Progress on algorithms for high-precision evaluation of special functions

Fredrik Johansson

October 23, 2013

RISC Algorithmic Combinatorics Seminar, Winter 2013/14

Contents

1. *Evaluating parametric holonomic sequences using rectangular splitting* (`arxiv.org/abs/1310.3741`)

2. *Rigorous high-precision computation of the Hurwitz zeta function and its derivatives* (`arxiv.org/abs/1309.2877`)

Part 1: Evaluating parametric holonomic sequences using rectangular splitting

Evaluating holonomic sequences

Linear recurrence of order r: $c(i+1) = M(i)c(i)$
where M is an $r \times r$ matrix of polynomials.

With $r = 2$:

$$\begin{bmatrix} c_1(i+1) \\ c_2(i+1) \end{bmatrix} = \begin{bmatrix} a_{11}(i) & a_{12}(i) \\ a_{21}(i) & a_{22}(i) \end{bmatrix} \begin{bmatrix} c_1(i) \\ c_2(i) \end{bmatrix}$$

Goal: compute $c(n)$ efficiently, for large n.

Beating the naive algorithm

Example (Fibonacci numbers):

$$\begin{bmatrix} F(i+1) \\ F(i+2) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} F(i) \\ F(i+1) \end{bmatrix}$$

Compute M^n using binary exponentiation ($O(\log n)$ operations). Works for constant M only.

In general: compute $M(n-1)M(n-2)\cdots M(0)$, exploit structure of matrix product.

Binary splitting

Recursively split product in half:

$$\begin{array}{ccccccc} & & & M(3)M(2)M(1)M(0) & & & \\ & & \nearrow & & \nwarrow & & \\ & M(3)M(2) & & & & M(1)M(0) & \\ \nearrow & & \nwarrow & & \nearrow & & \nwarrow \\ M(3) & & M(2) & & M(1) & & M(0) \end{array}$$

Useful if **the cost grows with the entry sizes**:

$R[x]$: $O(\mathsf{M}(n) \log n) = O(n^{1+\varepsilon})$ R-operations
$\mathbb{Z}$: $O(\mathsf{M}(n \log n) \log n) = O(n^{1+\varepsilon})$ bit ops

Fast multipoint evaluation

$p \in R[x]$ of degree n can be evaluated at n points using $O(\mathrm{M}(n) \log n) = O(n^{1+\varepsilon})$ R-operations.

$$p \bmod (x - c_0)(x - c_1)(x - c_2)(x - c_3)$$

$$\swarrow \qquad \searrow$$

$$p \bmod (x - c_0)(x - c_1) \qquad p \bmod (x - c_2)(x - c_3)$$

$$\swarrow \quad \searrow \qquad \swarrow \quad \searrow$$

$$p \bmod (x - c_0) \quad p \bmod (x - c_1) \quad p \bmod (x - c_2) \quad p \bmod (x - c_3)$$

Sequence evaluation using F.M.E.

$\prod_{i=0}^{n-1} M(i), M \in R[k]^{r \times r}$, $n = 16$, step length $m = \sqrt{n} = 4$

1. $P = M(k+3)M(k+2)M(k+1)M(k)$ by binary splitting.
2. $P(12), P(8), P(4), P(0)$ by fast multipoint evaluation.
3. $P(12)P(8)P(4)P(0)$ by repeated multiplication

Useful if **arithmetic operations have fixed cost**: $O(\mathrm{M}(n^{1/2}) \log n) = O(n^{1/2+\varepsilon})$ R-operations

Parametric sequences

Definition: We call $(c(x,n))_{n=0}^{\infty}$ a *parametric holonomic sequence* over C (with parameter x) if

$$c(x, i+1) = M(x,i)c(x,i)$$

where $M \in C[x][k]^{r \times r}$.

Example: rising factorials

$$c(x,n) = x(x+1)\dots(x+n-1)$$

with $M = (x+k-1)$ and $C = \mathbb{Z}$.

Cheap and expensive operations

Two rings $C \subset H$, distinguish between:

- *Coefficient operations* in C
- *Scalar operations* in H
  - Additions in H
  - Multiplications $C \times H \to H$
- *Nonscalar multiplications* $H \times H \to H$

We want to evaluate $c(x, n)$ at $x = z \in H$, but avoid as much work in H as possible.

