Master's Project

COMPUTATIONAL METHODS FOR THE RIEMANN ZETA FUNCTION

under the guidance of
Prof. Frank Massey
Department of Mathematics and Statistics
The University of Michigan − Dearborn

Matthew Kehoe
mskehoe@umd.umich.edu

In partial fulfillment of the requirements
for the degree of
MASTER of SCIENCE
in Applied and Computational Mathematics

December 19, 2015

Acknowledgments

I would like to thank my supervisor, Dr. Massey, for his help and support while completing the project. I would also like to thank all of the professors at the University of Michigan − Dearborn who continually teach mathematics. Many authors, including Andrew Odlyzko, Glen Pugh, Ken Takusagawa, Harold Edwards, and Aleksandar Ivic assisted me by writing articles and books about the Riemann zeta function. I am also grateful to the online communities of Stack Overflow and Mathematics Stack Exchange. Both of these communities assisted me in my studies and are helpful for all students.

Abstract

The purpose of this project is to explore different computational methods for the Riemann zeta function. Modern analysis shows that a variety of different methods have been used to compute values for the zeta function and its zeroes. I am looking to develop multiple programs that will find non-trivial zeroes and specific function values. All computer implementations have been done inside the Java programming language. One could easily transfer the programs written inside the appendix to their preferred programming language. My main motivation for writing these programs is because I am interested in learning more about the mathematics behind the zeta function and its zeroes.

The zeta function is an extremely important part of mathematics and is formally denoted by $\zeta(s)$. The Riemann hypothesis states that all non-trivial zeroes of $\zeta(s)$ lie on the critical line where $\Re(s) = 1/2$. It is important to note that the critical strip is the region between $0 < \Re(s) < 1$ that contains all non-trivial zeroes. Modern research and advanced algorithms have not been able to disprove the hypothesis. Recent research has produced what is known as the Odlyzko−Schönhage algorithm. The algorithm has been used to numerically verify the first 10^{13} zeroes of the Riemann hypothesis.

The first three programs calculate specific function values for $\zeta(s)$. My first program directly calculates $\zeta(s)$ for any value of s provided $s \in \mathbb{R} \setminus \{1\}$. This program applies the standard form of $\zeta(s)$ alongside recursion through the functional equation. The main difficulty deals with implementing a recursive method which will handle all values calculated between $-1 \leq \Re(s) \leq 2$. I used an alternative method known as the Cauchy−Schlömilch transformation. The second program handles computation of $\zeta(s)$ for all $s \in \mathbb{C} \setminus \{1\}$. A summation formula known as the Abel−Plana formula calculates the zeta function for $s \in \mathbb{C}$. An integral estimate is done through what is known as adaptive quadrature. The third program calculates complex values for $\zeta(s)$ through the Dirichlet series summation formula.

The fourth program applies the well known Riemann−Siegel formula. Prior to the work done inside the Odlyzko−Schönhage algorithm, this was the method of choice for finding zeroes of $\zeta(s)$. The algorithm itself is heavily dependent on locating sign changes for the Riemann−Siegel Z function which is known as $Z(t)$. The sign changes of $Z(t)$ directly relate to zeroes on the critical line. Loosely speaking, one can derive this formula from a direct relationship that was originally developed by Siegel. The relationship shows that $Z(t) = e^{i\theta(t)}\zeta\left(\frac{1}{2} + it\right)$. I have also decided to investigate what is known as Lehmer's phenomenon. This was found by D. H. Lehmer in 1956, the phenomenon references when two zeros of the zeta function are so close together that it is unusually difficult to find a sign change between them.

The final program calculates the zeta function for integer values. Two summation formulas are referenced that calculate $\zeta(s)$ for even and odd values through the use of Bernoulli numbers. Positive even values are calculated by a summation formula originally developed by Leonard Euler. The formula states that $\zeta(2n) = (-1)^{n+1}B_{2n}(2\pi)^{2n}/\ 2(2n)!$ and is used for calculating values of $\zeta(s)$ where $s \equiv 0\, mod\, 2$. Negative odd values are calculated through a similar summation formula which states that $\zeta(-n) = -B_{n+1}/(n+1)$ for all $n \geq 1$. There is no known formula for calculating $\zeta(2n+1)$ where $n > 1$. A detailed analysis of all Java programs is discussed inside the appendix.

While writing computer programs is enjoyable, it is important to point out an important proof made by G.H. Hardy in 1915. This is known as Hardy's theorem and states that an infinite amount of zeroes lie on the critical line where $\Re(s) = 1/2$. Considering we have already found 10^{13} zeroes, it is highly unlikely modern computers will disprove the hypothesis.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

The zeta function was originally introduced by Bernhard Riemann in his classic 8-paper paper [33] labeled "On the Number of Primes Less Than a Given Magnitude." In this paper, Riemann expands upon work done previously by Euler, Gauss, and Dirichlet. He starts off by considering the Euler Product Formula.

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_{p \in P} \frac{1}{1 - p^{-s}} \qquad \forall\ \Re(s) > 1$$

which can also be written as

$$\zeta(s) = \sum_{n=1}^{\infty} n^{-s} = \prod_{p \in P} \left(1 - p^{-s}\right)^{-1}$$

The RHS of the above equation is known as Euler's identity. Taking the logarithm of both sides and then differentiating produces

$$-\frac{\zeta'(s)}{\zeta(s)} = \sum_{p \in P} \frac{\log p}{p^s - 1} = \sum_{p \in P} \sum_{n=1}^{\infty} (\log p)\, p^{-ns}$$

An important trick is to recognize that

$$\prod_{p} \frac{1}{\left(1 - \frac{1}{p^s}\right)} = \prod_{p} \left(1 + \frac{1}{p^s} + \frac{1}{(p^2)^s} + \frac{1}{(p^3)^s} + \cdots\right) = \sum_{n} \frac{1}{(p_1^{n_1} p_2^{n_2} \cdots p_r^{n_r})^s}$$

because $p_1, p_2, \cdots, p_r$ are distinct primes and $n_1, n_2, \cdots, n_r$ are natural numbers (see Appendix A). Euler originally used the fundamental theorem of arithmetic when writing the infinite sum for $\sum n^{-s}$.

Another form of the equation above references

$$-\frac{\zeta'(s)}{\zeta(s)} = \sum_{n=1}^{\infty} \Lambda(n)\, n^{-s}$$

$\Lambda(n)$ is known as the von Mangoldt function and is defined as

$$\Lambda(n) = \begin{cases} log\, p, & n = p^k \quad (p\, is\, prime,\ k \geq 1) \\ 0, & otherwise \end{cases}$$

To match what is used in modern notation, I will introduce the integral form of the gamma function.

$$\Gamma(t) = \int_0^{\infty} x^{t-1} e^{-x}\, dx \qquad \forall\ \Re(t) > 0$$

It is straightforward [16] to derive the following relationship between the zeta function and the gamma function.

$$\zeta(s) = \frac{1}{\Gamma(s)} \int_0^\infty \frac{x^{s-1}}{e^x - 1} \mathrm{d}x \qquad \forall\ \Re(s) > 1$$

By applying methods from the theory of complex variables, Riemann was able to derive what is now known as the functional equation. This equation [20] is

$$\zeta(s) = 2^s \pi^{s-1}\ \sin\left(\frac{\pi s}{2}\right)\ \Gamma(1-s)\ \zeta(1-s)$$

There are many different forms of the functional equation. Another form [20] is

$$\pi^{-(1-s)/2}\Gamma\left(\frac{1-s}{2}\right)\zeta(1-s) = \pi^{-s/2}\Gamma\left(\frac{s}{2}\right)\zeta(s).$$

Which shows that the equation itself is unchanged by substituting $s \to (1-s)$. A third form [20] of the functional equation is

$$\zeta(s) = \chi(s)\zeta(1-s)\ \ where\ \ \chi(s) = \frac{(2\pi)^s}{2\Gamma(s)cos(\frac{\pi s}{2})}$$

By defining [20] $\Psi(x)$ as $\Psi(x) = x - \lfloor x \rfloor - \frac{1}{2}$, one may write

$$\zeta(s) = -s \int_0^\infty \Psi(x) x^{-s-1} dx \qquad \forall\ \ -1 < \Re(s) < 0$$

The first form of the functional equation can be derived through a Fourier expansion. By the Fourier expansion, $\Psi(x) = -\sum_{n=1}^{\infty}(n\pi)^{-1} sin(2n\pi x)$, the equation above can be [20] expanded to

$$\begin{aligned}
\zeta(s) &= \pi^{-1} s \sum_{n=1}^{\infty} n^{-1} \int_0^\infty sin(2n\pi x) x^{-s-1} dx \qquad \forall\ \ -1 < \Re(s) < 0 \\
&= \pi^{-1} s \sum_{n=1}^{\infty} (2n\pi)^s n^{-1} \int_0^\infty (sin\ y) y^{-s-1} dy \\
&= \pi^{-1} s (2\pi)^s \{-\Gamma(-s)\} sin(\frac{\pi s}{2}) \zeta(1-s)
\end{aligned}$$

which reduces to

$$\zeta(s) = 2^s \pi^{s-1}\ \sin\left(\frac{\pi s}{2}\right)\ \Gamma(1-s)\ \zeta(1-s)$$

In zeta-function theory, $N(T)$ is known as the number of zeroes for $\zeta(s)$ in the region of $0 < \Re(s) < 1$ and height of $0 < t \leq T$. It is also common to denote $N_0(T)$ as the number of zeroes in the form $\rho = \frac{1}{2} + it$ for which $0 < t \leq T$. Riemann had originally estimated the number of zeroes [33] between 0 and T as

$$N(T) = \frac{T}{2\pi} \log\left(\frac{T}{2\pi}\right) - \frac{T}{2\pi} + \mathcal{O}(\log T)$$

A change was made by Hans von Mangoldt who compacted the estimate to

$$N(T) = \frac{T}{2\pi} \log\left(\frac{T}{2\pi e}\right) + \mathcal{O}(\log T)$$

In 1915, G.H. Hardy showed that $N_0(T) \to \infty$ as $T \to \infty$. He revised Riemann's estimate to form

$$N_0(T) >\ CT \qquad (C > 0, T \geq T_0)$$

Another improvement was made in 1942 by A. Selberg which states

$$N_0(T) > \ CT \ \log T \qquad (C > 0, T \geq T_0)$$

Norman Levinson also increased progress in 1974 by forming ($C = \frac{1}{3}$)

$$N_0(T+U) - N_0(T) \ > \ C(N(T+U) - N(T))$$
$$U = TL^{-10}, \ L = \log(T/2\pi)$$

Further improvements are being researched today. The precise location of zeroes for $\zeta(s)$ remains one of the greatest unsolved problems in analytic number theory. The Riemann hypothesis itself may be thought of as $N(T) = N_0(T)$ for all $T > 0$, where $N_0(T)$ represents the number of zeroes for $\zeta(s)$ in the form $\rho = \frac{1}{2} + it$, $0 < t \leq T$.

The Riemann zeta function is part of a larger group of functions known as Dirichlet L-functions. The most common of these can be [6] represented by

$$L(s, \chi) = \sum_{n=1}^{\infty} \chi(n) n^{-s} \qquad \forall \ \Re(s) > 1$$

where χ is known as a Dirichlet character. By letting $s = \sigma + it$, it is possible to show that $L(s, \chi)$ is absolutely convergent for $\sigma > 1$. Similar to the zeta function, $L(s, \chi)$ [6] can be extended to the entire complex plane as a meromorphic function through the use of analytic continuation. Dirichlet L-functions are part of a larger set of functions known as Dirichlet series. An ordinary Dirichlet series references

$$\sum_{n=1}^{\infty} \frac{a_n}{n^s} \qquad \forall \ \sigma > \sigma_a$$

provided that s is complex and a_n is an arithmetical function. The σ_a above is known as the abscissa of absolute convergence. The ordinary Dirichlet series is absolutely convergent [6] for $\sigma > \sigma_a$. An important Dirichlet series is known as the Dirichlet eta function. It is common to denote this as the alternating zeta function, by which

$$\eta(s) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n^s} \qquad \forall \ \Re(s) > 0$$

A direct relationship [4] can be formed between $\zeta(s)$ and $\eta(s)$

$$\eta(s) = \left(1 - 2^{1-s}\right) \zeta(s)$$

The Mellin transform [31] is an integral transform that is defined as

$$\{\mathcal{M}f\}(s) = \varphi(s) = \int_0^{\infty} x^{s-1} f(x) dx$$

Through the use of the gamma function [16], one can represent $\eta(s)$ as a Mellin transform

$$\eta(s) = \frac{1}{\Gamma(s)} \int_0^{\infty} \frac{x^{s-1}}{e^x + 1} dx \qquad \forall \ \Re(s) > 0$$

This is of interest to me due to the derivation of the Cauchy−Schlömilch transformation. Starting at

$$\Gamma(s)\eta(s) = \int_0^\infty \frac{x^{s-1}}{e^x+1}dx$$

one can apply integration by parts [2] to form

$$2^{1-s}\,\Gamma(s+1)\,\eta(s) = 2\int_0^\infty \frac{x^{2s+1}}{\cosh^2(x^2)}\,dx = \int_0^\infty \frac{t^s}{\cosh^2(t)}\,dt \qquad \forall\ \Re(s) > -1$$

While this is important, it is not the only way to arrive at the desired outcome for the Cauchy−Schlömilch transformation. Many of these equations can be used to create computer programs which calculate values for $\zeta(s)$. To my own surprise, the main difficulty I ran into while creating these programs was not due to any limitation of mathematical tools. Nearly every issue I encountered was due to the calculation of floating point arithmetic inside the Java programming language. I will go into further details below. My first implementation dealt with calculating $\zeta(s)$ for all $s \in \mathbb{R} \setminus \{1\}$ through the Cauchy−Schlömilch transformation.

Chapter 2

Theory For Zeta Programs

2.1 The Cauchy-Schlömilch transformation and zeta(s)

My first implementation deals with a direct implementation of $\zeta(s)$ provided $s \in \mathbb{R}$. The starting point for this analysis stems from the well known equation

$$\zeta(s) = \sum_{n=1}^{\infty} n^{-s} \qquad \forall\ \Re(s) > 1$$

Using this equation, one can in principle implement a computer program to find all values of $\zeta(s)$ provided $\Re(s) > 1$. The first step deals with applying the integral test from calculus. The integral test is a direct way to analyze convergence. It can be shown that x^{-s} is positive and decreasing for all $s > 1$. Setting $a_n = f(n)$ and $x > 1$, we can directly compare convergence between an infinite sum and improper integral by

$$\sum_{n=1}^{\infty} a_n \; converges \iff \int_1^{\infty} f(x)\, dx \; is \; convergent$$

The infinite series can then be estimated by a remainder term. It is necessary to define S_n as the nth partial sum of the series and S as the actual value of the series. Denoting R_n as the remainder term, we can write

$$R_n = S - S_n$$

Because the function is continuous, positive, and decreasing, we can fully utilize the integral test. Setting $x \geq n$, we can bound the remainder term between the sum of two improper integrals.

$$\int_{n+1}^{\infty} f(x)\, dx \leq \ R_n \ \leq \int_n^{\infty} f(x)\, dx$$

These steps are necessary because they allow the computer to approximate the value of the remainder by computing multiple integral approximations. It is not difficult to show that the remainder takes the form

$$R_n \leq \frac{1}{s-1} n^{s-1}$$

This is all that is necessary to compute values for $\zeta(s)$ where $\Re(s) > 1$. However, I decided to use the summation formula only when $\Re(s) \geq 2$. It is not practical to use the summation formula for $\zeta(s)$ near $1 < \Re(s) < 2$ because of the number of necessary terms to sum. My implementation continuously evaluates the remainder term and then stops the computation when the relative error is less than 2×10^{-7}. To find values for $\zeta(s)$ where $\Re(s) < 1$, it is necessary to reference the functional equation. Recall that this equation takes the following form

$$\zeta(s) = 2^s \pi^{s-1} \ \sin\left(\frac{\pi s}{2}\right) \ \Gamma(1-s)\ \zeta(1-s)$$

Through the use of an alternative equation for $\zeta(s)$, the functional equation can be shown to provide analytic continuation for all $s \neq 1$ and can be used to find values of $\zeta(s)$ for all real values of s. Using this equation, we can calculate values for $\zeta(s)$ in the critical strip of $0 < \Re(s) < 1$ and when $\Re(s) < 0$. To this extent, I introduce what is known as the Cauchy−Schlömilch transformation. In terms of the zeta function, this can be directly stated [26] as

$$\zeta(s) = \frac{2^{s-1}}{(1-2^{1-s})\Gamma(s+1)} \int_0^\infty \frac{t^s}{\cosh^2(t)}\, dt \qquad \forall\ \Re(s) > -1$$

Further information regarding the derivation of the equation can be found inside the appendix. Through the use of the functional equation and the Cauchy−Schlömilch transformation, I was able to write a recursive method that will calculate $\zeta(s)$ for all $s \in \mathbb{R} \setminus \{1\}$. My program uses a method which specifically targets values where recursion is necessary. It is possible to use other integral transformations to calculate $\zeta(s)$, the most interesting one I was able to find was the Cauchy−Schlömilch transformation.

The recursive method that I created is dependent on the following four conditions:

• If $\Re(s) \geq 2$, use the infinite sum defined by $\sum_{n=1}^{\infty} n^{-s}$. Stop calculating future values for $\zeta(s)$ when the relative error between two terms is less than 2×10^{-7}.

• Else if $0 \leq \Re(s) < 2$, calculate $\zeta(s)$ by the Cauchy−Schlömilch transformation. It is important to note that this does cause performance issues. The function $t^s/\cosh^2(t)$ is not differentiable with respect to t at $t = 0$ for $0 \leq s < 1$. The evaluation of $\zeta(s)$ near s = 0 takes about 3 seconds.

• Else if $-141 \leq \Re(s) < 0$, calculate $\zeta(s)$ by recursion through the functional equation. The Lanczos approximation is used to evaluate $\Gamma(1-s)$.

• Else compute $\zeta(s)$ by recursion through the functional equation. The Stirling approximation is used to evaluate $\Gamma(1-s)$.

The implementation of the Cauchy−Schlömilch transformation creates a certain amount of error in the approximation for $\int_0^\infty t^s/\cosh^2(t)\, dt$. The value of the integral can be directly computed through Simpson's method. Simpson's method is commonly used in numerical integration and approximates a definite integral through a standard partition of $\Delta x = \frac{b-a}{n}$. Since $t^s/\cosh^2(t)$ is continuous in t from $0 \leq s < 2$, we can apply a direct integral approximation from $[a, b]$. The standard implementation of Simpson's method can be shown [13] to take the following form

$$\int_a^b f(x)\, dx \approx \tfrac{h}{3}\left[f(x_0) + 2\sum_{j=1}^{n/2-1} f(x_{2j}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(x_n)\right]$$

Denoting ES_n as the error function for Simpson's method, one can write

$$|ES_n| \ \leq\ \tfrac{h^4}{180}(b-a) \max_{\xi \in [a,b]} |f^{(4)}(\xi)| \ \ for\ \ a \leq \xi \leq b$$

In order to approximate the error involved with Simpson's method, it is necessary to calculate up to the fourth derivative. The $t^s/\cosh^2(t)$ term is the integrand of the integral $\int_0^\infty t^s/\cosh^2(t)\, dt$. So, it is necessary to calculate the fourth derivative of $t^s/\cosh^2(t)$. The $dfdt4$ command in Mathematica produces the following result

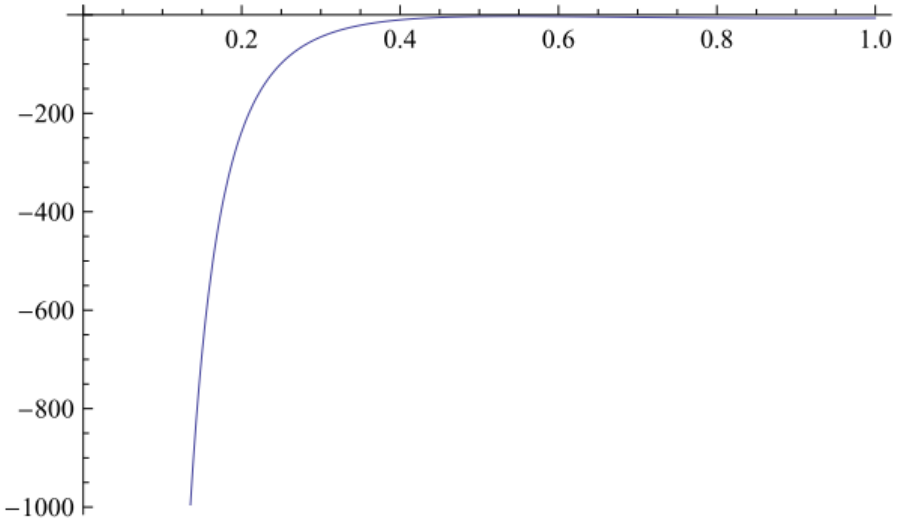
$$\begin{aligned}\frac{\partial^4}{\partial t^4}\left(\frac{t^s}{\cosh^2(t)}\right) &= (s-3)(s-2)(s-1)\, s\, t^{s-4} \operatorname{sech}^2(t) - \\ &\quad 8(s-2)(s-1)\, s\, t^{s-3} \tanh(t) \operatorname{sech}^2(t) + 4\, s\, t^{s-1}\left(16 \tanh(t) \operatorname{sech}^4(t) - 8 \tanh^3(t) \operatorname{sech}^2(t)\right) + \\ &\quad 6(s-1)\, s\, t^{s-2}\left(4 \tanh^2(t) \operatorname{sech}^2(t) - 2 \operatorname{sech}^4(t)\right) + \\ &\quad t^s\left(16 \operatorname{sech}^6(t) - 88 \tanh^2(t) \operatorname{sech}^4(t) + 16 \tanh^4(t) \operatorname{sech}^2(t)\right)\end{aligned}$$

The expanded form of the fourth derivative would be difficult to implement inside my program. To make matters worse, the fourth derivative requires Java to evaluate about twenty different functions. An alternate form for the equation on the RHS is

$$
\begin{aligned}
&s^4\, t^{s-4}\, \operatorname{sech}^2(t) + s^3 \operatorname{sech}^2(t)\left(-6\, t^{s-4} - 8\, t^{s-3} \tanh(t)\right) + \\
&s^2\left(\frac{96\, e^{2t}\left(-4\, e^{2t} + e^{4t} + 1\right) t^{s-2}}{\left(e^{2t}+1\right)^4} + \operatorname{sech}^2(t)\left(11\, t^{s-4} + 24\, t^{s-3} \tanh(t)\right)\right) + \\
&s\left(e^{2t}\left(-\frac{96\left(-4\, e^{2t} + e^{4t} + 1\right) t^{s-2}}{\left(e^{2t}+1\right)^4} - \frac{128\left(e^{2t}-1\right)\left(-10\, e^{2t} + e^{4t} + 1\right) t^{s-1}}{\left(e^{2t}+1\right)^5}\right) + \right. \\
&\qquad \left. \operatorname{sech}^2(t)\left(-6\, t^{s-4} - 16\, t^{s-3} \tanh(t)\right)\right) + \\
&t^s\left(\frac{16\left(e^{2t}-1\right)^4 \operatorname{sech}^2(t)}{\left(e^{2t}+1\right)^4} - \frac{128\, e^{4t}\left(-30\, e^{2t} + 11\, e^{4t} + 11\right)}{\left(e^{2t}+1\right)^6}\right)
\end{aligned}
$$

Thankfully, there is an alternative solution. There is no need to reference either form of the fourth derivative inside Java. The main problem deals with the calculation of the fourth derivative near $t = 0$ when $0 \le s < 2$. Directly plotting the fourth derivative in Mathematica with $s = 1/2$ shows a vertical asymptote at $t = 0$.

Figure 2.1: The fourth derivative of $t^s/\cosh^2(t)$ near $t = 0$



Assuming that $\int_0^{20} t^s/\cosh^2(t)\, dt$ is a correct integral approximation for $\int_0^{\infty} t^s/\cosh^2(t)\, dt$, the improper integral can be broken into the sum of two integrals for $\int_0^1 t^s/\cosh^2(t)\, dt + \int_1^{20} t^s/\cosh^2(t)\, dt$. I then used Mathematica to find the number of subintervals for a relative error of 2×10^{-7}. Simpson's rule worked fine for evaluating $\int_1^{20} t^s/\cosh^2(t)\, dt$; at $s = 2$ it took a total of $n = 256$ subintervals. At $s = 1/2$, the evaluation of $\int_0^1 t^s/\cosh^2(t)\, dt$ is significantly worse and took a total of $n = 8192$ subintervals. I decided to work around this by creating a power series for $\operatorname{sech}^2(t)$. This can be evaluated by

$$
\begin{aligned}
\int_0^1 t^{1/2}\operatorname{sech}^2(t)dt &= \int_0^1 t^{1/2}\left[\operatorname{sech}^2(t) - 1 + t^2 - \frac{2t^4}{3}\right]dt + \int_0^1 t^{1/2}\left[1 - t^2 + \frac{2t^4}{3}\right]dt \\
&= \int_0^1 t^{1/2}\left[\operatorname{sech}^2(t) - 1 + t^2 - \frac{2t^4}{3}\right]dt + \int_0^1 \left[t^{1/2} - t^{5/2} + \frac{2t^{9/2}}{3}\right]dt \\
&= \int_0^1 t^{1/2}\left[\operatorname{sech}^2(t) - 1 + t^2 - \frac{2t^4}{3}\right]dt + \left[\frac{2}{3}\, t^{3/2} - \frac{2}{7}\, t^{7/2} + \frac{4t^{11/2}}{27}\right]\Bigg|_0^1 \\
&= \int_0^1 t^{1/2}\left[\operatorname{sech}^2(t) - 1 + t^2 - \frac{2t^4}{3}\right]dt + \left[\frac{2}{3} - \frac{2}{7} + \frac{4}{27}\right] \\
&= \int_0^1 t^{1/2}\left[\operatorname{sech}^2(t) - 1 + t^2 - \frac{2t^4}{3}\right]dt + \left[\frac{100}{189}\right]
\end{aligned}
$$

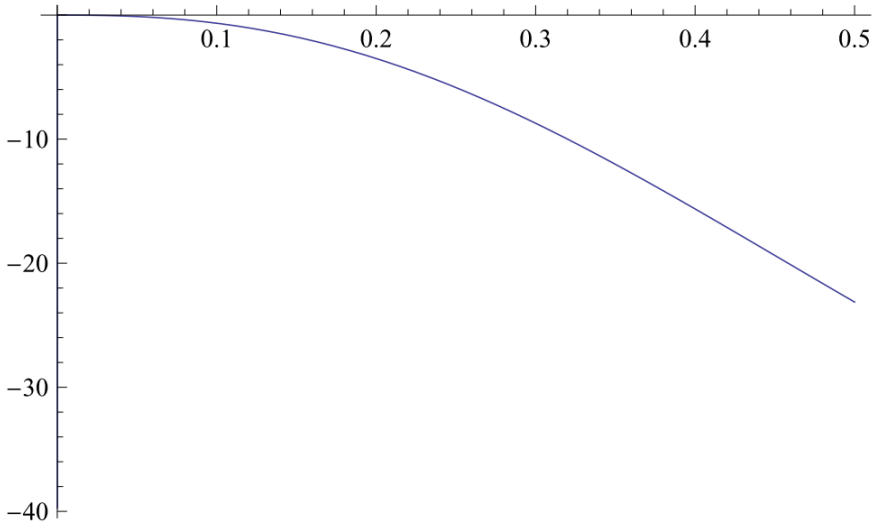
The power series approximation for $\int_0^1 t^{1/2}\left(\operatorname{sech}^2(t) - 1 + t^2 - \frac{2t^4}{3}\right)dt$ is ideal for Simpson's rule. Mathematica shows that the same relative error of 2×10^{-7} only requires $n = 32$ subintervals. It also has a slightly friendlier fourth derivative.

```
dfdt4[t_, s_] = D[f[t, s], {t, 4}]
```

$$(-3+s)\,(-2+s)\,(-1+s)\,s\,t^{-4+s}\left(-1+t^2-\frac{2\,t^4}{3}+\text{Sech}[t]^2\right)+$$
$$4\,(-2+s)\,(-1+s)\,s\,t^{-3+s}\left(2\,t-\frac{8\,t^3}{3}-2\,\text{Sech}[t]^2\,\text{Tanh}[t]\right)+$$
$$6\,(-1+s)\,s\,t^{-2+s}\left(2-8\,t^2-2\,\text{Sech}[t]^4+4\,\text{Sech}[t]^2\,\text{Tanh}[t]^2\right)+$$
$$4\,s\,t^{-1+s}\left(-16\,t+16\,\text{Sech}[t]^4\,\text{Tanh}[t]-8\,\text{Sech}[t]^2\,\text{Tanh}[t]^3\right)+$$
$$t^s\left(-16+16\,\text{Sech}[t]^6-88\,\text{Sech}[t]^4\,\text{Tanh}[t]^2+16\,\text{Sech}[t]^2\,\text{Tanh}[t]^4\right)$$

The vertical asymptote at $t = 0$ is now removed. Simpson's rule is able to calculate $\zeta(s)$ for $s = 1/2$ efficiently.

Figure 2.2: The power series approximation for $t^s/\cosh^2(t)$ near $t = 0$



My program does not properly calculate $\zeta(s)$ for large negative values of s. I originally thought that the problem was caused by the reference of the functional equation and the limitation of the double data type in Java. The max value for a double in Java is around 1.797×10^{308}, the minimum value is about 4.9×10^{-324}. Initial testing showed that the program stopped working after $s < -141$. Directly analyzing $\zeta(-141)$ by the functional equation equates to $2^{-141} \cdot \pi^{-142} \cdot \Gamma(142) \sin\left(\frac{\pi(-141)}{2}\right) \zeta(142) \approx 1.73 \times 10^{130}$. Because the double value should calculate up to 1.797×10^{308}, the problem must be due to a different section of the program.

After thinking about my design, I realized that the approximation of the gamma function was the cause of the issue. The Lanczos approximation for the gamma function is represented by

$$\Gamma(z+1) = \sqrt{2\pi}\left(z+g+\frac{1}{2}\right)^{z+\frac{1}{2}} e^{-\left(z+g+\frac{1}{2}\right)} A_g(z)$$

where

$$A_g(z) = \frac{1}{2}p_0(g) + p_1(g)\frac{z}{z+1} + p_2(g)\frac{z(z-1)}{(z+1)(z+2)} + \cdots$$

My program was written with $g = 7$. Plugging in $\Gamma(141+1)$,

$$\Gamma(142) = \sqrt{2\pi}\left(141+7+\frac{1}{2}\right)^{141+\frac{1}{2}} e^{-\left(141+7+\frac{1}{2}\right)} A_7(141)$$

will compute to a number just below 1.797×10^{308}. I then decided to look for an alternative approximation for the gamma function. Stirling's approximation is represented by

$$(n!) \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

The new approximation extended my program from $s < -141$ up until $s < -171$. At $s = -171$,

$$(-171!) \approx \sqrt{2\pi - 171}\left(\frac{-171}{e}\right)^{-171} \approx -8.6618 \times 10^{-307}$$

This is above the lower bound of 4.9×10^{-324}, the functional equation is likely causing the value to compute lower. Because I was heavily reading about complex numbers, I decided not to investigate any further. More recent memories about complex variable theory stimulated interest in calculating $\zeta(s) \,\forall\, s \in \mathbb{C} \setminus \{1\}$. A well known integral approximation for complex values of $\zeta(s)$ is known as the Abel−Plana formula.

2.2 The Abel-Plana formula for zeta(s)

The Abel−Plana formula is a summation formula [[35],[3]] used to compare the difference between a sum over discrete values and an integral of a function. It was developed by Niels Abel and Giovanni Plana in the early 1820s. The derivation of the formula is rather interesting, so I have decided to write a general outline.

The derivation of the Abel−Plana formula starts by letting $f(z)$ be an analytic function and $g(z)$ be a meromorphic function for $a \leq x \leq b$ inside the complex plane where $z = x + iy$. By assuming that the functions $f(z)$ and $g(z)$ satisfy

$$\lim_{h\to\infty} \int_{a\pm ih}^{b\pm ih} [g(z) \pm f(z)]\, dz = 0$$

one can write the following formula

$$\int_a^b f(x)dx = \pi i \sum_k Res_{z_k} g(z) - \frac{1}{2}\int_{-i\infty}^{+i\infty} [g(u) + sgn(Imz) f(u)]_{u=a+z}^{u=b+z}\, dz$$

by which z_k are the poles of $g(z)$ contained in the region of $a < x < b$.

The proof of this result is heavily dependent on both the residue theorem and Cauchy's integral theorem. Suppose that we construct a rectangle C_h with vertices at $a \pm ih$ and $b \pm ih$. By the residue theorem,

$$\int_{C_h} g(z)\, dz = 2\pi i \sum_k Res_{z_k} g(z)$$

where the sum on the right is over the residues inside C_h. Next, denote C_h^+ and C_h^- as the upper and lower halves of the contour. We can now break up the integral to form

$$\begin{aligned}\int_{C_h} g(z)\, dz &= \int_{C_h^+} g(z)\, dz + \int_{C_h^+} f(z)\, dz + \int_{C_h^-} g(z)\, dz - \int_{C_h^-} f(z)\, dz - \int_{C_h^+} f(z)\, dz + \int_{C_h^-} f(z)\, dz \\ &= \int_{C_h^+} [g(z) + f(z)]\ dz + \int_{C_h^-} [g(z) - f(z)]\ dz - \int_{C_h^+} f(z)\, dz + \int_{C_h^-} f(z)\, dz\end{aligned}$$

Notice that the path of $C_h^{\pm}$ is broken up into six subintervals. Let $D_h^{\pm}$ be a new path which closes the path in $C_h^{\pm}$ by ±[a,b]. Because f is analytic, we can apply Cauchy's integral theorem to write

$$\int_{D_h^{\pm}} f(z)\, dz = 0$$

which implies that

$$\int_{D_h^-} f(z)\,dz - \int_{D_h^+} f(z)\,dz = \Big[\int_{C_h^-} f(z)\,dz - \int_a^b f(z)\,dz\Big] - \Big[\int_{C_h^+} f(z)\,dz + \int_a^b f(z)\,dz\Big] = 0$$

This allows one to break up the integration limits for $\int_{C_h} g(z)\,dz$. The first equation can be broken into

$$\int_{C_h^+} [g(z)+f(z)]\ dz = \int_b^{b+ih} [g(z)+f(z)]\ dz - \int_{a+ih}^{b+ih} [g(z)+f(z)]\ dz - \int_a^{a+ih} [g(z)+f(z)]\ dz$$

The second equation can be broken into

$$\int_{C_h^-} [g(z)-f(z)]\ dz = -\int_b^{b-ih} [g(z)-f(z)]\ dz + \int_{a-ih}^{b-ih} [g(z)-f(z)]\ dz + \int_a^{a-ih} [g(z)-f(z)]\ dz$$

These two equations can be compacted to form

$$\int_{C_h^\pm} [g(z)\pm f(z)]\,dz = \pm\int_0^{\pm ih} [g(u)\pm f(u)]_{u=a+z}^{u=b+z}\,dz \mp \int_{a\pm ih}^{b\pm ih} [g(z)\pm f(z)]\,dz$$

which, letting $h\to\infty$, is then further reduced [3] to the original form shown inside the $\int_a^b f(x)\,dx$ above.

