Population genomics of the critically endangered spoon-billed sandpiper
Model calibration

Raimundo Saona[*] Anastasia Lyulina[†] Fyodor Kondrashov[*‡]

September 24, 2024

Abstract

In this report we present the numerical approximation of the expectation of polymorphisms, also called heterozygosity, for a model of random selection coefficient. We present the model, and how to approximate all involved quantities step-by-step.

Contents

[*]Institute of Science and Technology Austria
[†]Stanford University
[‡]Okinawa Institute of Science and Technology

1 Modeling

We use the following notation.

- selection coefficient s
- dominance coefficient h
- allele frequency x
- mutation rate U
- population size N

Our parametric model for the distribution of the allele frequency is the following.

$$f(x \mid s, h) \propto \exp(2Ns(x^2 + 2hx(1-x)))\, x^{4NU-1}\,(1-x)^{4NU-1}\,.$$

where we model s by a mixture of a Gamma distribution with shape α and scale $1/\beta$ (with negative sign), and a point mass at s_0 with probability p. In other words,

$$S \sim \begin{cases} \delta_{s_0} & p \\ -\Gamma(\alpha, \beta) & 1-p \end{cases}$$

where p is the probability of the mixture, and the Gamma distribution with shape α and scale $1/\beta$ is given by

$$f_{\Gamma(\alpha,\beta)}(s) = x^{\alpha-1}\exp(-\beta s)\frac{\beta^\alpha}{\Gamma(\alpha)}\mathbb{1}[s > 0]\,.$$

The dominance coefficient h is related to the selection coefficient as follows.

$$h = h(s) = \begin{cases} 0 & s = s_0 \\ 0.5 & s \neq s_0 \end{cases}$$

We fix the following values.

- $s_0 = -1 \cdot 10^{-2}$
- $N = 2 \cdot 10^4$
- $U = 1.2 \cdot 10^{-8}$

so the model is parameterized by p, α, and β.

2 Polymorphisms

Given a allele frequency x, a measure of polymorphisms is $2x(1-x)$. In our model, the allele frequency is a random variable X. Therefore, the expected number of polymorphisms is

$$\mathbb{E}[poly] = \mathbb{E}_S\left[\mathbb{E}_X\left[2X(1-X) \mid S\right]\right]\,.$$

So $\mathbb{E}[poly]$ can be approximated using a Monte Carlo approach by simulating the random selection coefficient S.

3 Numerical approximation

The following are key to derive formal error bounds on integrals.

Integration bounds. The following are simple but powerful results to numerically approximate integrals.

Proposition 3.1. *Consider $t > 0$ and a function $f\colon [0,t] \to \mathbb{R}$. If $\|f - f(0)\|_\infty \le L$, then*

$$\left| \int_{[0,t]} f(x)dx - f(0)t \right| \le Lt\,.$$

Proposition 3.2. *Consider $t > 0$ and functions $f, g\colon [0,t] \to \mathbb{R}$. If $\|f - f(0)\|_\infty \le L$ and $\|g - g(0)\|_\infty \le L$, then*

$$\left| \int_{[0,t]} f(x)g(x)\,dx - f(0)g(0)t \right| \le L(f(0) + g(0))t + L^2 t\,.$$

Floating points. The IEEE Standard for Floating-Point Arithmetic (IEEE 754-2008) defines 64-bit floating point numbers, called "binary64". These represent numbers in scientific notation in base 2 with 1 sing bit, 11 exponent bits, and 53 significand precision bits. In other words, there are 15.95 decimal digits of accuracy, for numbers whose exponent in base 10 is in $[-308, 308]$. The operations one can perform over floating point numbers losing only one binary digit of accuracy include addition, substraction, multiplication, division, exponentiation [Gol91, Tan89], logarithmization [DDLM07, LMBDDM16], and arbitrary sum [ZH10].

For example, for all $n \ge 1$ and $x_1, x_2, \ldots, x_n \in \mathbb{R}$ such that $10^{-308} \le |x_i| \le 308$, we can compute

$$\sum_{i=0}^{n} \exp(x_i)$$

losing only at most two binary digits of accuracy.