Example: rising factorials, $C = \mathbb{Z}$, $H = \mathbb{R}$.

Rectangular splitting

Paterson-Stockmeyer (1973): $p \in C[x]$ of degree n can be evaluated for $x = z \in H$ using $O(n)$ scalar operations and $O(n^{1/2})$ nonscalar multiplications.

Example:

$$
\begin{array}{cccccccccccc}
(& p_0 & + & p_1 x & + & p_2 x^2 & + & p_3 x^3 &) & & + \\
(& p_4 & + & p_5 x & + & p_6 x^2 & + & p_7 x^3 &) & x^4 & + \\
(& p_8 & + & p_9 x & + & p_{10} x^2 & + & p_{11} x^3 &) & x^8 & \\
(& p_{12} & + & p_{13} x & + & p_{14} x^2 & + & p_{15} x^3 &) & x^{12} &
\end{array}
$$

Rectangular splitting for sequences

Example:

$$M = \begin{bmatrix} k+x & kx \\ k+1 & 1 \end{bmatrix} \in \mathbb{Z}[x][k]^{2\times 2}, \quad n = 9.$$

We compute

$$M(x,8)M(x,7)\cdots M(x,1)M(x,0) = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

where $a, b, c, d \in \mathbb{Z}[x]$ have degree ≤ 9, then evaluate a, b, c, d using Paterson-Stockmeyer.

Example

$$a = x^9 + 240x^8 + 15708x^7 + 335985x^6 + 2388848x^5 + 5774126x^4 + 5013705x^3 + 1574619x^2 + 149920x$$

$$b = x^8 + 236x^7 + 14789x^6 + 282148x^5 + 1515299x^4 + 2212956x^3 + 949355x^2 + 109600x$$

$$c = 9x^8 + 1520x^7 + 65163x^6 + 836457x^5 + 3288691x^4 + 4143766x^3 + 1704538x^2 + 193839x + 1$$

$$d = 9x^7 + 1484x^6 + 59452x^5 + 630520x^4 + 1592613x^3 + 985557x^2 + 141692x + 1$$

This is bad

Assume $x \in \mathbb{R}$, with $O(n)$ bits of precision

Expanded coefficients: $O(n \log n)$ bits

This is **slower than the naive algorithm**

Memory-hungry: $O(n)$ coefficients, $O(n^2 \log n)$ bits

Poor numerical stability with negative numbers

Improved rectangular splitting

$$\begin{array}{ccc} (M(x,8) & M(x,7) & M(x,6)) \quad \times \\ (M(x,5) & M(x,4) & M(x,3)) \quad \times \\ (M(x,2) & M(x,1) & M(x,0)) \end{array}$$

$$=$$

$$\begin{bmatrix} x^3 + 134x^2 + 978x + 336 & 6x^3 + 481x^2 + 400x \\ 9x^2 + 566x + 433 & 54x^2 + 489x + 1 \end{bmatrix} \times$$

$$\begin{bmatrix} x^3 + 53x^2 + 222x + 60 & 3x^3 + 106x^2 + 85x \\ 6x^2 + 143x + 91 & 18x^2 + 111x + 1 \end{bmatrix} \times$$

$$\begin{bmatrix} x^3 + 8x^2 + 6x & x^2 + 4x \\ 3x^2 + 8x + 1 & 3x + 1 \end{bmatrix}$$

This is good

Typical case: $x = z \in \mathbb{R}$, with $O(n)$ bits of precision

With $m \sim n^{1/2}$, we **still only do $O(n^{1/2})$ nonscalar multiplications**

Coefficients have $O(m \log m) = O(n^{1/2} \log n)$ bits

Memory consumption is $O(mn) = O(n^{3/2})$ bits

Difference version

Generate

$$\Delta_m = \prod_{i=0}^{m-1} M(x, k+m+i) - \prod_{i=0}^{m-1} M(x, k+i) \in C[x][k]^{r \times r}$$

$$M(5)M(4)M(3) = M(2)M(1)M(0) + \Delta_3(x, 0)$$

$$M(8)M(7)M(6) = M(5)M(4)M(3) + \Delta_3(x, 3)$$

Rising factorials, $m = 4$: D. M. Smith (2001).

