MATHEMATICS

Faster Convergence for the Smaller Values of the Riemann Zeta Function [duplicate]

Asked 4 years, 4 months ago Active 4 years, 4 months ago Viewed 514 times

3

This question already has answers here:

How to evaluate Riemann Zeta function (2 answers)

Closed 3 years ago.

I have a C++ program that uses the equation

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

to calculate the Riemann zeta function.

This equation converges fast for larger values, like 183, but converges much slower for smaller values, like 2. For example, calculating the value of $\zeta(2)$ took an hour to be accurate to 5 digits, but one second for $\zeta(183)$ to be accurate to 100 digits.

Are there any equations for calculating the Riemann zeta function that are faster **for calculating smaller values**?

Because I am coding in C++, I cannot use $\int$ (without implementing external libraries, which is not really an option here).

convergence-divergence riemann-zeta rate-of-convergence

Share Cite Follow

edited Oct 28 '16 at 1:27

asked Oct 28 '16 at 0:19

esote

1,083 1 10 23

The product formula might be faster:

$$\prod_{p\text{ prime}} \left(1-\frac{1}{p^s}\right)^{-1}$$

Of course, the speed of the convergence depends on s - if s is close to 1, you won't get fast convergence. – Thomas Andrews Oct 28 '16 at 0:30

Just to clarify - and it might well be irrelevant - but what do you mean by "without implementing external libraries"? Do you mean you can't #include? – peter a g Oct 28 '16 at 0:43

3 The faster formula that is easy to prove is $\eta(s) = \sum_{n=0}^\infty \frac{1}{2^{n+1}} \sum_{k=0}^n (-1)^k \binom{n}{k} \frac{1}{(k+1)^s}$ And then you can prove a similar formula for $\zeta(s)$ – reuns Oct 28 '16 at 0:47

@peterag I just want avoid using libraries like the Boost C++ math library. Built-in includes (like cmath and complex) are fine. – esote Oct 28 '16 at 0:48

1 After an hour it was only accurate to 5 digits? What are you running, Windows 95? I definitely remember programming my own that could do several digits in seconds. Maybe post yours on C++ stack exchange? – Kaynex Oct 28 '16 at 1:09

2 Answers

Active Oldest Votes

5

There is a derivation of $(1-2^{1-s})\zeta(s) = \eta(s) = \sum_{n=0}^\infty \frac{1}{2^{n+1}} \sum_{k=0}^n \binom{n}{k} \frac{(-1)^k}{(k+1)^s}$ converging in $\mathcal{O}(2^{-N})$ for every $s \in \mathbb{C}$

- $$\sum_{k=0}^\infty x^k = \frac{1}{1-x} = \frac{1/2}{1-\frac{1+x}{2}} = \sum_{n=0}^\infty \frac{(1+x)^k}{2^{n+1}}$$
  $$= \sum_{n=0}^\infty \frac{1}{2^{n+1}} \sum_{k=0}^n \binom{n}{k} x^k = \sum_{k=0}^\infty x^k \sum_{n=k}^\infty \binom{n}{k} \frac{1}{2^{n+1}}$$
  $$\text{identifying the coefficients : } \sum_{n=k}^\infty \frac{1}{2^{n+1}} \binom{n}{k} = 1$$
- For $\text{Re}(s) > 1$ where everything converges absolutely :

$$\sum_{n=0}^\infty \frac{1}{2^{n+1}} \sum_{k=0}^n \binom{n}{k} \frac{(-1)^k}{(k+1)^s} = \sum_{k=0}^\infty \frac{(-1)^k}{(k+1)^s} \sum_{n=k}^\infty \frac{1}{2^{n+1}} \binom{n}{k} = \sum_{k=0}^\infty \frac{(-1)^k}{(k+1)^s} = \eta(s)$$