The Abel−Plana formula is generalized by adding an additional assumption to the original hypothesis of $\lim_{h\to\infty}\int_{a\pm ih}^{b\pm ih}[g(z)\pm f(z)]\,dz = 0$. Suppose that $f(z)$ and $g(z)$ also satisfy

$$\lim_{b\to\infty}\int_b^{b\pm i\infty} [g(z)\pm f(z)]\,dz = 0$$

then

$$\lim_{b\to\infty}\Big[\int_a^b f(x)dx - \pi i\sum_k Res_{z_k g(z)}\Big] = \frac{1}{2}\int_{a-i\infty}^{a+i\infty} [g(z)+sgn(Imz)f(z)]\,dz$$

because

$$-\frac{1}{2}\int_{-i\infty}^{+i\infty} [g(u)+sgn(Imz)f(u)]_{u=a+z}^{u=b+z}\,dz = -\frac{1}{2}\Big\{\int_b^{b+i\infty}[g(z)+f(z)]\,dz - \int_b^{b-i\infty}[g(z)-f(z)]\,dz\Big\} + \frac{1}{2}\int_{a-i\infty}^{a+i\infty}[g(z)+sgn(Imz)f(z)]\,dz$$

So, taking the limit [3] as $b\to\infty$ forms the RHS for

$$\frac{1}{2}\int_{a-i\infty}^{a+i\infty} [g(z)+sgn(Imz)f(z)]\,dz$$

A corollary is used to restrict the generalized Abel−Plana formula to $0<a<1$. With the same hypotheses as before, $\lim_{h\to\infty}\int_{a\pm ih}^{b\pm ih}[g(z)\pm f(z)]\,dz = 0$ and $\lim_{b\to\infty}\int_b^{b\pm i\infty}[g(z)\pm f(z)]\,dz = 0$, the corollary forms

$$\lim_{n\to\infty}\Big[\sum_{s=1}^{n} f(s) - \int_a^{n+a} f(x)dx\Big] = \frac{1}{2i}\int_a^{a-i\infty} f(z)\,(cot\pi z - i)\,dz - \frac{1}{2i}\int_a^{a+i\infty} f(z)\,(cot\pi z + i)\,dz$$

The details behind the derivation are shown in [3]. It is necessary to substitute $b = n+a$ and $g(z) = -if(z)\cot(\pi z)$.

The end of the derivation involves taking the limit as $a \to 0$. It is important to note that we add an additional term, $f(0)/2$, because $-if(z)\cot(\pi z)$ has a pole at zero. Taking the limit as $n \to \infty$ and letting $a \to 0$ changes the LHS to

$$\frac{f(0)}{2} + \sum_{n=1}^{\infty} f(n) - \int_0^{\infty} f(x)\,dx$$

A final substitution [3] is made for $z = it$ and $dz = idt$, letting $a \to 0$ changes the RHS to

$$\begin{aligned}
&\frac{1}{2}\int_0^{-\infty} f(it)\left(\cot(\pi it) - i\right)dt - \frac{1}{2}\int_0^{+\infty} f(it)\left(\cot(\pi it) + i\right)dt \\
&= -\frac{1}{2}\int_0^{+\infty} f(-it)\left(\cot(-\pi it) - i\right)dt - \frac{1}{2}\int_0^{+\infty} f(it)\left(\cot(\pi it) + i\right)dt \\
&= \frac{1}{2}\int_0^{+\infty} f(-it)\left(\cot(\pi it) + i\right)dt - \frac{1}{2}\int_0^{+\infty} f(it)\left(\cot(\pi it) + i\right)dt \\
&= -\frac{1}{2}\int_0^{+\infty} \Big[f(it) - f(-it)\Big]\left(\cot(\pi it) + i\right)dt
\end{aligned}$$

We can now add $f(0)/2$ to both sides

$$\sum_{n=0}^{\infty} f(n) - \int_0^{\infty} f(x)\,dx = \frac{1}{2}f(0) - \frac{1}{2}\int_0^{\infty} \left[f(it) - f(-it)\right]\left[cot(\pi it) + i\right]dt$$

It is then necessary to apply the following identity

$$cot(\pi it) + i = \frac{-2i}{e^{2\pi t} - 1}$$

Which will yield the known equation for the Abel−Plana formula.

$$\sum_{n=0}^{\infty} f(n) - \int_0^{\infty} f(x)\,dx = \frac{f(0)}{2} + i\int_0^{\infty} \frac{f(it) - f(-it)}{e^{2\pi t} - 1}\,dt$$

Abel published another version of the Abel-Plana formula which is known as the alternating series version. This version is obtained [8] by applying the Abel−Plana formula to the function $z \mapsto 2f(2z)$ and then subtracting the resulting equation from the Abel−Plana formula.

$$\sum_{n=0}^{\infty} (-1)^n f(n) = \frac{f(0)}{2} + i\int_0^{\infty} \frac{f(it) - f(-it)}{2\sinh(\pi t)}\,dt$$

I found further information by reviewing additional theory about Cauchy's argument principle [34]. The argument principle states that if $f(z)$ is meromorphic inside a closed contour C, then

$$\oint_C \frac{f'(z)}{f(z)}\,dz = 2\pi i(N - P)$$

where N are the complex roots for $f(z)$ in C and P are the number of poles for $f(z)$ in C.

Introducing g(z) as a meromorphic function allows one to write [7] the complex integral of the logarithmic derivative of $g(z)$ multiplied by $f(z)$ as

$$\oint_C f(z)\frac{g'(z)}{g(z)}\,dz$$

which, taken with specific choices for f and g, will form [29] the generalized Abel−Plana formula

$$\sum_{n=0}^{\infty} f(n) - \int_0^{\infty} f(x)\,dx = \frac{f(0)}{2} + i\int_0^{\infty} \frac{f(it) - f(-it)}{e^{2\pi t} - 1}\,dt$$

I will now focus on the zeta function. Provided that the function is both holomorphic and analytic in the region of interest, it is possible to extend the Abel−Plana formula to $\zeta(s)$. This creates the following integral transformation [38]

$$\zeta(s) = \frac{2^{s-1}}{s-1} - 2^s \int_0^{\infty} \frac{\sin(s\arctan t)}{(1+t^2)^{\frac{s}{2}}(\mathrm{e}^{\pi t}+1)}\,\mathrm{d}t \qquad \forall\ s \in \mathbb{C}\setminus\{1\}$$

where s is assumed to take the form of $s = a + ib$ provided $a, b \in \mathbb{R}$. This is an alternative method to directly calculate any value of $\zeta(s)$. It was interesting to find out that Mathematica displayed other relationships that involved the Abel−Plana formula for $\zeta(s)$. The first relationship can be plainly stated as

$$\int_0^{\infty} \frac{\sin(s\arctan t)}{(1+t^2)^{\frac{s}{2}}(\mathrm{e}^{\pi t}+1)} = \frac{1}{2(s-1)} - 2^{-s}\zeta(s)$$

The second relationship is an alternate form for the integrand.

$$\frac{\sin(s\arctan t)}{(1+t^2)^{\frac{s}{2}}(\mathrm{e}^{\pi t}+1)} = \frac{i\,(t^2+1)^{-s/2}\,e^{\frac{1}{2}s(\log(1-it)-\log(1+it))}}{2(e^{\pi t}+1)} - \frac{i\,(t^2+1)^{-s/2}\,e^{-\frac{1}{2}s(\log(1-it)-\log(1+it))}}{2(e^{\pi t}+1)}$$

A third relationship generates a Taylor series expansion at s = 0.

$$\frac{s\tan^{-1}(t)}{e^{\pi t}+1} - \frac{s^2\left(\log\left(t^2+1\right)\tan^{-1}(t)\right)}{2\left(e^{\pi t}+1\right)} + \frac{s^3\left(3\log^2\left(t^2+1\right)\tan^{-1}(t) - 4\tan^{-1}(t)^3\right)}{24\left(e^{\pi t}+1\right)} + \frac{s^4\left(4\log\left(t^2+1\right)\tan^{-1}(t)^3 - \log^3\left(t^2+1\right)\tan^{-1}(t)\right)}{48\,e^{\pi t}+48} + \frac{s^5\tan^{-1}(t)\left(5\log^4\left(t^2+1\right) - 40\log^2\left(t^2+1\right)\tan^{-1}(t)^2 + 16\tan^{-1}(t)^4\right)}{1920\left(e^{\pi t}+1\right)} + O\left(s^6\right)$$

While it would be interesting to compute the Taylor series through a method inside Java, it is not necessary for the implementation of the Abel−Plana formula. I wrote a method inside Java that is heavily dependent on the calculation of complex numbers and complex arithmetic. This includes the calculation of the modulus, principle argument, and complex trigonometric functions. Similar to the computation of $\zeta(s)$ through the Cauchy−Schlömilch transformation, it is necessary to use recursion inside the approximation of $\zeta(s)$ through adaptive quadrature. Because $\zeta(s)$ is calculated $\forall\ s \in \mathbb{C}$, it is necessary to reference two conditional statements

• If $s = a + bi$ and $\zeta(s)$ is calculated as $\zeta(1 + 0i)$, log an error message. The Riemann zeta function is holomorphic everywhere except at the pole where $s = 1$. However, $\zeta(1 + it)$ where $t \in \mathbb{R}\setminus\{0\}$ is certainly valid.

• Else use the Abel−Plana formula for all other calculations of $s \in \mathbb{C}$. The integrand of $\sin(s\arctan t)$ $/(1+t^2)^{\frac{s}{2}}(\mathrm{e}^{\pi t}+1)$ will need to be approximated by numerical integration.

The grand majority of the error inside the approximation of $\zeta(s)$ stems from the numerical approximation of $\int_0^{\infty} \sin(s\arctan t)/(1+t^2)^{\frac{s}{2}}(\mathrm{e}^{\pi t}+1)\,\mathrm{d}t$. I decided to use a slightly more advanced form of Simpson's method known as adaptive quadrature. The method itself [28] starts by letting $f(x)$ be a real-valued function of a real variable, which is defined inside a finite interval for $a \le x \le b$. This leads to the direct value of the integral for $\int_b^a f(x)dx$. By carefully selecting where $f(x)$ is sampled, the quadrature method attempts to evaluate as

few points as possible. The additive property of definite integrals is applied as the basis for quadrature. By letting $f(x)$ be continuous on [a, b] and c be a point where $a \leq c \leq b$, the additive property shows that

$$\int_a^b f(x)\,dx = \int_a^c f(x)\,dx + \int_c^b f(x)\,dx$$

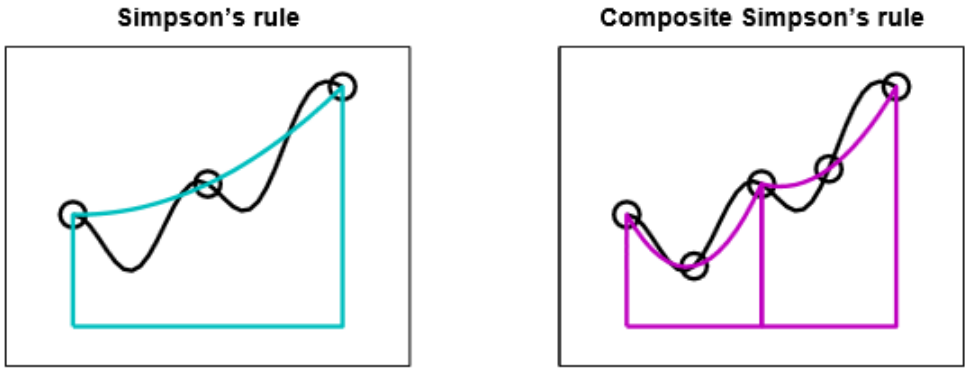
If each of the two integrals on the right is approximated to a specific tolerance, the sum of these will produce the desired output. Recursion is necessary when the sum is not within the desired output. The algorithm itself will recursively compute the value of each integral for [a,c] and [b,c] until the sum reaches a specific tolerance.

The same method is applied in many different algorithms and is commonly known as partitioning into subintervals. It is also common to represent these subintervals by writing $\Delta x_i = x_i - x_{i-1}$. Observing that individual subintervals can be written as $[x_o, x_1]$, $[x_1, x_2]$, $[x_2, x_3]$ allows the ith subinterval to be written as $[x_{i-1}, x_i]$. The standard form of Simpson's rule interpolates the integrand at two endpoints for a and b. One can then define a midpoint, labeled c, where $c = (a + b)/2$. The midpoint of any segment in n-dimensional space whose endpoints are $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$ forms $(A + B)/2$. It is customary to denote S_1 as the sum and write Simpson's rule as

$$S_1 = \frac{h}{6}\left(f(a) + 4f(c) + f(b)\right)$$

The h is the length of the interval and can be thought of as the difference between two endpoints where $h = b - a$. Applying the same procedure again creates two midpoints and breaks the interval into two halves labeled [a,c] and [c,b]. Two new points are created, $d = (a + c)/2$ and $e = (c + b)/2$; which are new midpoints. These new points can be shown graphically by

Figure 2.3: Simpson's rule versus Composite Simpson's rule



Adaptive quadrature is also known as the Composite Simpson's rule. As shown above, the Composite Simpson's rule breaks the interval into two segments and then defines two midpoints d and e. Applying Simpson's rule to both subintervals creates a new sum, denoted S_2 [28] where

$$S_2 = \frac{h}{12}\left(f(a) + 4f(d) + 2f(c) + 4f(e) + f(b)\right)$$

Because S_1 and S_2 approximate the same integral, the error from the approximation is their difference

$$E = (S_2 - S_1)$$

A more accurate approximation [28] is formed by adding an additional variable named Q. Both S_1 and S_2 are fourth order, but S_2 has half the step size of S_1. This means that S_2 is 2^4 times more accurate. Applying the difference with Q forms

$$Q - S_1 = 16(Q - S_2)$$

Adding S1 to the RHS generates

$$Q = 16Q - 16S_2 + S_1$$

the end result is

$$Q = \frac{16S_2 - S_1}{15} = S_2 + \frac{S_2 - S_1}{15}$$

Adaptive quadrature breaks the interval into two subintervals and then applies a weighted combination of the five function values for $f(a)$, $f(b)$, $f(c)$, $f(d)$, and $f(e)$. All literature which I have read shows that this method is only applicable to real-valued functions. I was able to use the Composite Simpson's rule inside the calculation of $\zeta(s)$ for all $s \in \mathbb{C}$. I'm not certain why this method is restricted to real-value functions. I developed a program in Java that returns correct values for complex functions. I was curious about this result and made similar programs for the Trapezoidal rule, Simpson's rule, and the Romberg algorithm. These all produced correct results.

The computer implementation of the Abel−Plana formula suffered from the same shortcomings as the previously described Cauchy−Schlömilch transformation. One major limitation is due to the evaluation of $(2^{s-1}/(s-1)) - (2^s)$. At high values of s, the real part approaches one while the imaginary part approaches zero. As an example, $\zeta(100 - 7i) = 1 + 0.1097998919157576 \times 10^{30} - 7.8118212442515740797416 \times 10^{31}i$. The level of precision necessary to calculate large values of s is much higher than the maximum provided by the double data type. To work around this, I found a similar integral transformation originally published by Ernst Lindelöf. He [24] found that

$$\zeta(s) = \frac{1}{2} + \frac{1}{s-1} + 2\int_0^\infty \frac{\sin(s \arctan x)}{(1+x^2)^{s/2}(e^{2\pi x} - 1)} dx.$$

It is interesting to note that Lindelöf published this integral representation in 1905 while the DLMF lists the discovery of the Abel−Plana formula for $\zeta(s)$ in 2001. The new integral representation alleviates the previous evaluation of $(2^{s-1}/(s-1)) - (2^s)$. Because it was necessary to integrate the complex integrand with a large amount of precision, I was forced to develop a new method that would calculate everything through the BigDecimal format in Java. The Apache Commons Math Library for Java has a rather extensive list of methods that can be used for numerical calculations. Although, most of these classes are evaluated through the double data type which is not practical when performing calculations with high precision. Some interesting results from my program include

$$\zeta(3 - 4i) = 0.8905549069649760163418883739789070190 + 0.008075945424226249347 5 * i$$
$$\zeta(22 + 28i) = 1.000000202208084723712501681438 1488 - 1.2633541222053952 \times 10^{-7} * i$$
$$\zeta(15.2 - 23.6i) = 0.999978894762091281269006499024 0755 - 1.604400130991151 \times 10^{-5} * i$$

Because all BigDecimal calculations in Java are slow, precision of these calculations could be increased at a significant reduction of speed. I'm certainly not the first person who has ran across this issue, IBM developed a brand new library labeled Enhanced BigDecimal. At the time of writing this section of my paper, I was unable to successfully implement large values of $\zeta(s)$ using the Abel−Plana formula. The program is able to calculate $\zeta(200 + 0i)$ but fails when calculating $\zeta(230 + 30i)$. I will explain these failures more explicitly inside the appendix. Another interesting integral transformation that Lindelöf [24] published was

$$\zeta(s) = \frac{2^{s-1}}{1 - 2^{1-s}} \int_0^\infty \frac{\cos(s \arctan x)}{(1+x^2)^{s/2} \cosh\left(\frac{1}{2}\pi x\right)} dx.$$

According to Apostol's notes inside the DLMF, the formula was also published in 1905.

2.3 Dirichlet series for zeta(s)

To reiterate my previous problem, the Abel−Plana formula for $\zeta(s)$ lead to two issues that resulted in numerical instability. The first major problem I encountered was the evaluation of $\zeta(s)$ through a form which required a large amount of functional evaluations. As shown in [22], it is difficult to implement high precision calculations through floating-point arithmetic inside Java. The integrand of $\sin(s \arctan t) \ /(1+t^2)^{\frac{s}{2}}(\mathrm{e}^{\pi t}+1)$ is an increasingly oscillatory function because of the sine inside the numerator. As s increases, the number of evaluations also increases and significantly slows down the speed of computation. The second problem is due to the evaluation of $2^{s-1}/(s-1)-2^s$. The computation of $\zeta(100)$ alone would require about 45 digits of decimal precision in order to produce correct results. Both of these issues become significantly worse as either the real or imaginary part of $\zeta(s)$ increase.

I needed some time away from the Abel−Plana formula to think about an alternative solution. A nudge in the right direction aligned my attention towards infinite series. In particular, the Dirichlet series representation for $\zeta(s)$. The Dirichlet series itself can be shown to form

$$\zeta(s) = \frac{1}{s-1}\sum_{n=1}^{\infty}\left(\frac{n}{(n+1)^s} - \frac{n-s}{n^s}\right) \qquad \forall\ \Re(s) > 0$$

The derivation of the series is definitely of interest. It starts off by pondering how $\sum_{k=1}^{\infty} k^{-s}$ grows if $\mathrm{Re}(s) < 1$. By acknowledging that $\sum_{k=1}^{n} \frac{1}{n}\left(\frac{k}{n}\right)^{-s}$ is the Riemann sum for $\int_0^1 x^{-s}\,\mathrm{d}x$, one may write

$$\sum_{k=1}^{n}\frac{1}{k^s} = n^{1-s}\sum_{k=1}^{n}\frac{1}{n}\left(\frac{k}{n}\right)^{-s} \sim n^{1-s}\int_0^1 x^{-s}\mathrm{d}x = \frac{n^{1-s}}{1-s}$$

which is valid for $\mathrm{Re}(1-s) > 0 \iff \mathrm{Re}(s) < 1$. This means that

$$\sum_{k=1}^{n}\frac{1}{k^s} \approx \frac{n^{1-s}}{1-s} + \zeta(s)$$

for $\mathrm{Re}(s) > 0, \neq 1$. It is important to note that the line above is formed by combining the evaluation of $\zeta(s)$ through $0 < \mathrm{Re}(s) < 1$ and $\mathrm{Re}(s) > 1$ separately. Because $\zeta(s)$ has a pole at $s = 1$, it is worthwhile to evaluate $(s-1)\zeta(s)$ instead. Let's denote this as c_n and define it as the limit of

$$c_n = (s-1)\sum_{k=1}^{n}\frac{1}{k^s} + n^{1-s}$$

Apply the forward difference method to produce

$$\begin{aligned} c_{n+1} - c_n &= \frac{s-1}{(n+1)^s} + \frac{n+1}{(n+1)^s} - \frac{n}{n^s} \\ &= \frac{s}{(n+1)^s} + n\left(\frac{1}{(n+1)^s} - \frac{1}{n^s}\right) \end{aligned}$$

Therefore, add $c_1 + (c_2 - c_1) + \cdots + (c_{n+1} - c_n)$ and let $c_1 = s$ to form

$$\begin{aligned} c_{n+1} &= s + \sum_{k=1}^{n}\left[\frac{s}{(k+1)^s} + k\left(\frac{1}{(k+1)^s} - \frac{1}{k^s}\right)\right] \\ &= \frac{s}{(n+1)^s} + \sum_{k=1}^{n}\left[\frac{s}{k^s} + k\left(\frac{1}{(k+1)^s} - \frac{1}{k^s}\right)\right] \end{aligned}$$

Now, apply the limit so that $n \to \infty$. The term for $\dfrac{s}{(n+1)^s}$ will be deleted. Substitute back in c_n

$$(s-1)\zeta(s) = \sum_{k=1}^{\infty}\left[\frac{s}{k^s} + k\left(\frac{1}{(k+1)^s} - \frac{1}{k^s}\right)\right]$$

Which further reduces to

$$\begin{aligned}\zeta(s) &= \frac{1}{s-1}\sum_{k=1}^{\infty}\left[\frac{s}{k^s} + k\left(\frac{1}{(k+1)^s} - \frac{1}{k^s}\right)\right] \\ &= \frac{1}{s-1}\sum_{k=1}^{\infty}\left(\frac{k}{(k+1)^s} - \frac{k-s}{k^s}\right)\end{aligned}$$

The calculation of $\zeta(s)$ through the Dirichlet series is ideal for computation. While it is impossible to work around high precision calculations by the BigDecimal class, it is certainly straightforward to evaluate $\zeta(s)$ by a series summation. Combining this equation with any form of the functional equation would produce a computer program that could certainly handle $\zeta(s)$ for all $s \in \mathbb{C} \setminus \{1\}$.

The Riemann zeta function is heavily connected with Dirichlet series and has many interesting applications. By functional notation, the transformation of $f \mapsto D(f, s)$ can be defined as the following Dirichlet series

$$D(f, s) = \sum_{n=1}^{\infty} f(n) n^{-s}$$

From which [5] the following transformative property can be derived

$$\begin{aligned}D(f, s) \cdot D(g, s) = \Big(\sum_{n=1}^{\infty}\frac{f(n)}{n^s}\Big)\Big(\sum_{n=1}^{\infty}\frac{g(n)}{n^s}\Big) &= \sum_{n=1}^{\infty}\frac{\sum_{(d|n)} f(d)g(n/d)}{n^s} \\ &= D(f * g, s)\end{aligned}$$

The above result is known as the Dirichlet convolution of arithmetical functions. Note that the Dirichlet convolution is strange and actually means to multiply two functions. The Dirichlet convolution applies to both the equality of functions for complex variables and an analytical proof of the Möbius inversion formula. If we let $f = 1$, we can define the Riemann zeta function as

$$\zeta(s) = D(1, s) = \sum_{n=1}^{\infty}\frac{1}{n^s}$$

By letting $f(n) = d(n)$, one can define [6] divisor function such that $D(d, s) = \sum_n \frac{d(n)}{n^s}$. The convolution shows that $d = 1 * 1$ so we multiply $D(d, s) = D(1, s)D(1, s)$ to form

$$D(d, s) = \zeta(s) \cdot \zeta(s) = \zeta^2(s)$$

The Möbius function isn't left out. Because $\delta(1) = 1$ and $\delta(n) = 0$ for all $n > 1$, we can write

$$D(\delta, s) = \frac{1}{1^s} + \sum_{n=2}^{\infty}\frac{0}{n^s} = 1$$

The above identity allows one to find the Dirichlet series for $D(\mu, s)$. By noting that convolution means multiplication, $\mu * 1 = \delta$ produces

$$1 = D(\mu, s)D(1, s) = D(\mu, s)\zeta(s)$$

The identity above is very useful because it directly relates combinatorial methods with analytical methods. The theory of Dirichlet series also applies to many other interesting parts of the zeta function. One most prominent of these includes the Euler product expansion,

$$D(f, s) = \prod_{p}\Big(1 - \frac{f(p)}{p^s}\Big)^{-1}$$

2.4 Riemann-Siegel formula for zeroes of zeta(s)

The Riemenn-Siegel formula is a method for finding non-trivial zeroes on the critical line. The method itself was originally developed by Riemann and was later published by Siegel in the early 1930s. Through careful analysis of Riemenn's original formula for $\zeta(s)$, Siegel was able to improve the previously developed Euler-McLaurin Summation formula. To facilitate a general understanding of how the formula was derived, I will outline the general approach first taken by Riemann. He starts off by deriving [12] the following formula which is valid for all s

$$\zeta(s) = \frac{\Pi(-s)}{2\pi \mathrm{i}} \int_{+\infty}^{+\infty} \frac{(-x)^s}{e^x - 1} \cdot \frac{dx}{x}$$

The two limits of integration represent what is now known as a contour integral. As is standard with similar problems, one can show that $(-x)^s$ can be written as $(-x)^s = e^{slog(-x)}$. $Log(-x)$ is then defined as log z for $z \notin \mathbb{R}^-$. It is then necessary to apply a branch cut that winds around 0 in the counterclockwise direction. In Riemann's original paper, this can be shown to start at $+\infty$, descend left downwards on the positive real axis, circles the origin once counterclockwise, and then moves right on the positive real axis to $+\infty$. In modern notation, a direct relationship between the Pi function and the gamma function is known to be $\Pi(s) = \Gamma(s+1)$. Further analysis can be performed through a direct analysis of the functional equation. Applying $s = \frac{1}{2} + it$ and $\Pi(s) = \Gamma(s+1)$ allows one to rewrite [12] the functional equation as

$$\begin{aligned}
\xi\left(\frac{1}{2} + it\right) &= \frac{s}{2} \cdot \Pi\left(\frac{s}{2} - 1\right) \pi^{-s/2}(s-1) \cdot \zeta(s) \\
&= e^{\log \Pi(\frac{s}{2}-1)\pi^{-s/2}} \cdot \frac{s(s-1)}{2} \cdot \zeta(s) \\
&= \left[e^{\Re \log \Pi(\frac{s}{2}-1)\pi^{-1/4}} \cdot \frac{-t^2 - \frac{1}{4}}{2}\right] \times \left[e^{i\Im \log \Pi(\frac{s}{2}-1)\pi^{-it/2}} \cdot \zeta\left(\frac{1}{2} + it\right)\right] \\
&= -\, e^{i\Im \log \Pi(\frac{s}{2}-1)\pi^{-it/2}} \cdot \zeta\left(\frac{1}{2} + it\right)
\end{aligned}$$

where the last equation is simplified as a negative sign due to the $\left(-t^2 - \frac{1}{4}\right)$ in the equation above. It is standard to label the right hand side of this equation as $Z(t)$. This is known as the Riemann−Siegel Z function. It is possible to rearrange terms to find

$$Z(t) = e^{i\vartheta(t)} \zeta\left(\frac{1}{2} + it\right)$$

The $\vartheta(t)$ is known as the Riemann−Siegel theta function. This can be directly stated as

$$\vartheta(t) = \arg\left(\Gamma\left(\frac{2it+1}{4}\right)\right) - \frac{\log \pi}{2} t$$

Using Stirling's approximation, one can approximate the value of $\vartheta(t)$ as

$$\vartheta(t) = \frac{t}{2} \log \frac{t}{2\pi} - \frac{t}{2} - \frac{\pi}{8} + \frac{1}{48t} + \frac{7}{5760t^3} + \cdots$$

The error from this approximation is small and is no larger than $1/t^5$. Riemann showed that you could further simplify the $Z(t)$ function [12] to form the following equation (L is a line segment which approximates the integral by the line of steepest decent through the saddle point)

$$Z(t) = 2 \sum_{n^2 < t/2\pi} n^{-1/2} \cos(\theta(t) - t \log n) + \frac{e^{-i\vartheta(t)} e^{-t\pi/2} (-a)^{-(1/2)+it} e^{-Na}}{(2\pi)^{1/2} (2\pi)^{it} e^{-i\pi/4} (1 - ie^{-t\pi})} \int_L \frac{e^{i(x-a)^2/4\pi} e^{p(x-a)} \sum_{n=0}^{\infty} b_n (x-a)^n dx}{e^x - 1}$$

The important part of the equation above is the infinite series on the right hand side of the equation. This is shown as $\sum_{n=0}^{\infty} b_n(x-a)^n$. The series can be truncated at the nth term to form the following approximation

$$Z(t) = 2\sum_{n^2<t/2\pi} n^{-1/2}\cos(\theta(t) - t\log n) \; + (-1)^{N-1}(\frac{t}{2\pi})^{-1/4} \cdot U\left[b_0c_0 + b_1c_1 + ... + b_nc_n\right]$$

Riemann was able to derive this using the logarithmic derivative of $\sum_{n=0}^{\infty} b_n(x-a)^n$. He then truncated all of the terms past b_0c_0. This allowed him to form crude estimates for the locations of the the first two roots. Labeling the roots as α_j, he was able to estimate the values of the first two roots as $\alpha_1 = 14.1$ and $\alpha_2 = 20.7$. As will be shown below, this is fairly accurate. In the year 1903, Gram published results for the location of the first 15 non-trivial zeroes for $\zeta(s)$. This was later reviewed by Carl Siegel who published the Riemann−Siegel formula in 1932. Siegel was able to show that the formula takes the following form

$$Z(t) = 2\sum_{n^2<t/2\pi} n^{-1/2}\cos(\theta(t) - t\log n) + R_t$$

where

$$R_t \sim (-1)^{N-1}(\frac{t}{2\pi})^{-1/4} \times [C_0 \; + C_1(\frac{t}{2\pi})^{-1/2} \; + C_2(\frac{t}{2\pi})^{-2/2} \; + C_3(\frac{t}{2\pi})^{-3/2} \; + C_4(\frac{t}{2\pi})^{-4/2} \,]$$

The R_t term truncates the infinite series which was defined as $\sum_{n=0}^{\infty} b_n(x-a)^n$. Most computer implementations are designed to calculate a combination of C_0, C_1, C_2, C_3, and C_4 for the remainder terms. This attributes to a fair amount of error inside numerical calculations. The program that I wrote in Java uses the C_0, C_1, C_2, C_3, and C_4 terms in the approximation. Before going into further details, it is important to note a derivation that was done by Riemann. The derivation [12] shows that

$$C_0 = \Psi(p) = e^{i\pi/8}e^{-2\pi ip^2} \cdot \frac{1}{2\pi i}\int_L \frac{e^{iu^2/4\pi}e^{2pu}du}{e^u - 1} = \frac{\cos 2\pi(p^2 - p - 1/16)}{\cos 2\pi p}$$

By setting $N = \left\lfloor \frac{t}{2\pi} \right\rfloor^{1/2}$ and $p = \left(\frac{t}{2\pi}\right)^{1/2} - \left\lfloor \frac{t}{2\pi} \right\rfloor^{1/2}$, one can form

$$\begin{aligned}
C_1 &= -\frac{\Psi^3(p)}{96\pi^2} \\
C_2 &= \frac{\Psi^2(p)}{64\pi^2} \; + \; \frac{\Psi^6(p)}{18432\pi^4} \\
C_3 &= -\frac{\Psi'(p)}{64\pi^2} \; - \; \frac{\Psi^5(p)}{3840\pi^4} \; - \; \frac{\Psi^9(p)}{5308416\pi^6} \\
C_4 &= \frac{\Psi(p)}{128\pi^2} \; + \; \frac{19\Psi^4(p)}{24576\pi^4} \; + \; \frac{11\Psi^8(p)}{5898240\pi^6} \; + \; \frac{\Psi^{12}(p)}{2038431744\pi^8}
\end{aligned}$$

Additional remainder terms can be calculated through recursion, although my program is not designed to calculate past C_4. In comparing these calculations with the values provided by Andrew Odlyzko, all values for the non-trivial zeroes are accurate to about seven decimals of precision. I originally tried to calculate the remainder terms using finite difference approximations. After spending a large amount of time on this, I realized that it was senseless to try to calculate the remainder terms inside Java. My program started to become less accurate as I tried to apply the finite difference approximation past the fifth derivative. I originally didn't understand why this occurred and started to read more about finite difference theory. I was fortunate to discover the cause of the issue. It is not practical to use higher order finite difference approximations inside computer programs.

After initial investigation of the finite difference method, I was able to locate the following approximations

$$
\begin{aligned}
f'(x) &\approx \frac{-f(x+2h)+8f(x+h)-8f(x-h)+f(x-2h)}{12h} \\
f''(x) &\approx \frac{-f(x+2h)+16f(x+h)-30f(x)+16f(x-h)-f(x-2h)}{12h^2} \\
f^{(3)}(x) &\approx \frac{f(x+2h)-2f(x+h)+2f(x-h)-f(x-2h)}{2h^3} \\
f^{(4)}(x) &\approx \frac{f(x+2h)-4f(x+h)+6f(x)-4f(x-h)+f(x-2h)}{h^4}
\end{aligned}
$$

The error involved in these approximations is of $O(h^2)$ and $O(h^4)$. They are found inside the Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables [1]. However, it is possible to generalize the finite difference approximations and form an even finite difference operator.

This can be found by first defining a finite difference operator as Δ

$$\Delta f(x) = \frac{f(x+h/2)-f(x-h/2)}{h}.$$

It's a second order first derivative operator. After applying it multiple times, one finds that

$$
\begin{aligned}
\Delta^2 f(x) = \Delta\Delta f(x) &= \Delta\left(\frac{f(x+h/2)-f(x-h/2)}{h}\right) \\
&= \frac{f(x+h)-f(x)}{h^2} - \frac{f(x)-f(x-h)}{h^2} \\
&= \frac{f(x+h)-2f(x)+f(x-h)}{h^2} \sim f''(x)
\end{aligned}
$$

which can be further simplified [40] to

$$\Delta^{2k} f(x) = \frac{1}{h^{2k}} \sum_{m=-k}^{k} \binom{2k}{k+m} (-1)^{m+k} f(x+mh)$$

The truncation error involved with the approximation is

$$\Delta^{2k} f(x) - f^{(2k)}(x) = \frac{kh^2}{12} f^{(2k+2)}(x) + O(h^4)$$

Using this method, one can represent the finite difference approximation [40] of a twelve order derivative as

$$f^{(12)}(x) \approx \frac{1}{h^{12}} \sum_{k=-6}^{6} \binom{12}{6+k} (-1)^k f(x+kh)$$

This was my original strategy for calculating the remainder terms, the initial implementation had a method which calculated up to the sixth derivative. When the same method tried to calculate the functional equation, an error message was returned specifying NaN (not a number). Double precision in Java is limited to a mantissa of 53 bits of precision. Further analysis shows that finite difference approximations past the fourth derivative are rarely used in numerical calculations. Numerical experiments have shown that these calculations are only accurate to about two decimal points [40] when using double precision. This explains why it was difficult for me to find representations past the fourth derivative in literature. Any program that applies high order approximations of the finite difference schemes would need to evaluate numerical calculations through extremely high precision. It is possible to do this inside Java through the BigDecimal class.