Numerical evaluation

Bit complexity of scalar multiplications, $m \sim n^{\alpha}$, $0 < \alpha < 1$, x is a p-bit floating-point number:

$$O\left(n\,p\,\frac{\mathrm{M}(m \log m)}{m \log m}\right)$$

Mult. algorithm	Scalar multiplications	Naive
Classical	$\tilde{O}(n^{1+\alpha} p)$	$\tilde{O}(np^2)$
Karatsuba	$\tilde{O}(n^{1+0.585\alpha} p)$	$\tilde{O}(np^{1.585})$
FFT	$\tilde{O}(np)$	$\tilde{O}(np)$

Fast multipoint evaluation: $\tilde{O}(n^{0.5} p)$

Hypergeometric series

Smith (1989): use *scalar divisions* to remove content in the Paterson-Stockmeyer algorithm:

$$\begin{aligned}\exp(x) \approx \quad & \left(1 + \tfrac{1}{1}\left(x + \tfrac{1}{2}\left(x^2 + \tfrac{1}{3}x^3\right)\right)\right) \\ +\tfrac{x^4}{4!} \quad & \left(1 + \tfrac{1}{5}\left(x + \tfrac{1}{6}\left(x^2 + \tfrac{1}{7}x^3\right)\right)\right) \\ +\tfrac{x^8}{8!} \quad & \left(1 + \tfrac{1}{9}\left(x + \tfrac{1}{10}\left(x^2 + \tfrac{1}{11}x^3\right)\right)\right) \\ +\tfrac{x^{12}}{12!} \quad & \left(1 + \tfrac{1}{13}\left(x + \tfrac{1}{14}\left(x^2 + \tfrac{1}{15}x^3\right)\right)\right)\end{aligned}$$

Generalizing Smith's summation algorithm

Smith's algorithm:

- Coefficients with $O(\log n)$ bits
- Only $c(x, n) = \sum_{k=0}^{n} b(k) x^k$, b hypergeometric
- Requires divisions

Our algorithm:

- Coefficients with $O(n^{1/2} \log n)$ bits
- Any parametric sequence $c(x, n)$
- Avoids divisions

Both:

- Speedup with non-FFT multiplication

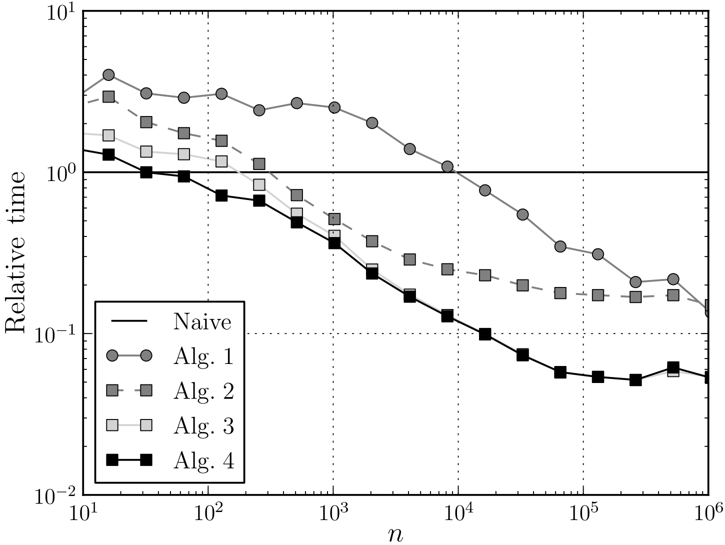
Rising factorial algorithms

- Naive algorithm
- Algorithm 1: fast multipoint evaluation
- Algorithm 2: poor rectangular splitting
- Algorithm 3: improved rectangular splitting
- Algorithm 4: difference version

In Algorithm 3 and 4, $m = \min(0.2p^{0.4}, n^{0.5})$.

Benchmark problem: $p = 4n$.