Normalizing constant. Consider

$$C(s,h) = \int_{[0,1]} g(x \mid s,h)\,dx = \int_{[0,1]} \exp(2Ns(x^2 + 2hx(1-x)))\, x^{4NU-1}\,(1-x)^{4NU-1}\,dx\,,$$

the normalizing constant of $f(x \mid s,h)$.

Note that $g(\,\cdot \mid s,h)$ is a function with singularities at 0 and 1. To numerically deal with this, $C(s,h)$ can be written as a numerically stable expression as follows.

$$\begin{aligned}
C(s,h) &= \int_{[0,1]} g(x \mid s,h)\,dx \\
&= \int_{[0,1/2]} g(x \mid s,h)\,dx + \int_{[1/2,1]} g(x \mid s,h)\,dx \\
&= \int_{[0,1/2]} g(x \mid s,h) - x^{4NU-1}\,dx + \int_{[0,1/2]} x^{4NU-1}\,dx \\
&\quad + \int_{[1/2,1]} g(x \mid s,h) - \exp(2Ns)(1-x)^{4NU-1}\,dx + \exp(2Ns)\int_{[1/2,1]} (1-x)^{4NU-1}\,dx \\
&= \int_{[0,1/2]} g(x \mid s,h) - x^{4NU-1}\,dx + \int_{[1/2,1]} g(x \mid s,h) - \exp(2Ns)(1-x)^{4NU-1}\,dx \\
&\quad + (1 + \exp(2Ns)) \int_{[0,1/2]} x^{4NU-1}\,dx \\
&= \int_{[0,1/2]} g(x \mid s,h) - x^{4NU-1}\,dx + \int_{[1/2,1]} g(x \mid s,h) - \exp(2Ns)(1-x)^{4NU-1}\,dx \\
&\quad + \big(1 + \exp(2Ns)\big) \frac{1}{4NU} \left(\frac{1}{2}\right)^{4NU}\,.
\end{aligned}$$

This expression is stable because all integrands are continuous and have no singularities. See Figure 1 for an illustration.

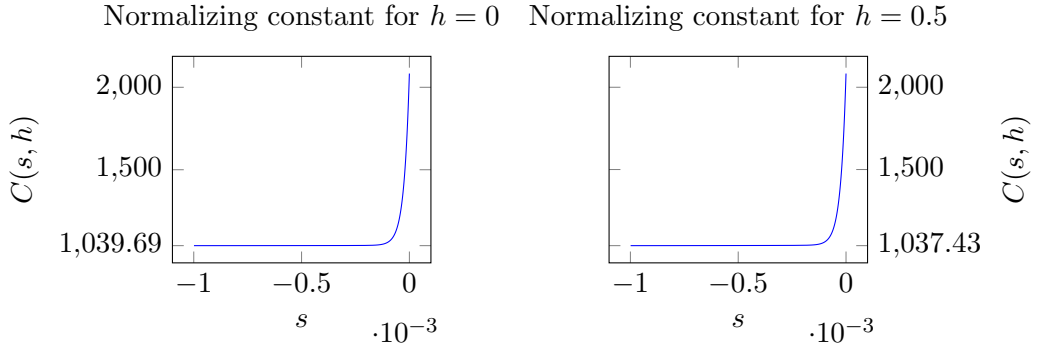

Figure 1: Normalizing constant in terms of s

Fixed s and h. Fix s, h such that $|2Ns| < 1000$, i.e., $|s| < 1/40 = 25 \cdot 10^{-2}$. We are interested in in computing

$$\begin{aligned}
\mathbb{E}\left[poly \mid s,h\right] &= \mathbb{E}_X\left[2X(1-X) \mid s,h\right] \\
&= \int_{[0,1]} 2x(1-x)\frac{f(x \mid s,h)}{C(s,h)}dx \\
&= \int_{[0,1]} 2x(1-x)\frac{\exp(2Ns(x^2+2hx(1-x)))\, x^{4NU-1}\,(1-x)^{4NU-1}}{C(s,h)}\,dx \\
&= \frac{2}{C(s,h)}\int_{[0,1]} \exp(2Ns(x^2+2hx(1-x)))\, x^{4NU}\,(1-x)^{4NU}\,dx \\
&= \frac{2}{C(s,h)}\int_{[0,1]} \exp(4NU\ln(x(1-x)) + 2Ns(x^2+2hx(1-x)))\,dx\,,
\end{aligned}$$