I later found out that one could calculate the remainder terms through a direct Taylor series expansion. The C_0 term above can be expressed as a truncated Taylor series which can then be used to find the first twelve derivatives. A table for this approximation is provided in Edward's book [12] and was presented by Haselgrove in 1960. Similar work was done by Glen Pugh [32] and Ken Takusagawa [39] who used coefficients of Chebyshev polynomials in opposition to Taylor polynomials.

In order to approximate the value of zeroes, I used a variation of the root-finding algorithm known as Brent's method. Brent's method was published by Richard Brent in 1979, it combines features from root bracketing, bisection, and inverse quadratic interpolation. One improvement is made directly inside the constructor. The maximum order of the inverted polynomial root is user-defined, which allows the user to specify individualized functional evaluations.

The notation used by Gram to calculate roots on the critical line is defined as $\rho = \frac{1}{2} + i\alpha_j$. Using the same α_j defined above, the first twelve values of α_j are approximately

$$\begin{array}{lll} \alpha_1 = 14.134725142 & \alpha_2 = 21.022039639 & \alpha_3 = 25.010857580 \\ \alpha_4 = 30.424876126 & \alpha_5 = 32.935061588 & \alpha_6 = 37.586178159 \\ \alpha_7 = 40.918719012 & \alpha_8 = 43.327073281 & \alpha_9 = 48.005150881 \\ \alpha_{10} = 49.7738324781 & \alpha_{11} = 52.970321478 & \alpha_{12} = 56.446247697 \end{array}$$

All of these roots are directly where $Z(t)$ changes sign. The computer implementation that I wrote can be used to compute millions of zeroes. For large scale computations, one can reference what is known as Gram's law. Gram's law is a direct observation that zeroes of the Riemann−Siegel $Z(t)$ function tend to alternate with what are known as Gram points. This is not an actual law and can be shown to fail indefinitely. In order to explain this observation, I will refer back to

$$Z(t) = e^{i\vartheta(t)}\zeta\left(\frac{1}{2} + it\right)$$

One can manipulate the equation so that $\zeta\left(\frac{1}{2} + it\right) = e^{-i\vartheta(t)}\dot{Z(t)}$. Using Euler's identity, $e^{i\vartheta} = \cos\vartheta + i\sin\vartheta$, the equation can be divided into $\zeta\left(\frac{1}{2} + it\right) = \cos\vartheta(t)\dot{Z(t)} - i\sin\vartheta(t)\dot{Z(t)}$. Provided $t \in \mathbb{R}$, this equation can be further simplified to $\operatorname{Re}\zeta\left(\frac{1}{2} + it\right) = \cos\vartheta(t)\dot{Z(t)}$. This means that a zero of a real value of the zeta function on the critical line is directly related to both $\cos\vartheta(t)$ and $Z(t)$. Alternatively, it can also be shown that $\operatorname{Im}\zeta\left(\frac{1}{2} + it\right) = -\sin\vartheta(t)\dot{Z(t)}$. This implies that a sign change of $\operatorname{Im}\zeta(s)$ corresponds to a sign change of either $Z(t)$ or $\sin\vartheta(t)$. Gram's observations showed that $Z(t)$ changes sign for about the first 50 intervals of $\vartheta(t)$ because t is nowhere near an integer multiple of π. Therefore, one can think of the Riemann−Siegel formula as a direct relationship between a sign change of $Z(t)$ and a root where $\operatorname{Re}(s) = 1/2$. Further analysis shows that the $\operatorname{Re}\zeta\left(\frac{1}{2} + it\right)$ is generally positive while $\operatorname{Im}\zeta\left(\frac{1}{2} + it\right)$ alternates between positive and negative values. Focusing on $\operatorname{Im}\zeta\left(\frac{1}{2} + it\right)$, I will refer back to $\operatorname{Im}\zeta\left(\frac{1}{2} + it\right) = -\sin\vartheta(t)\dot{Z(t)}$. It can be shown that the zeroes of $Z(t)$ tend to alternate with the zeroes of $\sin\vartheta(t)$. This is known as Gram's law. We can then define a Gram point by noting that $\sin\vartheta(t)$ will be zero at integer multiples of π. If $\vartheta(t)$ is the Riemann−Siegel theta function, the t can be replaced by g_n to denote a Gram point as

$$\vartheta\left(g_n\right) = n\pi \qquad\qquad \forall \quad n \geq -1$$

It can then [23] be shown that

$$\zeta\left(\frac{1}{2} + ig_n\right) = \cos(\vartheta(g_n))Z(g_n) = (-1)^n Z(g_n)$$

and that

$$(-1)^n Z(g_n) > 0$$

will hold if and only if g_n is a Gram point. All the points in which $(-1)^n Z(g_n) > 0$ are known as "good" and all the points where $(-1)^n Z(g_n) \leq 0$ are known as "bad." A process to handle the bad points uses what is known as a Gram block. A Gram block is an interval defined by $g_n \leq t \leq g_{n+k}$ in which g_n and g_{n+k} are good and where $g_{n+1}, g_{n+2}, g_{n+2}, g_{n+3} \ldots g_{n+k-1}$ are bad. The total number of Gram points, both good and bad, can be referenced to count the number of zeroes for $Z(t)$ for an interval of $0 \leq t \leq g_n$.

A method for finding these zeroes is known as Backlund's method. His method was developed around 1912 and is based off of work done inside Riemann's original paper labeled *On the Number of Primes Less Than a Given Magnitude*. Riemann had originally estimated the number of roots for $\xi(t)$ to be approximately

$$N(T) \approx \frac{T}{2\pi} \log \frac{T}{2\pi} - \frac{T}{2\pi}$$

It can then be shown that the right hand side of the approximation is greater than a constant times $T \log T \; as \; T \to \infty$. Defining the number of roots as ρ, and the range as $0 < \operatorname{Im} s < T$, one can [12] find

$$N(T) = \pi^{-1}\vartheta(T) \; + \; 1 \; + \pi^{-1} \operatorname{Im} \int_C \frac{\zeta'(s)}{\zeta(s)} \, ds$$

Where Stirling's approximation once again applies to the $\vartheta(t)$ term and is defined as

$$\vartheta(T) = \frac{T}{2} \log \frac{T}{2\pi} - \frac{T}{2} - \frac{\pi}{8} + \frac{1}{48T} + \frac{7}{5760T^3} + \cdots$$

Using properties from $\xi\left(\frac{1}{2} + it\right)$ and $\log \xi\left(\frac{1}{2} + it\right)$, the two equations above can be combined [12] to show

$$N(T) - \left[\frac{T}{2\pi} \log \frac{T}{2\pi} - \frac{T}{2\pi}\right] = \left[-\frac{1}{8} + \frac{1}{48\pi T} + \cdots\right] + \; 1 \; + \pi^{-1} \operatorname{Im} \int_C \frac{\zeta'(s)}{\zeta(s)} \, ds$$

Backlund was then able to estimate the following bound

$$\left| N(T) - \left(\frac{T}{2\pi} \log \frac{T}{2\pi} - \frac{T}{2\pi} + \frac{7}{8}\right)\right| < 0.137 \; \log T \; + \; 0.443 \; \log \log T \; + \; 4.350 \qquad \forall \;\; T \geq 2$$

Another function, labeled $S(T)$, was discovered by John Littlewood. This function is defined as

$$S(T) = N(T) - \pi^{-1}\vartheta(T) - 1$$

Rearranging $S(T)$ and $N(T)$ creates the formation of $N(T) = \vartheta(T)\pi^{-1} + S(T) + 1$. Various other methods can be used to show that both $S(g_n)$ and $N(g_n)$ are bounded. Littlewood relied on these bounds to prove $N(g_n) \; \leq \; \pi^{-1}\vartheta(g_n) + 1 \; \leq \; n + 1$. This is a direct consequence of Gram's law, it can be shown that all of the zeroes between $0 < \operatorname{Im} s < T$ where $\Re(s) = 1/2$ are included through Gram points. Counting the number of good and bad Gram points is equivalent to verifying the Riemann hypothesis up to a given height where $N(T) = N(g_n)$.

While this is interesting, I have chosen not to implement Gram's law in any of my programs. Glen Pugh wrote a comprehensive program to calculate the first 12 million zeroes using both the Riemann−Siegel formula and Gram points. I find another part of Edward's book to be of much greater interest. This is something which is known as Lehmer's phenomenon. It is important to anyone who calculates zeroes for the zeta function, as it can be shown to nearly disprove the Riemann hypothesis. The phenomenon also provides an additional way to think about the Riemann hypothesis.

2.5 Lehmer's phenomenon

Lehmer's phenomenon was originally found by D. H. Lehmer while computing non-trivial zeroes for the zeta function in 1956. In his own estimates for the zeroes of $\zeta(s)$, Lehmer was only able to estimate the C_0 term in the Riemann-Siegel formula described above. While performing these calculations, he noticed that some of the sign changes between consecutive roots of $Z(t)$ appeared to be extremely small. This is now known as Lehmer's phenomenon. The phenomenon is a direct observation that the $|Z'(t)|$ is extremely small at consecutive t-values between two zeroes. It is important to describe Lehmer's observations in detail. As I will describe below, a single counter example can be shown to disprove the Riemann hypothesis. Therefore, $Z(t)$ must have a positive local maximum followed by a negative local minimum. Two consecutive zeroes, ρ_n and ρ_{n+1}, must have a local maximum or minimum between them which crosses the t-axis for the Riemann−Siegel $Z(t)$ function.

A direct theorem [[19], [30], [21]] states that if the RH is true, the graph of $Z'(t)/Z(t)$ must be monotonically decreasing between the zeroes of $Z(t)$ for all $t \geq t_o$. The first way to support this claim assumes that a contradictory statement is true. Suppose that $Z(t)$ has a positive local minimum or a negative local maximum between two consecutive roots labeled ρ_1 and ρ_2. By definition, $Z'(t)$ would have two consecutive zeroes for t_1 and t_2 in which $t_1 < t_2$ in (ρ_1, ρ_2). The direct calculation of $Z'(t)/Z(t)$ would also provide two distinct zeroes. However, the claim would suggest

$$\frac{Z'(t_1)}{Z(t_1)} > \frac{Z'(t_2)}{Z(t_2)}$$

This leads to a contradiction since $Z'(t_1) = Z'(t_2) = 0$. For all $t \geq t_o$, there must be precisely one zero of $Z'(t)$ between the two consecutive zeroes of $Z(t)$. Another way to think about this stems from the previously defined functional equation

$$\xi(s) = \frac{1}{2}s(s-1)\pi^{-s/2}\Gamma\left(\frac{s}{2}\right)\zeta(s)$$

where $\xi(s)$ is entire and can be shown to represent a symmetric mapping of the zeta function. Through applying the standard log rules, noting that $\int \frac{\xi'(s)}{\xi(s)}ds = \log(\xi(s)) + C(s)$, we can [30] write

$$\frac{\xi'(s)}{\xi(s)} = A + \sum_{\rho}\left(\frac{1}{s-\rho} + \frac{1}{\rho}\right), \quad A = \log 2 + \frac{1}{2}\log\pi - 1 - \frac{1}{2}C_0$$

where ρ is a zero for $\zeta(s)$ and C_0 is Euler's constant. It is customary to denote this constant by the lowercase Greek letter gamma,

$$\gamma = C_0 = \lim_{n\to\infty}\left(\sum_{k=1}^{n}\frac{1}{k} - \ln(n)\right) = \int_1^{\infty}\left(\frac{1}{\lfloor s\rfloor} - \frac{1}{s}\right)ds$$

An alternative way to write the function equation uses what is known as the Chi function. By [20], this is directly equivalent to

$$\zeta(s) = \chi(s)\zeta(1-s), \quad \chi(s) = 2^s\pi^{s-1}\sin\left(\frac{\pi s}{2}\right)\Gamma(1-s)$$

By applying the Riemann−Siegel $Z(t)$ function described inside the previous section, one can combine the functional equation and the Chi function [30] to produce

$$Z(t) = \chi^{-1/2}\left(\frac{1}{2}+it\right)\zeta\left(\frac{1}{2}+it\right) = \frac{\pi^{-it/2}\,\Gamma\left(\frac{1}{4}+\frac{1}{2}it\right)\zeta\left(\frac{1}{2}+it\right)}{\left|\Gamma\left(\frac{1}{4}+\frac{1}{2}it\right)\right|}$$

We can now write a new equation for $f(t)$. From the relationship for $Z(t)$ above,

$$\xi\left(\frac{1}{2}+it\right) = -f(t)Z(t)$$

and

$$f(t) = \frac{1}{2}\pi^{-1/4}\left(t^2+\frac{1}{4}\right)\left|\Gamma\left(\frac{1}{4}+\frac{1}{2}\,it\right)\right|$$

Using the log rules to differentiate generates

$$\frac{Z'(t)}{Z(t)} \;=\; -\,\frac{f'(t)}{f(t)} \;+\; i\frac{\xi'\left(\frac{1}{2}+it\right)}{\xi\left(\frac{1}{2}+it\right)}$$

If one substitutes the previous equation for $\xi'(s)/\xi(s)$, evaluating the roots as $\rho = \frac{1}{2}+i\gamma$ and $s = \frac{1}{2}+it$, the following relationship [30] can be derived

$$\left(i\frac{\xi'\left(\frac{1}{2}+it\right)}{\xi\left(\frac{1}{2}+it\right)}\right)' = -\sum_{\gamma}\frac{1}{(t-\gamma)^2} < -\,C(\log\log t)^2 \qquad\qquad \forall\ c>0,\ t\neq\gamma$$

As described in the start of the claim, $t \geq t_0$ for this to be true. One can then use Stirling's approximation for the gamma function to obtain

$$n! = \Gamma(n+1) \approx \frac{n^n}{e^n}\sqrt{2\pi n}$$

Combining both equations, it can be shown [30] that

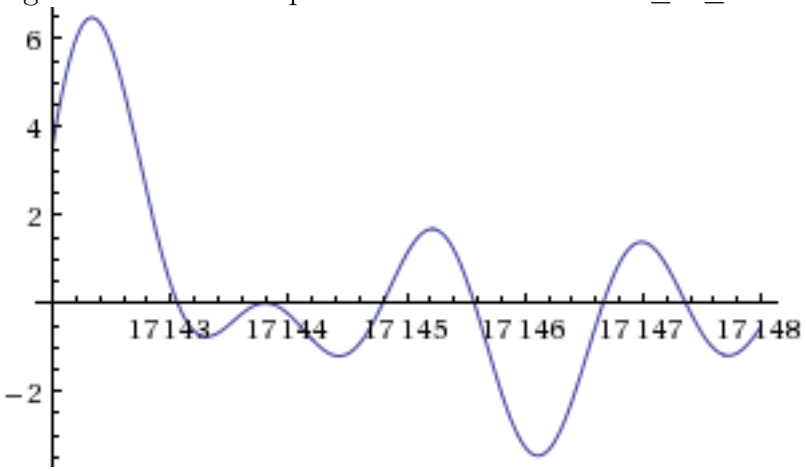
$$\frac{d}{dt}\left(\frac{f'(t)}{f(t)}\right) \;\ll\; \frac{1}{t}$$

and that

$$\left(\frac{Z'(t)}{Z(t)}\right)' \;<\; 0 \;\iff\; t \geq t_0$$

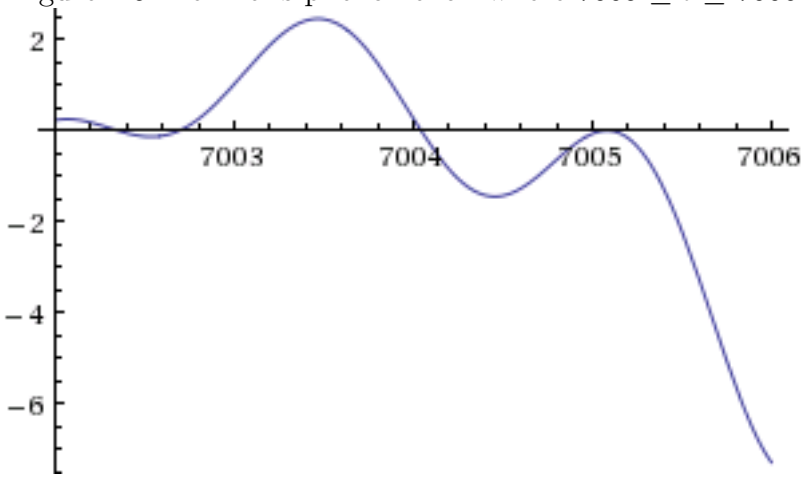
As $Z'(t)/Z(t)$ is always negative, it follows that the graph is monotonically decreasing for zeroes where $t \geq t_0$. Many sources choose to omit this analysis since the phenomenon itself is much easier to explain through visual methods. This can be best described through graphing values for $Z(t)$ where the hypothesis nearly fails. One of these values can be computed between t-values of $17143 \leq t \leq 17144$. A direct graph between $Z(t)$ and t shows the following

Figure 2.4: Lehmer's phenomenon where $17143 \leq t \leq 17144$



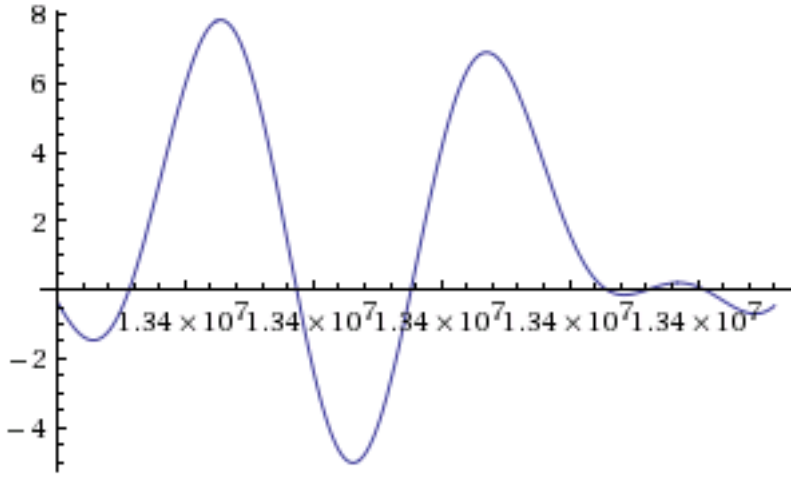
At around a t-value of 17143.8, the vertical height above the t axis is about 0.00397. Simply moving this value -0.00398 units downward would contradict the Riemann hypothesis. Another example occurs between the t-values of $7005 \leq t \leq 7006$.

Figure 2.5: Lehmer's phenomenon where $7005 \leq t \leq 7006$



To the best of my knowledge, numerical calculations have shown that this phenomenon is likely to continue up until $0 \leq t \leq \infty$. Rosser, Yohe, and Schoenfield [12] found a pair of zeroes which vary by only $4.4\times\ 10^{-4}$ near the vicinity of the 13,400,000th zero. I verified this through Mathematica and found that the two zeroes are near a t-value of 13,999,997.3.

Figure 2.6: Lehmer's phenomenon where $13999997 \leq t \leq 13999998$



Other results come from Odlyzko and Montgomery. In 1973, Montgomery provided a new theory which is known as Montgomery's conjecture [14]. Denoting $R_2(u)$ as the pair correlation function between non-trivial zeroes where $\Re(s) = 1/2$, one can write

$$R_2(u) = 1 - \operatorname{sinc}^2 u + \delta(u) = 1 - \left(\frac{\sin \pi u}{\pi u}\right)^2 + \delta(u), \qquad \delta_n = (\gamma_{n+1} - \gamma_n)\frac{\log \frac{\gamma_n}{2\pi}}{2\pi}$$

Odlyzko supported this conjecture through large scale computations of non-trivial zeroes for $\zeta(s)$ in the 1980s. The same pair correlation function is used for random Hermitian matrices. Since Hardy showed that the zeta function has an infinite number of zeroes on the critical line, one could assume that there is an infinite amount of zeroes that are close together.

All of this information made me engrossed with improving the program I wrote for the Riemann−Siegel formula. I made a few modifications and then checked to see where Lehmer's phenomenon would occur. I analyzed the first five million t-values values of $Z(t)$ and then computed zeroes in the range of $0 < Z(t) < 5000000$. I have posted a (fairly large) table of zeroes where $|\rho_{n+1} - \rho_n| < 0.025$ inside the last section of the appendix. The most interesting thing to think about with Lehmer's phenomenon deals with the spacing between zeroes. How can anyone be certain that they have found every non-trivial zero on the basis of output from a computer program? Perhaps it will be possible to modify root-finding algorithms so that they return every zero within the desired tolerance.

Chapter 3

Theory For Zeta Values

3.1 Zeta(k) for even k

Provided that n is a natural number and $|z| < \pi$, the following [36] series representation holds

$$z \cot z = z\frac{\cos z}{\sin z} = 1 - 2\sum_{n=1}^{\infty} \frac{z^2}{n^2\pi^2 - z^2}$$

Leonard Euler was able to reduce this representation through properties of the geometric sum and absolute convergence. Following these techniques, the above representation can be simplified [11] to

$$\begin{aligned} 1 - 2\sum_{n=1}^{\infty} \frac{z^2}{n^2\pi^2 - z^2} &= 1 - 2\sum_{n=1}^{\infty} \frac{z^2}{n^2\pi^2} \frac{1}{1 - (\frac{z}{n\pi})^2} \\ &= 1 - 2\sum_{n=1}^{\infty} \frac{z^2}{n^2\pi^2} \sum_{k=0}^{\infty} \Big(\frac{z}{n\pi}\Big)^{2k} \\ &= 1 - 2\Big(\sum_{n=1}^{\infty} \frac{1}{n^{2k+2}}\Big) \sum_{k=0}^{\infty} \frac{z^{2k+2}}{\pi^{2k+2}} \\ &= 1 - 2\Big(\sum_{n=1}^{\infty} \frac{1}{n^{2k}}\Big) \sum_{k=1}^{\infty} \frac{z^{2k}}{\pi^{2k}} \end{aligned}$$

Another way to represent $z \cot z$ involves Euler's formula. This leads to an alternative representation through what are known as Bernoulli numbers.

$$\begin{aligned} z \cot z = z\frac{\cos z}{\sin z} = iz\frac{2\cos z}{i2sinz} = iz\frac{e^{iz} + e^{-iz}}{e^{iz} - e^{-iz}} &= iz + \frac{2iz}{e^{2iz} - 1} \\ &= 1 + \sum_{k=2}^{\infty} B_k \frac{(2iz)^k}{k!} \end{aligned}$$

The B_k are known as Bernoulli numbers and are calculated [11] through the power series

$$\frac{z}{e^z - 1} = \sum_{k=0}^{\infty} B_k \frac{(z)^k}{k!} \qquad \forall \ \ |z| < 2\pi$$

The first few Bernoulli numbers B_k are

$$B_0 = 1 \qquad B_1 = -\frac{1}{2} \qquad B_2 = \frac{1}{6}$$
$$B_4 = -\frac{1}{30} \qquad B_6 = \frac{1}{42} \qquad B_8 = -\frac{1}{30}$$
$$B_{10} = \frac{5}{66} \qquad B_{12} = \frac{-691}{2730} \qquad B_{14} = \frac{7}{6}$$
$$B_{16} = -\frac{3617}{510} \qquad B_{18} = \frac{43867}{798} \qquad B_{20} = -\frac{174611}{330}$$

By comparing the two values of $z \cot z$ written above, we can write

$$z \cot z = 1 - 2\Big(\zeta(2k)\Big) \sum_{k=1}^{\infty} \frac{z^{2k}}{\pi^{2k}} = 1 + \sum_{k=2}^{\infty} B_k \frac{(2iz)^k}{k!}$$

The coefficients of powers for z^k form the known relationship

$$\zeta(2k) = \frac{(-1)^{k+1} B_{2k} (2\pi)^{2k}}{2(2k)!}$$

The value of the first even positive integer, $\zeta(2)$, is known as the Basel problem and is arguably the most famous of all values for $\zeta(s)$. Euler and others have developed multiple solutions to the Basel problem. The one that is most interesting to me includes three precursory results. The first of these results [10] references the following identity

$$\frac{1}{2} \left(sin^{-1} x\right)^2 = \int_0^x \frac{\sin^{-1} t}{\sqrt{1-t^2}} dt$$

which follows from the u substitution of $u = sin^{-1} t$. The second result deals with the series expansion for $\sin^{-1} x$. Noting that

$$\sin^{-1} x = \int_0^x \frac{1}{\sqrt{1-t^2}} dt = \int_0^x \left(1-t^2\right)^{-1/2} dt$$

and that Isaac Newton wrote the general form of the binomial series as

$$(1+x)^r = 1 + rx + \frac{r(r-1)}{2 \cdot 1} x^2 + \frac{r(r-1)(r-2)}{3 \cdot 2 \cdot 1} x^3 + \frac{r(r-1)(r-2)(r-3)}{4 \cdot 3 \cdot 2 \cdot 1} x^4 + \cdots$$

Inserting Newton's series into the previous equation produces

$$\begin{aligned}
\int_0^x \left(1-t^2\right)^{-1/2} dt &= \int_0^x \Big(1 + \frac{1}{2} t^2 + \frac{1 \cdot 3}{2^2 \cdot 2!} t^4 + \frac{1 \cdot 3 \cdot 5}{2^3 \cdot 3!} t^6 + \frac{1 \cdot 3 \cdot 5 \cdot 7}{2^4 \cdot 4!} t^8 + \cdots \Big) dt \\
&= t + \frac{1}{2} \times \frac{t^3}{3} + \frac{1 \cdot 3}{2 \cdot 4} \times \frac{t^5}{5} + \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} \times \frac{t^7}{7} + \frac{1 \cdot 3 \cdot 5 \cdot 7}{2 \cdot 4 \cdot 6 \cdot 8} \times \frac{t^9}{9} + \cdots \Big|_0^x \\
&= x + \frac{1}{2} \times \frac{x^3}{3} + \frac{1 \cdot 3}{2 \cdot 4} \times \frac{x^5}{5} + \frac{1 \cdot 3 \cdot 5}{2 \cdot 4 \cdot 6} \times \frac{x^7}{7} + \frac{1 \cdot 3 \cdot 5 \cdot 7}{2 \cdot 4 \cdot 6 \cdot 8} \times \frac{x^9}{9} + \cdots
\end{aligned}$$

The expanded series of $sin^{-1} x$ is the second result. The third and final result is taken [10] from another integral relationship.

$$\int_0^1 \frac{t^{n+2}}{\sqrt{1-t^2}} dt = \frac{n+1}{n+2} \int_0^1 \frac{t^n}{\sqrt{1-t^2}} dt \qquad \forall \; n \geq 1$$

Defining J as

$$J = \int_0^1 \frac{t^{n+2}}{\sqrt{1-t^2}} dt$$

Allows one to apply integration by parts substituting $u = t^{n+1}$ and $dv = \left(t/\sqrt{1-t^2}\right) dt$ to form

$$\begin{aligned} J &= \left(-t^{n+1}\sqrt{1-t^2}\right)\Big|_0^1 + (n+1)\int_0^1 t^n\sqrt{1-t^2}\, dt \\ &= 0 + (n+1)\int_0^1 \frac{t^n\left(1-t^2\right)}{\sqrt{1-t^2}}\, dt \\ &= (n+1)\int_0^1 \frac{t^n}{\sqrt{1-t^2}}\, dt - (n+1)J \end{aligned}$$

Moving J to the RHS produces

$$\begin{aligned} J + (n+1)J &= (n+1)\int_0^1 \frac{t^n}{\sqrt{1-t^2}}\, dt \\ (n+2)J &= (n+1)\int_0^1 \frac{t^n}{\sqrt{1-t^2}}\, dt \\ J &= \frac{n+1}{n+2}\int_0^1 \frac{t^n}{\sqrt{1-t^2}}\, dt \end{aligned}$$

By which the definition of J is the third result. Combining all three of these results will provide a rather ingenuous solution to the Basel problem. The first step refers back to the identity for $1/2(\sin^{-1} x)^2$. Replacing x with 1 produces

$$\frac{\pi^2}{8} = \frac{1}{2}\left(sin^{-1}1\right)^2 = \int_0^1 \frac{sin^{-1}t}{\sqrt{1-t^2}} dt$$

The second step deals with the series expansion that was found inside the second result. Replacing $sin^{-1}t$ with the Newtonian series expansion generates

$$\begin{aligned} \frac{\pi^2}{8} = &\int_0^1 \frac{t}{\sqrt{1-t^2}} dt + \frac{1}{2\cdot 3}\int_0^1 \frac{t^3}{\sqrt{1-t^2}} dt + \frac{1\cdot 3}{2\cdot 4\cdot 5}\int_0^1 \frac{t^5}{\sqrt{1-t^2}} dt \\ &+ \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6\cdot 7}\int_0^1 \frac{t^7}{\sqrt{1-t^2}} dt + \cdots \end{aligned}$$

Knowing that

$$\int_0^1 \frac{t}{\sqrt{1-t^2}} dt = 1$$

and that the third relationship [10] provides a way to recursively calculate $\pi^2/8$ through J, one can write

$$\begin{aligned} \frac{\pi^2}{8} &= 1 + \frac{1}{2\cdot 3}\left[\frac{2}{3}\right] + \frac{1\cdot 3}{2\cdot 4\cdot 5}\left[\frac{2}{3}\times\frac{4}{5}\right] + \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6\cdot 7}\left[\frac{2}{3}\times\frac{4}{5}\times\frac{6}{7}\right] + \cdots \\ &= 1 + \frac{1}{9} + \frac{1}{25} + \frac{1}{49} + \cdots \end{aligned}$$

Which is is a general summation formula for the reciprocal of odd squares. Euler recognized that this could be used to show that

$$\begin{aligned}\sum_{k=1}^{\infty}\frac{1}{k^2} &= \left[1+\frac{1}{9}+\frac{1}{25}+\frac{1}{49}+\cdots\right]+\left[\frac{1}{4}+\frac{1}{16}+\frac{1}{36}+\frac{1}{64}+\cdots\right]\\ &= \left[1+\frac{1}{9}+\frac{1}{25}+\frac{1}{49}+\cdots\right]+\frac{1}{4}\left[\frac{1}{4}+\frac{1}{9}+\frac{1}{16}+\frac{1}{25}+\cdots\right]\\ &= \frac{\pi^2}{8}+\frac{1}{4}\sum_{k=1}^{\infty}\frac{1}{k^2}\end{aligned}$$

Rearranging terms produces

$$\sum_{k=1}^{\infty}\frac{1}{k^2} = \frac{\pi^2}{8}+\frac{1}{4}\sum_{k=1}^{\infty}\frac{1}{k^2}$$

$$\sum_{k=1}^{\infty}\frac{1}{k^2} - \frac{1}{4}\sum_{k=1}^{\infty}\frac{1}{k^2} = \frac{\pi^2}{8}$$

$$\frac{3}{4}\sum_{k=1}^{\infty}\frac{1}{k^2} = \frac{\pi^2}{8}$$

Thus

$$\sum_{k=1}^{\infty}\frac{1}{k^2} = \frac{\pi^2}{6}$$

Another interesting fact deals with the calculation of $\zeta(-2k)$. Starting at

$$\zeta(s) = 2^s\pi^{s-1}\sin\left(\frac{\pi s}{2}\right)\Gamma(1-s)\zeta(1-s)$$

Substitute $s=-2k$ to form

$$\begin{aligned}\zeta(-2k) &= -2^{-2k}\pi^{-2k-1}\sin(\pi k)\Gamma(1+2k)\zeta(1+2k)\\ &= -2^{-2k}\pi^{-2k-1}\sin(\pi k)\cdot(2k)!\cdot\zeta(1+2k)\end{aligned}$$

Because all other functions are finite, $\zeta(-2k)=0$ for all $k\in\mathbb{N}^+$. The sine function will always be zero at multiples of $k\pi$. A slightly more interesting fact looks at the value of $\zeta'(-2k)$. Apply the finite argument for the derivative [17] of the functional equation,

$$\zeta'(s) = 2^s\pi^{s-1}\left(\frac{1}{2}\pi\cos\left(\frac{\pi s}{2}\right)\right)\Gamma(1-s)\zeta(1-s)$$

which can be extended to $\zeta'(-2k)$ to produce

$$\begin{aligned}\zeta'(-2k) &= \frac{1}{2}\cdot 2^{-2k}\pi^{-2k-1}\pi\cos(\pi k)\Gamma(1+2k)\zeta(1+2k)\\ &= \frac{1}{2}\cdot(2\pi)^{-2k}\cos(\pi k)\Gamma(1+2k)\zeta(1+2k)\\ &= \frac{(-1)^k}{2(2\pi)^{2k}}\cdot(2k)!\cdot\zeta(1+2k)\end{aligned}$$

Some initial values for these first derivatives include

$$\zeta'(-2) = -\frac{\zeta(3)}{4\pi^2}, \quad \zeta'(-4) = \frac{3}{4\pi^4}\zeta(5), \quad \zeta'(-6) = -\frac{45}{8\pi^6}\zeta(7), \quad \zeta'(-8) = \frac{315}{4\pi^8}\zeta(9)$$

The Java program I created applies the Bernoulli numbers to calculate the values for $\zeta(2k)$. This was best handled by applying the BigFraction class which is included inside the Apache Commons Math Library. Another method uses the BigRational class in order to calculate the Bernoulli numbers. For anyone who is curious, there is about seven or eight different ways to calculate the Bernoulli numbers in Java.

While reading about alternative ways to find the summation formula for $\zeta(2k)$, I found [9]

$$\zeta(2k) = \frac{(-1)^{k+1}(2\pi)^{2k}}{2(2k)!}\frac{1}{(k+1)(2k+1)}\left(k - \sum_{i=1}^{k-1}\binom{2k+2}{2i}B_{2i}\right)$$

which could be used to evaluate $\zeta(2k)$ differently.

