A New Formulation for the Riemann Zeta Function

Vilas Winstein, Ghaith Hiary

The Ohio State University

Introduction

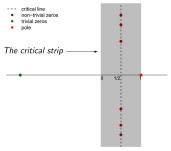
The Riemann zeta function is an important function in Number Theory, and is central to the Riemann hypothesis, which is considered one of the greatest unsolved problems in mathematics. We are implementing a new method for computing the Riemann zeta function with very high accuracy.

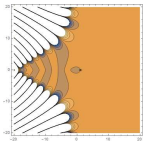
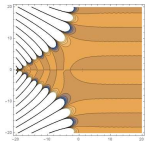

First studied by Riemann and Euler (pictured above), the Riemann zeta function is defined by the analytic continuation of the series below, and is also related to the primes as below:

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

$$\zeta(s) = \prod_{p \in P} \frac{1}{1 - \frac{1}{p^s}}$$

Where P is the set of prime numbers. As such, the zeta function is relevant to number theory. One particular example is the Riemann hypothesis which concerns the location of the zeroes of the zeta function. It states that all nontrivial zeroes of ζ have a real part of $\frac{1}{2}$. In order to observe zeroes of ζ, a way to numerically evaluate the function is necessary, as it can be used to approximate zeroes via the intermediate value theorem.



The critical strip | $\mathrm{Re}[\zeta(s)]$ | $\mathrm{Im}[\zeta(s)]$

These zeroes lie on the so-called "critical strip" which consists of points with real part between 0 and 1 (pictured above). However, the formulation of ζ shown above only holds for numbers with real part greater than 1. An alternate formulation below works for the critical strip.

$$\zeta(s) = \frac{1}{1 - 2^{1-s}} \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n^s}, \qquad Re(s) > 0$$

But, since this formulation (as well as others) uses an infinite series (the one shown above), it cannot be numerically evaluated exactly for all input values.

Objectives

Current methods for computing the zeta function suffer from either low speed or limited accuracy. The method we are implementing does not suffer from limited accuracy, and is comparatively faster than current methods for large input values.

The main objective is to obtain a working program that will compute the Riemann zeta function according to the new formulation, and to optimize the program to run efficiently and effectively on a variety of computer systems.

In order to achieve this goal, many numerical methods must be implemented and tested, and a framework for working with and using the program must be constructed.

Future goals include generalizing this method to a class of functions called Dirichlet L-functions, and distributing the method online so that it is available for use by researchers in the field.

Methods

The program is primarily written in the C programming language, with some supplementary files written in the Wolfram language (Mathematica scripts) and the Python programming language. In C, the GNU MPFR (Multiple Precision Floating-point Reliably) library is used to achieve arbitrarily high precision results.

First, since the zeta function involves a complex variable and gives a complex result, a library for multiple precision complex numbers (numbers of the form $a + bi$ where $i^2 = -1$) was necessary. Since no such library was readily available, we created our own library of code including functions for basic complex operations (like addition, subtraction, multiplication, and division) and more advanced operations (like exponentiation).

Initially, we worked directly with an existing formulation of the zeta function (the Euler-Maclaurin formula) which is slow but does not suffer from limited accuracy. Modifications to this formulation give the new formulation which is faster for large input values.

The main modification of the existing formulation is the approximation of segments of the main infinite sum by a quicker-to-compute and closed-form function shown below

Methods (continued)

$$(*) \qquad \sum_{n=v_r}^{v+K_r-1} \frac{1}{n^s} \approx \frac{B_r(s,m)}{v_r^s}$$

$$B_r(s,m) \coloneqq \sum_{j=0}^{m} \frac{f_s^{(j)}(0)}{j!} \frac{g_{K_r}^{(j)}(-s/v_r)}{v_r^j}$$

$$f_S(z) \coloneqq \frac{e^{sz}}{(1+z)^s}, \qquad f(0) = 1$$

$$g_k(z) \coloneqq \sum_{k=0}^{K-1} e^{kz} = \frac{e^{Kz} - 1}{e^z - 1}, \qquad z \notin 2\pi i\mathbb{Z}$$

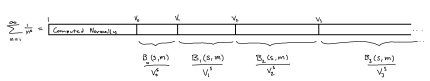
The derivatives of f and g used above in the formula for B_r involve mainly polynomials whose coefficients can be pre-computed and stored in a C header file for quick access. This is done using a Mathematica script that uses the number-theoretical properties of the coefficients of the involved polynomials.

The v_r in the formulas above are simply a sequence of integers that demarcate where to cut the main infinite sum, and K_r is simply the number of terms in the main sum block that are being approximated by $B_r(s,m)$. The m in this expression is simply the number of derivatives of f and g that are used in the computation of the function. This must be determined by the program in order to ensure that the approximation $(*)$ is good enough. This is done with a simple binary search through the list of possible values for m.

Other correction terms (from the original Riemann-Siegel formulation) must be added in, since even truncating the original sum without replacing any blocks by the B_r approximation leaves too much error. In full, the formula we are using boils down to this:

$$\zeta(s) = \sum_{n=1}^{v_0-1} \frac{1}{n^s} + \sum_{r=0}^{R} \frac{B_r(s,m)}{v_r^s} + \frac{M^{-s}}{2} + \frac{M^{1-s}}{s-1} + \varepsilon_1 + \varepsilon_2$$

Where M is the truncation point of the main sum (before approximation) and the ε_1 and ε_2 are error terms that can be controlled by taking M or m higher.



approximation of the main sum by closed-form funcions

Results

Currently the program is working correctly and gives correct values (meaning values of arbitrarily large accuracy) for $\zeta(s)$ whenever the imaginary part of s is not too close to a multiple of 2π.

The reason for this inaccuracy when the imaginary part of s is too close to a multiple of 2π is because of the g_k function used in approximating ζ. This function includes a division by $e^z - 1$, which, if z is a multiple of $2\pi i$, is zero (and even when z is sufficiently close to a multiple of $2\pi i$, this division by a very small number causes error to accumulate and give an inaccurate answer).

The program also has a user interface via the command line, so it can be worked into new programs and used for research. The command line output can be trimmed and tailored for specific use cases.

Conclusions

Work still needs to be done in order to make the program accurate for values of s with imaginary part close to a multiple of 2π, and once this is complete the program will need to be optimized for use on various computer systems.

Once this optimization is complete, the program will be available for use by researchers.

In the future, generalizations of the method can be used to compute Dirichlet L-functions.

References

[1] G. A. Hiary, *An alternative to Riemann-Siegel type formulas*, ArXiv e-prints (2014).