We show how to approximate

$$\int_{[0,1]} \exp(4NU\ln(x(1-x))+)\,dx\,.$$

First, we notice that we are interested in $s \in [-0.01, 0]$ and $h \in [0, 1/2]$. Second, the integration in $[0, 1/2]$ is harder to approximate than the integration in $[1/2, 1]$ since the exponential $\exp(2Ns(x^2 + 2hx(1-x)))$ is smaller for x large. Therefore, we present in detail how to approximate the integration over $[0, 1/2]$. Note that, for $s \le 0$, $h \in [0, 1/2]$, and $x \in [0, 1]$, we have that

$$\begin{aligned}
\exp(2Ns(x^2+2hx(1-x))) &\le 1 \\
x^{4NU} &\le 1 \\
(1-x)^{4NU} &\le 1\,.
\end{aligned}$$

Therefore,

$$\mathbb{E}\left[poly \mid s,h\right] \le \frac{2}{C(s,h)}\,.$$

The integral over $[0, 1/2]$ can be approximated as usual by a sum over a grid $0 = x_0 < x_1 < \ldots < x_n = 1/2$. Note that, for every $0 < x_1 < x_2 < 1/2$,

$$\sup_{x\in[x_1,x_2]} \left|\frac{\partial}{\partial x}\exp(2Ns(x^2+2hx(1-x)))\, x^{4NU}\,(1-x)^{4NU}\right| \le 4N\left(\frac{U}{x_1}+s\right)\,.$$

Therefore, by Proposition 3.1,

$$\int_{[x_1,x_2]} \exp(2Ns(x^2+2hx(1-x)))\, x^{4NU}\,(1-x)^{4NU}\,dx \\ \approx (x_2-x_1)\exp(2Ns(x_1^2+2hx_1(1-x_1)))\, x_1^{4NU}\,(1-x_1)^{4NU}\,,$$

where the approximation error is no larger than $(x_2-x_1)^2 4N(U/x_1+|s|)$. We control this error by choosing an appropiate grid. For the initial and last intervals in our grid, by a simple bound for $s<0$,

$$\left|\int_{[0,x_1]} \exp(2Ns(x^2+2hx(1-x)))\, x^{4NU}\,(1-x)^{4NU}\,dx\right| < x_1\,.$$

Therefore, the overall error by using a grid and the simplest approximation of the integral is

$$x_1 + \sum_{i=1}^{n-1}(x_{i+1}-x_i)^2 4N\left(\frac{U}{x_i}+|s|\right)\,.$$

For $\varepsilon\in(0,1)$, define $x_1 \coloneqq \varepsilon$ and

$$x_{i+1} \coloneqq \sqrt{\varepsilon}\sqrt{x_i}+x_i\,.$$

This grid satisfies that $(x_{i+1}-x_i)^2/x_i = \varepsilon$. Therefore, this grid incurs in at most the following approximation error of the integral

$$\varepsilon + \sum_{i=1}^{n(\varepsilon)-1} \varepsilon 4N(U+|s|) < \varepsilon\,(1+n(\varepsilon)4N(U+|s|))\,,$$

where $n(\varepsilon)$ is number of points required so that $x_{n(\varepsilon)} \geq 1/2$, which converges to zero as ε converges to zero.

Note that, the smaller ε, the smaller the numbers x_1 we need to consider. As we work with 64-bits floating points, there is a limit to how small we can consider ε. We keep track of the error estimation to ensure that the overall error is less than 10^{-6}. See Figure 2 for a plot of the behaviour of the approximation error of the integral in terms of ε.