3.2 Zeta(k) for odd k

Starting from the standard form of the functional equation

$$\begin{aligned}\zeta(s) &= 2^s\pi^{s-1}\ \sin\left(\frac{\pi s}{2}\right)\ \Gamma(1-s)\ \zeta(1-s)\\ &= 2(2\pi)^{s-1}\ \sin\left(\frac{\pi s}{2}\right)\ \Gamma(1-s)\ \zeta(1-s)\end{aligned}$$

one can substitute $s = (-2k+1)$ to form

$$\begin{aligned}\zeta(-2k+1) &= 2(2\pi)^{-2k}\ \sin\left(\frac{\pi\,(-2k+1)}{2}\right)\ \Gamma(2k)\ \zeta(2k)\\ &= 2(2\pi)^{-2k}\ (-1)^k\ (2k-1)!\,\zeta(2k)\end{aligned}$$

Which can be used to evaluate $\zeta(s)$ for negative odd integers. To take this one step further, Leonard Euler developed a method that will calculate odd integer values for $\zeta(-k)$. Starting with the identity

$$t + t^2 + t^3 + t^4 + \cdots = (t - t^2 + t^3 - t^4 + \cdots) + 2(t^2 + t^4 + t^6 + t^8 + \cdots)$$

one can apply the operator $(t\frac{d}{dt})^k$, differentiating first [37] and then multiplying by t for a total of k times to form

$$1^k t + 2^k t^2 + 3^k t^3 + 4^k t^4 + \cdots = \left(t\frac{d}{dt}\right)^k\left(\frac{t}{1+t}\right) + 2^{k+1}(1^k t^2 + 2^k t^4 + 3^k t^6 + 4^k t^8 + \cdots)$$

Heuristically, evaluating this new equation at $t = 1$ produces

$$\zeta(-k) = \left[\left(t\frac{d}{dt}\right)^k\left(\frac{t}{1+t}\right)\right]_{t=1} + 2^{k+1}(\zeta(-k))$$

such that

$$\zeta(-k) - 2^{k+1}(\zeta(-k)) = \left[\left(t\frac{d}{dt}\right)^k\left(\frac{t}{1+t}\right)\right]_{t=1}$$

Thus

$$\zeta(-k) = (1-2^{k+1})^{-1}\left[\left(t\frac{d}{dt}\right)^k\left(\frac{t}{1+t}\right)\right]_{t=1}$$

Let $t = e^X$ and observe that $t\frac{d}{dt} = \frac{d}{dX}$ forms

$$\zeta(-k)(1-2^{k+1}) = \left[\left(\frac{d^k}{dX^k}\right)\left(\frac{e^X}{e^X+1}\right)\right]_{X=0} \qquad \forall\ k \in \mathbb{N}$$

The above equation is a Taylor series,

$$\frac{e^X}{e^X+1} = \sum_{n=0}^{\infty}\frac{(1-2^{k+1})\zeta(-k)}{k!}X^k$$

This can be rewritten [37] as the function of a complex variable, $X = 2\pi iz$, producing

$$F(z) = \frac{e^{2\pi iz}}{e^{2\pi iz}+1} = \sum_{k=0}^{\infty}\frac{(1-2^{k+1})\zeta(-k)}{k!}(2\pi iz)^k = \sum_{k=0}^{\infty}\frac{(1-2^{k+1})\zeta(-k)\,(2\pi i)^k}{k!}(z)^k$$

Where $F(z)$ is a complex function which generates values for $\zeta(-k)$ across all natural numbers. A new function, G, can be introduced such that $G = \pi\cot\pi z$. This produces

$$\begin{aligned} G(z) = \pi\cot\pi z = \frac{i\pi\,(1+e^{2i\pi z})}{(-1+e^{i\pi z})\,(1+e^{i\pi z})} &= \pi i\frac{(e^{2i\pi z}+1)}{(e^{2i\pi z}-1)} \\ &= \frac{1}{z} - 2\sum_{k=1}^{\infty}\zeta(2k)z^{2k-1} \end{aligned}$$

From which both F(z) and G(z) are fractional linear functions of $e^{2\pi iz}$. Comparing the two functions in the linear form of $e^{2\pi iz}$ forms

$$F(z) = \frac{e^{2\pi iz}}{e^{2\pi iz}+1}, \qquad G(z) = \pi i\frac{(e^{2i\pi z}+1)}{(e^{2i\pi z}-1)}$$

$$\frac{1}{\pi i}\Big(G(z) - 2G(2z)\Big) = -F(z) + F(-z)$$

Euler recognized that these two complex functions are related to $\zeta(1-k)$ and $\zeta(k)$ for even $k \geq 2$. He equated the coefficients [37] between F and G to produce

$$\zeta(1-k) = 2\frac{\Gamma(k)}{(2\pi i)^k}\zeta(k) \qquad \forall\ even\ k \geq 2$$

A known relationship for $\zeta(k)$ can be stated as

$$\zeta(k) = -\frac{1}{2}\cdot\frac{(2\pi ik)^k B_k}{k!} \qquad \forall\ even\ k \geq 2$$

Compare the two equations for $\zeta(k)$ and $\zeta(1-k)$,

$$\zeta(1-k) = 2\frac{\Gamma(k)}{(2\pi i)^k}\left(-\frac{1}{2}\cdot\frac{(2\pi ik)^k B_k}{k!}\right) = (k-1)!\left(-\frac{B_k}{k!}\right) = -\frac{B_k}{k} \qquad \forall\ even\ k \geq 2$$

Substituting $k = k+1$ leads to

$$\zeta(-k) = -\frac{B_{k+1}}{k+1} \qquad \forall\ k \in \mathbb{N}$$

An alternative derivation is shown in [15].

I created a new method inside Java to calculate these negative Bernoulli numbers. Odd values of $\zeta(2k+1)$ where $k > 1$ are unknown in the sense of a general formula involving π. Known relationships have been formed to compute

$$\zeta(3) = \sum_{k=1}^{\infty} \frac{1}{k^3} = \frac{5}{2} \sum_{k=1}^{\infty} \frac{(-1)^k}{k^3 \binom{2k}{k}}, \quad \zeta(5) = \sum_{k=1}^{\infty} \frac{1}{k^5} = \frac{32}{31} \sum_{k=0}^{\infty} \frac{1}{(1+2k)^5}$$

While no one has been able to find an exact value for $\zeta(3)$ involving π, Leonard Euler [10] successfully analyzed the following series to form

$$\sum_{k=0}^{\infty} (-1)^k \frac{1}{(2k+1)^3} = 1 - \frac{1}{27} + \frac{1}{125} - \frac{1}{343} + \cdots = \frac{\pi^3}{32}$$

3.3 Some thoughts about Apéry's constant

The Dirichlet eta function is defined as

$$\eta(s) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n^s} \qquad \forall \ \Re(s) > 0$$

It is straightforward to derive the following relationship between the zeta function [18] and the Dirichlet eta function.

$$\begin{aligned} \zeta(s) &= \sum_{n=1}^{\infty} \frac{1}{n^s} \\ &= \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n^s} + 2 \sum_{n=1}^{\infty} \frac{1}{(2n)^s} \\ &= \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n^s} + 2^{1-s}\, \zeta(s) \end{aligned}$$

the end result is

$$(1 - 2^{1-s})\zeta(s) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n^s} \qquad \forall \ \Re(s) > 1$$

I noticed a similar pattern when analyzing individual terms in $\zeta(3)$. Observe that the $1/8$ factors with all other even terms.

$$\begin{aligned} \zeta(3) = \sum_{n=1}^{\infty} \frac{1}{n^3} &= 1 + \frac{1}{8} + \frac{1}{27} + \frac{1}{64} + \frac{1}{125} + \frac{1}{216} + \frac{1}{343} + \frac{1}{512} + \frac{1}{729} + \frac{1}{1000} + \cdots \\ &= \frac{1}{8} \left(1 + \frac{1}{8} + \frac{1}{27} + \frac{1}{64} + \frac{1}{125} + \frac{1}{216} + \cdots \right) + \left(1 + \frac{1}{27} + \frac{1}{125} + \frac{1}{343} + \frac{1}{729} + \cdots \right) \\ &= \frac{1}{8}\, \zeta(3) + \left(1 + \frac{1}{27} + \frac{1}{125} + \frac{1}{343} + \frac{1}{729} + \cdots \right) \\ &= \frac{1}{8}\, \zeta(3) + \sum_{n=0}^{\infty} \frac{1}{(2n+1)^3} \\ &= \frac{8}{7} \sum_{n=0}^{\infty} \frac{1}{(2n+1)^3} \end{aligned}$$

Starting from the previous step, follow the method displayed above and insert Euler's summation formula.

$$\begin{aligned}\zeta(3) &= \frac{1}{8}\,\zeta(3) + \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)^3} + 2\sum_{n=0}^{\infty} \frac{1}{(4n+3)^3} \\ &= \frac{1}{8}\,\zeta(3) + \frac{\pi^3}{32} + 2\sum_{n=0}^{\infty} \frac{1}{(4n+3)^3}\end{aligned}$$

Unfortunately, this doesn't appear to lead anywhere. Reducing the equation further produces

$$\frac{7}{16}\,\zeta(3) - \frac{\pi^3}{64} = \sum_{n=0}^{\infty} \frac{1}{(4n+3)^3}$$

by which one can change the infinite sum on the RHS to form

$$\frac{7}{16}\,\zeta(3) - \frac{\pi^3}{64} = \sum_{n=1}^{\infty} \frac{1}{(4n-1)^3}$$

Apply the same procedure again and expand the alternating zeta function,

$$\frac{1}{64}\left(28\,\zeta(3) - \pi^3\right) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{(4n-1)^3} + 2\sum_{n=1}^{\infty} \frac{1}{(8n-1)^3}$$

Notice that the RHS of the new equation can be expanded by the Hurwitz zeta function.

$$\begin{aligned}\frac{1}{64}\left(28\,\zeta(3) - \pi^3\right) &= \frac{1}{512}\left(\zeta\left(3, \frac{3}{8}\right) - \zeta\left(3, \frac{7}{8}\right)\right) + 2\sum_{n=1}^{\infty} \frac{1}{(8n-1)^3} \\ 8\left(28\,\zeta(3) - \pi^3\right) &= \left(\zeta\left(3, \frac{3}{8}\right) - \zeta\left(3, \frac{7}{8}\right)\right) + 1024\sum_{n=1}^{\infty} \frac{1}{(8n-1)^3} \\ \zeta(3) &= \frac{\pi^3}{28} + \frac{1}{224}\left(\zeta\left(3, \frac{3}{8}\right) - \zeta\left(3, \frac{7}{8}\right)\right) + \frac{32}{7}\sum_{n=1}^{\infty} \frac{1}{(8n-1)^3}\end{aligned}$$

Although, this also doesn't appear to lead anywhere. The Hurwitz zeta function is derived [27] through the multiplication theorem. Starting with

$$\zeta(s, q) = \sum_{n=0}^{\infty} \frac{1}{(q+n)^s} \qquad\qquad \forall\ \Re(s) > 1, \Re(q) > 0$$

substitute mq in place of q

$$\begin{aligned}\zeta(s, mq) &= \sum_{n=0}^{\infty} \frac{1}{(mq+n)^s} \\ &= \frac{1}{m^s}\sum_{n=0}^{\infty} \frac{1}{(q+\frac{n}{m})^s}\end{aligned}$$

Next, write $n = n'm + n''$, where n' will range from 0 to ∞ and n'' will range from 0 to $m-1$. Putting this above gives

$$\begin{aligned}\zeta(s, mq) &= \frac{1}{m^s} \sum_{\substack{n'\geq 0 \\ n'' \bmod m}} \frac{1}{(q+\frac{n''}{m}+n')^s} \\ &= \frac{1}{m^s}\sum_{k=0}^{m-1}\sum_{n'\geq 0} \frac{1}{(q+\frac{k}{m}+n')^s} \\ &= \frac{1}{m^s}\sum_{k=0}^{m-1} \zeta\left(s, q+\frac{k}{m}\right)\end{aligned}$$

A more simplified form deals with setting $q = 0$,

$$\zeta(s) = \frac{1}{m^s} \sum_{k=1}^{m} \zeta\left(s,\ \frac{k}{m}\right)$$

I observed a rather intriguing similarity when analyzing a different form of $\zeta(3)$. Writing out $\zeta(3)$ and $\zeta(6)$,

$$\zeta(3) = \sum_{n=1}^{\infty} \frac{1}{n^3} = 1 + \frac{1}{8} + \frac{1}{27} + \frac{1}{64} + \frac{1}{125} + \frac{1}{216} + \frac{1}{343} + \frac{1}{512} + \frac{1}{729} + \frac{1}{1000} + \cdots$$

$$\zeta(6) = \sum_{n=1}^{\infty} \frac{1}{n^6} = 1 + \frac{1}{64} + \frac{1}{729} + \frac{1}{4096} + \frac{1}{15625} + \frac{1}{46656} + \frac{1}{117649} + \frac{1}{262144} + \cdots$$

Notice that all of the square terms cancel because

$$\zeta(6) = \sum_{n=1}^{\infty} \frac{1}{n^6} = \sum_{n=1}^{\infty} \frac{1}{(n^3)^2}$$

This allows one to write $\zeta(3)$ as

$$\begin{aligned}
\zeta(3) = \sum_{n=1}^{\infty} \frac{1}{n^3} &= \left(1 + \frac{1}{64} + \frac{1}{729} + \frac{1}{4096} + \frac{1}{15625} + \cdots\right) + \left(\frac{1}{8} + \frac{1}{27} + \frac{1}{125} + \frac{1}{216} + \frac{1}{343} + \cdots\right) \\
&= \zeta(6) + \left(\frac{1}{8} + \frac{1}{27} + \frac{1}{125} + \frac{1}{216} + \frac{1}{343} + \cdots\right) \\
&= \frac{\pi^6}{945} + \left(\frac{1}{8} + \frac{1}{27} + \frac{1}{125} + \frac{1}{216} + \frac{1}{343} + \cdots\right)
\end{aligned}$$

which reduces the problem to finding the sum of non squares,

$$\zeta(3) - \frac{\pi^6}{945} = \left(\frac{1}{8} + \frac{1}{27} + \frac{1}{125} + \frac{1}{216} + \frac{1}{343} + \cdots\right)$$

The sequence of numbers inside the denominator of the RHS match OEIS A179125.

$$\zeta(3) - \frac{\pi^6}{945} = \sum_{n=1}^{\infty} \frac{1}{\left(n + \left\lfloor 1/2 + \sqrt{n} \right\rfloor\right)^3}$$

After spending several days investigating the various sequences, I was unable to find a pattern that would reduce the equation further.

The Bernoulli numbers are used to form integral representations for the zeta function. Two notable integral representations are defined by

$$n!\,\zeta(n+1) = \int_0^{\infty} \frac{t^n}{e^t - 1} dt, \qquad \zeta(3) = \frac{1}{2} \int_0^{\infty} \frac{t^2}{e^t - 1} dt$$

and

$$n!\,\eta(n+1) = \int_0^{\infty} \frac{t^n}{e^t + 1} dt, \qquad \eta(3) = \frac{1}{2} \int_0^{\infty} \frac{t^2}{e^t + 1} dt$$

These are useful because they form a relationship between the zeta function, factorial function, and the alternating zeta function.

A final integral representation [25] between the zeta function and Bernoulli numbers is defined as

$$\zeta(2n+1) = \frac{(-1)^{n+1}(2\pi)^{2n+1}}{2(2n+1)!} \int_0^1 B_{2n+1}(x) \cot(\pi x) dx$$

by which $\zeta(3)$ forms

$$\zeta(3) = \frac{(2\pi)^3}{2(3)!} \int_0^1 B_3(x) \cot(\pi x) dx$$

Chapter 4

Appendices

4.1 Appendix A - Proof of the Euler Product Formula

The Euler Product Formula states that

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_{p \in P} \frac{1}{1 - p^{-s}} \qquad \forall\ \Re(s) > 1$$

Starting from the RHS, one can write

$$\prod_{p \in P} \frac{1}{1 - p^{-s}} = \prod_{i=1} \sum_{k_i\,=\,0}^{\infty} \left(\frac{1}{p_i^s}\right)^{k_i} = \prod_{i=1} \sum_{k_i\,=\,0}^{\infty} \frac{1}{p_i^{sk_i}}$$

Where the second step follows from the geometric sum of $(1 - p^{-s})^{-1}$ and is valid since $|p^{-s}| < 1$. Expanding the third equation generates

$$\prod_{i=1} \sum_{k_i\,=\,0}^{\infty} \frac{1}{p_i^{sk_i}} = \sum_{k_1\,=\,0}^{\infty} \frac{1}{p_1^{sk_1}} \cdot \sum_{k_2\,=\,0}^{\infty} \frac{1}{p_2^{sk_2}} \cdots \sum_{k_n\,=\,0}^{\infty} \frac{1}{p_n^{sk_n}} \cdots$$

The RHS can be reduced, rather cleverly to

$$\sum_{k_1\,=\,0}^{\infty} \frac{1}{p_1^{sk_1}} \cdot \sum_{k_2\,=\,0}^{\infty} \frac{1}{p_2^{sk_2}} \cdots \sum_{k_n\,=\,0}^{\infty} \frac{1}{p_n^{sk_n}} \cdots = \sum_{k_1\,=\,0}^{\infty} \sum_{k_2\,=\,0}^{\infty} \cdots \sum_{k_n\,=\,0}^{\infty} \cdots \left(\frac{1}{p_1^{k_1}} \frac{1}{p_2^{k_2}} \cdots \frac{1}{p_n^{k_n}} \cdots\right)^s$$

The last step creates an infinite sum for the inverse of prime numbers. The fundamental theorem of arithmetic states that if n is a natural number, it is possible to write n as a product of primes with corresponding powers. The product represented this way is unique. To match the above equation, we can reference n as

$$n = p_1^{k_1} p_2^{k_2} \cdots p_n^{k_n} = \prod_{i=1}^{n} p_i^{k_i}$$

Because the product is unique, it is possible to represent all integers greater than one as a product of prime numbers. Taking the inverse of the product of prime numbers would create the inverse for the summation of integers. This reduces the equation to

$$\sum_{k_1\,=\,0}^{\infty} \sum_{k_2\,=\,0}^{\infty} \cdots \sum_{k_n\,=\,0}^{\infty} \cdots \left(\frac{1}{p_1^{k_1}} \frac{1}{p_2^{k_2}} \cdots \frac{1}{p_n^{k_n}} \cdots\right)^s = \sum_{k=1}^{\infty} \frac{1}{k^s} = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

The Euler Product formula shows that there exists an infinite amount of prime numbers. It also provides a way of thinking about the relationship between the inverse summation of integers and the distinct representation of prime numbers.

4.2 Appendix B - Derivation of the Cauchy-Schlomilch transformation for zeta(s)

Starting from the well known form of the zeta function,

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} \qquad \forall\ \sigma > 1$$

one can deduce the [2] following relationship between the gamma function and the zeta function.

$$\zeta(s) = \frac{1}{(1-2^{1-s})\,\Gamma(s)} \int_0^{\infty} \frac{t^{s-1}}{1+e^t} dt$$

Substituting $t = y^2$ produces

$$\zeta(s) = \frac{2}{(1-2^{1-s})\,\Gamma(s)} \int_0^{\infty} \frac{y^{2s-1}}{1+e^{y^2}} dy$$

such that

$$\frac{1}{2}\left(1-2^{1-s}\right)\Gamma(s)\zeta(s) = \int_0^{\infty} \frac{y^{2s-1}}{1+e^{y^2}} dy$$

We then need to rely on a theorem which was developed by Cauchy and Schlömilch. Assume that $a, b > 0$ and let f be a continuous function in which the below intervals converge. This is known as the Cauchy−Schlömilch transformation, it states that

$$\int_0^{\infty} f\Big(\left(ax - bx^{-1}\right)^2\Big) dx = \frac{1}{a}\int_0^{\infty} f(y^2) dy$$

The RHS of the transformation corresponds to the integral above for $\int_0^{\infty} \frac{y^{2s-1}}{1+e^{y^2}} dy$. By [2], the transformation changes the integral relationship to

$$\int_0^{\infty} \frac{x^{2s+1}}{cosh^2\left(x^2\right)} dx = 2^{-s}\left(1-2^{1-s}\right)\Gamma(s+1)\zeta(s)$$

Isolating the terms for $\zeta(s)$ generates

$$\zeta(s) = \frac{2^s}{(1-2^{1-s})\,\Gamma(s+1)} \int_0^{\infty} \frac{x^{2s+1}}{cosh^2\left(x^2\right)} dx$$

The change of variables for $t = x^2$ produces

$$\zeta(s) = \frac{2^{s-1}}{(1-2^{1-s})\,\Gamma(s+1)} \int_0^{\infty} \frac{t^s}{cosh^2\left(t\right)} dt \qquad \forall\ \sigma > -1$$

which is the known result for the Cauchy−Schlömilch transformation of the zeta function. Another interesting result [26] from the transformation is

$$\eta(s) = \frac{1}{(2^{1-s})\,\Gamma(s+1)} \int_0^{\infty} \frac{t^s}{cosh^2\left(t\right)} dt \qquad \forall\ \sigma > -1$$

The relationship is formed through substituting the alternating zeta function,

$$\eta(s) = \left(1-2^{1-s}\right)\zeta(s)$$

4.3 Appendix C - Java program overview

• **CauchySchlomich.java** was the first program I developed after initially learning about the zeta function. I was cognizant of the standard summation formula and the known integral relationship between $\zeta(s)$ and $\Gamma(s)$,

$$\zeta(s) = \frac{1}{\Gamma(s)} \int_0^\infty \frac{x^{s-1}}{e^x - 1} \mathrm{d}x \qquad \forall \;\; \Re(s) > 1$$

I didn't use this relationship because I knew that $-1 < \Re(s) \leq 0$ would be problematic and would require a separate function inside Java. The Cauchy−Schlömilch transformation is much more interesting and can be extended to all values for $s \in \mathbb{R}$ outside of the simple pole at $s = 1$. I ran across three different complications while writing the Java program. The first problem dealt with evaluating all functions through the double data type in Java. As a reference,

$$Java\ double\ maximum = 1.797 \times 10^{308}$$
$$Java\ double\ minimum = 4.9 \times 10^{-324}$$

The functional equation will reach this rather quickly since $\Gamma(1 - s)$ will go above the maximum. $\Gamma(200)$ is 3.94×10^{372} and would not be supported by the double data type. I could have worked around this by initially writing the functional equation and gamma approximation in BigDecimal format. These two methods are labeled riemannFuncForm and lancGamma in my Java program. The second problem dealt with the approximation of $\int_0^\infty t^s / \cosh^2(t)\, dt$ through Simpson's rule. Because of the functional equation, it is only necessary to apply the Cauchy−Schlömilch transformation between $0 \leq \Re(s) < 2$. As I explained inside the third section, I could have broke the integral into $\int_0^1 t^{1/2} \mathrm{sech}^2(t)\, dt$ and then applied the power series approximation. It is not necessary to do this as the range of integration is isolated to a small subset.

The third problem dealt with the approximation of the gamma function. I originally used the Lanczos approximation,

$$\Gamma(z+1) = \sqrt{2\pi}\left(z + g + \frac{1}{2}\right)^{z+\frac{1}{2}} e^{-\left(z+g+\frac{1}{2}\right)} A_g(z)$$
$$A_g(z) = \frac{1}{2}p_0(g) + p_1(g)\frac{z}{z+1} + p_2(g)\frac{z(z-1)}{(z+1)(z+2)} + \cdots$$

After I realized that I was unable to calculate $s < -141$, I switched to Stirling's approximation

$$\Gamma(n+1) = n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

which extended the calculation to $s > -170$. My program does not support calculations where s is less than -170. I could have improved the program by writing every method inside the BigDecimal class in Java.

• **RiemannSiegel.java** was the second program I developed in Java. It is important to note the the main difficulty of Riemann−Siegel formula deals with the calculation of the remainder terms. A condensed form of the Riemann−Siegel formula for $Z(t)$ is

$$Z(t) = 2 \sum_{n^2 < t/2\pi} n^{-1/2} \Big[\cos(\theta(t) - t \log n)\Big] + R$$

where R is the remainder term. My original intention was to calculate the remainder terms through writing nine different finite difference approximations. It was necessary to write all nine derivatives because the coefficients of the remainder terms apply nine out of twelve derivatives. This is certainly not something

to be repeated, although it is helpful in general comprehension of floating-point arithmetic. After doing a fair bit of reading, I realized that the best method was truncating the series so that Java would reference the remainder coefficients for C_0, C_1, C_2, C_3, and C_4. In 1960, C. B. Haselgrove produced an article labeled "Tables of the Riemann zeta Function." These are listed inside Edward's *Riemann's Zeta Function* and are ideal for computation. Both Glen Pugh [32] and Ken Takusagawa [39] found suitable workarounds. Ken uses coefficients of Chebyshev polynomials instead of Taylor polynomials. Through the use of a Taylor series expansion, the zeroes generated by the the Riemann−Siegel formula are accurate to an error of about $1e-7$.

Another problem to think about is the creation of a root-finding algorithm. I originally developed a variation of Newton's method to find zeroes. I noticed that Netwon's method was slow and then found a faster method through the Apache Commons Math library. If you go to their web site,

`http://commons.apache.org/proper/commons-math/userguide/analysis.html`

scroll down to the bottom and read the last section which explains how the UnivariateFunctionDifferentiator interface works. Just in case you are curious, I was the person who originally attempted to calculate the zeroes of $Z(t)$ through the Newton-Raphson solver. The bracketing n^{th} order Brent method is the algorithm that is used inside RiemannSiegel.java. The algorithm itself is an improvement to Brent's method and is rather impressive. The root-finding algorithm was able to find the first 10143 zeroes in ten seconds.

My main motivation for investigating the Riemann−Siegel formula wasn't to find non-trivial zeroes for $\zeta(s)$. The exact values of these zeroes is already well known and has been published in detail by A. Odlyzko. My main interest with $Z(t)$ emerged from Lehmer's phenomenon. Lehmer's phenomenon can be thought of in many different ways, one interpretation is the size of $|Z(t)|$ associated with extremely close zeroes. This provides a direct connection to the distribution of zeroes inside the $Z(t)$ function. It may also be connected with the distribution of prime numbers. There seems to be a lot of unresolved mysteries with Lehmer's phenomenon.

Near the end of the project, I developed a way to find all consecutive roots in the specified range for Lehmer's phenomenon. I will note that I commented out this section of the code in RiemannSiegel.java. My original thoughts are different than what is written in the program. I originally thought about finding all the zeroes through calculating where the derivative is zero. If you are curious, you can read a conversation about some of these thoughts here .

• **AbelPlana.java** was built from the integration formula used to calculate $\zeta(s)$. I highly recommend not to calculate $\zeta(s)$ by the Abel−Plana formula. The Abel−Plana formula for $\zeta(s)$ converges slowly and requires decimal calculations of hundreds to thousands of decimal precision.

Since this is now stated clearly, I am going to explain everything which I did wrong. To start, the first two variations of the Abel−Plana formula are

$$\zeta(s) = \frac{2^{s-1}}{s-1} - 2^s \int_0^\infty \frac{\sin(s \arctan t)}{(1+t^2)^{\frac{s}{2}}(\mathrm{e}^{\pi\, t}+1)}\,\mathrm{d}t \qquad \forall\ s \in \mathbb{C} \setminus \{1\}$$

and

$$\zeta(s) = \frac{1}{2} + \frac{1}{s-1} + 2\int_0^\infty \frac{\sin(s \arctan x)}{(1+x^2)^{s/2}(e^{2\pi x}-1)}dx. \qquad \forall\ s \in \mathbb{C} \setminus \{1\}$$

Notice that we are evaluating $s = a + ib$ so the integration method must be able to handle complex arithmetic. Java doesn't have a complex variable library, so you are forced to turn to the Apache Commons Math library or Apfloat. I prefer to write my own programs, so I created a helper library that would handle the evaluation of complex functions inside Java. Writing the method to calculate BigDecimal.pow (BigDecimal BG) is much harder than it would initially seem. Turning back to the Abel−Plana formula, I will focus on the three major problems I encountered.

The first problem deals with the calculation of BigDecimals in Java. BigDecimals are immutable and will eat up memory. You can use Java's MathContext to increase decimal precision at the cost of speed. If you look at the numbers being calculated, a MathContext of 512 decimal precision is hardly satisfactory. The integration itself slows down as the value of s increases. It isn't practical to write a program this way, the evaluation of $\zeta(400 + 55i)$ will take hours before the desired level of convergence is reached.

The second problem is independent to the original implementation of the Abel−Plana formula for $\zeta(s)$. For $s = a + 0i$, the values of the two constants terms, $(2^{s-1}/(s-1)) - (2^s)$, slowly converge to 1.

a	C1	C2
10	5.689E1	1.024E3
20	2.759E4	1.048E6
30	1.851E7	1.073E9
40	1.409E10	1.099E12
60	9.770E15	1.152E18
100	6.402E27	1.267E30

At higher inputs of a, $\zeta(a + 0i)$ will decrease towards 1 and then further decrease as $a \to \infty$. As the value of a increases, the program recursively integrates a value near 1 and then checks to see if the difference between C1 and C2 is less than the desired error term, labeled EPSILON. Since the difference converges to 1, the integration will never reach the desired error defined by EPSILON.

The third problem is due to the oscillatory behavior of the integrand. The sine function causes the integrand to oscillate in both formulas for $\zeta(s)$. Because of this pattern, increasing the bounds of integration would increase the number of zeroes found. My program returned incorrect results when evaluating an interval larger than [0, 8]. I may be wrong, I would guess that any program which did numerical integration in a similar manner would encounter the same issue.

I was fortunate to receive help while writing AbelPlana.java. If you are curious, please review the discussion on Stack Overflow .

• **DirichletZeta.java** is the preferred method for numerical evaluation of the zeta function. The standard summation formula for $\zeta(s)$ is

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} \qquad \forall\ \Re(s) > 1$$

which could be used to find values for $\Re(s) > 1$. However, this would be extremely boring. The Dirichlet series formula for $\zeta(s)$ references

$$\zeta(s) = \frac{1}{s-1} \sum_{n=1}^{\infty} \left(\frac{n}{(n+1)^s} - \frac{n-s}{n^s} \right) \qquad \forall\ \Re(s) > 0$$

By any form of the functional equation, the program could recursively extend to negative values where $\Re(s) < 0$. I will note that the simple pole at $s = 1$ evaluates to infinity.

The output produced in Java is extremely accurate and can be computed at higher precision by increasing both the PRECISION and EPSILON constants in DirichletZeta.java. As an example, truncating a lot of terms for these two values produces

$$\zeta(23 + 453i) = 1.000000117620530737559373956908457 8... + 1.940465406934971750707906130476 26...10^{-8} * i$$

$$\zeta(97 - 33i) = 0.999999999999999999999999999995992 1... + -4.87491868428770955400197157448 02..10^{-30} * i$$

CAS such as MATLAB or Mathematica must use some variation of the summation formulas. Python has a nice alternative known as mpmath .

• **Complex.java** is a program that uses BigDecimal.java to calculate common complex functions such as atan(), mod(), log(), sqrt(), and sinh(). The most difficult complex function to write is BigDecimal.pow (BigDecimal BG). Other libraries in Java have various representations of complex arithmetic. Two most notable of these are Apfloat and the Apache Commons Math library . While the Apache Commons Math library does have a lot of good software, the use of double format for high precision algorithms is undesirable.

• **BernoulliZeta.java** was the last program I wrote in Java. My main motivation for writing this program was due to the calculation of Bernoulli numbers. There are several ways to calculate them in Java, I strongly urge the reader to try a different method. The recursion formula for $\zeta(2k)$ is listed as

$$\zeta(2k) = \frac{(-1)^{k+1} B_{2k} (2\pi)^{2k}}{2(2k)!} \qquad \forall\ k \in \mathbb{N}$$

Inside BernoulliZeta.java, BigInteger.pow(BigInteger BI) does significantly slow down as the value of 2k increases. I could have made an improvement in the design of the .pow() method I was using. The .pow() method in Java is definitely a source of major frustration.

Here are various web sites and libraries that are helpful in evaluating the zeta function. I find that reading extra information is always beneficial in the long run.

