\geqslant\sum_{k=m}^{n-1}\int_{k}^{k+1}f(k)\mathrm{d}x-\int_{m}^{n}f(x)\mathrm{d}x\geqslant 0,$$

$$\sum_{k=m}^{n}f(k)-\int_{m}^{n}f(x)\mathrm{d}x\leqslant f(m)+\sum_{k=m+1}^{n}\int_{k-1}^{k}f(k)\mathrm{d}x-\int_{m}^{n}f(x)\mathrm{d}x\leqslant f(m).$$

又因为 $f(n)-\int_{n-1}^{n} f(x)\mathrm{d}x<0$，故 $\sum_{k=m}^{n} f(k)-\int_{m}^{n} f(x)\mathrm{d}x$ 单调递减，因此由定理1.5，存在 $\alpha\in[0,f(m)]$，使得 $\lim_{n\to\infty}\left(\sum_{k=m}^{n} f(k)-\int_{m}^{n} f(x)\mathrm{d}x\right)=\alpha$. □

在面积原理中，如果取 f 为满足一些特殊条件的函数，我们可以得到一些推论. 下面是一条比较重要的推论.

定理 2.3 (Cauchy 积分判别法) 设 $x\geqslant m\in\mathbf{N}^*$ 时 f 是非负递减函数，则级数 $\sum_{n=m}^{\infty} f(n)$ 与反常积分 $\int_{m}^{+\infty} f(x)\mathrm{d}x$ 同敛散.

利用上述结论，我们容易得到 Riemann-ζ 函数的部分和的最高阶项估计：

定理 2.4 设 $p>0$，记 $\zeta_N(p)=\sum_{n=1}^{N}\frac{1}{n^p}$，则：

(1) 当 $p\neq 1$ 时，$\lim_{N\to\infty}\left(\zeta_N(p)-\frac{N^{1-p}}{1-p}\right)$ 存在，即 $\zeta(p)$.

(2) 当 $p=1$ 时，$\lim_{N\to\infty}(\zeta_N(1)-\log N)$ 存在，即 Euler 常数 γ.

事实上，上述结论的第 (2) 条已经证明了定理1.8.

二、$p=1$ 时的精细估计

本小节考察 $p=1$ 时的情况，下一小节以此作为引子，仿照着给出 $p\neq 1$ 的一般理论. 由定理1.8，我们有

$$\sum_{n=1}^{N}\frac{1}{n}=\log N+\gamma+\mathrm{o}(1)\ (N\to\infty).$$

为了研究 o(1) 的具体情况，我们猜测其可能为 $\mathrm{O}(\frac{1}{N})$，因此利用 O'Stolz 公式尝试计算

$$\lim_{N\to\infty}\frac{\sum_{n=1}^{N}\frac{1}{n}-\log N-\gamma}{\frac{1}{N}}=\lim_{N\to\infty}\frac{\frac{1}{N}-\log N+\log(N-1)}{-\frac{1}{N^2}}$$

$$=-\lim_{N\to\infty}N\left(1+N\log(1-\frac{1}{N})\right)=-\lim_{N\to\infty}N\left(1+N(-\frac{1}{N}-\frac{1}{2N^2}+\mathrm{o}(\frac{1}{N^2}))\right)=\frac{1}{2}.$$

因此 $\sum_{n=1}^{N}\frac{1}{n}=\log N+\gamma+\frac{1}{2N}+\mathrm{o}(\frac{1}{N})\ (N\to\infty)$. 继续猜测余项为为 $\mathrm{O}(\frac{1}{N^2})$，从而继续计算

$$\lim_{N\to\infty}\frac{\sum_{n=1}^{N}\frac{1}{n}-\log N-\gamma-\frac{1}{2N}}{\frac{1}{N^2}}=\lim_{N\to\infty}\frac{\frac{1}{N}-\log N+\log(N-1)-\frac{1}{2N}+\frac{1}{2(N-1)}}{-\frac{2}{N^3}}$$

$$=-\lim_{N\to\infty}\frac{N^2}{2}\left(1+N\log(1-\frac{1}{N})+\frac{1}{2(N-1)}\right)$$

$$=-\lim_{N\to\infty}\frac{N^2}{2}\left(\frac{1}{2(N-1)}-\frac{1}{2N}-\frac{1}{3N^2}+\mathrm{o}(\frac{1}{N^2})\right)=-\frac{1}{12}.$$

反复做下去，我们可以得到

$$\sum_{n=1}^{N}\frac{1}{n}=\log N+\gamma+\frac{1}{2N}-\frac{1}{12N^2}+\frac{1}{120N^4}-\frac{1}{252N^6}+\frac{1}{240N^8}-\frac{1}{132N^{10}}+\mathrm{O}(\frac{1}{N^{12}}).$$

先忽略掉余项，取 $N=100$，代入上式，两侧取 28 位小数近似值，解得 $\gamma\approx 0.5772156649015328606065121112$，它的绝对误差约为 2.11×10^{-26}. 有了展开式中的常数的近似值，我们就可以用它来估计有限项的近似值. 例如，要计算调和级数的前 1000000 项和，只需将 $N=1000000$ 代入，γ 的近似值也代入，即可求得 $\sum_{n=1}^{1000000}\frac{1}{n}\approx 14.392726722865723631381127514 3$，绝对误差依旧约为 2.11×10^{-26}，这的确不难理解，因为展开式本身只能产生 10^{-72} 量级的误差，而这主要误差是上一步中 γ 的计算带来的.