Comparison of rising factorial algorithms



Fast gamma function

$$\Gamma(z) \approx \gamma(z, N) = z^{-1} N^z e^{-N} {}_1F_1(1, 1+z, N)$$

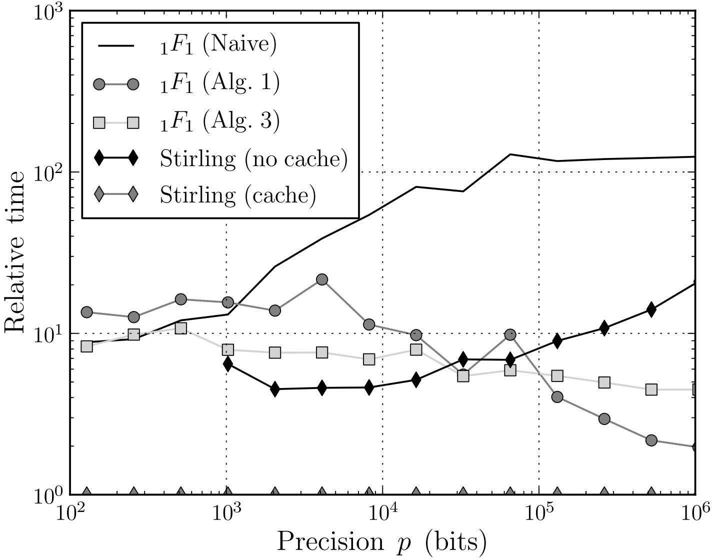
Take $t_k = N^k / (z(z+1)\cdots(z+k))$, $s_n = \sum_{k=0}^{n} t_k$,
$N \approx p \log 2$, $n \approx (e \log 2) p$

$$\begin{bmatrix} s_k \\ t_{k+1} \end{bmatrix} = \frac{M(k)}{q(k)} \frac{M(k-1)}{q(k-1)} \cdots \frac{M(0)}{q(0)} \begin{bmatrix} 0 \\ 1/z \end{bmatrix}$$

$$M(k) = \begin{bmatrix} 1+k+z & 1+k+z \\ 0 & N \end{bmatrix}, \quad q(k) = 1+k+z$$

$O(p^{1.5+\varepsilon})$ with fast multipoint evaluation

Comparison of gamma function algorithms



Summary

Our rectangular splitting algorithm:

- Avoids large coefficients
- Generalizes two different algorithms by Smith
- Simple, can be applied systematically to a general class of sequences
- Asymptotically slower, but in practice competitive, with fast multipoint evaluation

Part 2: Rigorous high-precision computation of the Hurwitz zeta function and its derivatives

The Hurwitz zeta function

$$\zeta(s, a) = \sum_{k=0}^{\infty} \frac{1}{(k+a)^s}$$

Analytic continuation to $s, a \in \mathbb{C}$.

Special cases: Riemann zeta ($a = 1$), Dirichlet L-functions, polygamma functions, polylogarithms, special values of hypergeometric functions

Goal

Compute $\zeta(s, a)$ and **derivatives** with respect to s, to **very high precision**, with **rigorous error bounds**.

Motivation:

- Rigorous analytic computations
- Studying asymptotics
- Determinant approximations for zeros (current work by Yuri Matiyasevich and Gleb Beliakov)
- LMFDB.org: database of L-functions and modular forms

The Euler-Maclaurin formula

$$\sum_{k=N}^{U} f(k) = I + T + R$$

$$\begin{aligned}
I &= \int_N^U f(t)\,dt \\
T &= \frac{1}{2}(f(N) + f(U)) \\
&\quad + \sum_{k=1}^{M} \frac{B_{2k}}{(2k)!}\left(f^{(2k-1)}(U) - f^{(2k-1)}(N)\right) \\
R &= -\int_N^U \frac{\tilde{B}_{2M}(t)}{(2M)!} f^{(2M)}(t)\,dt
\end{aligned}$$

Computing $\zeta(s, a)$ using Euler-Maclaurin

$$\zeta(s,a) = \underbrace{\sum_{k=0}^{N-1} f(k)}_{S} + \underbrace{\sum_{k=N}^{\infty} f(k)}_{I+T+R}, \quad f(k) = \frac{1}{(a+k)^s}$$

For derivatives, substitute $s \to s + x \in \mathbb{C}[[x]]$:

$$f(k) = \frac{1}{(a+k)^{s+x}} = \sum_{i=0}^{\infty} \frac{(-1)^i \log^i(a+k)}{i!(a+k)^s} x^i \in \mathbb{C}[[x]]$$