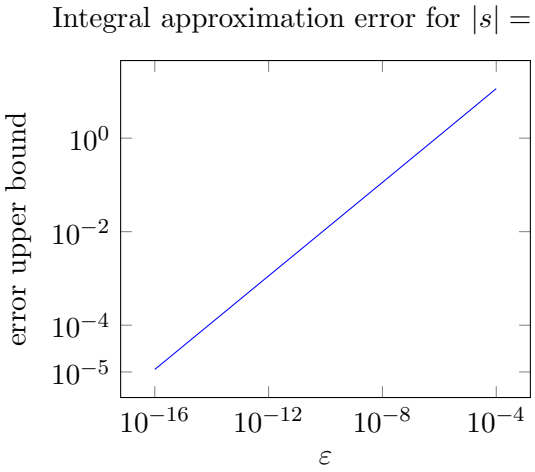

Figure 2: Plot of the normalizing constant

Note that, if $\varepsilon \geq 10^{-16}$, then $n(\varepsilon) < 2\cdot 10^8$, so the required number of evaluations is manageable in modern computers. Recall that, each evaluation must be within the range of

64-bit floating point numbers. This is the case because, for $s \in [-1 \cdot 10^{-2}, 0]$ and $h \in [0, 0.5]$, we have that, for all $x \in [10^{-16}, 1/2]$,

$$|4NU \ln(x(1-x)) + 2Ns(x^2 + 2hx(1-x))| \leq |4NU \ln(x)| + |2Ns(x^2 + x)| \leq 301\,,$$

so the integrand involving the exponential is within the 64-bit floating point range.

In summary, taking $\varepsilon = 10^{-16}$, we have shown that, for $s \in [-1 \cdot 10^{-2}, 0]$ and $h \in [0, 1/2]$,

$$\begin{aligned}&\int_{[0,1/2]} \exp(4NU \ln(x(1-x)) + 2Ns(x^2 + 2hx(1-x)))\, dx \\ &\quad\approx \sum_{i=1}^{n(10^{-15})-1} (x_{i+1} - x_i) \exp(4NU \ln(x_i(1-x_i)) + 2Ns(x_i^2 + 2hx_i(1-x_i)))\,,\end{aligned}$$

and the approximation error is less than 10^{-5}.

To obtain this sum with an appropiate accuracy, note that x_i is represented in 64-bit floating point, which has at least 15.9 digits of accuracy. Then, while computing this expression naively, we lose at most 12 binary digits, which corresponds to at most 4 digits of accuracy. In other words, we can obtain this approximation with 11 digits of accuracy.

The integral over $[1/2, 1]$ can be approximated using similar bounds. Finally, together with an appropiate approximation of $C(s, h)$, we can compute $\mathbb{E}[poly \mid s, h]$ with an error of at most 10^{-5}. Figure 3 and Figure 4 visualize the approximation for $h = 0.0$ and $h = 0.5$ respectively.

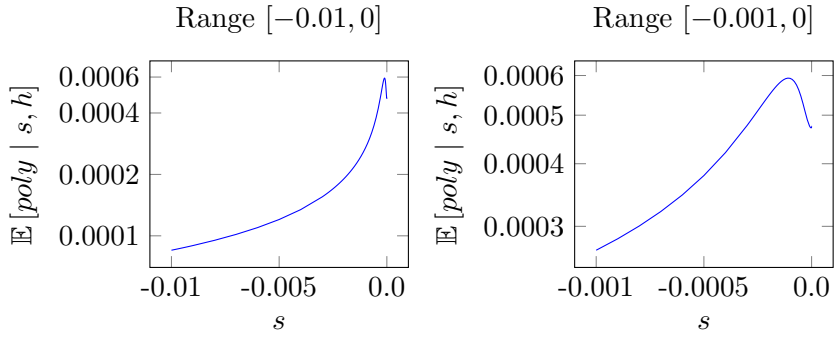

Figure 3: Expected polymorphisms in terms of the selection coefficient for $h = 0.0$