首先介绍一个显然的引理，它将用于定理的证明.

定理 2.5 设 $\alpha\in\mathbf{R}$，则 $1-\left(1-\frac{1}{N}\right)^{\alpha}=\sum_{j=1}^{\infty}\binom{\alpha}{j}(-1)^{j-1}\frac{1}{N^j}\ (N\in\mathbf{N}^*)$.

下面给出此方法对应的定理及其证明：

定理 2.6 (调和级数部分和逼近定理) 存在 $\{c_m^{(1)}\}_{m=1}^{\infty}$，使得 $\forall s\geqslant 2$且$s\in\mathbf{N}^*$，

$$\sum_{n=1}^{N}\frac{1}{n}=\log N+\gamma+\frac{1}{2}N^{-1}+\sum_{m=1}^{s-1}c_m^{(1)}N^{-2m}+\mathrm{O}(N^{-2s})\ (N\to\infty).$$

更进一步，$c_1^{(1)}=-\frac{1}{12}$，$\forall s\geqslant 2$且$s\in\mathbf{N}^*$，

$$c_s^{(1)}=\frac{1}{2s}\left(\frac{1}{1+2s}-\frac{1}{2}-\sum_{m=1}^{s-1}\binom{2s}{2m-1}c_m^{(1)}\right).$$

证明 假设 $\forall 1 \leqslant j \leqslant s-1$，$c_j^{(1)} = \frac{1}{2j}\left(\frac{1}{1+2j} - \frac{1}{2} - \sum_{m=1}^{j-1}\binom{2j}{2m-1}c_m^{(1)}\right)$，且满足 $\sum_{n=1}^{N}\frac{1}{n} = \log N + \gamma + \frac{1}{2}N^{-1} + \sum_{m=1}^{s-1}c_m^{(1)}N^{-2m} + \mathrm{O}(N^{-2s})\ (N \to \infty)$. 则取

$$\begin{aligned} c_s^{(1)} &= \lim_{N\to\infty}\frac{1}{N^{-2s}}\left(\sum_{n=1}^{N}\frac{1}{n} - \log N - \gamma - \frac{1}{2}N^{-1} - \sum_{m=1}^{s-1}c_m^{(1)}N^{-2m}\right) \\ &= \lim_{N\to\infty}\frac{1}{(-2s)N^{-2s-1}}\Big(\frac{1}{N} + \log(1-\frac{1}{N}) - \frac{1}{2}N^{-1}\big(1-(1-\frac{1}{N})^{-1}\big) \\ &\quad - \sum_{m=1}^{s-1}c_m^{(1)}N^{-2m}\big(1-(1-\frac{1}{N})^{-2m}\big)\Big) \end{aligned}$$

由假设条件，只需考察分子的 N^{-2s-1} 项即可. 利用定理2.5有：

$$\begin{aligned} c_s^{(1)} &= \lim_{N\to\infty}\frac{1}{(-2s)N^{-2s-1}}\Big(\frac{1}{N} - \sum_{j=1}^{\infty}\frac{1}{j}N^{-j} + \frac{1}{2}\sum_{j=1}^{\infty}N^{-j-1} \\ &\quad + \sum_{m=1}^{s-1}c_m^{(1)}\sum_{j=1}^{\infty}\binom{2m+j-1}{2m-1}N^{-2m-j}\Big) \\ &= \frac{1}{2s}\left(\frac{1}{1+2s} - \frac{1}{2} - \sum_{m=1}^{s-1}\binom{2s}{2m-1}c_m^{(1)}\right). \end{aligned}$$

由归纳公理知，所证结论成立. □

事实上，展开式的系数的绝对值是逐渐增大的，并且通过观测，很可能为指数增长的速度. 但是，系数的增长规律极为复杂，目前尚不清楚其具体规律. 不过通过多次实验发现，$c_m^{(1)} = \zeta(1-2m)$，且 Riemann-ζ 函数在负偶数处的值全部为 0，这也与展开式中每隔一项就消失一项的现象不谋而合. 展开式系数的问题深究下去，甚至会与 Riemann-ζ 函数的许多未解决的性质有关. 不过通过大量的实验表明，只要 N 取到 1000 以上，N^{-m} 足以抵消掉系数增长带来的误差，因此算法依旧具有稳定性. 不过从严密性的角度来讲，算法的稳定性的证明需要研究系数的变化规律，但限于目前研究的进度，本文对于此算法的稳定性不做讨论，只讨论算法理论上的证明和运行效果.