I The Riemann zeta research project

II The Home Page for Andrew Odlyzko

III The Home Page for Glen Pugh

IV Analysis of Riemann-Siegel done by Ken Takusagawa

V The Apache Commons Math library

VI Apfloat for Java

VII Various programs written by Jose Menes

VIII Various programs written by miraclefoxx

IX The Java Numerics website

X Everything about BigDecimalMath

XI Integral forms of the zeta function

XII Princeton Java website

XIII MrYouMath zeta function on Youtube

XIV The Riemann zeta function by T. Majlth

XV Riemann zeta in various languages

XVI Riemann zeta in four dimensions

XVII A comical approach to the Riemann hypothesis

XVIII The Mystery of the Land of Riemannia

XIX Information about the Riemann zeta function by Terence Tao

XX The Riemann Hypothesis FAQ

XXI Matt's Java Programs

4.4 Appendix D - CauchySchlomich.java

```
/*************************************************************************
**
**    Euler-Riemann Zeta Function
**
*************************************************************************
**    Matthew Kehoe
**    06/20/2015
**
**    This program computes the value for Zeta(s) using the standard form
**    of Zeta(s), the Riemann functional equation, and the Cauchy-Schlomilch
**    transformation. A recursive method named riemannFuncForm has been created
**    to handle computations of Zeta(s). Simpson's method is used to
**    approximate the definite integral calculated by the Cauchy-Schlomilch
**    transformation.
*************************************************************************/

import java.util.*;

public class CauchySchlomich {

    // Main method
    public static void main(String[] args) {
        ZetaMain();
    }

    /**
   * Calculates Zeta(s) for s in real.
    */
    public static void ZetaMain() {
        double s = 0;
        double start, stop, totalTime;
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter the value of s inside the Riemann Zeta " +
                "Function: ");
        try {
                s = scan.nextDouble();
        }
        catch (Exception e) {
           System.out.println("Please enter a valid number for s.");
        }
        start = System.currentTimeMillis();
        System.out.println("Value for the Zeta Function = "
                    + riemannFuncForm(s));
        stop = System.currentTimeMillis();
        totalTime = (double) (stop-start) / 1000.0;
        System.out.println("Total time taken is " + totalTime + " seconds.");

    }

    /**
   * The standard summation of an infinite series for the Zeta function.
         * <br> Stop the summation when the relative error is less than 2*10^-7
   * @param s - the value of s.
   * @return the value of Zeta(s) through an infinite series
         * approximation.
    */
    public static double standardZeta(double s) {
        int n = 1;
        double currentSum = 0;
        double relativeError = 1;
        double error = 2.0 * 10E-7;
        double remainder;

```

```
64          while (relativeError > error) {
65              currentSum = Math.pow(n, -s) + currentSum;
66              remainder = 1 / ((s-1)* Math.pow(n, (s-1)));
67              relativeError =  remainder / currentSum;
68              n++;
69          }
70          System.out.println("The number of terms summed was " + n + ".");
71          return currentSum;
72      }
73
74      /**
75      * Approximation of the Gamma function by the Lanczos Approximation.
76      * @param s - the value of s.
77      * @return the approximate value of Gamma(s).
78       */
79      public static double lancGamma(double s){
80                      double[] p = {0.99999999999980993, 676.5203681218851,
81                          -1259.1392167224028, 771.32342877765313,
82                          -176.61502916214059, 12.507343278686905,
83                          -0.13857109526572012, 9.9843695780195716e-6,
84                          1.5056327351493116e-7};
85
86                      int g = 7;
87
88                      // Implements Euler's Reflection Formula.
89                      if(s < 0.5) return Math.PI / (Math.sin(Math.PI * s)
90                              *lancGamma(1-s));
91
92                      s -= 1;
93                      double a = p[0];
94                      double t = s + g + 0.5;
95                      for(int i = 1; i < p.length; i++){
96                              a += p[i] / (s+i);
97                      }
98
99                      return Math.sqrt(2*Math.PI)*Math.pow(t, s+0.5)
100                             *Math.exp(-t)*a;
101             }
102
103     /**
104     * Approximation of the Gamma function by the Stirling Approximation.
105     * @param s - the value of s.
106     * @return the approximate value of Gamma(s).
107      */
108     public static double strlGamma(double s){
109     return Math.sqrt(2 * Math.PI/s) *
110                     Math.pow((s/Math.E), s);
111 }
112
113     /**
114     * Riemann's Functional Equation - Directly calculates the value of
115           Zeta(s).
116
117         1. The first if statement handles when s < 0 and s is a multiple
118            of 2k. These are trivial zeroes where Zeta(s) = 0.
119
120         2. The second if statement handles when s < -170. This
121            implementation of Zeta(s) does not handle values below Zeta(-170).
122            The specific cause is due to the double data type and the Stirling
123            Approximation for Gamma(s).
124
125         3. The third if statement handles when s < -141. The value of Zeta(s)
126            is calculated by recursion through the functional equation and
127            the Stirling Approximation for Gamma(s).
128
```

```
129         4. The fourth if statement handles when s <= -1. The value of Zeta(s)
130            is calculated by recursion through the functional equation and
131            the Lanczos Approximation for Gamma(s).
132
133         5. The fifth if statement handles the values of -1 < s < 0. Recursion
134            is used alongside an approximation through Simpson's method.
135
136         6. The sixth if statement handles the values of 0 <= s < 2. Simpson's
137            method is used to numerically compute an approximation of the
138            definite integral.
139
140         7. The last if statement directly calculates Zeta(s) through the
141            standard form where s > 1.
142
143    * @param s - the value of s.
144    * @return the value of Zeta(s)
145     */
146     public static double riemannFuncForm(double s) {
147         double term = Math.pow(2, s)*Math.pow(Math.PI, s-1)*
148             (Math.sin((Math.PI*s)/2));
149
150         if(s <= -2 && s % 2 == 0)
151             return 0;
152         else if(s < -170) {
153             System.out.println("This implementation of Zeta(s) does not " +
154                     "support values where s < -170.");
155             return term*standardZeta(1-s)*strlGamma(1-s);
156         }
157         else if(s <= -141)
158             return term*standardZeta(1-s)*strlGamma(1-s);
159         else if(s <= -1)
160             return term*standardZeta(1-s)*lancGamma(1-s);
161         else if (s > -1 && s < 0)
162             return term*getSimpsonSum(1-s)*lancGamma(1-s);
163         else if (s >= 0 && s < 2)
164             return getSimpsonSum(s);
165         else
166             return standardZeta(s);
167     }
168
169     /**
170    * Returns the function referenced inside the right hand side of the
171          * Cauchy-Schlomilch transformation for Zeta(s).
172    * @param s - the value of s.
173    * @return the value of the function inside the integrand.
174     */
175     public static double function(double x, double s) {
176         double sech = 1 / Math.cosh(x); // Hyperbolic cosecant
177         double squared = Math.pow(sech, 2);
178         return ((Math.pow(x, s)) * squared);
179     }
180
181     /**
182    * Simpson's rule for evaluating a definite integral.
183    * @param a - the start of the integral approximation.
184          * @param b - the end of the integral approximation.
185          * @param s - the value of s.
186          * @param n - initially the number of terms to sum, should be even.
187    * @return the integral approximation through Simpson's method.
188     */
189     public static double SimpsonsRule(double a, double b, double s, int n) {
190         double simpson, dx, x, sum4x, sum2x;
191
192         dx = (b-a) / n;
193         sum4x = 0.0;
```

```
194         sum2x = 0.0;
195
196         // 4/3 terms
197         for (int i = 1; i < n; i += 2) {
198             x = a + i * dx;
199             sum4x += function(x,s);
200         }
201
202         // 2/3 terms
203         for (int i = 2; i < n-1; i += 2) {
204             x = a + i * dx;
205             sum2x += function(x,s);
206         }
207
208         // Compute the integral approximation.
209         simpson = function(a,s) + function(a,b);
210         simpson = (dx / 3)*(simpson + 4 * sum4x + 2 * sum2x);
211         return simpson;
212     }
213
214     /**
215     * Handles the error inside Simpson's rule for for
216           * f(x) = t^s * sech(t)^2. The integration is done from 0 to 100.
217           * Stop Simspson's Method when the relative error is less than
218           * 2 * 10^-7.
219     * @param a - the start of the integral approximation.
220           * @param b - the end of the integral approximation.
221           * @param s - the value of s.
222           * @param n - initially the number of terms to sum, should be even.
223     * @return the integral approximation through Simpson's method.
224      */
225     public static double SimpsonError(double a, double b, double s, int n)
226     {
227         double futureVal;
228         double absError = 1.0;
229         double finalValueOfN;
230         double numberOfIterations = 0.0;
231         double currentVal = SimpsonsRule(a,b,s,n);
232
233         while (absError / currentVal > 0.000001) {
234             n = 2*n;
235             futureVal = SimpsonsRule(a,b,s,n);
236             absError = Math.abs(futureVal - currentVal) / 15;
237             currentVal = futureVal;
238         }
239
240         // Find the number of iterations. N starts at 8 and doubles
241         // every iteration.
242         finalValueOfN = n / 8;
243         while (finalValueOfN % 2 == 0) {
244             finalValueOfN = finalValueOfN / 2;
245             numberOfIterations++;
246         }
247         System.out.println("The number of iterations is "
248                 + numberOfIterations + ".");
249         return currentVal;
250     }
251
252     /**
253     * Returns an approximate sum of Zeta(s) through Simpson's rule.
254           * @param s - the value of s.
255     * @return the approximate sum by Simpson's rule.
256      */
257     public static double getSimpsonSum(double s) {
258         double constant = Math.pow(2, (2*s)-1) / (((Math.pow(2, s)) -2)*
```

```
                (lancGamma(1+s)));
        double seriesApprox = Math.pow(1/Math.cosh(1.0), 2) + 2/3 + 100/189;
        System.out.println("Did Simpson's Method.");
        if (s > 0 && s < 1)
            return seriesApprox*SimpsonError(0, 100, s, 8);
        else
            return constant*SimpsonError(0, 100, s, 8);
    }
}
```

4.5 Appendix E - RiemannSiegel.java

```
/*************************************************************************
**
**    Riemann-Siegel Formula for roots of Zeta(s) on critical line.
**
*************************************************************************
**    Matthew Kehoe
**    07/31/2015
**
**    This program finds the roots of Zeta(s) using the well known Riemann-
**    Siegel formula. The Riemann Siegel theta function is approximated
**    using Stirling's approximation. It also uses an interpolation method to
**    locate zeroes. The coefficients for R(t) are handled by the Taylor
**    Series approximation originally listed by Haselgrove in 1960. It is
**    necessary to use these coefficients in order to increase computational
**    speed. Lehmer's Phenomenon can be explored through the commented code
**    inside the findRoots() method.
*************************************************************************/

import java.util.LinkedHashSet;
//These two imports are from the Apache Commons Math library
import org.apache.commons.math3.analysis.UnivariateFunction;
import org.apache.commons.math3.analysis.solvers.BracketingNthOrderBrentSolver;
import java.util.*;

public class RiemannSiegel{
    public static void main(String[] args){
        SiegelMain();
    }

    // Main method
    public static void SiegelMain() {
        System.out.println("Zeroes inside the critical line for " +
                "Zeta(1/2 + it). The t values are referenced below.");
        System.out.println();
        findRoots();
    }

    /**
   * The sign of a calculated double value.
   * @param x - the double value.
   * @return the sign in -1,  1, or 0 format.
    */
    private static int sign(double x) {
  if (x < 0.0)
            return -1;
        else if (x > 0.0)
            return 1;
        else
            return 0;
    }

    /**
   * Finds the roots of a specified function through the
```

```
321         * BracketingNthOrderBrentSolver from the Apache Commons Math library.
322         * See http://commons.apache.org/proper/commons-math/apidocs/org/
323         * apache/commons/math3/analysis/solvers/BracketingNthOrderBrentSolver
324         * .html
325   * The zeroes inside the interval of 0 < t < 10000 are printed from
326         * a LinkedHashSet.
327    */
328    public static void findRoots() {
329  BracketingNthOrderBrentSolver f = new
330           BracketingNthOrderBrentSolver(1e-10, 10);
331        UnivariateFunction func = (double x) -> RiemennZ(x, 4);
332        LinkedHashSet<Double> set = new LinkedHashSet<>();
333        double i = 1.0;
334        while (i < 10000) {
335            i+= 0.1;
336            if(sign(func.value(i)) != sign(func.value(i+0.1))) {
337            set.add(f.solve(1000, func, i, i+0.1));
338            }
339        }
340        set.stream().filter((s) -> (s > 0)).forEach((s) -> {
341            System.out.println(s);
342        });
343
344        // Uncomment the code below to investigate Lehmer's Phenomenon
345        // EPSILON can be made smaller to see zeroes which are closer together
346        /*
347        double EPSILON = .05;
348        Iterator<Double> itr = set.iterator();
349        Double prevVal = null;
350        //current zero
351        int j = 0;
352        while(itr.hasNext()) {
353            j++;
354            Double currentVal = itr.next();
355            if (prevVal != null) {
356                if(currentVal - prevVal < EPSILON ) {
357                    // To be copied into a file or pasted into Excel
358                    // Column Headers - See Appendix J
359                    // Root 1, Root 2, Value of Root 1, Value of Root 2,
360                    // Distance between Roots
361                    System.out.print((j-1) + ", " + j + ", ");
362                    System.out.print(prevVal + ", " + currentVal + ", ");
363                    System.out.print(currentVal - prevVal);
364                    System.out.println();
365                }
366            }
367        prevVal = currentVal;
368        }
369        */
370    }
371
372    /**
373   * Riemann-Siegel theta function using the approximation by the
374         * Stirling series.
375   * @param t - the value of t inside the Z(t) function.
376   * @return Stirling's approximation for theta(t).
377    */
378    public static double theta (double t) {
379        return (t/2.0 * Math.log(t/(2.0*Math.PI)) - t/2.0 - Math.PI/8.0
380                + 1.0/(48.0*Math.pow(t, 1)) + 7.0/(5760*Math.pow(t, 3)));
381    }
382
383    /**
384   * Computes Math.Floor of the absolute value term passed in as t.
385   * @param t - the value of t inside the Z(t) function.
```

```
386     * @return Math.floor of the absolute value of t.
387      */
388      public static double fAbs(double t) {
389          return Math.floor(Math.abs(t));
390
391      }
392
393      /**
394     * Riemann-Siegel Z(t) function implemented per the Riemenn Siegel
395           * formula. See http://mathworld.wolfram.com/Riemann-SiegelFormula.html
396           * for details
397     * @param t - the value of t inside the Z(t) function.
398           * @param r - referenced for calculating the remainder terms by the
399           * Taylor series approximations.
400     * @return the approximate value of Z(t) through the Riemann-Siegel
401           * formula
402      */
403      public static double RiemennZ(double t, int r) {
404
405          double twopi = Math.PI * 2.0;
406          double val = Math.sqrt(t/twopi);
407          double n = fAbs(val);
408          double sum = 0.0;
409
410          for (int i = 1; i <= n; i++) {
411            sum += (Math.cos(theta(t) - t * Math.log(i))) / Math.sqrt(i);
412          }
413          sum = 2.0 * sum;
414
415          double remainder;
416          double frac = val - n;
417          int k = 0;
418          double R = 0.0;
419
420          // Necessary to individually calculate each remainder term by using
421          // Taylor Series co-efficients. These coefficients are defined below.
422          while (k <= r) {
423              R = R + C(k, 2.0*frac-1.0) * Math.pow(t / twopi,
424                      ((double) k) * -0.5);
425              k++;
426          }
427
428          remainder = Math.pow(-1, (int)n-1) * Math.pow(t / twopi, -0.25) * R;
429          return sum + remainder;
430      }
431
432      /**
433     * C terms for the Riemann-Siegel formula. See
434           * https://web.viu.ca/pughg/thesis.d/masters.thesis.pdf for details.
435           * Calculates the Taylor Series coefficients for C0, C1, C2, C3,
436           * and C4.
437     * @param n - the number of coefficient terms to use.
438           * @param z - referenced per the Taylor series calculations.
439     * @return the Taylor series approximation of the remainder terms.
440      */
441      public static double C (int n, double z) {
442          if (n==0)
443              return(.38268343236508977173 * Math.pow(z, 0.0)
444              +.43720468077520449360 * Math.pow(z, 2.0)
445              +.13237657548034352332 * Math.pow(z, 4.0)
446              -.01360502604767418865 * Math.pow(z, 6.0)
447              -.01356762197010358089 * Math.pow(z, 8.0)
448              -.00162372532314446528 * Math.pow(z,10.0)
449              +.00029705353733379691 * Math.pow(z,12.0)
450              +.00007943300879521470 * Math.pow(z,14.0)
```

```
451             +.00000046556124614505 * Math.pow(z,16.0)
452             -.00000143272516309551 * Math.pow(z,18.0)
453             -.00000010354847112313 * Math.pow(z,20.0)
454             +.00000012357927083 86 * Math.pow(z,22.0)
455             +.00000001788108385 80 * Math.pow(z,24.0)
456             -.00000000033914143 90 * Math.pow(z,26.0)
457             -.00000000016326633 90 * Math.pow(z,28.0)
458             -.00000000000378510 93 * Math.pow(z,30.0)
459             +.00000000000093274 23 * Math.pow(z,32.0)
460             +.00000000000005221 84 * Math.pow(z,34.0)
461             -.00000000000000335 07 * Math.pow(z,36.0)
462             -.00000000000000034 12 * Math.pow(z,38.0)
463             +.00000000000000000 58 * Math.pow(z,40.0)
464             +.00000000000000000 15 * Math.pow(z,42.0));
465       else if (n==1)
466             return(-.02682510262837534703 * Math.pow(z, 1.0)
467             +.01378477342635185305 * Math.pow(z, 3.0)
468             +.03849125048223508223 * Math.pow(z, 5.0)
469             +.00987106629906207647 * Math.pow(z, 7.0)
470             -.00331075976085840433 * Math.pow(z, 9.0)
471             -.00146478085779541508 * Math.pow(z,11.0)
472             -.00001320794062487696 * Math.pow(z,13.0)
473             +.00005922748701847141 * Math.pow(z,15.0)
474             +.00000598024258537345 * Math.pow(z,17.0)
475             -.00000096413224561698 * Math.pow(z,19.0)
476             -.00000018334733722714 * Math.pow(z,21.0)
477             +.00000004446708756272 * Math.pow(z,23.0)
478             +.00000000270963508218 * Math.pow(z,25.0)
479             +.00000000007785288654 * Math.pow(z,27.0)
480             -.00000000002343762601 * Math.pow(z,29.0)
481             -.00000000000158301728 * Math.pow(z,31.0)
482             +.00000000000012119942 * Math.pow(z,33.0)
483             +.00000000000001458378 * Math.pow(z,35.0)
484             -.00000000000000028786 * Math.pow(z,37.0)
485             -.00000000000000008663 * Math.pow(z,39.0)
486             -.00000000000000000084 * Math.pow(z,41.0)
487             +.00000000000000000036 * Math.pow(z,43.0)
488             +.00000000000000000001 * Math.pow(z,45.0));
489       else if (n==2)
490             return(+.00518854283029316849 * Math.pow(z, 0.0)
491             +.00309465838808347 46 * Math.pow(z, 2.0)
492             -.01133594107822937338 * Math.pow(z, 4.0)
493             +.00223304574195814477 * Math.pow(z, 6.0)
494             +.00519663740886233021 * Math.pow(z, 8.0)
495             +.00034399144076208337 * Math.pow(z,10.0)
496             -.00059106484274705828 * Math.pow(z,12.0)
497             -.00010229972547935857 * Math.pow(z,14.0)
498             +.00002088839221699276 * Math.pow(z,16.0)
499             +.00000592766549309654 * Math.pow(z,18.0)
500             -.00000016423838362436 * Math.pow(z,20.0)
501             -.00000015161199700941 * Math.pow(z,22.0)
502             -.00000000590780369821 * Math.pow(z,24.0)
503             +.00000000209115148595 * Math.pow(z,26.0)
504             +.00000000017815649583 * Math.pow(z,28.0)
505             -.00000000001616407246 * Math.pow(z,30.0)
506             -.00000000000238069625 * Math.pow(z,32.0)
507             +.00000000000005398265 * Math.pow(z,34.0)
508             +.00000000000001975014 * Math.pow(z,36.0)
509             +.00000000000000023333 * Math.pow(z,38.0)
510             -.00000000000000011188 * Math.pow(z,40.0)
511             -.00000000000000000416 * Math.pow(z,42.0)
512             +.00000000000000000044 * Math.pow(z,44.0)
513             +.00000000000000000003 * Math.pow(z,46.0));
514       else if (n==3)
515             return(-.00133971609071945690 * Math.pow(z, 1.0)
```

```
            +.00374421513637939370 * Math.pow(z, 3.0)
            -.00133031789193214681 * Math.pow(z, 5.0)
            -.00226546607654717871 * Math.pow(z, 7.0)
            +.00095484999985067304 * Math.pow(z, 9.0)
            +.00060100384589636039 * Math.pow(z,11.0)
            -.00010128858286776622 * Math.pow(z,13.0)
            -.00006865733449299826 * Math.pow(z,15.0)
            +.00000059853667915386 * Math.pow(z,17.0)
            +.00000333165985123995 * Math.pow(z,19.0)
            +.00000021919289102435 * Math.pow(z,21.0)
            -.00000007890884245681 * Math.pow(z,23.0)
            -.00000009941468508130 * Math.pow(z,25.0)
            +.00000000095701162109 * Math.pow(z,27.0)
            +.00000000018763137453 * Math.pow(z,29.0)
            -.00000000000443783768 * Math.pow(z,31.0)
            -.00000000000224267385 * Math.pow(z,33.0)
            -.00000000000003627687 * Math.pow(z,35.0)
            +.00000000000001763981 * Math.pow(z,37.0)
            +.00000000000000079608 * Math.pow(z,39.0)
            -.00000000000000009420 * Math.pow(z,41.0)
            -.00000000000000000713 * Math.pow(z,43.0)
            +.00000000000000000033 * Math.pow(z,45.0)
            +.00000000000000000004 * Math.pow(z,47.0));
      else
            return(+.00046483389361763382 * Math.pow(z, 0.0)
            -.00100566073653404708 * Math.pow(z, 2.0)
            +.00024044856573725793 * Math.pow(z, 4.0)
            +.00102830861497023219 * Math.pow(z, 6.0)
            -.00076578610717556442 * Math.pow(z, 8.0)
            -.00020365286803084818 * Math.pow(z,10.0)
            +.00023212290491068728 * Math.pow(z,12.0)
            +.00003260214424386520 * Math.pow(z,14.0)
            -.00002557906251794953 * Math.pow(z,16.0)
            -.00000410746443891574 * Math.pow(z,18.0)
            +.00000117811136403713 * Math.pow(z,20.0)
            +.00000024456561422485 * Math.pow(z,22.0)
            -.00000002391582476734 * Math.pow(z,24.0)
            -.00000000750521420704 * Math.pow(z,26.0)
            +.00000000013312279416 * Math.pow(z,28.0)
            +.00000000013440626754 * Math.pow(z,30.0)
            +.00000000000351377004 * Math.pow(z,32.0)
            -.00000000000151915445 * Math.pow(z,34.0)
            -.00000000000008915418 * Math.pow(z,36.0)
            +.00000000000001119589 * Math.pow(z,38.0)
            +.00000000000000105160 * Math.pow(z,40.0)
            -.00000000000000005179 * Math.pow(z,42.0)
            -.00000000000000000807 * Math.pow(z,44.0)
            +.00000000000000000011 * Math.pow(z,46.0)
            +.00000000000000000004 * Math.pow(z,48.0));
    }
}
```

4.6 Appendix F - AbelPlana.java

```
/**************************************************************************
**
**    Abel-Plana Formula for the Zeta Function
**
**************************************************************************
**    Matthew Kehoe
**    08/16/2015
**
**    This program computes the value for Zeta(z) using a definite integral
**    approximation through the Abel-Plana formula. The Abel-Plana formula
**    can be shown to approximate the value for Zeta(s) through a definite
```

```
**    integral. The integral approximation is handled through the Composite
**    Simpson's Rule known as Adaptive Quadrature.
**********************************************************************/

/* The Abel-Plana formula will calculate small values of the zeta function.
   If s = a + i*b, the program is designed to handle -30 < a < 30 and
   -30 < b < 30. The program will work for s = 100 + 0*i but will converge
   very slowly as either a or b increases. */

import java.util.*;
import java.math.*;


public class AbelPlana extends Complex {
    private static MathContext MC = new MathContext(512,
            RoundingMode.HALF_EVEN);
    public static void main(String[] args) {
        AbelMain();
    }

    // Main method
    public static void AbelMain() {
        double re = 0, im = 0;
        double start, stop, totalTime;
        Scanner scan = new Scanner(System.in);
        System.out.println("Calculation of the Riemann Zeta " +
                "Function in the form Zeta(s) = a + ib.");
        System.out.println();
        System.out.print("Enter the value of [a] inside the Riemann Zeta " +
                "Function: ");
        try {
                re = scan.nextDouble();
        }
        catch (Exception e) {
           System.out.println("Please enter a valid number for a.");
        }
        System.out.print("Enter the value of [b] inside the Riemann Zeta " +
                "Function: ");
        try {
                im = scan.nextDouble();
        }
        catch (Exception e) {
           System.out.println("Please enter a valid number for b.");
        }
        start = System.currentTimeMillis();
        Complex z = new Complex(new BigDecimal(re), new BigDecimal(im));
        System.out.println("The value for Zeta(s) is " + AbelPlana(z));
        stop = System.currentTimeMillis();
        totalTime = (double) (stop-start) / 1000.0;
        System.out.println("Total time taken is " + totalTime + " seconds.");
    }

    /**
     * The definite integral for Zeta(z) in the Abel-Plana formula.
         * <br> Numerator = Sin(z * arctan(t))
         * <br> Denominator = (1 + t^2)^(z/2) * (e^(2*pi*t) - 1)
     * @param t - the value of t passed into the integrand.
         * @param z - The complex value of z = a + i*b
     * @return the value of the complex function.
    */
    public static Complex f(double t, Complex z) {
        Complex num = (z.multiply(Math.atan(t))).sin();
        Complex D1 = new Complex(1 + t*t).pow(z.divide(TWO));
        Complex D2 = new Complex(Math.pow(Math.E, 2.0*Math.PI*t) - 1.0);
        Complex den = D1.multiply(D2);
```

```
643         return num.divide(den, MC);
644     }
645
646     /**
647      * Adaptive quadrature - See http://www.mathworks.com/moler/quad.pdf
648      * @param a - the lower bound of integration.
649          * @param b - the upper bound of integration.
650          * @param z - The complex value of z = a + i*b
651      * @return the approximate numerical value of the integral.
652     */
653     public static Complex adaptiveQuad(double a, double b, Complex z) {
654         double EPSILON = 1E-13;
655         double step = b - a;
656         double c = (a + b) / 2.0;
657         double d = (a + c) / 2.0;
658         double e = (b + c) / 2.0;
659
660         Complex S1 = (f(a, z).add(f(c, z).multiply(FOUR)).add(f(b, z))).
661                 multiply(step / 6.0);
662         Complex S2 = (f(a, z).add(f(d, z).multiply(FOUR)).add(f(c, z).multiply
663                 (TWO)).add(f(e, z).multiply(FOUR)).add(f(b, z))).multiply
664                 (step / 12.0);
665         Complex result = (S2.subtract(S1)).divide(FIFTEEN, MC);
666
667         if(S2.subtract(S1).mod() <= EPSILON)
668             return S2.add(result);
669         else
670             return adaptiveQuad(a, c, z).add(adaptiveQuad(c, b, z));
671     }
672
673     /**
674      * The definite integral for Zeta(z) in the Abel-Plana formula.
675          * <br> value =  1/2 + 1/(z-1) + 2 * Integral
676          * @param z - The complex value of z = a + i*b
677      * @return the value of Zeta(z) through value and the
678          * quadrature approximation.
679     */
680     public static Complex AbelPlana(Complex z) {
681         Complex C1 = ONEHALF.add(ONE.divide(z.subtract(ONE), MC));
682         Complex C2  =  TWO.multiply(adaptiveQuad(1E-16, 100.0, z));
683         if ( z.real().doubleValue() == 0 && z.imag().doubleValue() == 0)
684             return new Complex(0.0, 0.0);
685         else if ( z.real().doubleValue() == 1 && z.imag().doubleValue() == 0) {
686             System.out.println("The zeta function is undefined.");
687             return new Complex(Double.POSITIVE_INFINITY, 0.0);
688         }
689         else
690             return C1.add(C2);
691     }
692 }
```

4.7 Appendix G - DirichletZeta.java

```
693 /*************************************************************************
694 **
695 **    Dirichlet series for the Zeta function
696 **
697 *************************************************************************
698 **    Matthew Kehoe
699 **    09/20/2015
700 **
701 **    This program computes the value for Zeta(z) using the Dirichlet series
702 **    summation. The Dirichlet series summation is valid for Re(s) > 0 and
703 **    quickly approaches convergence. The BigDecimal class and methods are
704 **    referenced through the Apfloat library for Java.
```

```
705 ************************************************************************/
706
707 import org.apfloat.ApcomplexMath;
708 import org.apfloat.Apcomplex;
709 import org.apfloat.Apfloat;
710 import java.util.*;
711
712 public class DirichletZeta {
713     final private static long PRECISION = 500;
714     final private static Apfloat ONE = new Apfloat("1", PRECISION);
715
716     public static void main(String[] args) {
717         DirichletMain();
718     }
719
720     // Main method
721     public static void DirichletMain() {
722         float re = 0, im = 0;
723         double start, stop, totalTime;
724         Scanner scan = new Scanner(System.in);
725         System.out.println("Calculation of the Riemann Zeta "
726                 + "Function in the form Zeta(s) = a + ib.");
727         System.out.println();
728         System.out.print("Enter the value of [a] inside the Riemann Zeta "
729                 + "Function: ");
730         try {
731             re = scan.nextFloat();
732         } catch (Exception e) {
733             System.out.println("Please enter a valid number for a.");
734         }
735         System.out.print("Enter the value of [b] inside the Riemann Zeta "
736                 + "Function: ");
737         try {
738             im = scan.nextFloat();
739         } catch (Exception e) {
740             System.out.println("Please enter a valid number for b.");
741         }
742         start = System.currentTimeMillis();
743         Apfloat real = new Apfloat(re, PRECISION);
744         Apfloat ima = new Apfloat(im, PRECISION);
745         Apcomplex s = new Apcomplex(real,ima);
746         System.out.println("The value for Zeta(s) is " +
747                 String.format("%.100s", calculateDirichletZeta(s)));
748         stop = System.currentTimeMillis();
749         totalTime = (double) (stop - start) / 1000.0;
750         System.out.println("Total time taken is " + totalTime + " seconds.");
751     }
752
753     /**
754      * The calculation of the Dirichlet series summation for Zeta(s)
755      * Calculation = 1/(s-1) * series summation.
756      * Stop the summation when the convergence is reached through the
757      * desired Apfloat EPSILON.
758      * @param s the user defined value for re(s) and ima(s)
759      * @return the calculated value of Zeta(s)
760     */
761     private static Apcomplex calculateDirichletZeta(Apcomplex s) {
762         Apcomplex sum = Apcomplex.ZERO;
763         Apcomplex cons = (ONE.divide(s.subtract(ONE)));
764         Apfloat EPSILON = new Apfloat("1e-6", PRECISION);
765         int stop_condition = 1;
766         int n = 1;
767         while (stop_condition > 0)
768         {
769             Apcomplex nextIter = zetaSum(n, s);
```

```
            stop_condition = ApcomplexMath.abs(nextIter).compareTo(EPSILON);
            sum = sum.add(nextIter);
            {
                System.out.println("Value at interation " + n + " : "
                        + String.format("%.100s", cons.multiply(sum)));
            }
            n++;
        }
        return cons.multiply(sum);
    }

    /**
     * The Dirichlet series summation formula for Zeta(s)
     * Summation = n/(n+1)^s - (n-s)/n^s
     * @param this_n the current index for the summation
     * @param s the user defined value for re(s) and ima(s)
     * @return the series summation
    */
    public static Apcomplex zetaSum(long this_n, Apcomplex s) {
         Apfloat n = new Apfloat(this_n, PRECISION);
         Apfloat nPlusOne = n.add(ONE);

         Apcomplex firstTerm = n.divide(ApcomplexMath.pow(nPlusOne, s));
         Apcomplex secondTerm = n.subtract(s).divide(ApcomplexMath.pow
                                (n, s));
         return firstTerm.subtract(secondTerm);
    }
}
```