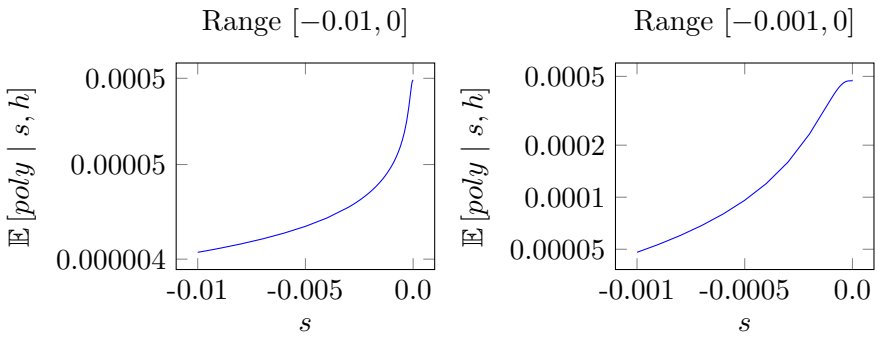

Figure 4: Expected polymorphisms in terms of the selection coefficient for $h = 0.5$

Random s. To compute $\mathbb{E}[poly]$ where s is random and h is a function of s, we simply integrate our approximation of $\mathbb{E}[poly \mid s, h]$ with the corresponding density function of s. In other words, we compute the following expression

$$\mathbb{E}[poly] = \int_{\mathbb{R}} \mathbb{E}[poly \mid s, h(s)]\, f_s(s)\, ds\,.$$

Point mass. Consider

$$s \equiv s_0$$
$$h \equiv 0.0$$

Then,

$$\mathbb{E}[poly] = \mathbb{E}\left[poly \mid S_0, h(S_0)\right] \approx 0.000085\,.$$

Gamma distribution. Consider

$$s \sim -\Gamma(\alpha, \beta)$$
$$h \equiv 0.5$$

Therefore,

$$\mathbb{E}[poly] = \int_{-\infty}^{0} \mathbb{E}\left[poly \mid s, h(s)\right] f_{\Gamma(\alpha,\beta)}(-s)\, ds\,,$$

Note that the infinite integral can be approximated by imposing a lower bound since $\Gamma(\alpha, \beta)$ is exponentially decreasing and $\mathbb{E}\left[poly \mid s, h(s)\right] \in [0, 1]$, and the bound depends on α and β. For example, by Chebyshev's inequality, we have that

$$\mathbb{P}(\Gamma(\alpha, \beta) \geq t) \leq \frac{\alpha}{(t\beta)^2}\,,$$

so the lower bound -0.01 implies an error of at most 10^{-5} as long as $\alpha \leq \beta^2 \cdot 10^{-9}$. By further considering a bound on the value of $\mathbb{E}[poly|s, h(s)]$, we can deal with more parameters α, β.

Lastly, while the density $f_{\Gamma(\alpha,\beta)}(\text{cot})$ is generally difficult to evaluate, the cummulative density $F_{\Gamma(\alpha,\beta)}(\text{cot})$ is easier to evaluate numerically. Therefore, we consider the following approximation

$$\mathbb{E}[poly] \approx \sum_{i=1}^{n-1} \big(F_{\Gamma(\alpha,\beta)}(s_{i+1}) - F_{\Gamma(\alpha,\beta)}(s_i)\big)\mathbb{E}\left[poly \mid -s_i, h(-s_i)\right]\,,$$

where $0 = s_0 < s_1 < \ldots < s_n$ is an appropiate grid. This approximation correspond to approximate $s \mapsto \mathbb{E}\left[poly \mid s, h(s)\right]$ by a function that is constant by parts. Considering previous approximations, we can choose a grid so that the overall error is less than 10^{-5}. See Figure 5 for an illustration.