三、$p \neq 1$ 时的精细估计

本小节即仿照上一小节的方法，给出 $p \neq 1$ 的情况下，Riemann-ζ 函数的部分和渐近展开式.

定理 2.7 (Riemann-ζ 函数部分和逼近定理)　设 $p \neq 1$，则存在 $\{c_m^{(p)}\}_{m=1}^{\infty}$，使得 $\forall s \geqslant 2$且$s \in \mathbf{N}^*$，

$$\sum_{n=1}^{N} \frac{1}{n^p} = \frac{N^{1-p}}{1-p} + \zeta(p) + \frac{1}{2}N^{-p} + \sum_{m=1}^{s-1} c_m^{(p)} N^{-p-2m+1} + \mathrm{O}(N^{-p-2s+1}) \ (N \to \infty).$$

更进一步，$c_1^{(p)} = -\dfrac{p}{12}$，$\forall s \geqslant 2$且$s \in \mathbf{N}^*$，

$$c_s^{(p)} = \frac{1}{-p-2s+1}\left(-\frac{1}{1-p}\binom{1-p}{2s+1} + \frac{1}{2}\binom{-p}{2s} - \sum_{m=1}^{s-1} c_m^{(p)}\binom{-p-2m+1}{2s-2m+1}\right).$$

证明　假设 $\forall 1 \leqslant j \leqslant s-1$，

$$c_j^{(p)} = \frac{1}{-p-2j+1}\left(-\frac{1}{1-p}\binom{1-p}{2j+1} + \frac{1}{2}\binom{-p}{2j} - \sum_{m=1}^{j-1} c_m^{(p)}\binom{-p-2m+1}{2j-2m+1}\right)$$

且满足

$$\sum_{n=1}^{N} \frac{1}{n^p} = \frac{N^{1-p}}{1-p} + \zeta(p) + \frac{1}{2}N^{-p} + \sum_{m=1}^{s-1} c_m^{(p)} N^{-p-2m+1} + \mathrm{O}(N^{-p-2s+1}) \ (N \to \infty).$$

仿照 $p = 1$ 的情况，考察

$$\begin{aligned}
c_s^{(p)} &= \lim_{N\to\infty} \frac{1}{N^{-p-2s+1}}\left(\sum_{n=1}^{N}\frac{1}{n^p} - \frac{N^{1-p}}{1-p} - \gamma(p) - \frac{1}{2}N^{-p} - \sum_{m=1}^{s-1} c_m^{(p)} N^{-p-2m+1}\right) \\
&= \lim_{N\to\infty} \frac{1}{(-p-2s+1)N^{-p-2s}}\Big(\frac{1}{N^p} - \frac{N^{1-p}}{1-p}\big(1-(1-\frac{1}{N})^{1-p}\big) - \frac{1}{2}N^{-p}\big(1-(1-\frac{1}{N})^{-p}\big) \\
&\quad - \sum_{m=1}^{s-1} c_m^{(p)} N^{-p-2m+1}\big(1-(1-\frac{1}{N})^{-p-2m+1}\big)\Big) \\
&= \lim_{N\to\infty} \frac{1}{(-p-2s+1)N^{-p-2s}}\Big[\frac{1}{N^p} - \frac{N^{1-p}}{1-p}\sum_{j=1}^{\infty}\binom{1-p}{j}(-1)^{j-1}\frac{1}{N^j} \\
&\quad - \frac{1}{2}N^{-p}\sum_{j=1}^{\infty}\binom{-p}{j}(-1)^{j-1}\frac{1}{N^j} - \sum_{m=1}^{s-1} c_m^{(p)} N^{-p-2m+1}\sum_{j=1}^{\infty}\binom{-p-2m+1}{j}(-1)^{j-1}\frac{1}{N^j}\Big] \\
&= \frac{1}{-p-2s+1}\left(-\frac{1}{1-p}\binom{1-p}{2s+1} + \frac{1}{2}\binom{-p}{2s} - \sum_{m=1}^{s-1} c_m^{(p)}\binom{-p-2m+1}{2s-2m+1}\right).
\end{aligned}$$

我们只需考察 $c_1^{(p)}$ 即可：

$$\begin{aligned}
c_1^{(p)} &= \lim_{N\to\infty} \frac{1}{N^{-p-1}}\left(\sum_{n=1}^{N}\frac{1}{n^p} - \frac{N^{1-p}}{1-p} - \gamma(p) - \frac{1}{2}N^{-p}\right) \\
&= \lim_{N\to\infty} \frac{1}{(-p-1)N^{-p-2}}\Big(\frac{1}{N^p} - \frac{N^{1-p}}{1-p}\big(1-(1-\frac{1}{N})^{1-p}\big) - \frac{1}{2}N^{-p}\big(1-(1-\frac{1}{N})^{-p}\big)\Big) \\
&= \frac{1}{(-p-1)}\left(-\frac{1}{1-p}\binom{1-p}{3} + \frac{1}{2}\binom{-p}{2}\right) = -\frac{1}{12p}.
\end{aligned}$$