4.8 Appendix H - Complex.java

```
/*************************************************************************
**
**    Complex Numbers
**
*************************************************************************
**    Matthew Kehoe
**    08/20/2015
**
**    This class is necessary as a helper class for the calculation of
**    imaginary numbers. The calculation of Zeta(z) inside AbelMain is in
**    the form of z = a + i*b.
*************************************************************************/

import java.math.BigDecimal;
import java.math.MathContext;

public class Complex extends Object{
    private BigDecimal re;
    private BigDecimal im;

    /**
        BigDecimal constant for zero
    */
    final static Complex ZERO = new Complex(BigDecimal.ZERO) ;

    /**
        BigDecimal constant for one half
    */
    final static Complex ONEHALF = new Complex(new BigDecimal(0.5));

    /**
        BigDecimal constant for one
    */
    final static Complex ONE = new Complex(BigDecimal.ONE);
```

```
833
834     /**
835         BigDecimal constant for two
836     */
837     final static Complex TWO = new Complex(new BigDecimal(2.0));
838
839     /**
840         BigDecimal constant for four
841     */
842     final static Complex FOUR = new Complex(new BigDecimal(4.0)) ;
843
844     /**
845         BigDecimal constant for fifteen
846     */
847     final static Complex FIFTEEN = new Complex(new BigDecimal(15.0)) ;
848
849     /**
850         BigDecimal constant for PI
851         Accurate to 1000 decimal places
852     */
853     final static BigDecimal PI = new BigDecimal(
854         "3.14159265358979323846264338327950288419716939937510582097494459230781"+
855         "64062862089986280348253421170679821480865132823066470938446095505822"+
856         "23172535940812848111745028410270193852110555964462294895493038196442"+
857         "88109756659334461284756482337867831652712019091456485669234603486104"+
858         "45432664821339360726024914127372458700660631558817488152092096282925"+
859         "40917153643678925903600113305305488204665213841469519415116094330572"+
860         "70365759591953092186117381932611793105118548074462379962749567351885"+
861         "75272489122793818301194912983367336244065664308602139494639522473719"+
862         "07021798609437027705392171762931767523846748184676694051320005681271"+
863         "45263560827785771342757789609173637178721468440901224953430146549585"+
864         "37105079227968925892354201995611212902196086403441815981362977477130"+
865         "99605187072113499999983729780499510597317328160963185950244594553469"+
866         "08302642522308253344685035261931188171010003137838752886587533208381"+
867         "42061717766914730359825349042875546873115956286388235378759375195778"+
868         "18577805321712268066130019278766111959092164201989");
869
870     /**
871         BigDecimal constant for E
872         Accurate to 1000 decimal places
873     */
874     final static BigDecimal E = new BigDecimal(
875         "2.71828182845904523536028747135266249775724709369995957496696762772407"+
876         "66303535475945713821785251664274274663919320030599218174135966290435"+
877         "72900334295260595630738132328627943490763233829880753195251019011573"+
878         "83418793070215408914993488416750924476146066808226480016847741185374"+
879         "23454424371075390777449920695517027618386062613313845830007520449338"+
880         "26560297606737113200709328709127443747047230696977209310141692836819"+
881         "02555151086574637721112523897844250569536967707854499699679468644549"+
882         "05987931636889230098793127736178215424999229576351482208269895193662"+
883         "33182528869398496465105820939239829488793320362509443117301238197068"+
884         "41614039701983767932068328237646480429531180232878250981945581530175"+
885         "67173613320698112509961818815930416903515988885193458072738667385894"+
886         "22879228499892086805825749279610484198444363463244968487560233624827"+
887         "04197862320900216099023530436994184914631409343173814364054625315209"+
888         "61836908887070167683964243781405927145635490613031072085103837505118"+
889         "01157477041718986106873969655212671546889570350354");
890
891     /**
892         Default constructor equivalent to zero
893     */
894     public Complex() {
895         re = BigDecimal.ZERO;
896         im = BigDecimal.ZERO;
897     }
```

```
898
899     /**
900         Constructor with real part only
901         @param x Real part, BigDecimal
902     */
903     public Complex(BigDecimal x) {
904         re = x;
905         im = BigDecimal.ZERO;
906     }
907
908     /**
909         Constructor with real part only
910         @param x Real part, double
911     */
912     public Complex(double x) {
913         re = new BigDecimal(x);
914         im = BigDecimal.ZERO;
915     }
916
917     /**
918         Constructor with real and imaginary parts in double format.
919         @param x Real part
920         @param y Imaginary part
921     */
922     public Complex(double x, double y) {
923         re= new BigDecimal(x);
924         im= new BigDecimal(y);
925     }
926
927     /**
928         Constructor for the complex number z = a + i*b
929         @param re Real part
930         @param im Imaginary part
931     */
932     public Complex (BigDecimal re, BigDecimal im) {
933         this.re = re;
934         this.im = im;
935     }
936
937     /**
938         Real part of the Complex number
939         @return Re[z] where z = a + i*b.
940     */
941     public BigDecimal real() {
942         return re;
943     }
944
945     /**
946         Imaginary part of the Complex number
947         @return Im[z] where z = a + i*b.
948     */
949     public BigDecimal imag() {
950         return im;
951     }
952
953     /**
954         Complex conjugate of the Complex number
955         in which the conjugate of z is z-bar.
956         @return z-bar where z = a + i*b and z-bar = a - i*b
957     */
958     public Complex conjugate() {
959         return new Complex(re, im.negate());
960     }
961
962     /**
```

```
963      * Returns the sum of this and the parameter.
964
965        @param augend the number to add
966        @param mc the context to use
967        @return this + augend
968      */
969     public Complex add(Complex augend,MathContext mc)
970     {
971         //(a+bi)+(c+di) = (a + c) + (b + d)i
972         return new Complex(
973             re.add(augend.re,mc),
974             im.add(augend.im,mc));
975     }
976
977     /**
978        Equivalent to add(augend, MathContext.UNLIMITED)
979        @param augend the number to add
980        @return this + augend
981      */
982     public Complex add(Complex augend)
983     {
984         return add(augend, MathContext.UNLIMITED);
985     }
986
987     /**
988         Addition of Complex number and a double.
989         @param d is the number to add.
990         @return z+d where z = a+i*b and d = double
991     */
992     public Complex add(double d){
993         BigDecimal augend = new BigDecimal(d);
994         return new Complex(this.re.add(augend, MathContext.UNLIMITED),
995                 this.im);
996     }
997
998     /**
999      * Returns the difference of this and the parameter.
1000       @param subtrahend the number to subtract
1001       @param mc the context to use
1002       @return this - subtrahend
1003    */
1004    public Complex subtract(Complex subtrahend, MathContext mc)
1005    {
1006        //(a+bi)-(c+di) = (a - c) + (b - d)i
1007        return new Complex(
1008            re.subtract(subtrahend.re,mc),
1009            im.subtract(subtrahend.im,mc));
1010    }
1011
1012    /**
1013     * Equivalent to subtract(subtrahend, MathContext.UNLIMITED)
1014       @param subtrahend the number to subtract
1015       @return this - subtrahend
1016    */
1017    public Complex subtract(Complex subtrahend)
1018    {
1019        return subtract(subtrahend,MathContext.UNLIMITED);
1020    }
1021
1022    /**
1023        Subtraction of Complex number and a double.
1024        @param d is the number to subtract.
1025        @return z-d where z = a+i*b and d = double
1026    */
1027    public Complex subtract(double d){
```

```
        BigDecimal subtrahend = new BigDecimal(d);
        return new Complex(this.re.subtract(subtrahend, MathContext.UNLIMITED),
                this.im);
    }

    /**
     * Returns the product of this and the parameter.
       @param multiplicand the number to multiply by
       @param mc the context to use
       @return this * multiplicand
     */
    public Complex multiply(Complex multiplicand, MathContext mc)
    {
        //(a+bi)(c+di) = (ac - bd) + (ad + bc)i
        return new Complex(
            re.multiply(multiplicand.re,mc).subtract(im.multiply
                (multiplicand.im,mc),mc),
            re.multiply(multiplicand.im,mc).add(im.multiply
                (multiplicand.re,mc),mc));
    }

    /**
       Equivalent to multiply(multiplicand, MathContext.UNLIMITED)
       @param multiplicand the number to multiply by
       @return this * multiplicand
     */
    public Complex multiply(Complex multiplicand)
    {
        return multiply(multiplicand,MathContext.UNLIMITED);
    }

    /**
        Complex multiplication by a double.
        @param d is the double to multiply by.
        @return z*d where z = a+i*b and d = double
    */
    public Complex multiply(double d){
        BigDecimal multiplicand = new BigDecimal(d);
        return new Complex(this.re.multiply(multiplicand, MathContext.UNLIMITED)
                ,this.im.multiply(multiplicand, MathContext.UNLIMITED));
    }

    /**
        Modulus of a Complex number or the distance from the origin in
        * the polar coordinate plane.
        @return |z| where z = a + i*b.
    */
    public double mod() {
        if ( re.doubleValue() != 0.0 || im.doubleValue() != 0.0)
            return Math.sqrt(re.multiply(re).add(im.multiply(im))
                    .doubleValue());
        else
            return 0.0;
    }

    /**
   * Modulus of a Complex number squared
   * @param z = a + i*b
   * @return |z|^2 where z = a + i*b
    */
    public double abs(Complex z) {
        double doubleRe = re.doubleValue();
        double doubleIm = im.doubleValue();
        return doubleRe * doubleRe + doubleIm * doubleIm;
  }
```

```
1093
1094    public Complex divide(Complex divisor)
1095    {
1096        return divide(divisor,MathContext.UNLIMITED);
1097    }
1098
1099    /**
1100        * The absolute value squared.
1101        * @return The sum of the squares of real and imaginary parts.
1102        * This is the square of Complex.abs() .
1103        */
1104    public BigDecimal norm()
1105    {
1106                return re.multiply(re).add(im.multiply(im)) ;
1107    }
1108
1109    /** The inverse of the the Complex number.
1110          @param mc amount of precision
1111          @return 1/this
1112        */
1113    public Complex inverse(MathContext mc)
1114    {
1115                final BigDecimal hyp = norm() ;
1116                /* 1/(x+iy)= (x-iy)/(x^2+y^2 */
1117                return new Complex( re.divide(hyp,mc), im.divide(hyp,mc)
1118                        .negate() ) ;
1119    }
1120
1121    /** Divide through another BigComplex number.
1122        @param oth the other complex number
1123        @param mc amount of precision
1124        @return this/other
1125        */
1126    public Complex divide(Complex oth, MathContext mc)
1127    {
1128                /* implementation: (x+iy)/(a+ib)= (x+iy)* 1/(a+ib) */
1129                return multiply(oth.inverse(mc),mc) ;
1130    }
1131
1132   /**
1133        Division of Complex number by a double.
1134        @param d is the double to divide
1135        @return new Complex number z/d where z = a+i*b
1136    */
1137    public Complex divide(double d){
1138        BigDecimal divisor = new BigDecimal(d);
1139        return new Complex(this.re.divide(divisor, MathContext.UNLIMITED),
1140                this.im.divide(divisor, MathContext.UNLIMITED));
1141    }
1142
1143    /**
1144        Exponential of a complex number (z is unchanged).
1145        <br> e^(a+i*b) = e^a * e^(i*b) = e^a * (cos(b) + i*sin(b))
1146        @return exp(z) where z = a+i*b
1147    */
1148    public Complex exp () {
1149        return new Complex(Math.exp(re.doubleValue()) * Math.cos(im.
1150                doubleValue()), Math.exp(re.doubleValue()) *
1151                Math.sin(im.doubleValue()));
1152    }
1153
1154     /**
1155        The Argument of a Complex number or the angle in radians
1156        with respect to polar coordinates.
1157        <br> Tan(theta) = b / a, theta = Arctan(b / a)
```

```
        <br> a is the real part on the horizontal axis
        <br> b is the imaginary part of the vertical axis
        @return arg(z) where z = a+i*b.
    */
    public double arg() {
        return Math.atan2(im.doubleValue(), re.doubleValue());
    }

    /**
        The log or principal branch of a Complex number (z is unchanged).
        <br> Log(a+i*b) = ln|a+i*b| + i*Arg(z) = ln(sqrt(a^2+b^2))
        * + i*Arg(z) = ln (mod(z)) + i*Arctan(b/a)
        @return log(z) where z = a+i*b
    */
    public Complex log() {
        return new Complex(Math.log(this.mod()), this.arg());
    }

    /**
        The square root of a Complex number (z is unchanged).
        Returns the principal branch of the square root.
        <br> z = e^(i*theta) = r*cos(theta) + i*r*sin(theta)
        <br> r = sqrt(a^2+b^2)
        <br> cos(theta) = a / r, sin(theta) = b / r
        <br> By De Moivre's Theorem, sqrt(z) = sqrt(a+i*b) =
        * e^(i*theta / 2) = r(cos(theta/2) + i*sin(theta/2))
        @return sqrt(z) where z = a+i*b
    */
    public Complex sqrt() {
        double r = this.mod();
        double halfTheta = this.arg() / 2;
        return new Complex(Math.sqrt(r) * Math.cos(halfTheta), Math.sqrt(r) *
                Math.sin(halfTheta));
    }

    /**
        The real cosh function for Complex numbers.
        <br> cosh(theta) = (e^(theta) + e^(-theta)) / 2
        @return cosh(theta)
    */
    private double cosh(double theta) {
        return (Math.exp(theta) + Math.exp(-theta)) / 2;
    }

    /**
        The real sinh function for Complex numbers.
        <br> sinh(theta) = (e^(theta) - e^(-theta)) / 2
        @return sinh(theta)
    */
    private double sinh(double theta) {
        return (Math.exp(theta) - Math.exp(-theta)) / 2;
    }

     /**
        The sin function for the Complex number (z is unchanged).
        <br> sin(a+i*b) = cosh(b)*sin(a) + i*(sinh(b)*cos(a))
        @return sin(z) where z = a+i*b
    */
    public Complex sin() {
        return new Complex(cosh(im.doubleValue()) * Math.sin(re.doubleValue()),
                sinh(im.doubleValue())* Math.cos(re.doubleValue()));
    }

    /**
        The cos function for the Complex number (z is unchanged).
```

```
1223        <br> cos(a +i*b) = cosh(b)*cos(a) + i*(-sinh(b)*sin(a))
1224        @return cos(z) where z = a+i*b
1225     */
1226     public Complex cos() {
1227         return new Complex(cosh(im.doubleValue()) * Math.cos(re.doubleValue()),
1228                 -sinh(im.doubleValue()) * Math.sin(re.doubleValue()));
1229     }
1230
1231     /**
1232         The hyperbolic sin of the Complex number (z is unchanged).
1233         <br> sinh(a+i*b) = sinh(a)*cos(b) + i*(cosh(a)*sin(b))
1234         @return sinh(z) where z = a+i*b
1235     */
1236     public Complex sinh() {
1237         return new Complex(sinh(re.doubleValue()) * Math.cos(im.doubleValue()),
1238                 cosh(re.doubleValue()) * Math.sin(im.doubleValue()));
1239     }
1240
1241     /**
1242         The hyperbolic cosine of the Complex number (z is unchanged).
1243         <br> cosh(a+i*b) = cosh(a)*cos(b) + i*(sinh(a)*sin(b))
1244         @return cosh(z) where z = a+i*b
1245     */
1246     public Complex cosh() {
1247         return new Complex(cosh(re.doubleValue()) *Math.cos(im.doubleValue()),
1248                 sinh(re.doubleValue()) * Math.sin(im.doubleValue()));
1249     }
1250
1251     /**
1252         The tan of the Complex number (z is unchanged).
1253         <br> tan (a+i*b) = sin(a+i*b) / cos(a+i*b)
1254         @return tan(z) where z = a+i*b
1255     */
1256     public Complex tan() {
1257         return (this.sin()).divide(this.cos());
1258     }
1259
1260     /**
1261         The arctan of the Complex number (z is unchanged).
1262         <br> tan^(-1)(a+i*b) = 1/2 i*(log(1-i*(a+b*i))-log(1+i*(a+b*i))) =
1263         <br> -1/2 i*(log(i*a - b+1)-log(-i*a + b+1))
1264         @return arctan(z) where z = a+i*b
1265     */
1266     public Complex atan(){
1267         Complex ima = new Complex(0.0,-1.0);     //multiply by negative i
1268         Complex num = new Complex(this.re.doubleValue(),this.im.doubleValue()
1269                 -1.0);
1270         Complex den = new Complex(this.re.negate().doubleValue(),this.im
1271                 .negate().doubleValue()-1.0);
1272         Complex two = new Complex(2.0, 0.0);     // divide by 2
1273         return ima.multiply(num.divide(den).log()).divide(two);
1274     }
1275
1276     /**
1277    * The Math.pow equivalent of two Complex numbers.
1278    * @param z - the complex base in the form z = a + i*b
1279    * @return z^y where z = a + i*b and y = c + i*d
1280     */
1281     public Complex pow(Complex z){
1282         Complex a = z.multiply(this.log(), MathContext.UNLIMITED);
1283         return a.exp();
1284     }
1285
1286     /**
1287    * The Math.pow equivalent of a Complex number to the power
```

```
         * of a double.
   * @param d - the double to be taken as the power.
   * @return z^d where z = a + i*b and d = double
    */
     public Complex pow(double d){
         Complex a=(this.log()).multiply(d);
         return a.exp();
     }

    /**
        Override the .toString() method to generate complex numbers, the
        string representation is now a literal Complex number.
        @return a+i*b, a-i*b, a, or i*b as desired.
    */
    public String toString() {
        if (re.doubleValue() != 0.0 && im.doubleValue()  > 0.0) {
            return re + " + " + im +"*i";
        }
        if (re.doubleValue() != 0.0 && im.doubleValue() < 0.0) {
            return re + " - "+ (-im.doubleValue()) + "*i";
        }
        if (im.doubleValue() == 0.0) {
            return String.valueOf(re);
        }
        if (re.doubleValue() == 0.0) {
            return im + "*i";
        }
        return re + " + i*" + im;
    }}
```

4.9 Appendix I - BernoulliZeta.java

```
/*************************************************************************
**
**    Bernoulli numbers to calculate of Zeta(n)
**
*************************************************************************
**    Matthew Kehoe
**    09/15/2015
**
**    The Bernoulli numbers have multiple interesting relationships with the
**    Zeta function. One of these relationships, originally found by Euler,
**    states that Zeta(2n) = -1^(n+1) * B_2n * (2*pi)^2n / 2 * (2n)!. Another
**    relationship is defined by Zeta(-n) = - B_{n+1}/(n+1). This program is
**    designed to calculate even and odd integer values for the Zeta function.
*************************************************************************/


import java.math.BigInteger;
import org.apache.commons.math3.fraction.BigFraction;

public class BernoulliZeta {

    // Main method
    public static void main(String[] args) {
        System.out.println("The value of Zeta(2n) in the form \u03A0^2 /"
                + " fraction.");
        System.out.println();
        printZeta2N();

        // Uncomment to print out Zeta(-n)
        /* System.out.println("The value of Zeta(-N) in BigFraction form.");
         System.out.println();
         printZetaNegN();*/
    }
```

```
1351
1352     /**
1353    * Prints the values of Zeta(2n) in the format of
1354          * Numerator = BigInteger * pi^2n
1355          * Denominator = BigInteger
1356     */
1357     public static void printZeta2N(){
1358         String pi = "\u03A0";
1359
1360         for (int i = 1; i < 1000; i++) {
1361
1362             if( evenZeta(i).getNumerator().doubleValue() > 1)
1363                 System.out.println("Zeta of " + 2*i + " = " +
1364                         evenZeta(i).getNumerator() + " " + pi + "^" + 2*i +
1365                         " / " + evenZeta(i).getDenominator());
1366             else
1367                 System.out.println("Zeta of " + 2*i + " = " + pi +
1368                         "^" + 2*i + " / " + evenZeta(i).getDenominator());
1369         }
1370     }
1371
1372     /**
1373    * Prints the values of Zeta(-n) in the format of
1374          * Numerator = BigFraction Bern_n+1
1375          * Denominator = BigInteger n+1
1376     */
1377     public static void printZetaNegN(){
1378         for (int i = 1; i < 1000; i++) {
1379             System.out.println("Zeta of " + -i + " = "
1380                 + oddZeta(i));
1381         }
1382     }
1383
1384
1385     /**
1386    * BigFraction representation of the Zeta value for Zeta(2N).
1387          * @param n the int value for Zeta(2n)
1388          * Numerator = (-1)^n+1 * B_2n * (2pi)^2n
1389          * Denominator = 2*(2n)!
1390    * @return the value of Zeta(2n)
1391     */
1392     public static BigFraction evenZeta(int n){
1393
1394         BigInteger negOne = BigInteger.valueOf(-1);
1395         BigInteger two = BigInteger.valueOf(2);
1396         BigInteger j = BigInteger.valueOf(n);
1397
1398         // Sign
1399         BigInteger sign = negOne.pow(n+1);
1400
1401         // Numerator
1402         BigFraction bernNum = bernoulli(2*n);
1403         BigFraction num = bernNum.multiply(two.pow(2*n)).multiply(sign);
1404
1405         // Denominator
1406         BigInteger twoN = two.multiply(BigInteger.valueOf(n));
1407         BigInteger den = two.multiply(factorial(twoN));
1408
1409         BigFraction zeta2N = num.divide(den);
1410         return zeta2N;
1411     }
1412
1413     /**
1414    * BigFraction representation of the Zeta value for Zeta(-n).
1415          * @param n the int value for Zeta(-n)
```

```
         * - B_n+1/(n+1)
    * @return the value of Zeta(-n)
     */
    public static BigFraction oddZeta(int n){

        BigInteger negOne = BigInteger.valueOf(-1);
        BigFraction bernNum = bernoulli(n+1);
        BigInteger j = BigInteger.valueOf(n+1);
        return bernNum.divide(j).multiply(negOne);
    }

    /**
     * The factorial function returning a BigInteger
     * @param n the BigInteger to pass through for the factorial(n)
     * @return the factorial of n
    */
    public static BigInteger factorial(BigInteger n) {
        BigInteger result = BigInteger.ONE;

        while (!n.equals(BigInteger.ZERO)) {
            result = result.multiply(n);
            n = n.subtract(BigInteger.ONE);
        }

        return result;
    }

    /* Generates the Bernoulli number, B_n, by a double sum.
     * @param n The index of the Bernoulli number.
     * @return The Bernoulli number at n.
     */
    private static BigFraction bernoulli(int n) {
        BigFraction result = BigFraction.ZERO;
        for (int k = 0; k <= n; k++) {
            BigFraction jSum = BigFraction.ZERO;
            BigInteger bInt = BigInteger.ONE;
            for (int j = 0; j <= k; j++) {
                BigInteger jPowN = (new BigInteger("" + j))
                        .pow(n);
                if (j % 2 == 0) {
                    jSum = jSum.add(bInt.multiply(jPowN));
                }
                else {
                    jSum = jSum.subtract(bInt.multiply(jPowN));
                }

                /* updates binomial(k,j) recursively
                 */
                bInt = bInt.multiply(new BigInteger("" + (k - j))).
                        divide(new BigInteger("" + (j + 1)));
            }
            result = result.add(jSum.divide(new BigInteger("" + (k + 1)))
            );
        }
        return result;
    }
}
```

4.10 Appendix J - Table of zeroes for Lehmer's phenomenon

The following table is due to my investigation of Lehmer's phenomenon through RiemannSiegel.java. I calculated all consecutive zeroes where $|\rho_{n+1} - \rho_n| < 0.025$.

Table 4.1: Consecutive zeroes less than 0.025 apart within $0 < t < 5000000$

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
82552	82553	63137.21153	63137.23238	0.020849501
87761	87762	66678.07586	66678.09534	0.019484631
95248	95249	71732.90121	71732.91591	0.014701475
200371	200372	139735.4983	139735.5205	0.022193481
254019	254020	173042.4722	173042.4885	0.016287013
262371	262372	178168.1651	178168.1878	0.022713403
269053	269054	182258.5504	182258.5681	0.017647035
285008	285009	191989.9288	191989.9515	0.022747537
293696	293697	197268.9603	197268.9768	0.016493141
307833	307834	205829.7413	205829.7623	0.020929496
329526	329527	218900.2218	218900.2425	0.02061241
332254	332255	220538.853	220538.8702	0.017246952
335501	335502	222487.1519	222487.1757	0.023891669
354769	354770	234016.895	234016.908	0.013053218
378381	378382	248072.7441	248072.7639	0.019821864
384080	384081	251453.7823	251453.8036	0.021385919
390607	390608	255321.3153	255321.3368	0.021534942
415587	415588	270071.2941	270071.3027	0.008636391
420891	420892	273193.6631	273193.6688	0.005703705
427431	427432	277038.9687	277038.9892	0.020540295
450021	450022	290282.0112	290282.0263	0.015101511
467285	467286	300365.0045	300365.0283	0.023801496
471291	471292	302700.3008	302700.3104	0.009564464
480268	480269	307927.1672	307927.1884	0.02120095
484839	484840	310585.556	310585.5794	0.023389019
490691	490692	313985.8784	313985.8954	0.017065426
509087	509088	324652.6202	324652.6423	0.022072184
576841	576842	363671.1596	363671.175	0.015327476
594017	594018	373500.2921	373500.3136	0.021437384
599731	599732	376764.9388	376764.9609	0.022118771
600914	600915	377440.6954	377440.719	0.023617951
640646	640647	400067.997	400068.0162	0.019194723
640807	640808	400159.3223	400159.34	0.017758604
643745	643746	401828.139	401828.1485	0.009497832
652695	652696	406907.1308	406907.1542	0.023377626
658789	658790	410362.026	410362.0408	0.014834329
658891	658892	410419.6988	410419.7219	0.023077232
664997	664998	413878.9052	413878.9277	0.022470975
674151	674152	419059.834	419059.8561	0.022092047
692679	692680	429528.1705	429528.1938	0.023285299
703892	703893	435852.8393	435852.8572	0.017928007
707515	707516	437894.4613	437894.4806	0.0193435
710560	710561	439609.6516	439609.6735	0.02187573

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
713556	713557	441296.9992	441297.0149	0.015731596
714183	714184	441649.8183	441649.8273	0.008925753
715868	715869	442598.6213	442598.6426	0.021311183
718123	718124	443867.7169	443867.7333	0.01632961
735329	735330	453540.824	453540.8368	0.012811027
744979	744980	458957.8785	458957.8961	0.01757116
747591	747592	460422.9208	460422.9302	0.009387113
751881	751882	462828.771	462828.7905	0.019458264
753237	753238	463589.1406	463589.1617	0.021158173
757260	757261	465843.3937	465843.4181	0.02437901
806436	806437	493326.1632	493326.1825	0.0193001
830305	830306	506616.5065	506616.5305	0.024027127
830921	830922	506959.3064	506959.327	0.020554907
838236	838237	511026.176	511026.194	0.017943864
839026	839027	511464.8957	511464.9025	0.006807653
841640	841641	512917.3306	512917.353	0.022434008
849671	849672	517377.3899	517377.4015	0.011650372
856943	856944	521412.7924	521412.8097	0.017263982
860964	860965	523642.8499	523642.8744	0.0244727
861550	861551	523967.7954	523967.8131	0.017775119
878627	878628	533429.1589	533429.1723	0.013367381
891788	891789	540711.6302	540711.654	0.02382436
899614	899615	545037.3444	545037.3656	0.021221324
917906	917907	555136.9163	555136.9315	0.015177435
930374	930375	562011.4591	562011.4683	0.009254825
949213	949214	572385.354	572385.3785	0.024521012
957675	957676	577039.2584	577039.2749	0.016527699
968024	968025	582727.1338	582727.1503	0.016523612
989804	989805	594681.0082	594681.0327	0.024498315
999404	999405	599943.3697	599943.3795	0.009846958
1004470	1004471	602718.6948	602718.714	0.019173876
1013225	1013226	607512.5508	607512.5734	0.022596915
1026628	1026629	614844.7766	614844.7992	0.022599819
1028621	1028622	615934.2628	615934.2873	0.024507719
1050144	1050145	627691.1972	627691.2072	0.009948276
1053074	1053075	629290.1815	629290.1947	0.01320463
1061111	1061112	633674.277	633674.2974	0.020416742
1074066	1074067	640736.047	640736.0707	0.023722141
1076882	1076883	642269.8596	642269.8811	0.021578973
1082401	1082402	645275.9393	645275.9467	0.007390148
1097797	1097798	653654.0131	653654.0267	0.013580461
1115578	1115579	663318.5083	663318.5113	0.002958679
1137218	1137219	675064.2749	675064.2909	0.015971725
1137375	1137376	675149.609	675149.6213	0.012273705
1162322	1162323	688668.6609	688668.682	0.021080706
1164883	1164884	690054.8601	690054.8775	0.017380658
1169624	1169625	692620.9588	692620.9806	0.021776941
1169838	1169839	692736.741	692736.7631	0.022071795
1179154	1179155	697776.878	697776.9026	0.024572289
1186489	1186490	701742.6585	701742.6705	0.011987069
1205484	1205485	712004.0005	712004.0069	0.006348226

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
1240514	1240515	730894.9679	730894.988	0.020093024
1248419	1248420	735152.48	735152.4965	0.016538365
1266638	1266639	744955.8132	744955.8378	0.024590158
1293943	1293944	759628.023	759628.0351	0.012108264
1302749	1302750	764354.8704	764354.8909	0.020531364
1317074	1317075	772038.531	772038.5505	0.019437651
1320093	1320094	773657.1461	773657.1559	0.009833234
1339135	1339136	783859.0949	783859.1062	0.011327984
1360444	1360445	795262.268	795262.2885	0.020453828
1374512	1374513	802782.8379	802782.8481	0.01016582
1377748	1377749	804511.9719	804511.9842	0.012288183
1393625	1393626	812991.0081	812991.0245	0.016448652
1395207	1395208	813835.1174	813835.1366	0.019205471
1413964	1413965	823841.8997	823841.9209	0.02116325
1430207	1430208	832498.9751	832498.9999	0.024812777
1436559	1436560	835882.0002	835882.0237	0.023501627
1440239	1440240	837841.7498	837841.7697	0.019889009
1441606	1441607	838569.6143	838569.6379	0.02359673
1457772	1457773	847172.8025	847172.8171	0.014655715
1457942	1457943	847263.1402	847263.1502	0.009969334
1475683	1475684	856695.4176	856695.4362	0.018605211
1480212	1480213	859102.5	859102.5212	0.021217065
1488697	1488698	863609.4003	863609.4235	0.023211147
1526054	1526055	883430.1301	883430.1526	0.022555738
1532715	1532716	886959.8914	886959.9116	0.020146599
1536185	1536186	888798.6659	888798.69	0.024083218
1557463	1557464	900065.5828	900065.6073	0.024525339
1557823	1557824	900256.1081	900256.1225	0.014375979
1560037	1560038	901427.6003	901427.6202	0.019913923
1584920	1584921	914586.8487	914586.8735	0.024867148
1589544	1589545	917030.3214	917030.3322	0.01072752
1633106	1633107	940024.1821	940024.2059	0.023791599
1646544	1646545	947107.8201	947107.8325	0.01242246
1668773	1668774	958815.9051	958815.9271	0.02200368
1670348	1670349	959644.8221	959644.8371	0.015055745
1671680	1671681	960346.2086	960346.2279	0.019347341
1686735	1686736	968267.725	968267.7442	0.019206299
1689687	1689688	969820.2674	969820.2888	0.02144303
1693111	1693112	971621.0091	971621.0331	0.023973114
1698836	1698837	974630.8876	974630.8984	0.010831646
1702136	1702137	976365.6598	976365.6741	0.014356263
1710913	1710914	980978.0779	980978.0912	0.013319331
1718524	1718525	984976.3876	984976.4087	0.021066873
1727569	1727570	989726.3817	989726.4049	0.02325436
1734211	1734212	993212.907	993212.9222	0.015210132
1737933	1737934	995166.4751	995166.4998	0.024746152
1740144	1740145	996326.6879	996326.709	0.021070426
1743531	1743532	998103.8133	998103.8368	0.023466891
1751445	1751446	1002254.889	1002254.902	0.013493482
1757395	1757396	1005375.226	1005375.246	0.01954172
1770407	1770408	1012196.445	1012196.468	0.023317273

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
1784770	1784771	1019720.689	1019720.71	0.020789367
1790601	1790602	1022774.321	1022774.337	0.015965384
1831653	1831654	1044250.016	1044250.036	0.020466295
1853009	1853010	1055407.79	1055407.813	0.023025292
1853487	1853488	1055657.193	1055657.216	0.023308602
1855472	1855473	1056694.248	1056694.266	0.018057352
1872256	1872257	1065454.857	1065454.877	0.019603107
1873931	1873932	1066329.318	1066329.332	0.014365614
1886393	1886394	1072829.839	1072829.853	0.014695199
1892310	1892311	1075915.16	1075915.178	0.018094293
1897158	1897159	1078442.784	1078442.803	0.01921118
1909529	1909530	1084889.826	1084889.85	0.024216939
1910997	1910998	1085654.814	1085654.838	0.02410434
1921598	1921599	1091176.795	1091176.809	0.014860077
1927523	1927524	1094262.125	1094262.132	0.00678618
1928049	1928050	1094535.978	1094536	0.021744302
1938950	1938951	1100210.05	1100210.073	0.022387084
1940164	1940165	1100841.885	1100841.899	0.013583981
1948489	1948490	1105173.844	1105173.862	0.018298764
1959057	1959058	1110670.415	1110670.44	0.024401449
1962887	1962888	1112661.837	1112661.861	0.023693101
1965293	1965294	1113913.059	1113913.079	0.020134805
1967979	1967980	1115309.383	1115309.406	0.023096341
1981332	1981333	1122248.837	1122248.862	0.024842836
1988927	1988928	1126194.45	1126194.472	0.021553037
2001185	2001186	1132559.978	1132559.998	0.02004583
2009961	2009962	1137116.071	1137116.089	0.018503048
2050466	2050467	1158122.147	1158122.169	0.022181438
2050815	2050816	1158302.96	1158302.972	0.011616387
2051152	2051153	1158477.529	1158477.551	0.022446021
2071825	2071826	1169186.579	1169186.596	0.017200241
2083140	2083141	1175044.41	1175044.434	0.024215251
2085435	2085436	1176232.242	1176232.252	0.009652807
2086313	2086314	1176686.955	1176686.98	0.024638676
2111074	2111075	1189495.888	1189495.908	0.020185362
2148094	2148095	1208625.586	1208625.605	0.018895078
2155672	2155673	1212538.44	1212538.453	0.013250527
2163215	2163216	1216432.57	1216432.58	0.010198602
2213296	2213297	1242258.648	1242258.666	0.018633355
2214413	2214414	1242833.998	1242834.02	0.021704326
2217173	2217174	1244255.865	1244255.884	0.01896944
2219761	2219762	1245589.018	1245589.036	0.017649967
2232353	2232354	1252074.323	1252074.347	0.024192358
2237216	2237217	1254578.364	1254578.383	0.018445469
2243172	2243173	1257644.03	1257644.054	0.023927743
2251237	2251238	1261794.883	1261794.902	0.019406835
2261276	2261277	1266960.067	1266960.092	0.024653869
2277451	2277452	1275278.285	1275278.309	0.024064365
2290412	2290413	1281940.449	1281940.469	0.019649411
2326271	2326272	1300358.484	1300358.503	0.018281911
2326565	2326566	1300509.59	1300509.613	0.022872211

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
2332792	2332793	1303705.702	1303705.713	0.011309108
2347208	2347209	1311102.12	1311102.138	0.018766859
2366495	2366496	1320993.306	1320993.324	0.017660991
2368001	2368002	1321765.453	1321765.47	0.01692339
2390906	2390907	1333502.965	1333502.987	0.022103799
2407298	2407299	1341897.946	1341897.966	0.019191986
2411958	2411959	1344283.74	1344283.754	0.014486739
2423459	2423460	1350170.084	1350170.1	0.015481247
2448777	2448778	1363121.6	1363121.619	0.018365362
2484410	2484411	1381332.786	1381332.8	0.013941265
2496534	2496535	1387524.654	1387524.666	0.012264149
2500512	2500513	1389555.694	1389555.711	0.017716617
2530619	2530620	1404920.129	1404920.152	0.022820471
2536614	2536615	1407977.89	1407977.913	0.022981227
2544599	2544600	1412049.975	1412049.991	0.016325583
2597349	2597350	1438925.829	1438925.849	0.0196364
2598394	2598395	1439457.546	1439457.567	0.021662809
2635662	2635663	1458420.51	1458420.533	0.02334999
2637618	2637619	1459415.381	1459415.406	0.024961431
2651449	2651450	1466447.355	1466447.375	0.019944349
2662350	2662351	1471987.724	1471987.74	0.015992164
2663490	2663491	1472566.982	1472567.006	0.024162364
2677096	2677097	1479479.535	1479479.559	0.02444829
2679229	2679230	1480563.271	1480563.295	0.023428075
2681474	2681475	1481703.654	1481703.672	0.017434421
2687611	2687612	1484820.353	1484820.372	0.019899325
2701906	2701907	1492077.94	1492077.961	0.020961143
2708410	2708411	1495379.233	1495379.254	0.020927001
2709806	2709807	1496087.94	1496087.958	0.017840188
2713528	2713529	1497976.671	1497976.693	0.022604622
2716545	2716546	1499507.683	1499507.707	0.023997724
2719103	2719104	1500805.69	1500805.697	0.007592808
2722877	2722878	1502720.053	1502720.075	0.021938314
2732243	2732244	1507471.013	1507471.022	0.00864007
2754390	2754391	1518700.806	1518700.823	0.016808543
2779538	2779539	1531443.948	1531443.971	0.022743152
2825904	2825905	1554916.003	1554916.014	0.010407676
2833891	2833892	1558956.387	1558956.41	0.02319627
2843831	2843832	1563983.894	1563983.918	0.023903553
2860222	2860223	1572270.957	1572270.98	0.022387514
2863315	2863316	1573834.334	1573834.344	0.010096335
2870844	2870845	1577639.311	1577639.322	0.010799135
2874330	2874331	1579400.943	1579400.968	0.024743644
2874964	2874965	1579721.076	1579721.097	0.02038246
2877423	2877424	1580963.616	1580963.638	0.022254941
2877871	2877872	1581190.114	1581190.134	0.019720607
2893193	2893194	1588930.016	1588930.032	0.016354147
2896001	2896002	1590347.894	1590347.919	0.024919019
2898395	2898396	1591557.309	1591557.322	0.012942455
2935244	2935245	1610156.488	1610156.51	0.021935437
2939458	2939459	1612282.493	1612282.505	0.012094858