4 Examples

In this section we compute $\mathbb{E}[poly]$ explictly for one parameter choice. Consider the case $p = 0.5$, $\alpha = 10$, and $\beta = 2 \cdot 10^4$. Note that

$$\mathbb{E}[poly] = p\mathbb{E}[poly|s = S_0, h = 0] + (1 - p)\mathbb{E}_{-s\sim\Gamma(\alpha,\beta)}[\mathbb{E}[poly|s, h = 0.5]]\,.$$

Since

$$\mathbb{E}[poly|s = S_0, h = 0] \approx 0.0000850151$$
$$\mathbb{E}_{-s\sim\Gamma(\alpha,\beta)}[\mathbb{E}[poly|s, h = 0.5]] \approx 0.0001073323$$

we get that

$$\mathbb{E}[poly] \approx 0.0000961737\,.$$

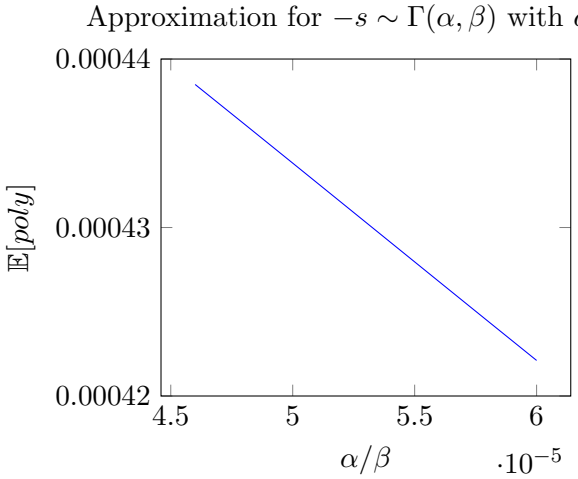

Figure 5: Expected polymorphisms for gamma distributed selection coefficient

5 Numerical evaluations

Our parametric model depends on p, α and β. We explore the space of all possible values by computing the expected polymorphisms only on a selected choice of parameters.

p	α	$1/\beta$	expected polymorphisms
0	300.0000000000001	0.0000001541467657920456	0.0004530
0	300.0000000000001	0.0000001541467657920456	0.0004530
0	290.8506518295917	0.000000158489319246111 4	0.0004532
0	225.56161320737237	0.000000205986394261687 5	0.0004528
0	182.5276007813775	0.000000251188643150958 23	0.0004534
0	169.59347117570735	0.000000270803644216005 5	0.0004533
0	127.51258982610176	0.000000365200481384398 6	0.0004525
0	115.93811031632079	0.000000398107170553496 87	0.0004530
0	95.87315155141827	0.000000483604804724923 4	0.0004527
0	72.81762821148743	0.00000063095734448019 3	0.0004530
0	72.08434242404265	0.000000637678477626980 7	0.0004530
0	54.19820188053225	0.00000086204985602077 4	0.0004520
0	46.27986268057486	0.000000999999999999997	0.0004524
0	40.750112830372316	0.000001141635691819326 7	0.0004521
0	30.638870628004046	0.000001517464554436814 3	0.0004518
0	29.267338838171042	0.000001584893192461114	0.0004519
0	23.036510285681892	0.000002042611760959217 4	0.0004509
0	18.58524958360993	0.000002511886431509582 3	0.0004510
0	17.320508075688775	0.000002707639018832409 8	0.0004506
0	13.022805810412262	0.000003630064702417371 5	0.0004497
0	11.84596516073017	0.00000398107170553496 9	0.0004496
0	9.791483623609768	0.000004887475719264104	0.0004483
0	7.549938308749734	0.00000630957344480193	0.0004477
0	7.3619428061166206	0.000006487995060251171	0.0004475
0	5.535238985626912	0.000008806082396876 74	0.0004453
0	4.874872287742436	0.00001	0.0004447
0	4.161791450287818	0.000011941732589246017	0.0004429
0	3.151143795792919	0.000015848931924611138	0.0004411