由归纳公理知，所证结论成立. □

第三节　一般有理函数的逼近方法

考察本章开头假设条件下的有理级数 $\sum\limits_{n=1}^{N}\dfrac{u(n)}{v(n)}$. 由定义1.3可知，设 $-d=\deg\dfrac{u}{v}=\deg u-\deg v<0$，进而容易知道，取 $c_0=\lim\limits_{n\to\infty}\dfrac{n^d u(n)}{v(n)}$，则

$$\deg\left(\frac{u(n)}{v(n)}-\frac{c_0}{n^d}\right)\leqslant -d-1$$

因此，对于一般的有理级数，我们反复进行用一个 $\zeta_N(p)$ 型问题进行逼近，那么余项的次数会不断减小，当次数减小到一定程度的时候，我们可以将余项的前若干项 (当然是一个不太多的项数) 直接求和，然后与前面的逼近相加，即得到所求有理级数的近似值. 它是由下面这个定理保证的.

定理 2.8 (有理级数逼近定理)　设 $\dfrac{u(x)}{v(x)}$ 是有理函数，$-d=\deg u-\deg v<0$，则 $\exists\{c_m\}_{m=1}^{\infty}$，使得 $\forall s\in\mathbf{N}$，

$$\sum_{n=1}^{N}\frac{u(n)}{v(n)}=\sum_{m=0}^{s}c_m\zeta_N(d+m)+\sum_{n=1}^{N}R_s(n).$$

其中 $R_s(n)$ 为关于 n 的有理函数，且 $\deg R_s\leqslant -d-s-1$. 更进一步，

$$c_0=\lim_{n\to\infty}\frac{n^d u(n)}{v(n)},\ c_s=\lim_{n\to\infty}n^{d+s}\left(\frac{u(n)}{v(n)}-\sum_{m=0}^{s-1}\frac{c_m}{n^{d+m}}\right)(\forall s\in\mathbf{N}).$$

证明　c_0 在前面的叙述中已经证明. 现假设 $\forall 0\leqslant j\leqslant s-1$，

$$c_j=\lim_{n\to\infty}n^{d+j}\left(\frac{u(n)}{v(n)}-\sum_{m=0}^{j-1}\frac{c_m}{n^{d+m}}\right)$$

且满足 $\sum\limits_{n=1}^{N}\dfrac{u(n)}{v(n)}=\sum\limits_{m=0}^{s-1}c_m\zeta_N(d+m)+\sum\limits_{n=1}^{N}R_{s-1}(n)$，$\deg R_{s-1}\leqslant -d-s$. 考察

$$c_s=\lim_{n\to\infty}n^{d+s}R_{s-1}(n),\ R_s(n)=R_{s-1}(n)-\frac{c_s}{n^{d+s}}.$$

显然有 $\deg R_s\leqslant -d-s-1$，$c_s=\lim\limits_{n\to\infty}n^{d+s}\left(\dfrac{u(n)}{v(n)}-\sum\limits_{m=0}^{s-1}\dfrac{c_m}{n^{d+m}}\right)$. 由归纳公理知，所证结论成立. □

我们不妨考察最简单的一种形式：$\sum\limits_{n=1}^{N}\dfrac{1}{n^p-a}$ 型数值计算问题. 现在给出对应的理论推导.

定理 2.9 设 $a \in \mathbf{R}$，$p > 0$，且 $\sqrt[p]{a} \notin \mathbf{N}^*$，则 $\forall s \in \mathbf{N}^*$，

$$\frac{1}{n^p - a} = \sum_{m=1}^{s} \frac{a^{m-1}}{n^{mp}} + \frac{a^s}{n^{sp}(n^p - a)}.$$

证明 假设 $\dfrac{1}{n^p - a} = \sum\limits_{m=1}^{s-1} \dfrac{a^{m-1}}{n^{mp}} + \dfrac{a^{s-1}}{n^{(s-1)p}(n^p - a)}$, 考察

$$\frac{a^{s-1}}{n^{(s-1)p}(n^p - a)} - \frac{a^{s-1}}{n^{sp}} = \frac{a^s}{n^{sp}(n^p - a)}.$$

$\therefore \dfrac{1}{n^p - a} = \sum\limits_{m=1}^{s} \dfrac{a^{m-1}}{n^{mp}} + \dfrac{a^s}{n^{sp}(n^p - a)}$. 由归纳公理知，所证结论成立. □

上述定理中，$\sqrt[p]{a} \notin \mathbf{N}^*$ 起到保证 $\sum\limits_{n=1}^{N} \dfrac{1}{n^p - a} \neq \infty$ 的作用. 上述定理给出了逼近 $\sum\limits_{n=1}^{N} \dfrac{1}{n^p - a}$ 型数值计算问题的方法，但是当 $|a| > 1$ 时，a^m 会以指数速度增长，导致除去余项的部分呈现出极大的数值不稳定性，这也导致最终对于余项的求和项数选取要求增大，降低了算法的效率. 因此，这种算法并不具有普适性. 进而我们可以考虑对于每一个给定的函数，重新进行一遍类似于 $\zeta_N(p)$ 中的推导. 下面通过具体例子给出过程.