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
2966551	2966552	1625944.971	1625944.989	0.017868011
2968219	2968220	1626785.954	1626785.97	0.016006565
2973555	2973556	1629475.956	1629475.971	0.015332507
2977840	2977841	1631635.384	1631635.409	0.024648109
2986212	2986213	1635854.206	1635854.218	0.012136836
2994897	2994898	1640229.762	1640229.784	0.021731103
3003208	3003209	1644416.146	1644416.17	0.023679825
3024855	3024856	1655316.057	1655316.08	0.023037179
3039026	3039027	1662448.536	1662448.546	0.010121621
3039159	3039160	1662515.735	1662515.743	0.008012865
3051633	3051634	1668791.575	1668791.599	0.023874407
3066643	3066644	1676341.138	1676341.162	0.024287361
3080654	3080655	1683386.188	1683386.204	0.015810023
3090252	3090253	1688210.262	1688210.286	0.023700069
3115581	3115582	1700937.27	1700937.294	0.023797831
3141965	3141966	1714185.759	1714185.779	0.020024325
3145628	3145629	1716024.497	1716024.52	0.023602286
3168692	3168693	1727598.154	1727598.177	0.022772259
3180863	3180864	1733703.514	1733703.533	0.018477611
3186088	3186089	1736323.704	1736323.725	0.021838351
3189956	3189957	1738263.341	1738263.365	0.02323691
3190424	3190425	1738498.013	1738498.037	0.023764255
3200512	3200513	1743555.789	1743555.81	0.020621097
3201703	3201704	1744152.873	1744152.887	0.013334441
3201980	3201981	1744291.647	1744291.671	0.024497771
3213652	3213653	1750141.697	1750141.708	0.010751954
3215557	3215558	1751096.752	1751096.772	0.01965947
3236171	3236172	1761424.625	1761424.646	0.020618197
3260397	3260398	1773555.981	1773555.995	0.01449261
3261490	3261491	1774103.469	1774103.493	0.02315068
3271858	3271859	1779292.804	1779292.808	0.004161135
3298013	3298014	1792379.875	1792379.896	0.020095386
3336513	3336514	1811629.37	1811629.39	0.019722744
3337440	3337441	1812092.929	1812092.953	0.024465217
3339063	3339064	1812903.906	1812903.919	0.012971169
3345641	3345642	1816191.221	1816191.244	0.022324408
3352238	3352239	1819487.044	1819487.068	0.024482447
3354560	3354561	1820647.248	1820647.272	0.023870202
3375427	3375428	1831069.72	1831069.738	0.018462216
3376274	3376275	1831492.631	1831492.645	0.013520617
3380279	3380280	1833492.505	1833492.521	0.015957988
3382202	3382203	1834452.707	1834452.722	0.014554699
3385612	3385613	1836155.113	1836155.137	0.023932556
3407000	3407001	1846830.526	1846830.539	0.012342109
3414516	3414517	1850581.181	1850581.2	0.019647514
3431244	3431245	1858925.477	1858925.499	0.022090942
3455802	3455803	1871171.089	1871171.109	0.019819291
3458235	3458236	1872383.889	1872383.91	0.021078293
3466242	3466243	1876374.802	1876374.815	0.013743137
3470356	3470357	1878424.933	1878424.95	0.016939434
3472785	3472786	1879635.623	1879635.635	0.012620774

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
3481107	3481108	1883781.86	1883781.878	0.017849786
3509282	3509283	1897815.643	1897815.658	0.015026219
3512555	3512556	1899445.474	1899445.485	0.011417979
3513027	3513028	1899680.608	1899680.632	0.023576019
3515922	3515923	1901121.815	1901121.836	0.021071228
3550103	3550104	1918133.565	1918133.579	0.01380437
3561839	3561840	1923971.853	1923971.878	0.024866042
3562410	3562411	1924255.993	1924256.016	0.022967138
3566799	3566800	1926438.912	1926438.934	0.021330242
3567735	3567736	1926904.147	1926904.172	0.024059941
3569898	3569899	1927980.061	1927980.085	0.024513464
3576284	3576285	1931155.651	1931155.671	0.019779969
3584896	3584897	1935437.852	1935437.876	0.023993607
3589059	3589060	1937507.367	1937507.39	0.023098118
3590165	3590166	1938057.405	1938057.428	0.02278232
3602882	3602883	1944378.183	1944378.2	0.017086483
3628979	3628980	1957344.208	1957344.231	0.022685511
3628995	3628996	1957352.356	1957352.373	0.017465889
3637897	3637898	1961773.993	1961773.997	0.003259306
3637921	3637922	1961785.683	1961785.708	0.024605298
3647914	3647915	1966748.405	1966748.429	0.024149218
3658078	3658079	1971794.713	1971794.738	0.024810189
3658685	3658686	1972096.022	1972096.043	0.020865513
3660604	3660605	1973048.549	1973048.564	0.014529366
3665691	3665692	1975573.762	1975573.785	0.023161798
3670012	3670013	1977718.303	1977718.325	0.021351019
3679339	3679340	1982347.222	1982347.233	0.011406136
3686952	3686953	1986124.458	1986124.483	0.024564044
3699021	3699022	1992111.904	1992111.927	0.023843911
3707599	3707600	1996366.604	1996366.629	0.024962218
3710220	3710221	1997666.649	1997666.667	0.018503317
3711235	3711236	1998169.813	1998169.83	0.01754264
3715059	3715060	2000065.676	2000065.697	0.021000029
3727059	3727060	2006015.973	2006015.995	0.02173692
3735555	3735556	2010227.239	2010227.255	0.015843237
3736351	3736352	2010622.12	2010622.131	0.011451835
3742894	3742895	2013865.19	2013865.214	0.024012665
3744874	3744875	2014846.515	2014846.522	0.007488953
3747909	3747910	2016350.697	2016350.717	0.019285745
3748646	3748647	2016715.65	2016715.672	0.022500722
3754010	3754011	2019373.5	2019373.524	0.023898685
3754223	3754224	2019479.25	2019479.272	0.022042864
3762150	3762151	2023406.467	2023406.485	0.018189102
3772701	3772702	2028633.468	2028633.488	0.019878733
3773678	3773679	2029117.555	2029117.565	0.009688315
3777202	3777203	2030862.866	2030862.889	0.022972708
3781083	3781084	2032784.87	2032784.89	0.020741176
3784299	3784300	2034377.795	2034377.804	0.009072751
3808269	3808270	2046245.216	2046245.234	0.018499822
3814843	3814844	2049499.298	2049499.322	0.024384914
3823279	3823280	2053673.644	2053673.656	0.012861361

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
3826528	3826529	2055281.925	2055281.945	0.020220426
3832771	3832772	2058370.39	2058370.4	0.010647587
3838053	3838054	2060983.94	2060983.965	0.024658866
3842280	3842281	2063074.551	2063074.574	0.022982209
3852053	3852054	2067908.702	2067908.714	0.011378083
3859129	3859130	2071408.129	2071408.15	0.02081626
3915227	3915228	2099134.344	2099134.369	0.024858193
3918709	3918710	2100854.504	2100854.529	0.024844968
3925658	3925659	2104287.071	2104287.095	0.023783484
3944746	3944747	2113712.854	2113712.87	0.016537079
3945694	3945695	2114180.865	2114180.883	0.017684291
3946915	3946916	2114783.838	2114783.861	0.023708837
3953001	3953002	2117788.134	2117788.142	0.007890746
3965312	3965313	2123864.894	2123864.908	0.013693273
3966813	3966814	2124605.42	2124605.424	0.004136706
3976713	3976714	2129491.066	2129491.087	0.020742234
3991042	3991043	2136560.555	2136560.579	0.024623779
3995658	3995659	2138837.559	2138837.577	0.017883536
4003134	4003135	2142524.907	2142524.926	0.018990819
4022261	4022262	2151956.874	2151956.888	0.014487634
4031853	4031854	2156685.537	2156685.56	0.022759417
4032047	4032048	2156781.337	2156781.356	0.019240983
4038268	4038269	2159847.654	2159847.66	0.00662584
4049317	4049318	2165292.924	2165292.944	0.019438911
4064245	4064246	2172648.204	2172648.217	0.013666071
4075228	4075229	2178058.731	2178058.75	0.018840741
4078552	4078553	2179696.184	2179696.2	0.015826772
4090286	4090287	2185474.905	2185474.929	0.02342154
4094031	4094032	2187318.89	2187318.908	0.018144957
4097345	4097346	2188950.807	2188950.819	0.011225413
4102124	4102125	2191303.747	2191303.767	0.020274588
4114093	4114094	2197195.627	2197195.647	0.020577777
4121303	4121304	2200744.679	2200744.7	0.020725584
4156678	4156679	2218149.333	2218149.338	0.00485933
4166562	4166563	2223010.616	2223010.628	0.012093497
4179924	4179925	2229581.15	2229581.17	0.020502377
4196782	4196783	2237868.268	2237868.283	0.015102763
4205366	4205367	2242087.306	2242087.324	0.018332112
4210235	4210236	2244480.08	2244480.097	0.016371403
4243443	4243444	2260793.743	2260793.765	0.022580887
4263170	4263171	2270480.721	2270480.737	0.0158172
4273195	4273196	2275402.32	2275402.34	0.019785248
4273712	4273713	2275656.219	2275656.242	0.023354814
4275791	4275792	2276676.192	2276676.208	0.015501474
4286131	4286132	2281751.693	2281751.708	0.0147938
4293405	4293406	2285321.2	2285321.222	0.022695621
4297284	4297285	2287224.826	2287224.849	0.023315793
4299110	4299111	2288120.754	2288120.76	0.006459724
4301271	4301272	2289181.103	2289181.127	0.024619692
4303683	4303684	2290364.434	2290364.457	0.022540759
4307346	4307347	2292161.948	2292161.971	0.022547036

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
4313229	4313230	2295047.745	2295047.764	0.018661986
4317609	4317610	2297196.005	2297196.021	0.015854757
4318888	4318889	2297823.574	2297823.598	0.024418084
4320828	4320829	2298775.069	2298775.078	0.009166096
4334669	4334670	2305563.04	2305563.063	0.022552863
4339551	4339552	2307957.137	2307957.148	0.011126191
4350734	4350735	2313440.067	2313440.091	0.024529897
4357413	4357414	2316714.272	2316714.292	0.020572537
4366168	4366169	2321005.659	2321005.672	0.013428316
4369133	4369134	2322458.759	2322458.781	0.02198467
4371942	4371943	2323835.468	2323835.485	0.016869989
4374124	4374125	2324904.72	2324904.743	0.022486652
4374778	4374779	2325225.169	2325225.183	0.013802535
4377626	4377627	2326620.815	2326620.839	0.024104339
4392960	4392961	2334134.136	2334134.147	0.011005441
4398867	4398868	2337027.805	2337027.811	0.005471981
4401510	4401511	2338322.564	2338322.584	0.020419732
4404445	4404446	2339760.331	2339760.354	0.022943889
4405283	4405284	2340170.485	2340170.496	0.010674615
4410282	4410283	2342618.818	2342618.83	0.012506966
4410893	4410894	2342918.354	2342918.374	0.020184841
4418825	4418826	2346802.901	2346802.92	0.019677055
4431209	4431210	2352866.525	2352866.536	0.01132832
4444526	4444527	2359385.93	2359385.941	0.010981505
4447374	4447375	2360780.042	2360780.049	0.007335039
4463898	4463899	2368867.368	2368867.389	0.020577257
4470427	4470428	2372061.768	2372061.784	0.015383462
4471868	4471869	2372766.909	2372766.918	0.009227819
4482002	4482003	2377724.979	2377724.992	0.013055837
4484137	4484138	2378769.005	2378769.021	0.015235838
4484197	4484198	2378798.466	2378798.483	0.016837856
4491813	4491814	2382524.005	2382524.029	0.023815013
4508945	4508946	2390902.465	2390902.484	0.018741964
4514937	4514938	2393832.247	2393832.272	0.024918809
4525510	4525511	2399001.547	2399001.567	0.020704946
4536959	4536960	2404597.871	2404597.888	0.016646088
4539967	4539968	2406068.395	2406068.417	0.022243705
4544450	4544451	2408259.124	2408259.147	0.023435156
4552528	4552529	2412206.944	2412206.959	0.014350309
4556619	4556620	2414205.562	2414205.584	0.021703778
4562198	4562199	2416931.727	2416931.746	0.018879662
4569515	4569516	2420506.152	2420506.164	0.012770253
4572866	4572867	2422143.232	2422143.25	0.018861835
4580407	4580408	2425826.657	2425826.673	0.015704328
4605045	4605046	2437858.847	2437858.87	0.022371123
4609175	4609176	2439875.423	2439875.44	0.016670682
4617956	4617957	2444161.93	2444161.945	0.015239922
4620307	4620308	2445309.79	2445309.804	0.013795879
4621531	4621532	2445907.198	2445907.214	0.016168533
4652348	4652349	2460946.282	2460946.303	0.021005771
4658989	4658990	2464186.031	2464186.045	0.014117553

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
4661704	4661705	2465510.454	2465510.478	0.023344095
4664492	4664493	2466870.302	2466870.326	0.023735486
4674620	4674621	2471810.616	2471810.637	0.020839468
4679155	4679156	2474022.164	2474022.187	0.022578946
4694350	4694351	2481432.104	2481432.119	0.015199406
4706391	4706392	2487302.329	2487302.342	0.012595059
4707769	4707770	2487974.099	2487974.114	0.015211042
4717925	4717926	2492924.442	2492924.453	0.011330904
4727680	4727681	2497679.077	2497679.101	0.023611542
4733387	4733388	2500460.016	2500460.039	0.022781568
4743552	4743553	2505412.922	2505412.935	0.013275647
4750523	4750524	2508809.442	2508809.466	0.024240793
4754991	4754992	2510985.702	2510985.714	0.011671078
4766151	4766152	2516421.884	2516421.904	0.0203742
4768211	4768212	2517424.885	2517424.901	0.015871667
4773904	4773905	2520197.551	2520197.57	0.019678098
4776750	4776751	2521583.491	2521583.514	0.022912698
4777630	4777631	2522012.022	2522012.043	0.021246673
4777972	4777973	2522178.559	2522178.567	0.008069254
4786673	4786674	2526415.389	2526415.412	0.023579036
4790075	4790076	2528071.931	2528071.948	0.016931722
4802181	4802182	2533965.518	2533965.542	0.023782676
4807872	4807873	2536735.768	2536735.793	0.024649723
4808755	4808756	2537165.441	2537165.463	0.021767376
4810426	4810427	2537978.937	2537978.961	0.023565901
4813587	4813588	2539517.034	2539517.058	0.024319297
4819387	4819388	2542340.091	2542340.103	0.012717902
4819524	4819525	2542406.6	2542406.625	0.024845248
4821483	4821484	2543359.846	2543359.86	0.013922272
4832690	4832691	2548813.324	2548813.344	0.019439143
4842622	4842623	2553645.766	2553645.782	0.015714642
4849873	4849874	2557172.99	2557173.008	0.017416413
4871849	4871850	2567861.274	2567861.297	0.023537756
4874312	4874313	2569058.925	2569058.933	0.008621019
4874689	4874690	2569242.515	2569242.539	0.023858036
4882346	4882347	2572965.664	2572965.679	0.014410338
4882706	4882707	2573140.78	2573140.802	0.022234284
4888254	4888255	2575838.048	2575838.069	0.021263851
4888951	4888952	2576177.141	2576177.16	0.018451007
4893044	4893045	2578166.695	2578166.707	0.011781765
4904399	4904400	2583686.257	2583686.266	0.008580528
4906337	4906338	2584628.487	2584628.504	0.016756446
4921602	4921603	2592046.843	2592046.859	0.01550764
4924335	4924336	2593375.014	2593375.038	0.024012408
4929820	4929821	2596040.072	2596040.092	0.019852653
4934893	4934894	2598505.09	2598505.113	0.022462984
4942259	4942260	2602083.63	2602083.647	0.017174533
4947449	4947450	2604604.922	2604604.942	0.01943077
4948539	4948540	2605134.196	2605134.218	0.021911273
4954532	4954533	2608044.854	2608044.868	0.013745477
4957353	4957354	2609415.202	2609415.226	0.023437857

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
4962925	4962926	2612121.289	2612121.309	0.019634479
4963519	4963520	2612409.789	2612409.814	0.024612013
4965637	4965638	2613438.286	2613438.31	0.024823335
4972710	4972711	2616873.15	2616873.167	0.017296527
4980819	4980820	2620810.352	2620810.357	0.004883703
4983901	4983902	2622306.781	2622306.805	0.024431033
4991349	4991350	2625922.519	2625922.542	0.023638823
4991930	4991931	2626204.559	2626204.58	0.020581236
4993066	4993067	2626755.87	2626755.894	0.024145696
5000289	5000290	2630262.024	2630262.048	0.023923145
5041632	5041633	2650323.437	2650323.458	0.020368031
5042995	5042996	2650984.611	2650984.614	0.002970091
5060698	5060699	2659571.284	2659571.295	0.011114982
5069293	5069294	2663739.138	2663739.163	0.024831659
5095504	5095505	2676446.731	2676446.755	0.02477705
5101036	5101037	2679128.449	2679128.461	0.012297097
5108298	5108299	2682648.056	2682648.077	0.020381094
5113227	5113228	2685037.093	2685037.114	0.020229698
5119703	5119704	2688175.175	2688175.198	0.023430866
5121889	5121890	2689234.589	2689234.596	0.007040088
5145685	5145686	2700763.124	2700763.141	0.017291722
5146257	5146258	2701039.996	2701040.019	0.023437199
5185800	5185801	2720189.169	2720189.191	0.022036938
5198839	5198840	2726501.225	2726501.241	0.016194663
5208404	5208405	2731130.988	2731131.013	0.02498942
5210258	5210259	2732027.982	2732028.006	0.023759284
5226575	5226576	2739923.913	2739923.932	0.019388323
5232974	5232975	2743019.897	2743019.919	0.022117661
5236999	5237000	2744967.276	2744967.288	0.011966065
5239436	5239437	2746146.095	2746146.114	0.018358074
5248031	5248032	2750304.239	2750304.255	0.016592985
5282348	5282349	2766900.29	2766900.311	0.020974667
5286104	5286105	2768716.008	2768716.022	0.014190761
5295331	5295332	2773176.662	2773176.682	0.019881648
5301656	5301657	2776234.447	2776234.469	0.021524435
5302271	5302272	2776531.302	2776531.324	0.021686201
5316326	5316327	2783324.728	2783324.751	0.023505012
5321617	5321618	2785881.476	2785881.496	0.020548031
5322351	5322352	2786236.283	2786236.303	0.019654968
5328679	5328680	2789293.663	2789293.686	0.023297491
5333116	5333117	2791437.72	2791437.745	0.024958869
5345466	5345467	2797404.49	2797404.513	0.022488125
5369783	5369784	2809149.826	2809149.84	0.014688765
5374601	5374602	2811476.344	2811476.364	0.019676898
5377291	5377292	2812775.449	2812775.465	0.01573904
5409800	5409801	2828469.907	2828469.931	0.023984405
5414196	5414197	2830591.756	2830591.777	0.020863057
5415624	5415625	2831281.223	2831281.241	0.018198275
5419382	5419383	2833094.73	2833094.754	0.02379328
5420378	5420379	2833575.28	2833575.297	0.016929592
5421384	5421385	2834061.118	2834061.127	0.008617226

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
5430003	5430004	2838220.452	2838220.473	0.021656005
5432043	5432044	2839204.546	2839204.566	0.019533324
5439196	5439197	2842656.248	2842656.265	0.016918028
5452021	5452022	2848843.787	2848843.809	0.02248671
5452804	5452805	2849221.312	2849221.333	0.021463594
5487125	5487126	2865774.159	2865774.181	0.022255544
5496765	5496766	2870422.343	2870422.36	0.017172351
5497162	5497163	2870613.52	2870613.544	0.024650107
5518158	5518159	2880735.24	2880735.264	0.024307369
5518337	5518338	2880821.256	2880821.27	0.014599338
5521482	5521483	2882337.521	2882337.545	0.023274369
5539116	5539117	2890835.582	2890835.603	0.02081576
5547899	5547900	2895067.487	2895067.506	0.019102887
5549695	5549696	2895932.714	2895932.738	0.023894523
5549712	5549713	2895940.977	2895940.999	0.022010575
5570934	5570935	2906164.458	2906164.482	0.023635562
5579496	5579497	2910288.364	2910288.389	0.024351167
5581927	5581928	2911459.224	2911459.247	0.023300534
5582133	5582134	2911558.383	2911558.393	0.010761929
5587589	5587590	2914185.877	2914185.894	0.017142229
5588437	5588438	2914594.28	2914594.301	0.021385353
5605847	5605848	2922977.401	2922977.422	0.021403912
5608544	5608545	2924275.973	2924275.984	0.010578756
5619632	5619633	2929613.622	2929613.647	0.024794738
5642375	5642376	2940560.558	2940560.581	0.022511076
5643658	5643659	2941177.771	2941177.785	0.013523082
5649780	5649781	2944123.805	2944123.828	0.022984086
5650597	5650598	2944516.82	2944516.837	0.017337889
5660259	5660260	2949165.888	2949165.903	0.014160046
5668631	5668632	2953193.383	2953193.395	0.012218248
5674007	5674008	2955779.897	2955779.92	0.02292881
5674033	5674034	2955792.505	2955792.517	0.011372577
5675166	5675167	2956337.365	2956337.39	0.024883121
5679965	5679966	2958646.083	2958646.105	0.022175724
5691183	5691184	2964041.64	2964041.658	0.018494734
5691281	5691282	2964088.592	2964088.612	0.020631315
5693382	5693383	2965099.278	2965099.285	0.007429104
5693481	5693482	2965146.836	2965146.856	0.02050927
5731907	5731908	2983622.782	2983622.8	0.017340034
5733444	5733445	2984361.878	2984361.899	0.020759976
5739154	5739155	2987106.315	2987106.33	0.014651368
5752276	5752277	2993412.729	2993412.749	0.020003065
5758369	5758370	2996340.976	2996340.996	0.020295493
5763971	5763972	2999032.897	2999032.91	0.013221146
5766319	5766320	3000161.303	3000161.313	0.009015439
5771069	5771070	3002443.416	3002443.426	0.009707532
5775423	5775424	3004535.218	3004535.242	0.023427574
5777255	5777256	3005415.343	3005415.363	0.019880938
5785839	5785840	3009539.386	3009539.406	0.019599389
5790588	5790589	3011820.69	3011820.711	0.020758305
5815574	5815575	3023821.167	3023821.185	0.017592234

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
5840010	5840011	3035553.943	3035553.968	0.024215413
5843777	5843778	3037362.498	3037362.519	0.021530308
5845196	5845197	3038043.536	3038043.558	0.022209886
5857930	5857931	3044155.978	3044155.989	0.010094485
5868523	5868524	3049239.964	3049239.987	0.022432157
5870489	5870490	3050183.558	3050183.576	0.018073331
5873146	5873147	3051458.655	3051458.672	0.0177982
5881215	5881216	3055330.483	3055330.507	0.024412909
5885892	5885893	3057574.28	3057574.304	0.02347221
5889034	5889035	3059082.1	3059082.124	0.024274286
5892938	5892939	3060954.915	3060954.936	0.021260445
5895543	5895544	3062204.936	3062204.956	0.020239482
5903110	5903111	3065834.855	3065834.869	0.014023607
5910605	5910606	3069430.076	3069430.088	0.01180451
5923721	5923722	3075720.834	3075720.852	0.018231924
5923776	5923777	3075747.199	3075747.221	0.021501323
5925991	5925992	3076809.294	3076809.318	0.024364029
5947463	5947464	3087105.762	3087105.783	0.021198655
5963945	5963946	3095007.295	3095007.314	0.019309444
5987088	5987089	3106099.903	3106099.921	0.018393349
6005747	6005748	3115040.517	3115040.531	0.014655049
6014020	6014021	3119004.283	3119004.308	0.024817577
6020053	6020054	3121894.531	3121894.547	0.015927779
6024651	6024652	3124096.995	3124097.013	0.018377683
6033564	6033565	3128366.302	3128366.309	0.006892754
6040163	6040164	3131527.03	3131527.054	0.024240934
6043969	6043970	3133349.629	3133349.652	0.02326157
6044392	6044393	3133552.335	3133552.359	0.023837958
6057456	6057457	3139808.344	3139808.362	0.018494411
6067431	6067432	3144584.447	3144584.469	0.022194665
6070585	6070586	3146094.643	3146094.667	0.023523623
6076644	6076645	3148995.26	3148995.272	0.011835248
6077611	6077612	3149458.189	3149458.211	0.022594806
6079638	6079639	3150428.611	3150428.63	0.019070521
6092085	6092086	3156386.381	3156386.404	0.022352934
6104331	6104332	3162247.565	3162247.583	0.017855111
6109880	6109881	3164903.169	3164903.19	0.021196256
6126861	6126862	3173028.377	3173028.401	0.02380858
6134920	6134921	3176883.921	3176883.936	0.014853827
6135497	6135498	3177160.331	3177160.355	0.023848197
6138884	6138885	3178780.645	3178780.667	0.022011468
6144960	6144961	3181687.202	3181687.227	0.024277069
6181752	6181753	3199283.038	3199283.055	0.017781516
6183161	6183162	3199956.809	3199956.825	0.016186188
6190631	6190632	3203528.285	3203528.305	0.019901895
6191610	6191611	3203996.257	3203996.266	0.009206758
6204666	6204667	3210238.028	3210238.053	0.024540529
6204968	6204969	3210382.42	3210382.434	0.014281526
6207589	6207590	3211635.302	3211635.32	0.018613053
6217864	6217865	3216546.458	3216546.467	0.009062152
6224732	6224733	3219828.924	3219828.941	0.016451637

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
6226729	6226730	3220783.165	3220783.176	0.010539219
6232147	6232148	3223372.601	3223372.62	0.019235707
6235908	6235909	3225169.772	3225169.797	0.024658462
6245071	6245072	3229548.26	3229548.274	0.013992692
6255347	6255348	3234457.997	3234458.017	0.020259791
6263903	6263904	3238545.553	3238545.578	0.024607971
6264772	6264773	3238960.276	3238960.294	0.018337036
6268114	6268115	3240557.162	3240557.185	0.022900628
6270817	6270818	3241848.083	3241848.099	0.016262987
6272815	6272816	3242802.251	3242802.275	0.024224582
6276580	6276581	3244601.166	3244601.189	0.023378304
6281023	6281024	3246723.25	3246723.269	0.019463192
6283527	6283528	3247918.992	3247919.016	0.024276197
6291820	6291821	3251879.654	3251879.679	0.024843132
6293797	6293798	3252823.476	3252823.495	0.019311899
6303553	6303554	3257482.412	3257482.434	0.021964563
6325689	6325690	3268051.111	3268051.132	0.020566793
6329119	6329120	3269688.106	3269688.127	0.020627548
6331255	6331256	3270707.821	3270707.842	0.021014368
6332120	6332121	3271120.889	3271120.911	0.02209448
6335636	6335637	3272799.092	3272799.111	0.018658336
6355735	6355736	3282391.772	3282391.795	0.022921956
6358606	6358607	3283761.934	3283761.949	0.014440271
6363570	6363571	3286130.618	3286130.627	0.009302712
6365951	6365952	3287267.024	3287267.045	0.020733487
6366377	6366378	3287470.041	3287470.064	0.022298588
6370592	6370593	3289481.412	3289481.429	0.016595412
6380725	6380726	3294315.859	3294315.879	0.020380641
6388859	6388860	3298196.542	3298196.563	0.021214222
6400575	6400576	3303785.218	3303785.232	0.014648392
6403649	6403650	3305251.419	3305251.441	0.022018882
6418580	6418581	3312372.366	3312372.39	0.02409008
6423854	6423855	3314887.466	3314887.475	0.009033056
6461565	6461566	3332866.712	3332866.73	0.018231024
6485598	6485599	3344321.352	3344321.375	0.023393835
6488899	6488900	3345894.064	3345894.074	0.010622127
6488996	6488997	3345940.358	3345940.373	0.014662892
6499119	6499120	3350763.826	3350763.845	0.01922249
6502105	6502106	3352186.64	3352186.662	0.021120279
6503613	6503614	3352905.254	3352905.273	0.018954876
6523042	6523043	3362161.183	3362161.204	0.020743526
6554446	6554447	3377117.935	3377117.954	0.019475497
6555512	6555513	3377625.642	3377625.666	0.024338137
6556188	6556189	3377947.467	3377947.487	0.019821918
6560736	6560737	3380113.127	3380113.15	0.023623789
6571228	6571229	3385108.577	3385108.592	0.01509972
6571707	6571708	3385336.551	3385336.576	0.024930593
6586342	6586343	3392304.021	3392304.045	0.024293811
6587585	6587586	3392895.652	3392895.675	0.022371479
6602947	6602948	3400207.82	3400207.842	0.022172355
6607182	6607183	3402223.292	3402223.312	0.020167013

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
6608594	6608595	3402895.084	3402895.104	0.0200201
6618361	6618362	3407543.59	3407543.615	0.024466069
6620093	6620094	3408367.499	3408367.524	0.024583704
6621827	6621828	3409192.519	3409192.536	0.017298312
6623424	6623425	3409952.526	3409952.547	0.020813821
6647704	6647705	3421504.583	3421504.602	0.01945152
6648085	6648086	3421685.883	3421685.898	0.015238617
6650097	6650098	3422642.967	3422642.976	0.009826953
6653924	6653925	3424463.585	3424463.608	0.023016744
6665426	6665427	3429934.332	3429934.349	0.017132202
6678185	6678186	3436003.026	3436003.04	0.014417622
6685167	6685168	3439323.245	3439323.269	0.02483984
6703330	6703331	3447959.584	3447959.595	0.010638267
6715427	6715428	3453710.254	3453710.267	0.01275379
6722064	6722065	3456865.569	3456865.588	0.018599904
6732818	6732819	3461977.154	3461977.164	0.00966604
6746321	6746322	3468394.808	3468394.827	0.018927992
6751415	6751416	3470815.284	3470815.295	0.010699076
6757033	6757034	3473485.189	3473485.212	0.022150535
6758813	6758814	3474330.819	3474330.842	0.022744985
6760524	6760525	3475144.157	3475144.176	0.018447055
6763377	6763378	3476499.436	3476499.454	0.018414017
6788862	6788863	3488607.055	3488607.078	0.023297518
6833676	6833677	3509889.853	3509889.876	0.022858859
6834404	6834405	3510235.553	3510235.57	0.016502508
6853553	6853554	3519326.315	3519326.335	0.019988434
6880734	6880735	3532227.832	3532227.852	0.020530921
6882376	6882377	3533006.756	3533006.78	0.023811265
6894766	6894767	3538886.454	3538886.478	0.024272712
6900332	6900333	3541527.52	3541527.545	0.024332713
6905911	6905912	3544174.787	3544174.808	0.020970746
6915508	6915509	3548727.694	3548727.715	0.021202019
6917013	6917014	3549441.679	3549441.697	0.018182425
6919381	6919382	3550565.073	3550565.096	0.022646536
6940627	6940628	3560642.861	3560642.881	0.020195221
6943496	6943497	3562003.636	3562003.66	0.023185072
6945836	6945837	3563113.186	3563113.211	0.024671494
6947266	6947267	3563791.588	3563791.612	0.023911644
6947918	6947919	3564100.85	3564100.863	0.012410684
6957444	6957445	3568618.499	3568618.517	0.017897574
6974961	6974962	3576924.353	3576924.376	0.022878322
6989282	6989283	3583713.77	3583713.788	0.01826125
6990000	6990001	3584054.174	3584054.191	0.017263447
6990085	6990086	3584094.414	3584094.435	0.020471624
6996270	6996271	3587026.299	3587026.319	0.020405948
6999363	6999364	3588492.581	3588492.59	0.009572406
7002395	7002396	3589929.73	3589929.754	0.024077773
7013765	7013766	3595318.583	3595318.606	0.023393401
7021574	7021575	3599019.471	3599019.487	0.01608876
7023701	7023702	3600027.973	3600027.986	0.013033571
7031813	7031814	3603871.877	3603871.897	0.019927763

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
7035680	7035681	3605704.179	3605704.202	0.022872448
7041327	7041328	3608379.775	3608379.798	0.022699316
7042850	7042851	3609101.645	3609101.663	0.017985572
7051398	7051399	3613151.159	3613151.18	0.020954338
7055048	7055049	3614880.478	3614880.502	0.023589017
7057354	7057355	3615973.159	3615973.17	0.01070763
7060974	7060975	3617687.981	3617687.986	0.004940579
7065516	7065517	3619839.355	3619839.378	0.022222802
7071840	7071841	3622835.35	3622835.36	0.010334592
7075186	7075187	3624420.33	3624420.346	0.016828928
7082821	7082822	3628036.092	3628036.106	0.014027589
7087406	7087407	3630207.785	3630207.799	0.014107679
7087866	7087867	3630425.728	3630425.745	0.01752255
7089467	7089468	3631183.684	3631183.708	0.023971874
7092034	7092035	3632399.624	3632399.645	0.021370279
7092989	7092990	3632851.828	3632851.85	0.021706063
7098541	7098542	3635481.032	3635481.051	0.018974554
7105507	7105508	3638779.834	3638779.855	0.02162797
7106163	7106164	3639090.164	3639090.174	0.009882579
7107838	7107839	3639883.263	3639883.285	0.022169293
7108928	7108929	3640399.334	3640399.357	0.022948651
7130877	7130878	3650791.326	3650791.345	0.018788577
7133567	7133568	3652064.222	3652064.245	0.022766331
7135238	7135239	3652855.547	3652855.559	0.011652164
7139416	7139417	3654833.53	3654833.547	0.016638818
7155455	7155456	3662424.917	3662424.941	0.024543503
7159899	7159900	3664528.002	3664528.023	0.021012502
7160344	7160345	3664738.461	3664738.482	0.020217782
7167084	7167085	3667927.873	3667927.894	0.021386745
7167552	7167553	3668149.906	3668149.928	0.022091383
7176530	7176531	3672398.133	3672398.158	0.024337098
7177255	7177256	3672741.287	3672741.306	0.019458977
7183523	7183524	3675707.044	3675707.068	0.023734877
7185922	7185923	3676842.266	3676842.289	0.022772856
7200092	7200093	3683546.073	3683546.097	0.024595395
7200391	7200392	3683687.832	3683687.848	0.015804761
7202882	7202883	3684866.097	3684866.12	0.022833025
7210554	7210555	3688495.049	3688495.064	0.015345026
7211543	7211544	3688963.099	3688963.118	0.018983941
7221519	7221520	3693681.51	3693681.534	0.024719143
7225141	7225142	3695394.908	3695394.928	0.020027496
7237766	7237767	3701365.587	3701365.608	0.020951994
7239302	7239303	3702091.93	3702091.944	0.014489626
7245551	7245552	3705046.873	3705046.89	0.016403044
7248489	7248490	3706435.909	3706435.934	0.024986851
7250049	7250050	3707173.977	3707173.998	0.021149701
7260278	7260279	3712010.567	3712010.575	0.007542018
7263704	7263705	3713630.304	3713630.326	0.022749764
7265008	7265009	3714246.722	3714246.747	0.024638835
7268195	7268196	3715753.611	3715753.631	0.019998143
7268217	7268218	3715763.659	3715763.678	0.019370182