0	3.1291346445318977	0.000015980727030317896	0.0004410
0	2.3527088612123075	0.000022084851528262447	0.0004370
0	2.0796912080337586	0.00002511864315095822	0.0004359
0	1.7689360204744258	0.000030532846214531185	0.0004326
0	1.386785662694208	0.000039810717055349695	0.0004296
0	1.3300135414628025	0.000042040057592132016	0.0004285
0	1	0.000059268778545316244	0.0004230
0.1	300.0000000000001	0.00000012221738250849407	0.0004244
0.1	300.0000000000001	0.00000012221738250849407	0.0004244
0.1	225.56161320737237	0.000000157456144792358 95	0.0004251
0.1	224.1010946598654	0.000000158489319246111 4	0.0004251
0.1	169.59347117570732	0.000000214446555285611 3	0.0004245
0.1	143.34770274920493	0.000000251188643150958 23	0.0004248
0.1	127.51258982610176	0.000000285796816639251 66	0.0004245
0.1	95.87315155141827	0.000000374231514388811 07	0.0004248
0.1	90.0283006949983	0.000000398107170553496 9	0.0004248
0.1	72.08434242404266	0.000000508397614402201 2	0.0004242
0.1	57.05958587182932	0.000000630957344801 93	0.0004246
0.1	54.19820188053225	0.000000666696266644451 6	0.0004245
0.1	40.750112830372316	0.000000893069946651287 3	0.0004242
0.1	36.22876263209082	0.000001	0.0004242
0.1	30.63887062800405	0.000001199346818638652 1	0.0004237
0.1	23.036510285681896	0.000001571441616584578 4	0.0004239
0.1	22.837445924637542	0.000001584893192461114	0.0004239
0.1	17.320508075688775	0.000002130056253684444 7	0.0004231
0.1	14.603973433793314	0.000002511886431509582 3	0.0004231
0.1	13.02280581041226	0.000002841939853710307 7	0.0004227
0.1	9.791483623609768	0.000003778197598796755 3	0.0004222
0.1	9.284429639760333	0.000003981071705534969	0.0004221
0.1	7.36194280611662	0.000005109136821319283	0.0004211
0.1	5.947353627019855	0.000006309573444801 93	0.0004206
0.1	5.535238985626913	0.000006827122654861981	0.0004201
0.1	4.161791450287818	0.000009186279145629 49	0.0004187
0.1	3.8238639591838774	0.00001	0.0004184
0.1	3.129134644531898	0.000012482821607816822	0.0004168
0.1	2.475753088277439	0.000015848931924611 14	0.0004155
0.1	2.3527088612123075	0.000016813321581146697	0.0004149
0.1	1.7689360204744258	0.000022940154646508 34	0.0004123
0.1	1.6201389072592205	0.000025118864315095822	0.0004116
0.1	1.3300135414628023	0.000031680605285582 89	0.0004090
0.1	1.0756784679359217	0.000039810717055349695	0.0004069
0.1	1	0.000043676489470296766	0.0004056
0.2	256.2807710325777	0.0000001	0.0003918
0.2	256.2807710325777	0.0000001	0.0003918
0.2	225.56161320737237	0.000000113646961223048 18	0.0003918
0.2	169.59347117570735	0.000000151239537475959 92	0.0003918
0.2	161.96443250931463	0.000000158489319246111 4	0.0003918
0.2	127.51258982610179	0.000000200559421862730 17	0.0003918
0.2	102.14031138870588	0.000000251188643150958 23	0.0003918