例题 数值计算 $\sum\limits_{n=1}^{10^8} \dfrac{1}{n^2 + 2}$.

解 由于 $\sum\limits_{n=1}^{\infty} \dfrac{1}{n^2 + 2}$ 收敛，设这个极限为 α. 利用 O'Stolz 公式考察

$$\lim_{N\to\infty} N\left(\sum_{n=1}^{N} \frac{1}{n^2 + 2} - \alpha\right) = \lim_{N\to\infty} \frac{\frac{1}{N^2+2}}{-\frac{1}{N^2}} = -1.$$

重复上述过程，我们可以得到：当 $N \to \infty$ 时，

$$\sum_{n=1}^{N} \frac{1}{n^2 + 2} = \alpha - \frac{1}{N} + \frac{1}{2N^2} + \frac{1}{2N^3} - \frac{1}{N^4} - \frac{1}{10N^5} + \frac{2}{N^6} - \frac{17}{14N^7} - \frac{4}{N^8} + \frac{59}{10N^9} + \frac{8}{N^{10}} + o\left(\frac{1}{N^{10}}\right).$$

忽略小量，取 $N = 1000$，可以确定常数 α 的 36 位小数：

$$\alpha \approx 0.861028100573727922821332158518211121.$$

经检验，α 的绝对误差为 2.35×10^{-32}. 把 α 的近似值代入，有：

$$\sum_{n=1}^{10^8} \frac{1}{n^2 + 2} \approx 0.861028090573727972821332658518201121.$$

经检验，它的绝对误差为 2.35×10^{-32}. 事实上，最终的误差全部由 α 的误差引起，因为 $o(\frac{1}{N^{10}})$ 只能产生 10^{-80} 量级的误差.

经过上面的例题，我们从理论上依旧可以得到类似于 $\zeta_N(p)$ 的逼近，但是这个过程中取极限的操作是随着 u, v 的不同而不同的，我们无法类似于 $\zeta_N(p)$ 的情况来推导出一个通用的代数递推式. Mathematica 虽然是符号计算软件，但是它在用极限方法计算展开式高阶系数的时候，在速度上必然比不过常规的代数运算，并且每进行一步迭代计算系数的操作，所需时间也会越来越长. 由此可见，虽然对于一般有理级数的数值算法理论上已经成行，但是操作过程中还会遇到许多困难等待解决.

第三章　算法代码及实验结果

本章依照上一章理论推导的次序，给出 $\zeta_N(p)$ 在 $p=1$ 和 $p\neq 1$ 时的 Mathematica 实现代码、实验数据与结论分析. Mathematica 需要用到的代码语法如下[2]：

```
1  Zeta[x] (*Riemann-ζ函数*)
2  EulerGamma (*内置的Euler常数γ*)
3  f[x_]:= (*定义函数f，自变量为x*)
4  If[expr,s1,s2,s3] (*如果expr返回True，执行s1，如果expr返
       回False，执行s2，否则执行s3，s2、s3可以不写*)
5  IntegerQ[x] (*判断x是否为整数，是返回True，不是返回False
       *)
6  Positive[x] (*判断x是否为正数，是返回True，不是返回False
       *)
7  Floor[x] (*Gauss取整函数，向下取整*)
8  Sum[expr,{n,k,l}] (*对表达式expr中的变量n从k到l求和*)
9  N[expr,t] (*表达式expr保留t位十进制有效数字*)
10 Binomial[n,m] (*组合数C(n,m)*)
11 Print["expr"] (*直接打印expr*)
12 Dp[expr,{i,m,n,t}] (*从m到n，以t为步长，循环执行表达式
       expr*)
```

由于 Mathematica 在输入小数的时候，它只能保留到 16 位小数，为了实验中的高精度，我们不妨把所有输入数据取为能够表达为解析表达式的式子 (根号、分数等).

第一节　调和级数的部分和计算

一、实现代码

依照代码行数，分别作用如下：

1. 我们定义调和级数部分和数值计算函数，M 为求和项数、s 为有效数字位数. 在计算开始之前，首先判断：当 M，s 至少一个不是正整数时，返回第 6

行提示信息“求和项数和有效位数请输入正整数!”

2-3. 继续判断 M 的大小，如果较小的话，使用逐项相加、直接计算的方法，这样能够有效地提升算法稳定性及效率，避免过小的 M 无法抵消展开式系数增长速度过快的不稳定因素.

4. 如果 M 超过了阈值，取 k 为恰当的展开项数，算法中是假定展开式系数为 1 的情况下，求解出 k，即令 $M^{-k} \leqslant 10^{-s}$. 而实际计算中，这样的 k 偏小，会导致误差，因此取 2 倍. 此处对右侧的计算先取数值近似，再使用了 Gauss 取整函数，这样的做法会使得计算 k 的效率更高，也避免了 Mathematica 因为内部迭代算法的溢出报错. 当 k 过大时，迭代次数过多，导致程序很难在短时间内运行出结果，因此取 k 的上界为一个常数.