Parts to evaluate

$$S = \sum_{k=0}^{N-1} \frac{1}{(a+k)^{s+x}}$$

$$I = \int_N^\infty \frac{1}{(a+t)^{s+x}} dt = \frac{(a+N)^{1-(s+x)}}{(s+x)-1}$$

$$T = \frac{1}{(a+N)^{s+x}} \left(\frac{1}{2} + \sum_{k=1}^{M} \frac{B_{2k}}{(2k)!} \frac{(s+x)_{2k-1}}{(a+N)^{2k-1}} \right)$$

$$R = -\int_N^\infty \frac{\tilde{B}_{2M}(t)}{(2M)!} \frac{(s+x)_{2M}}{(a+t)^{(s+x)+2M}} dt \quad \text{(bound)}$$

Bounding the remainder

$$
\begin{aligned}
|R| &= \left| \int_N^\infty \frac{\tilde{B}_{2M}(t)}{(2M)!} \frac{(s+x)_{2M}}{(a+t)^{s+x+2M}} dt \right| \\
&\le \int_N^\infty \left| \frac{\tilde{B}_{2M}(t)}{(2M)!} \frac{(s+x)_{2M}}{(a+t)^{s+x+2M}} \right| dt \\
&\le \frac{4\,|(s+x)_{2M}|}{(2\pi)^{2M}} \int_N^\infty \left| \frac{dt}{(a+t)^{s+x+2M}} \right| \in \mathbb{R}[[x]]
\end{aligned}
$$

$$
\int_N^\infty \left| \frac{dt}{(a+t)^{s+x+2M}} \right| = \sum_{k=0}^{\infty} \left(\int_N^\infty \frac{dt}{k!} \left| \frac{\log(a+t)^k}{(a+t)^{s+2M}} \right| \right) x^k
$$

A sequence of integrals

For $k \in \mathbb{N}$, $A > 0, B > 1, C \geq 0$,

$$
\begin{aligned}
J_k(A, B, C) &\equiv \int_A^\infty t^{-B}(C + \log t)^k dt \\
&= \frac{L_k}{(B-1)^{k+1} A^{B-1}}
\end{aligned}
$$

where

$$
L_0 = 1,\; L_k = kL_{k-1} + D^k
$$

$$
D = (B-1)(C + \log A)
$$

Error bound

Theorem: for complex numbers $s = \sigma + \tau i$, $a = \alpha + \beta i$ and positive integers N, M such that $\alpha + N > 1$ and $\sigma + 2M > 1$,

$$|R| \le \frac{4\,|(s+x)_{2M}|}{(2\pi)^{2M}} \left| \sum_{k=0}^{\infty} R_k x^k \right| \in \mathbb{R}[[x]]$$

where $R_k \le (K/k!)\, J_k(N+\alpha, \sigma+2M, C)$ and K and C are certain numbers given explicitly in terms of s, a, N, M.

Evaluation steps

To evaluate $\zeta(s+x, a)$ with an error of 2^{-p}:

1. Choose $N, M = O(p)$, bound the error term R
2. Compute the power sum S
3. Compute the integral I
4. Compute the Bernoulli numbers
5. Compute the tail T

Asymptotically fast evaluation

Observation: the first n Taylor coefficients of $\zeta(s, a)$ can be simultaneously computed to $O(n)$ digits of precision in $O(n^{2+\varepsilon})$ time (**softly optimal**).

My implementation: $O(n^{3+\varepsilon})$, but supports **parallelization** ($\approx 16\times$ speedup on 16 cores).

Fast power series power sum

$$V = \begin{bmatrix} 1 & \log a & \cdots & \log^N a \\ 1 & \log(a+1) & \cdots & \log^N(a+1) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \log(a+N) & \cdots & \log^N(a+N) \end{bmatrix}$$

$$Y = \begin{bmatrix} a^{-s} & (a+1)^{-s} & \ldots & (a+N)^{-s} \end{bmatrix}^T$$

VY is **multipoint evaluation**. We want $V^T Y$.

A fast algorithm for $V^T Y$ exists by the **transposition principle**. (Not yet implemented.)

Fast power series tail

To compute: $\sum_{k=1}^{M} B_{2k} t(k) \in \mathbb{C}[[x]]$ where $t(k)$ is hypergeometric.

Binary splitting works. The terms are not holonomic due to the Bernoulli numbers, but close enough! In general:

$$t(k+1) = r(k)t(k)$$
$$s(k+1) = s(k) + b(k)t(k)$$

$$\begin{bmatrix} t(k+1) \\ s(k+1) \end{bmatrix} = \begin{bmatrix} r(k) & 0 \\ b(k) & 1 \end{bmatrix} \begin{bmatrix} t(k) \\ s(k) \end{bmatrix}$$

If we just want a few derivatives

Tricks to speed up the power sum when $a = 1$.