- The hard part is to show that for any compact $K \subset \mathbb{C}$, there is a constant α such that for every n and every $s \in K$: $\left|\sum_{k=0}^n \binom{n}{k} \frac{(-1)^k}{(k+1)^s}\right| < \alpha$, by noting this is the nth forward difference $\Delta^n(0)$ of the sequence $\left\{\frac{(-1)^k}{(k+1)^s}\right\}_{k\in\mathbb{N}}$, so that the series converges in $\mathcal{O}(2^{-N})$ for every $s \in \mathbb{C}$ and is entire, i.e. there is a continuous function $\alpha : \mathbb{C} \to \mathbb{R}^+$ such that

$$\forall s \in \mathbb{C}, \qquad \left|\eta(s) - \sum_{n=0}^N \frac{1}{2^{n+1}} \sum_{k=0}^n (-1)^k \binom{n}{k} \frac{1}{(k+1)^s}\right| < \alpha(s) 2^{-N}$$

Share Cite Follow edited Oct 28 '16 at 10:39 answered Oct 28 '16 at 10:09

reuns
53.5k ● 2 ■ 24 ▲ 92

5

As far as the speed of the algorithm **and** its implementation go, you probably won't find much better than implemented by the GNU scientific library. See the file `zeta.c`, line 776. The source is sometimes interspersed by relevant references to Abramowicz and Stegun and other sources. **Update**: See user1952009's comment below for the maths that's used there!

Albeit I understand that you prefer writing your own implementation, please only do so if that's a necessary exercise. It's a lot of work to do things effectively and keep all cases covered at the same time. It's easier to take the file from the GSL if you don't want the whole library (linking against which would lead to the necessity of including BLAS as well), it's designed to be separable into independent modules. You can also start exploring by modifying it to better suit your needs, in a situation similar to what you describe I once managed to shorten the code for some elliptic integrals to 5 lines or so of branchless code, executable on a GPU.

If you need some special additions like complex arguments or arbitrary precision, Pari/GP (ζ source) is similarly the place to go. But that's harder to understand and heavily relies on the stack model used in that framework. ~~I think they use Euler's identity with precomputed primes and use a smart trick utilizing Horner's scheme to reduce the number of operations. I once studied the internals of that but just forgot, I'll update my answer if I decrypt the formula from the code.~~

Update: I think Pari uses a literal implementation of the algorithm described here. But they have a precomputed list of both primes and Bernoulli numbers so that makes it easier.

First, the optimum bounds N and M are computed from the required precision. Then S (a partial sum of k^{-s}) is evaluated using

$$\sum_{k=1}^N \frac{1}{k^s} = \sum_{\substack{k=1 \\ k \text{ odd}}}^N \frac{1}{k^s} + \frac{1}{2^s} \sum_{k=1}^{\lfloor N/2 \rfloor} \frac{1}{k^s} = \sum_{\substack{k=1 \\ k \text{ odd}}}^N \frac{1}{k^s} + \frac{1}{2^s} \sum_{\substack{k=1 \\ k \text{ odd}}}^{\lfloor N/2 \rfloor} \frac{1}{k^s} + \frac{1}{(2^s)^2} \sum_{k=1}^{\lfloor N/4 \rfloor} \frac{1}{k^s} = \cdots$$

until the sum becomes trivial ($O(\log N)$ steps) using Horner scheme. To evaluate the finite sum over odd ks, they use prime factorization and a precomputed table of $\frac{1}{p^{ms}}$ for p prime. This is viable as long as N is moderately small. One can replace that by a finite version of the sum taken exactly from your former approach, just terminating much earlier.

I and T (the tail for $n > N$) are calculated together, evaluating the factorials and powers iteratively and using known B_{2n}. The error term R is dropped.

Obviously choosing a good (M, N) is crucial. I spent way too much time on the above to look up what they are, but I'm sure there's some freedom in how you pick them. You can choose M small so you don't need very many B_n, and then hardcode those.

Share Cite Follow

edited Apr 13 '17 at 12:20
Community ♦
1

answered Oct 28 '16 at 10:19
The Vee
2,791 12 30

2 See the comments at the beginning of the source code, this is an obfuscated version of en.wikipedia.org/wiki/Dirichlet_eta_function#Borwein.27s_method – reuns Oct 28 '16 at 10:25

@reuns, and Borwein's method is merely the application of a convergence acceleration algorithm to the η series; not what I'd call an obfuscation. – J. M. isn't a mathematician Aug 14 '17 at 7:39