5. 核心算法，其中的 Euler 常数 EulerGamma 直接使用了系统中的常数，展开式系数也直接调用了系统中的 Riemann-ζ 函数 Zeta[x_].

```
1 HarmonicSeries[M_, s_] := If[IntegerQ[M] && Positive[M]
     && IntegerQ[s] && Positive[s],
2   If[M <= 100000,
3    N[Sum[1/n, {n, 1, M}], s],
4    k = Min[Floor[N[2 s*Log[10]/Log[M], 20]] + 1, 6000];
5    N[Log[M] + EulerGamma + 1/2/M + Sum[Zeta[-m]/M^(m + 1)
        , {m, 1, k}], s]],
6   Print["求和项数和有效位数请输入正整数！"]];
```

二、实验数据

求和项数 M	有效位数 s	算法用时 (秒)	Mathematica 用时 (秒)
123456	100	<0.1	2.8
123456	1000	<0.1	2.8
123456	5000	0.4	2.8
123456	10000	1.7	2.8
10^8	10000	1.0	>60
10^{20}	10000	0.3	<0.1
10^{10000}	10000	<0.1	<0.1

三、结论分析

通过试验可以发现，存在 M_0，在 $M \leqslant M_0$ 的时候，Mathematica 的固有算法使用的是直接求和法，并且得到不同精度所用的时间无差别；而本文算法收到有效数字的影响，主要是因为展开项数 k 与有效位数 s 成正相关，但是本文算法在 $M \leqslant M_0$ 的时候全面超过 Mathematica 的固有算法. 当 $M > M_0$ 以后，Mathematica 的固有算法可能使用了本文展开式的一阶逼近，这是因为一阶逼近对于 $M > M_0$ 的情况已经足够达到精度要求，而本文的算法依旧使用逼近法，故多出了判别条件与迭代的步骤，从而速度略慢于固有算法. 当 M 极大 (比如试验中的 10^{10000}) 时，二者又已经没有区别.

第二节　$p \neq 1$ 时 Riemann-ζ 函数的部分和计算

一、实现代码

依照代码行数，分别作用如下：

1. 定义 Riemann-ζ 函数 $\zeta(p)$ 级数部分和数值计算函数，K 为求和项数、r 为有效数字位数. 在计算开始之前，首先判断：当 K，r 不是正整数，或 $p \leqslant 0$ 时，返回第 11 行提示信息“求和项数和有效位数请输入正整数，且 ζ 函数的参数 $p > 0$!”

2-3. 继续判断 K 的大小，如果较小的话，使用逐项相加、直接计算的方法，这样能够有效地提升算法稳定性及效率，避免过小的 K 无法抵消展开式系数增长速度过快的不稳定因素.

4-5. 如果 K 超过了阈值，判断 p 是否为 1，如果为 1，则调用调和级数的部分和 HarmonicSeries 函数.

6. 如果 $p \neq 1$，取 l 为恰当的展开项数，算法中是假定展开式系数为 1 的情况下，求解出 l，即令 $K^{1-p-2l} \leqslant 10^{-s}$. 而实际计算中，这样的 l 偏小，会导致误差，因此取 2 倍. 此处对右侧的计算先取数值近似，再使用了 Gauss 取整函数，这样的做法会使得计算 l 的效率更高，也避免了 Mathematica 因为内部迭代算法的溢出报错. 当 l 过大时，迭代次数过多，导致程序很难在短时间内运行出结果，因此取 l 的上界为一个常数.

7. 清理 coef 列表，避免多次调用不同 p 值下的 ZetaSeries 函数时产生冲突.

8-10. 结合第二章的理论，用递归的方式定义 coef 数列，用来表示展开式系

数 $c_m^{(p)}$.

11. 核心算法.

```
ZetaSeries[p_, K_, r_] :=If[IntegerQ[K] && Positive[K] &&
    IntegerQ[r] && Positive[r] && Positive[p],
 If[K <= 10000,
  N[Sum[1/n^p, {n, 1, K}], r],
  If[p == 1,
   HarmonicSeries[K, r],
   l = Min[Floor[N[r*Log[10]/Log[K] - p + 1, 20]] + 1,
      100];
   Clear[coef];
   coef = Table[0, {n, 1, l}];
   coef[[1]] = -p/12;
   Do[coef[[j]] = (Binomial[1 - p, 2 j + 1]/(p - 1) +
      Binomial[-p, 2 j]/2 - Sum[coef[[i]]*Binomial[-p - 2
       i + 1, 2 j - 2 i + 1], {i, 1, j - 1}])/(-p - 2 j +
       1);, {j, 2, l}];
   N[K^(1 - p)/(1 - p) + Zeta[p] + 1/2/K^p + Sum[coef[[m
      ]]/K^(2 m + p - 1), {m, 1, l}], r]]],
  Print["求和项数和有效位数请输入正整数，且ζ函数的参数p>0
    ! "]];
```

二、实验数据

下面通过取具有不同特点的几组 p 值，得到若干组实验数据.