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
7279823	7279824	3721250.896	3721250.9	0.003379533
7282259	7282260	3722402.304	3722402.322	0.017729648
7282796	7282797	3722655.916	3722655.928	0.012002591
7291025	7291026	3726545.542	3726545.562	0.020331195
7291693	7291694	3726861.579	3726861.602	0.02348033
7301648	7301649	3731566.575	3731566.592	0.016448419
7304525	7304526	3732926.268	3732926.283	0.015718893
7316035	7316036	3738365.697	3738365.717	0.01973774
7320895	7320896	3740662.121	3740662.135	0.013691793
7330905	7330906	3745392.051	3745392.072	0.021215427
7352193	7352194	3755449.085	3755449.109	0.023504739
7354962	7354963	3756757.539	3756757.561	0.021832083
7367691	7367692	3762769.731	3762769.752	0.020964468
7370297	7370298	3764000.521	3764000.539	0.018405596
7373755	7373756	3765634.002	3765634.024	0.022407756
7377686	7377687	3767490.58	3767490.598	0.018304604
7395308	7395309	3775812.389	3775812.413	0.024427562
7396807	7396808	3776520.111	3776520.132	0.021431496
7400833	7400834	3778421.252	3778421.271	0.018216434
7415532	7415533	3785360.854	3785360.873	0.019009368
7420486	7420487	3787699.653	3787699.675	0.021762375
7427484	7427485	3791003.534	3791003.554	0.019818752
7440450	7440451	3797123.645	3797123.668	0.022945595
7447047	7447048	3800236.984	3800237.007	0.023001565
7458105	7458106	3805456.099	3805456.116	0.017467813
7471603	7471604	3811826.044	3811826.058	0.014397128
7473449	7473450	3812696.752	3812696.777	0.02496802
7476736	7476737	3814247.638	3814247.661	0.022586864
7483719	7483720	3817542.213	3817542.235	0.022549248
7498517	7498518	3824523.8	3824523.804	0.003876552
7519456	7519457	3834400.494	3834400.513	0.019438419
7519532	7519533	3834436.263	3834436.286	0.02242857
7528108	7528109	3838481.029	3838481.053	0.023761012
7530586	7530587	3839650.015	3839650.037	0.021333739
7531356	7531357	3840012.875	3840012.891	0.015476986
7534418	7534419	3841457.254	3841457.269	0.015261345
7538538	7538539	3843399.823	3843399.843	0.019895395
7541664	7541665	3844874.103	3844874.12	0.016881707
7553031	7553032	3850233.99	3850234.003	0.013406305
7568449	7568450	3857503.075	3857503.097	0.022037271
7568593	7568594	3857570.866	3857570.885	0.018788663
7572658	7572659	3859487.104	3859487.127	0.022502443
7579589	7579590	3862754.56	3862754.572	0.01284705
7582196	7582197	3863983.641	3863983.659	0.017945521
7583167	7583168	3864441.19	3864441.213	0.023188857
7598012	7598013	3871438.43	3871438.451	0.021497468
7602846	7602847	3873716.441	3873716.462	0.020750093
7631985	7631986	3887447.7	3887447.719	0.019565253
7639208	7639209	3890851.142	3890851.151	0.008742726
7640288	7640289	3891359.602	3891359.619	0.017156709
7647679	7647680	3894841.823	3894841.84	0.0172761

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
7647726	7647727	3894863.853	3894863.875	0.022112093
7648040	7648041	3895012.055	3895012.074	0.018808334
7660113	7660114	3900699.008	3900699.029	0.020963375
7681660	7681661	3910847.919	3910847.935	0.015683483
7688580	7688581	3914106.554	3914106.575	0.02112608
7693614	7693615	3916477.369	3916477.389	0.020539021
7701705	7701706	3920287.134	3920287.141	0.006769232
7706906	7706907	3922735.849	3922735.871	0.022558386
7714290	7714291	3926212.628	3926212.65	0.022128834
7715860	7715861	3926951.578	3926951.601	0.022499014
7734456	7734457	3935706.341	3935706.362	0.020047007
7746149	7746150	3941210.171	3941210.185	0.013973264
7751321	7751322	3943644.591	3943644.613	0.022057658
7768071	7768072	3951527.199	3951527.224	0.024421976
7796671	7796672	3964984.639	3964984.654	0.015075576
7802225	7802226	3967597.378	3967597.403	0.02409456
7805800	7805801	3969279.444	3969279.462	0.017817318
7816663	7816664	3974389.37	3974389.379	0.009617709
7822054	7822055	3976925.111	3976925.126	0.015294482
7826634	7826635	3979079.66	3979079.677	0.016309945
7830860	7830861	3981067.05	3981067.065	0.015240452
7834247	7834248	3982660.149	3982660.164	0.014892892
7839130	7839131	3984956.612	3984956.636	0.023919179
7843709	7843710	3987110.276	3987110.293	0.016558009
7845837	7845838	3988110.729	3988110.753	0.02401064
7856671	7856672	3993205.22	3993205.238	0.018359824
7858627	7858628	3994125.366	3994125.386	0.019469346
7860530	7860531	3995019.851	3995019.873	0.022005592
7862510	7862511	3995950.886	3995950.897	0.0111098
7862759	7862760	3996067.897	3996067.92	0.022730006
7869819	7869820	3999387.532	3999387.549	0.017445978
7870917	7870918	3999903.731	3999903.75	0.019366135
7883060	7883061	4005612.571	4005612.595	0.023806701
7895010	7895011	4011229.927	4011229.947	0.02033382
7899343	7899344	4013266.755	4013266.78	0.02483235
7903338	7903339	4015144.771	4015144.791	0.020579242
7909123	7909124	4017863.429	4017863.448	0.019246113
7916568	7916569	4021362.667	4021362.686	0.018807152
7916820	7916821	4021481.067	4021481.091	0.02379992
7921840	7921841	4023840.229	4023840.249	0.019363247
7926260	7926261	4025917.428	4025917.448	0.019244086
7941627	7941628	4033138.457	4033138.464	0.006642374
7958206	7958207	4040927.982	4040928.005	0.023339688
7970596	7970597	4046748.467	4046748.491	0.024507478
7976153	7976154	4049358.797	4049358.818	0.020970161
7976622	7976623	4049578.938	4049578.952	0.014302413
7978168	7978169	4050305.257	4050305.279	0.022412403
7980196	7980197	4051257.623	4051257.641	0.018141037
7995296	7995297	4058350.157	4058350.177	0.019956485
7995611	7995612	4058497.827	4058497.85	0.023742746
8001205	8001206	4061125.332	4061125.347	0.014195779

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
8001332	8001333	4061184.724	4061184.745	0.02111014
8012790	8012791	4066565.572	4066565.595	0.023266524
8014320	8014321	4067283.801	4067283.822	0.020856578
8016104	8016105	4068121.971	4068121.98	0.009254236
8017867	8017868	4068949.61	4068949.619	0.009029758
8027524	8027525	4073483.555	4073483.573	0.018695746
8058709	8058710	4088123.539	4088123.556	0.017193109
8060871	8060872	4089138.798	4089138.817	0.018375064
8063976	8063977	4090596.188	4090596.2	0.012045187
8074699	8074700	4095628.981	4095629.003	0.022004938
8095801	8095802	4105532.121	4105532.143	0.021360627
8127737	8127738	4120515.796	4120515.815	0.019261045
8128215	8128216	4120739.864	4120739.884	0.020069665
8139967	8139968	4126252.735	4126252.76	0.024535864
8142663	8142664	4127517.217	4127517.236	0.018972309
8146017	8146018	4129090.549	4129090.574	0.024690148
8147097	8147098	4129597.295	4129597.304	0.008363798
8149375	8149376	4130665.594	4130665.608	0.013631251
8158442	8158443	4134918.189	4134918.207	0.018053317
8158878	8158879	4135122.569	4135122.583	0.014484628
8160319	8160320	4135798.351	4135798.375	0.024235101
8194628	8194629	4151886.585	4151886.605	0.019847179
8196117	8196118	4152584.709	4152584.732	0.022900733
8204118	8204119	4156335.712	4156335.727	0.014570243
8209840	8209841	4159018.147	4159018.165	0.017956372
8224976	8224977	4166113.649	4166113.668	0.019178981
8225191	8225192	4166214.135	4166214.158	0.022193379
8242771	8242772	4174453.979	4174453.999	0.020118372
8255069	8255070	4180217.338	4180217.354	0.015799428
8261945	8261946	4183439.277	4183439.298	0.021041706
8265495	8265496	4185102.988	4185103.012	0.023295186
8272512	8272513	4188390.915	4188390.935	0.019966977
8274569	8274570	4189354.631	4189354.656	0.02435619
8275995	8275996	4190022.868	4190022.876	0.008756188
8286626	8286627	4195003.399	4195003.423	0.024246676
8287487	8287488	4195407.031	4195407.042	0.01100526
8300974	8300975	4201724.938	4201724.963	0.024416988
8301595	8301596	4202016.016	4202016.039	0.023612086
8313103	8313104	4207406.407	4207406.422	0.01572845
8316335	8316336	4208919.77	4208919.788	0.018368444
8318572	8318573	4209968.127	4209968.15	0.022753716
8322766	8322767	4211932.236	4211932.254	0.018535815
8328707	8328708	4214714.816	4214714.84	0.02464402
8332836	8332837	4216648.407	4216648.42	0.013320415
8333626	8333627	4217018.281	4217018.298	0.017322533
8335148	8335149	4217730.725	4217730.741	0.016776381
8337162	8337163	4218674.047	4218674.056	0.008337815
8337903	8337904	4219021.49	4219021.513	0.022521873
8347435	8347436	4223484.717	4223484.731	0.014174541
8369992	8369993	4234046.211	4234046.231	0.020023533
8371104	8371105	4234566.996	4234567.019	0.022477114

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
8372769	8372770	4235346.368	4235346.383	0.014918253
8375956	8375957	4236838.107	4236838.127	0.019724923
8384416	8384417	4240798.619	4240798.634	0.015209832
8392585	8392586	4244622.613	4244622.626	0.012478821
8404716	8404717	4250300.558	4250300.573	0.015442234
8404889	8404890	4250381.573	4250381.587	0.013941013
8406769	8406770	4251261.314	4251261.335	0.021315376
8408105	8408106	4251886.529	4251886.553	0.024459777
8410612	8410613	4253059.983	4253060.008	0.024842445
8426260	8426261	4260383.334	4260383.355	0.021234793
8428492	8428493	4261427.255	4261427.275	0.020208457
8441861	8441862	4267683.163	4267683.176	0.013083549
8446166	8446167	4269697.164	4269697.188	0.024568632
8449272	8449273	4271150.191	4271150.214	0.022662845
8449825	8449826	4271409.42	4271409.431	0.011591656
8455092	8455093	4273873.232	4273873.255	0.022770719
8469111	8469112	4280431.536	4280431.561	0.024741188
8473762	8473763	4282607.08	4282607.09	0.010169744
8480576	8480577	4285794.623	4285794.645	0.022355545
8483024	8483025	4286939.67	4286939.694	0.02372484
8492399	8492400	4291324.539	4291324.557	0.018050987
8495541	8495542	4292794.06	4292794.069	0.008733322
8496796	8496797	4293380.897	4293380.908	0.011803466
8507089	8507090	4298194.63	4298194.651	0.020990713
8518925	8518926	4303729.609	4303729.627	0.01807215
8520683	8520684	4304551.095	4304551.118	0.022917949
8526785	8526786	4307404.372	4307404.388	0.016731459
8530513	8530514	4309147.711	4309147.726	0.015312275
8546035	8546036	4316404.623	4316404.643	0.020209508
8546950	8546951	4316832.467	4316832.469	0.002323279
8554299	8554300	4320267.817	4320267.838	0.021544002
8559611	8559612	4322751.196	4322751.22	0.023973501
8562633	8562634	4324163.766	4324163.788	0.022042082
8564512	8564513	4325041.875	4325041.896	0.021031222
8568936	8568937	4327109.996	4327110.012	0.016312424
8601600	8601601	4342375.283	4342375.299	0.016564817
8614041	8614042	4348188.407	4348188.426	0.018869973
8614267	8614268	4348294.196	4348294.213	0.017253844
8617233	8617234	4349680.16	4349680.172	0.0123565
8621596	8621597	4351718.429	4351718.447	0.01807678
8625915	8625916	4353736.295	4353736.315	0.019453915
8631973	8631974	4356566.147	4356566.156	0.008233797
8634490	8634491	4357742.316	4357742.338	0.022310822
8637189	8637190	4359003.113	4359003.135	0.021940872
8640675	8640676	4360631.679	4360631.7	0.020933183
8641542	8641543	4361036.665	4361036.689	0.023882423
8654060	8654061	4366884.209	4366884.227	0.017161451
8669815	8669816	4374242.703	4374242.719	0.015949115
8673743	8673744	4376077.459	4376077.48	0.021207921
8680382	8680383	4379177.767	4379177.786	0.01893199
8683460	8683461	4380614.775	4380614.795	0.020471023

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
8684100	8684101	4380913.8	4380913.821	0.020790758
8694755	8694756	4385889.105	4385889.118	0.012976457
8699617	8699618	4388159.672	4388159.68	0.008322325
8700094	8700095	4388382.206	4388382.22	0.014298337
8704856	8704857	4390605.678	4390605.7	0.021916287
8708085	8708086	4392113.384	4392113.408	0.024412991
8715372	8715373	4395515.534	4395515.558	0.02363408
8715538	8715539	4395593.215	4395593.233	0.017198689
8720845	8720846	4398070.272	4398070.295	0.023649856
8725850	8725851	4400407.255	4400407.28	0.024435345
8731014	8731015	4402817.776	4402817.784	0.008822444
8746536	8746537	4410063.016	4410063.036	0.020119691
8748793	8748794	4411116.578	4411116.596	0.017670356
8751778	8751779	4412509.855	4412509.876	0.021345907
8752659	8752660	4412920.935	4412920.957	0.022231794
8757134	8757135	4415009.366	4415009.382	0.016030164
8763423	8763424	4417944.37	4417944.39	0.020195507
8768277	8768278	4420209.643	4420209.663	0.019850781
8772793	8772794	4422317.472	4422317.494	0.021721228
8775736	8775737	4423690.721	4423690.745	0.023743276
8790885	8790886	4430759.419	4430759.439	0.019587507
8792377	8792378	4431455.765	4431455.789	0.024400572
8797281	8797282	4433743.744	4433743.763	0.01881567
8804711	8804712	4437210.444	4437210.466	0.02205279
8812535	8812536	4440860.235	4440860.249	0.014236274
8821498	8821499	4445041.601	4445041.613	0.01164367
8830310	8830311	4449151.9	4449151.924	0.023980531
8844324	8844325	4455688.339	4455688.36	0.021151415
8854013	8854014	4460207.303	4460207.322	0.019572783
8865761	8865762	4465685.761	4465685.782	0.021391307
8867226	8867227	4466368.649	4466368.673	0.024481735
8870798	8870799	4468034.188	4468034.207	0.018941709
8871017	8871018	4468136.523	4468136.538	0.01417323
8886300	8886301	4475262.644	4475262.662	0.017139218
8886683	8886684	4475441.112	4475441.134	0.022406819
8897015	8897016	4480258.261	4480258.284	0.02288405
8906485	8906486	4484673.053	4484673.077	0.023427618
8917334	8917335	4489730.33	4489730.353	0.023480101
8931032	8931033	4496114.757	4496114.77	0.012702831
8936114	8936115	4498483.34	4498483.351	0.010755615
8944425	8944426	4502356.957	4502356.974	0.017100918
8946811	8946812	4503468.507	4503468.525	0.018174217
8948733	8948734	4504364.564	4504364.587	0.023088393
8958314	8958315	4508829.158	4508829.179	0.021112488
8966181	8966182	4512494.924	4512494.943	0.019178255
8986274	8986275	4521856.807	4521856.828	0.020582119
8993192	8993193	4525079.66	4525079.682	0.021967802
8994730	8994731	4525796.097	4525796.117	0.02054465
8998201	8998202	4527413.326	4527413.35	0.024650982
8999852	8999853	4528182.287	4528182.308	0.020508304
9004523	9004524	4530358.302	4530358.325	0.022847133

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
9012086	9012087	4533881.005	4533881.029	0.024543911
9031475	9031476	4542911.81	4542911.829	0.019354147
9047647	9047648	4550442.61	4550442.615	0.005170356
9057216	9057217	4554898.846	4554898.87	0.024092423
9059857	9059858	4556128.643	4556128.652	0.009009335
9063409	9063410	4557782.212	4557782.228	0.01607387
9086817	9086818	4568680.138	4568680.153	0.014993067
9095438	9095439	4572693.506	4572693.526	0.020066419
9113272	9113273	4580994.591	4580994.614	0.022577359
9119937	9119938	4584096.97	4584096.994	0.023448913
9123467	9123468	4585739.646	4585739.669	0.023079741
9129428	9129429	4588513.758	4588513.778	0.020200375
9138132	9138133	4592564.16	4592564.183	0.023363414
9140415	9140416	4593626.642	4593626.657	0.014972719
9156649	9156650	4601180.938	4601180.959	0.020820592
9161908	9161909	4603627.942	4603627.966	0.02436457
9165435	9165436	4605268.743	4605268.752	0.008516129
9179532	9179533	4611826.974	4611826.993	0.018920882
9181013	9181014	4612515.801	4612515.825	0.02451861
9201029	9201030	4621826.462	4621826.477	0.014729566
9203662	9203663	4623051.106	4623051.115	0.009458249
9212487	9212488	4627155.799	4627155.82	0.020859282
9216191	9216192	4628878.535	4628878.558	0.023397378
9244056	9244057	4641836.82	4641836.845	0.024833771
9254359	9254360	4646627.015	4646627.038	0.02287218
9265193	9265194	4651664.184	4651664.204	0.019597035
9276043	9276044	4656708.104	4656708.124	0.020675199
9279755	9279756	4658433.941	4658433.959	0.018022725
9283276	9283277	4660070.599	4660070.621	0.022530287
9286428	9286429	4661535.9	4661535.915	0.015227297
9298601	9298602	4667193.762	4667193.785	0.022718729
9301538	9301539	4668559.184	4668559.204	0.020271479
9315391	9315392	4674997.484	4674997.505	0.021055675
9317281	9317282	4675875.634	4675875.653	0.018690139
9322125	9322126	4678126.806	4678126.83	0.024369461
9322603	9322604	4678349.246	4678349.267	0.021137543
9336560	9336561	4684834.648	4684834.659	0.01058231
9347690	9347691	4690006.302	4690006.325	0.022823895
9348903	9348904	4690569.985	4690569.997	0.012717953
9373408	9373409	4701954.258	4701954.275	0.017656774
9390574	9390575	4709928.009	4709928.03	0.020567641
9391193	9391194	4710215.533	4710215.549	0.01572003
9407641	9407642	4717854.89	4717854.912	0.02232093
9409636	9409637	4718781.253	4718781.272	0.018565417
9411039	9411040	4719432.92	4719432.945	0.024563925
9412285	9412286	4720011.76	4720011.779	0.018560357
9416903	9416904	4722156.185	4722156.202	0.017330273
9425868	9425869	4726319.341	4726319.351	0.010363786
9442023	9442024	4733820.639	4733820.663	0.023217821
9446146	9446147	4735735.065	4735735.081	0.015830969
9448140	9448141	4736660.771	4736660.791	0.019402729

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
9454076	9454077	4739416.989	4739417.001	0.011259471
9459212	9459213	4741801.267	4741801.277	0.009121919
9463522	9463523	4743801.821	4743801.832	0.011712972
9463576	9463577	4743827.363	4743827.376	0.013130198
9464808	9464809	4744399.057	4744399.076	0.018994748
9469220	9469221	4746447.184	4746447.207	0.022913427
9472460	9472461	4747951.319	4747951.344	0.024078133
9476329	9476330	4749747.154	4749747.17	0.015377986
9491402	9491403	4756743.602	4756743.626	0.023813348
9492140	9492141	4757086.248	4757086.269	0.021150745
9494093	9494094	4757992.752	4757992.766	0.013887077
9503246	9503247	4762240.451	4762240.463	0.012256538
9506903	9506904	4763938.136	4763938.157	0.02023812
9534186	9534187	4776598.594	4776598.615	0.020234383
9541448	9541449	4779967.906	4779967.929	0.023680829
9552595	9552596	4785139.652	4785139.676	0.02411765
9591518	9591519	4803195.026	4803195.046	0.02014554
9594547	9594548	4804599.839	4804599.864	0.02461839
9616166	9616167	4814625.822	4814625.838	0.016472019
9624029	9624030	4818272.248	4818272.272	0.024551916
9631221	9631222	4821607.135	4821607.158	0.023155129
9639712	9639713	4825544.17	4825544.187	0.016532392
9639849	9639850	4825607.635	4825607.659	0.023963942
9643681	9643682	4827384.231	4827384.25	0.019948255
9644877	9644878	4827938.801	4827938.814	0.013425502
9671865	9671866	4840450.048	4840450.063	0.015293999
9675303	9675304	4842043.866	4842043.87	0.003336424
9678549	9678550	4843548.305	4843548.316	0.011731911
9681709	9681710	4845013.185	4845013.21	0.024927772
9705573	9705574	4856073.57	4856073.588	0.018478929
9730782	9730783	4867755.086	4867755.093	0.007702009
9733798	9733799	4869152.721	4869152.745	0.024511813
9739584	9739585	4871833.403	4871833.416	0.013064409
9746364	9746365	4874974.69	4874974.708	0.01759675
9755069	9755070	4879007.647	4879007.666	0.019182886
9757774	9757775	4880260.7	4880260.724	0.024282416
9761233	9761234	4881863.222	4881863.242	0.020235244
9773490	9773491	4887540.881	4887540.9	0.019233854
9774979	9774980	4888230.746	4888230.755	0.009717599
9783829	9783830	4892330.105	4892330.113	0.007845907
9793085	9793086	4896617.099	4896617.118	0.018674813
9802396	9802397	4900929.13	4900929.152	0.022140327
9805216	9805217	4902235.142	4902235.16	0.018499848
9809767	9809768	4904343.2	4904343.212	0.012072248
9837538	9837539	4917202.1	4917202.123	0.022992682
9839460	9839461	4918092.37	4918092.389	0.01872108
9850973	9850974	4923422.63	4923422.653	0.022687507
9857599	9857600	4926490.1	4926490.103	0.003028281
9859464	9859465	4927353.414	4927353.421	0.00716954
9862913	9862914	4928949.753	4928949.777	0.023320946
9869843	9869844	4932158.043	4932158.062	0.018676634

Continued on next page

Root 1	Root 2	Value of Root 1	Value of Root 2	Distance between Roots
9873703	9873704	4933944.885	4933944.892	0.007786662
9875828	9875829	4934928.667	4934928.682	0.014309546
9877446	9877447	4935677.685	4935677.697	0.01147445
9877875	9877876	4935875.921	4935875.941	0.019721203
9881265	9881266	4937445.325	4937445.339	0.014224919
9882748	9882749	4938131.613	4938131.632	0.019237931
9889657	9889658	4941329.343	4941329.362	0.019448333
9923097	9923098	4956805.034	4956805.057	0.02309115
9923408	9923409	4956948.992	4956949.014	0.021865869
9926825	9926826	4958530.097	4958530.104	0.007241423
9933096	9933097	4961431.948	4961431.971	0.022405215
9935706	9935707	4962639.551	4962639.564	0.013172454
9942383	9942384	4965728.915	4965728.931	0.016581122
9958896	9958897	4973368.703	4973368.724	0.021245628
9976135	9976136	4981342.898	4981342.922	0.024143394
9982957	9982958	4984498.602	4984498.619	0.017025047
9995077	9995078	4990104.532	4990104.551	0.018941825
9999287	9999288	4992051.881	4992051.905	0.024547733
10001041	10001042	4992862.648	4992862.667	0.019773009
10002395	10002396	4993488.883	4993488.901	0.018636165
10009232	10009233	4996650.947	4996650.969	0.021896206
10010921	10010922	4997431.774	4997431.797	0.022975322

Here are some interesting facts about the roots found between $0 < t < 5000000$:

• According to my program, the total number of roots found inside the first five million t-values is 10016474.

• From the 10016474 total roots calculated, only 1187 pairs of roots were found within $|\rho_{n+1} - \rho_n| < 0.025$.

• This means that 2374 out of 10016474 roots, or roughly 0.000237%, were within my required distance for Lehmer's phenomenon.

• The average distance between two roots found from the table above is 0.018773133.

• The two roots with the third shortest distance between two consecutive t-values occurred between $\rho_{5042995}$ and $\rho_{5042996}$. The distance between these two roots is 0.002970091.

• The two roots with the second shortest distance between two consecutive t-values occurred between $\rho_{1115578}$ and $\rho_{1115579}$. The distance between these two roots is 0.002958679.

• The two roots with the shortest distance between two consecutive t-values occurred between $\rho_{8546950}$ and $\rho_{8546951}$. I have confirmed all of my findings with Mathematica and has posted a graphical representation on the following page.

Figure 4.1: Lehmer's phenomenon where $2650984 \leq t \leq 2650985$

Figure 4.2: Lehmer's phenomenon where $663315 \leq t \leq 663319$

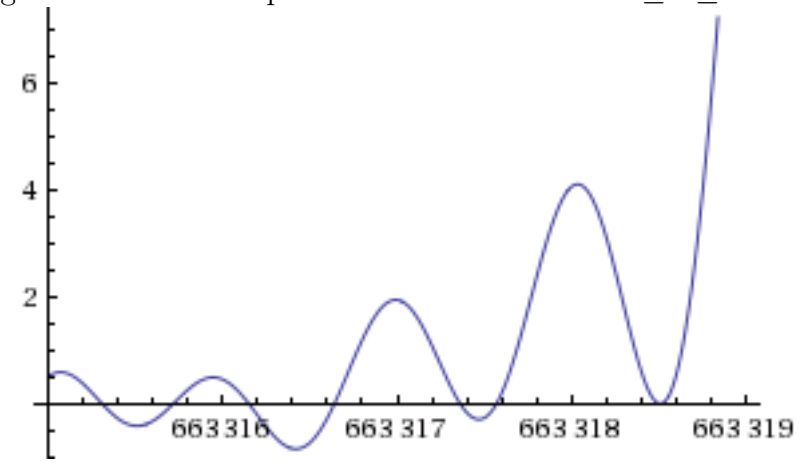
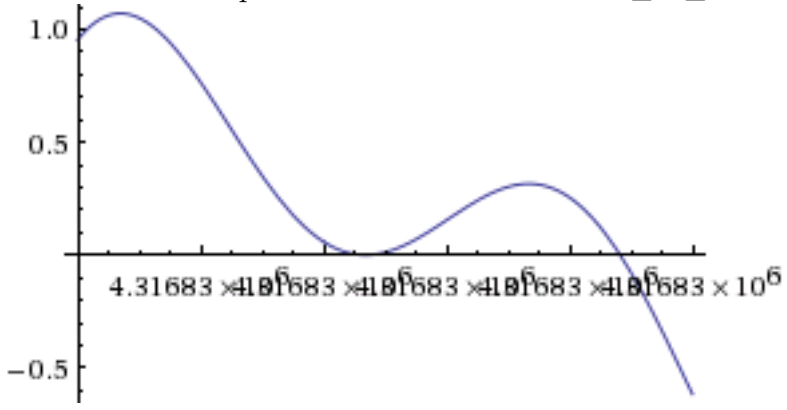


Figure 4.3: Lehmer's phenomenon where $4316832 \leq t \leq 4316833$



Chapter 5

Conclusion

My project combined applications of the Cauchy−Schlömilch transformation, Abel−Plana formula, and Riemann−Siegel formula in order to calculate values for $\zeta(s)$. It is possible to calculate $\zeta(s)$ through a variety of other equations, the most common of these are through the gamma function, Euler Maclaurin formula, and Laurent series. Through the use of Mathematica, one can apply hundreds of techniques to calculate specific values for $\zeta(s)$. Java is not the preferred language of choice for calculating arbitrary−precision arithmetic. Python is a better choice and has built in functions that will automatically extend precision. I was fortunate to find Apfloat, as it was extremely helpful in testing my programs.

The most interesting result that I found was obtained through analyzing Apéry's constant. As described inside [10], Leonard Euler developed a useful method for using geometric series in order to calculate $\zeta(s)$ for even values of s. The same method can be applied to $\zeta(3)$, which will form sequences for OEIS A179125, A007916, and A153147. One interesting result is

$$\zeta(3)-\zeta(6)=\frac{8}{63}+\frac{27}{728}+\frac{125}{15624}+\frac{216}{46655}+\frac{343}{117648}+\frac{1000}{999999}+\frac{1331}{1771560}+\frac{1728}{2985983}+\cdots$$

where, by an infinite summation formula, the numerator is A153147 and the denominator is $(A153147)^2-1$. Another sequence forms a relationship with π,

$$\begin{aligned}\sum_{n=1}^{\infty}\frac{1}{3n^2+3n-5}&=1+\frac{1}{13}+\frac{1}{31}+\frac{1}{55}+\frac{1}{85}+\frac{1}{121}+\frac{1}{163}+\cdots\\&=\frac{1}{5}+\frac{\pi\tan\left(\frac{1}{2}\sqrt{\frac{23}{3}}\,\pi\right)}{\sqrt{69}}\end{aligned}$$

The other series formed through manipulating $\zeta(3)$ don't distribute evenly into π. Perhaps it is possible to combine all of these series in order to find a relationship strictly with π.

Lehmer's phenomenon is an interesting concept that can, in theory, be extended to an infinite amount of zeroes for the zeta function. I was only able to scan through the first five million t-values due to time constraints and hardware constraints. It would certainly be interesting to analyze Lehmer's phenomenon for the first billion or trillion zeroes of the zeta function. Odlyzko notes that his own calculations become problematic near the 10^{20}th zero using twenty eight digits of decimal precision. With a proper implementation, using an arbitrary precision library, it would be possible to calculate zeta values at much higher ranges. These calculations would allow one to verify the Riemann hypothesis at remarkably large t-values.

Bibliography

[1] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Number 55. Courier Corporation, 1964.

[2] Tewodros Amdeberhan, ML Glasser, MC Jones, VH Moll, R Posey, and D Varela. The cauchy-schlomilch transformation. *arXiv preprint arXiv:1004.2445*, 2010.

[3] David Anderson. A review of the abel-plana formula. *Columbia University*, 2001.

[4] JG Andrews. All the zeros of the dirichlet eta function in the critical strip are on the critical line. Technical report, 2012.

[5] Tom M Apostol. *Modular functions and Dirichlet series in number theory*, volume 41. Springer Science & Business Media, 2012.

[6] Tom M Apostol. *Introduction to analytic number theory*. Springer Science & Business Media, 2013.

[7] James Ward Brown, Ruel Vance Churchill, and Martin Lapidus. *Complex variables and applications*, volume 7. McGraw-Hill New York, 1996.

[8] PL Butzer, PJSG Ferreira, G Schmeisser, and RL Stens. The summation formulae of euler–maclaurin, abel–plana, poisson, and their interconnections with the approximate sampling formula of signal analysis. *Results in Mathematics*, 59(3-4):359–400, 2011.

[9] Ji Chungang and Yonggao Chen. Euler's formula for zeta (2k), proved by induction on k. *Mathematics Magazine*, 73(2):154–155, 2000.

[10] William Dunham. *Euler: The master of us all*. Number 22. MAA, 1999.

[11] Roman J Dwilewicz and Ján Minác. Values of the riemann zeta function at integers. In *Materials matemàtics*, pages 0001–26, 2009.

[12] Harold M Edwards. *Riemann's zeta function*, volume 58. Courier Corporation, 2001.

[13] James F Epperson. *An introduction to numerical methods and analysis*. John Wiley & Sons, 2013.

[14] Nolan Essigmann. Montgomerys pair correlation conjecture and odlyzkos zeta function root finder. 2015.

[15] Graham Everest, Christian Röttger, and Tom Ward. The continuing story of zeta. *The Mathematical Intelligencer*, 31(3):13–17, 2009.

[16] Julian Havil and J Gamma. Exploring eulers constant, 2003.

[17] Semiclassical (http://math.stackexchange.com/users/137524/semiclassical). Proof that $\zeta'(-2n) = (-1)^n \frac{(2n)!}{2(2\pi)^{2n}}\zeta(2n+1)$. Mathematics Stack Exchange. URL:http://math.stackexchange.com/q/909824 (version: 2014-08-26).

[18] Raymond Manzoni (http://math.stackexchange.com/users/21783/raymond manzoni). Riemann zeta function manipulation. Mathematics Stack Exchange. URL:http://math.stackexchange.com/q/222588 (version: 2012-10-28).

[19] Aleksandar Ivić. On some reasons for doubting the riemann hypothesis. *arXiv preprint math/0311162*, 2003.

[20] Aleksandar Ivic. *The Riemann zeta-function: theory and applications*. Courier Corporation, 2003.

[21] Aleksandar Ivić. *The Theory of Hardy's Z-function*, volume 196. Cambridge University Press, 2012.

[22] William Kahan, Joseph D Darcy, Elect Eng, and High-Performance Network Computing. How javas floating-point hurts everyone everywhere. In *ACM 1998 workshop on java for high-performance network computing*, page 81. Stanford University, 1998.

[23] Maxim Korolev. Grams law and the argument of the riemann zeta function. *arXiv preprint arXiv:1106.0516*, 2012.

[24] Ernst Lindelöf. *Le Calcul des Résidus et ses Applications à la Théorie des Fonctions*. Gauthier-Villars, Paris, 1905. Reprinted by Éditions Jacques Gabay, Sceaux, 1989.

[25] Qiang Luo and Zhidan Wang. Numerical calculation of the riemann zeta function at odd-integer arguments: a direct formula method. *Mathematical Sciences*, 9(1):39–45, 2014.

[26] Michael S Milgram. Integral and series representations of riemann's zeta function and dirichlet's eta function and a medley of related results. *Journal of Mathematics*, 2013, 2013.

[27] mixedmath (http://math.stackexchange.com/users/9754/mixedmath). The multiplication formula for the hurwitz/generalized riemann zeta function. Mathematics Stack Exchange. URL:http://math.stackexchange.com/q/597891 (version: 2013-12-08).

[28] Cleve B Moler. *Numerical Computing with MATLAB: Revised Reprint*. Siam, 2008.

[29] Vladimir Mikhaĭ Mostepanenko. *The Casimir effect and its applications*.

[30] Yoichi Motohashi. *Analytic number theory*. Number 247. Cambridge University Press, 1997.

[31] Joubert Oosthuizen. The mellin transform. 2011.

[32] Glendon Ralph Pugh. *The Riemann-Siegel formula and large scale computations of the Riemann zeta function*. PhD thesis, University of British Columbia, 1998.

[33] Bernhard Riemann. Ueber die anzahl der primzahlen unter einer gegebenen grosse. *Ges. Math. Werke und Wissenschaftlicher Nachlaß*, 2:145–155, 1859.

[34] Walter Rudin. *Real and complex analysis (3rd)*. New York: McGraw-Hill Inc, 1986.

[35] Aram A Saharian. The generalized abel-plana formula with applications to bessel functions and casimir effect. *arXiv preprint arXiv:0708.1187*, 2007.

[36] Jerry Shurman. A series representation of the cotangent, February 2012.

[37] Jerry Shurman. Zeta at negative odd integers a la euler, February 2012.

[38] H. M. Srivastava and Junesang Choi. *Series Associated with the Zeta and Related Functions*. Kluwer Academic Publishers, Dordrecht, 2001.

[39] Ken Takusagawa. Tabulating values of the riemann-siegel z function along the critical line. 2002.

[40] uranix (http://math.stackexchange.com/users/200750/uranix). Looking for finite difference approximations past the fourth derivative. Mathematics Stack Exchange. URL:http://math.stackexchange.com/q/1381146 (version: 2015-08-01).