1. With $f(k) = k^{-(s+x)}$, $f(k_1 k_2) = f(k_1) f(k_2)$. Only need to evaluate $f(k)$ from scratch for **prime** k.

2. Fast logarithms of nearby integers (binary splitting):

$$\log(q) = \log(p) + 2 \operatorname{atanh}\left(\frac{q-p}{q+p}\right)$$

Some computational results

The Keiper-Li coefficients

Define $\{\lambda_n\}_{n=1}^{\infty}$ by

$$\log \xi\left(\frac{1}{1-x}\right) = \log \xi\left(\frac{x}{x-1}\right) = -\log 2 + \sum_{n=1}^{\infty} \lambda_n x^n$$

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

Keiper (1992): Riemann hypothesis $\Rightarrow \forall n : \lambda_n > 0$
Li (1997): Riemann hypothesis $\Leftarrow \forall n : \lambda_n > 0$

Keiper conjectured $2\lambda_n \approx (\log n - \log(2\pi) + \gamma - 1)$

Evaluating the Keiper-Li coefficients

Ingredients:

1. The series expansion of $\zeta(s)$ at $s = 0$
2. A series logarithm: $\log f(x) = \int f'(x)/f(x)dx$
3. Expansion of $\log \Gamma(1+s)$, essentially $\gamma, \zeta(2), \zeta(3), \zeta(4), \ldots$
4. Right-composing by $x/(x-1)$

A working precision of $\approx n$ bits is needed to get an accurate value for λ_n.

Fast composition

The *binomial transform* of $f = \sum_{k=0}^{\infty} a_k x^k$ is

$$T[f] = \frac{1}{1-x} f\left(\frac{x}{x-1}\right) = \sum_{n=0}^{\infty} \left(\sum_{k=0}^{n} (-1)^k \binom{n}{k} a_k \right) x^n$$

and the *Borel transform* is

$$B[f] = \sum_{k=0}^{\infty} \frac{a_k}{k!} x^k.$$

$T[f(x)] = B^{-1}[e^x B[f(-x)]]$, so we get $f\left(\frac{x}{x-1}\right)$ by a single power series multiplication!

Values of Keiper-Li coefficients

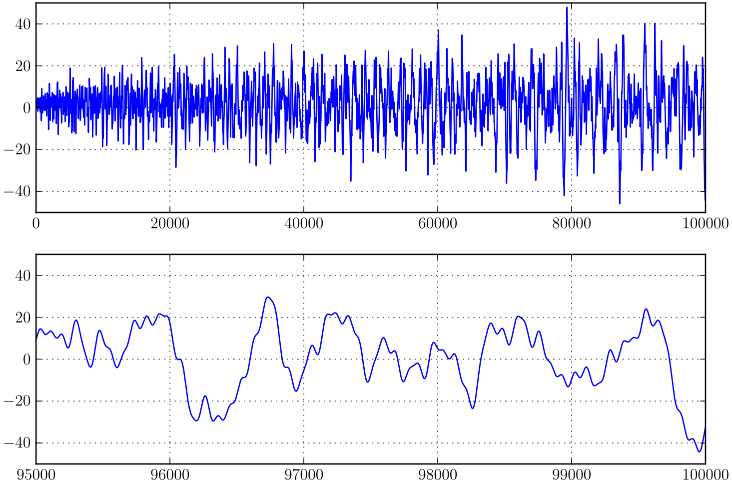
I have computed all λ_n up to $n = 100000$ using 110000 bits of precision. In particular,

$$\lambda_{100000} = 4.62580782406902231409416038\ldots$$

plus about 2900 more accurate digits.

Keiper's approximation suggests $\lambda_{100000} \approx 4.626132$.

Comparison with approximation formula



Plot of $n\,(\lambda_n - (\log n - \log(2\pi) + \gamma - 1)/2)$.