（1）$p=\frac{1}{12}$

求和项数 K	有效位数 r	算法用时 (秒)	Mathematica 用时 (秒)
12345	100	<0.1	0.7
12345	1000	0.3	0.7
12345	5000	2.7	0.9
12345	10000	35.2	2.7
10^{11}	10000	8.3	27.8
10^{50}	10000	2.6	27.8
10^{10000}	10000	<0.1	27.8

（2）$p=2$

求和项数 K	有效位数 r	算法用时 (秒)	Mathematica 用时 (秒)
100000	100	<0.1	3.7
100000	1000	<0.1	3.7
100000	5000	0.4	3.7
100000	10000	0.6	3.7
1000000	100	<0.1	>60
1000000	1000	<0.1	>60
1000000	5000	0.3	>60
1000000	10000	0.5	>60
10^{20}	100	<0.1	<0.1
10^{20}	1000	<0.1	<0.1
10^{20}	5000	<0.1	<0.1
10^{20}	10000	0.3	<0.1

（3）$p=\frac{10356}{359}$

求和项数 K	有效位数 r	算法用时 (秒)	Mathematica 用时 (秒)
10001	100	<0.1	0.8
10001	1000	5.0	2.0
10001	5000	19.6	8.2
10001	10000	>60	20.4
100000	100	<0.1	8.3
100000	1000	2.7	21.3
100000	5000	18.3	>60
100000	10000	40.2	>60
10^{10}	1000	<0.1	<0.1
10^{20}	1000	<0.1	<0.1

（4）$p=235$

求和项数 K	有效位数 r	算法用时 (秒)	Mathematica 用时 (秒)
100000	100	0.3	>60
100000	1000	0.3	>60
100000	5000	0.8	>60
100000	10000	3.7	>60
10^{8}	10000	<0.1	30 秒开始报错，误差较大
10^{20}	1000	<0.1	12 秒开始报错，误差较大

三、结论分析

通过实验我们得知，决定算法效率的主要因素是迭代次数 k，而它与求和项数 K 成负相关，与求和精度 r 成正相关. 对于不同的 p 值，都存在 K 的一个区域，在这片区域上，本文的算法要远远快于 Mathematica 的固有算法. 固有算法在 K 不太大的时候，直接求和计算，而本文提出的算法是通过考察 $K\to\infty$ 的极限状态，反向逼近有限项，因而会出现 K 越大，速度越快的情况. 不过，Mathematica 的固有算法，在 K 比较大的时候，也会使用本文的一阶逼近，但是如果把有效数字的位数要求提高，由于本文算法考察了详尽的高阶逼近，因此 Mathematica 的固有算法在精度上就比不上本文提出的算法了.

从 $p=\frac{1}{12}$ 的情况可以看出，当 p 比较小的时候，算法对于迭代次数 k 非常敏感；从 $p=2$ 的情况可以看出，算法在 p 接近 1(经过测试，大概在 $p\in[\frac{1}{2},3]$ 之间)的时候，各种情况下都表现出极为优异的性能；从 $p=235$ 的情况可以看出，算法在 p 极大的情况下，虽然偶尔会耗时会超过 3 秒，但是相比 Mathematica 的固有算法依旧具有不小的优势，并且在 Mathematica 固有算法出现溢出报错、精度降低的情况下，本文算法依旧能够保持高精度的准确计算. 特别地，在 $p=\frac{10356}{359}$ 的时候，本文算法的优势大大削弱，甚至有些情况不及 Mathematica 固有算法，这是因为 p 为相对复杂的分数，而 Mathematica 的固有算法对于分数的符号计算做了优化，而本算法中的迭代模块对于 p 是分数的情况未做优化，迭代算法本身更会使 p 为一个复杂分数情况下的复杂度大大增加.

本小节目前的算法实际上是与给定要求的精度 r 位有效数字有一定误差的，这是因为所取的迭代算法没有设置 l 的判别条件，只是粗略地给出了一个范围，这是因为如果给出严格的判别条件，会导致 l 比较大，以至于算法所需时间极长，这主要是因为目前的迭代算法没有做任何优化. 不过，这套没有优化的代码，在计算不超过几百位的精度的情况下在性能方面还是几乎全面超过 Mathematica 已有的算法的，由此可见，只要对于算法的迭代部分做更好的优化，此算法的潜力是极其巨大的，不过这也需要更多艰难的工作.

第四章　总结与展望

本文的算法在思想上是没有本质的困难，先通过 Stolz 公式给出 $\zeta_N(p)$ 的部分和渐近展开式，并将系数的递推式求解出来，之后便是设计算法加以实际运用. 不过在理论推导与编程实验的过程中，不仅有相对于已有算法优越之处，还有诸多悬而未解的问题，本章将这些问题罗列于此，并对算法未来的发展潜力做出展望.

算法的理论思想为递推与逼近思想，不过又与以往众多算法的思路有所不同. 大多数算法是直接正面逼近，而本文算法反其道而行之，先使用分析学的手段研究 $N \to \infty$ 时的极限情况，再减掉 $N+1$ 到 ∞ 的余项. 它增大了求极限的工作量，但极大地减少了正面数值计算的工作量，因此，本文的算法在面对 N 很大，要求精度较高的情况，较已有算法的优势更为明显.

此外，在研究调和级数的部分和展开系数时，通过大量数据观测发现 $c_m(1) = \zeta(1-2m)$. 更一般地，还观测发现 $c_m^{(p)}$ 与 $\zeta(1-2m)$ 之比可能与 $\sum_{k=1}^{m} k^p$ 有关，预计这需要涉及到大量 Riemann-ζ 函数、Euler-Γ 函数以及它们之间的对偶性有关的知识，远远超出了论文研究的范围. 一旦 $c_m^{(p)}$ 与 $\zeta(1-2m)$ 之间的关系被找到，本文的算法就可以有效地避免迭代，而通过直接调用 Riemann-ζ 函数进行计算，这将会使得现有算法的复杂度再次减小若干个数量级.

在讨论一般的有理级数数值计算问题的时候，我们通过使用已经研究过的 $\zeta_N(p)$ 进行逐次逼近，使得余项的有理级数次数越来越小，以至于最终可以在仅计算余项级数的前几千项和的情况下达到要求精度. 但是 $\sum_{n=1}^{N} \frac{1}{n^p - a}$ 型在 $|a| > 1$ 的时候，就已经指出此算法的不稳定性，因此我们需要寻找新的想法，即对于一般的有理级数，仍旧类似于逼近 $\zeta_N(p)$ 的方法对它直接使用 O'Stolz 公式不断渐近. 这种理论我们已经建立，但是想要编程实现，依旧存在着算法优化上的困难.

逼近一般有理级数的第一种思路虽然宣告失败，但是它利用了已有的特殊形式 $\zeta_N(p)$ 来逼近更一般的形式，这种化归转化思想不仅可以应用于此，还可以拓展到许多其他种类的数值计算问题：如果我们能够在一类数值计算问题中，找到一组稠密的项 (比如本文中的 $\zeta_N(p)$，它们可以逼近所有有理级数)，对于这些稠密的项，我们可以找到高效的算法，或者在它们可以取有限多个的时候，把它们的值提前运算并贮存起来，这样就能够得到该类数值计算问题的一般算法. 例

如，本文虽然只对有理级数给出了算法思路，但是对于分子分母均为有限个关于 n 的幂函数的线性组合形式的数列，它对应的级数部分和问题依旧可以利用定义 1.3 来不断逼近得到. 因此，它依旧是有价值的.

在算法实现部分，我们没能解决迭代次数较大时，算法效率不高的问题. 也正是如此，我们目前只能通过牺牲一些精度 (比如 1000 位以上) 来减少迭代次数的方法来保证运算速度.

总体来说，本文的算法研究过程中，提出了如下待解决的问题:

(1) $c_m^{(p)}$ 与 $\zeta(1-2m)$ 之间有着怎样的关系？

(2) 已知 1 是 Riemann-ζ 函数的 1 级极点，但是至今为止，$\lim\limits_{s\to 1}(1-s)\zeta(s)$ 依旧没有结果，是否可以通过此算法，从数值的角度为此问题的研究提供思路，进而推动 Riemann 猜想有关问题的发展？

(3) 能否研究出迭代次数 k 与求和项数 N、有效位数 s 之间更精确的关系？

(4) 数值积分通常会使用 Riemann 和进行逼近，而 Riemann 和在积分函数不太复杂的情况下，就会近似为有理级数. 当 Riemann 和所取的分割充分精细的时候，目前主流的算法复杂度一般都与分割的份数成正相关. 如果将本文的思想应用上去，是否可以完美地解决许多数值积分问题？

(5) 如何设计程序，能够自动判断给定有理函数的次数 $\deg\dfrac{u(x)}{v(x)}$？

(6) 如何对一般有理函数逼近中迭代过程进行优化以提升效率？

在理论上，本文的研究提出了上述关于 Riemann-ζ 函数的想法，并且能够从数值的角度给出 Riemann-ζ 函数一些更直观的性质，甚至可能为 Riemann 猜想有关问题提供研究方向，这深刻地体现出了基础数学不仅能够为应用数学提供理论支持，应用数学的研究也会反哺基础数学的发展. 在应用上，从工程应用的角度来讲，100 位有效数字基本上已经绰绰有余，而 100 位以内的精度，在许多固有算法效率较低，甚至无法达到精度要求的情况下，本文算法依旧能够完美的给出结果，因此本算法在工程应用中也具有比较重要的意义. 目前算法只是初步研究阶段，如果能把上述悬而未决的问题解决，算法的效率与运算必将继续大幅提高.

参 考 文 献

[1] 常庚哲，史济怀. 数学分析教程: 上册. 合肥: 中国科学技术大学出版社, 2012.

[2] 张韵华，王新茂. Mathematica 7 实用教程. 合肥: 中国科学技术大学出版社, 2011.