Timings for Keiper-Li coefficients (s)

	$n = 1000$	$n = 10000$	$n = 100000$
Error bound	0.017	1.0	97
Power sum	0.048	47	65402
(CPU time)	(0.65)	(693)	(1042210)
Bernoulli	0.0020	0.19	59
Tail	0.058	11	1972
Series log	0.047	8.5	1126
$\log \Gamma(1+x)$	0.019	3.0	1610
Composition	0.022	4.1	593
Total wall time	0.23	84	71051
RAM (MiB)	8	730	48700

Stieltjes constants

The **Stieltjes constants** are the coefficients $\gamma_n(a)$ ($\gamma_n(1) \equiv \gamma_n$) in the Laurent series

$$\zeta(s,a) = \frac{1}{s-1} + \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \gamma_n(a)\,(s-1)^n.$$

$\gamma_0 \approx +0.577216$ $\qquad \gamma_{10} \approx +0.000205$
$\gamma_1 \approx -0.072816$ $\qquad \gamma_{100} \approx -4.25340 \times 10^{17}$
$\gamma_2 \approx -0.009690$ $\qquad \gamma_{1000} \approx -1.57095 \times 10^{486}$

Asymptotics of Stieltjes constants

Open problem: **precise asymptotic bounds/series** for γ_n

Matsuoka (1985): $|\gamma_n| < 0.0001 e^{n \log \log n}, \; n \geq 10$

Knessl and Coffey (2011): asymptotic approximation formula (without explicit bound)

- Predicts sign oscillations
- Appears accurate even for small n
- Correct sign except for $n = 137$?

Knessl-Coffey approximation

$$\gamma_n \sim \frac{B}{\sqrt{n}} e^{nA} \cos(an + b)$$

$$A = \frac{1}{2}\log(u^2+v^2) - \frac{u}{u^2+v^2}, \quad B = \frac{2\sqrt{2\pi}\sqrt{u^2+v^2}}{[(u+1)^2+v^2]^{1/4}}$$

$$a = \tan^{-1}\left(\frac{v}{u}\right) + \frac{v}{u^2+v^2}, \quad b = \tan^{-1}\left(\frac{v}{u}\right) - \frac{1}{2}\left(\frac{v}{u+1}\right)$$

where $u = v \tan v$, and v is the unique solution of
$2\pi \exp(v \tan v) = (n/v)\cos(v), \quad 0 < v < \pi/2.$

Similar formula for $\gamma_n(a)$, $a \neq 1$.

Numerical values

Computed value of γ_{100000} ($>$10860 digits):
$1.99192730631254109565\ldots \times 10^{83432}$

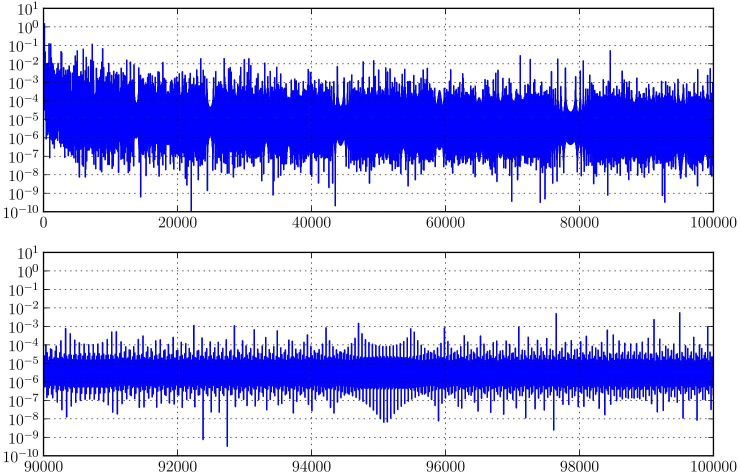
Knessl-Coffey approximation:
$1.9919333 \times 10^{83432}$

Matsuoka bound:
3.71×10^{106114}

Computed value of $\lambda_{50000}(1+i)$:
$(1.032502087431\ldots - 1.441962552840\ldots i) \times 10^{39732}$

Knessl-Coffey approximation:
$(1.0324943 - 1.4419586i) \times 10^{39732}$

Relative error of Knessl-Coffey formula



Nontrivial zeta zeros

I have computed the first nontrivial zero

$$0.5 + 14.13472514173\ldots i$$

of $\zeta(s)$ to over 300,000 digits.

Matiyasevich and Beliakov have now computed the first 40,000 zeros to 40,000 digits using my software (data soon to be published).

The end

Thank you!