0.2	95.87315155141827	0.00000026824058482269253	0.0003918
0.2	72.08434242404265	0.00000035497809161196566	0.0003918
0.2	64.36502993948046	0.00000039810717055349 69	0.0003918
0.2	54.19820188053225	0.00000047318691333941774	0.0003917
0.2	40.750112830372316	0.0000006294673535524286	0.0003917
0.2	40.65347306971581	0.0000006309573444801 93	0.0003917
0.2	30.63887062800405	0.00000083651446552443 01	0.0003916
0.2	25.639846151723688	0.000001	0.0003916
0.2	23.036510285681896	0.00000111780820960495 72	0.0003915
0.2	17.320508075688775	0.00000148221209068965 04	0.0003914
0.2	16.194502664483952	0.0000015848931924611 14	0.0003914
0.2	13.022805810412258	0.00000198362095700383 84	0.0003912
0.2	10.253734393489943	0.00000251188643150958 23	0.0003910
0.2	9.791483623609768	0.00000263841007779328 96	0.0003909
0.2	7.3619428061166206	0.0000035247279599420 3	0.0003906
0.2	6.510512597897407	0.00000398107170553496 9	0.0003905
0.2	5.535238985626911	0.00000472660670927546 5	0.0003901
0.2	4.161791450287818	0.00000627333608539715 86	0.0003897
0.2	4.137573074316612	0.0000063095734448019 3	0.0003896
0.2	3.129134644531898	0.0000084575805160051 8	0.0003888
0.2	2.643460952791566	0.00001	0.0003884
0.2	2.3527088612123075	0.00001134730137686367 1	0.0003879
0.2	1.7689360204744258	0.00001516480624590517 3	0.0003868
0.2	1.6917548381801009	0.0000158489319246111 4	0.0003867
0.2	1.3300135414628025	0.0000205617733729961 3	0.0003853
0.2	1.0917428030640703	0.0000251188643150958 2	0.0003844
0.2	1	0.00002778698158107633 3	0.0003837
0.3	131.73101632587696	0.0000001	0.0003553
0.3	131.73101632587696	0.0000001	0.0003553
0.3	127.51258982610177	0.00000010360313272253 67	0.0003552
0.3	95.87315155141827	0.00000013560818222650752	0.0003553
0.3	82.91681924936498	0.0000001584893192461114	0.0003553
0.3	72.08434242404265	0.00000018379256714765547	0.0003552
0.3	54.19820188053225	0.00000024049898594786135	0.0003553
0.3	52.162340819646396	0.00000025118864315095823	0.0003553
0.3	40.750112830372316	0.00000032064555654912 42	0.0003553
0.3	32.987505571800696	0.00000039810717055349 69	0.0003552
0.3	30.638870628004042	0.00000043005407274758674	0.0003552
0.3	23.036510285681896	0.00000056214156662142 32	0.0003553
0.3	20.647798282350223	0.0000006309573444801 93	0.0003552
0.3	17.320508075688775	0.00000075170420000203 7	0.0003552
0.3	13.023131078147005	0.000001	0.0003552
0.3	13.02280581041226	0.00000100002535401232 69	0.0003552
0.3	9.791483623609768	0.00000130849986639276 43	0.0003552
0.3	8.090646325469281	0.0000015848931924611 4	0.0003552
0.3	7.3619428061166206	0.00000173819399422295 66	0.0003552
0.3	5.535238985626912	0.0000022781928618455 2	0.0003551
0.3	5.01313317811982	0.00000251188643150958 23	0.0003551
0.3	4.161791450287818	0.00000299202386193602 8	0.0003551

0.3	3.129134644531898	0.000039411171707506 88	0.0003550
0.3	3.0963385271032418	0.000039810717055349 69	0.0003550
0.3	2.3527088612123075	0.000051142938718884704	0.0003550
0.3	1.883291180781875	0.00000630957344480193	0.0003549
0.3	1.7689360204744258	0.000066944182111722 88	0.0003549
0.3	1.3300135414628025	0.000086797882973864 42	0.0003547
0.3	1.1403134896010028	0.000099999999999999 99	0.0003547
0.3	1	0.000113052238957565 21	0.0003546

References

[DDLM07] Florent De Dinechin, Christoph Lauter, and Jean-Michel Muller. Fast and correctly rounded logarithms in double-precision. *RAIRO - Theoretical Informatics and Applications*, 41(1):85–102, January 2007.

[Gol91] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, March 1991.

[LMBDDM16] Julien Le Maire, Nicolas Brunie, Florent De Dinechin, and Jean-Michel Muller. Computing floating-point logarithms with fixed-point operations. In *2016 IEEE 23nd Symposium on Computer Arithmetic (ARITH)*, pages 156–163, Santa Clara, CA, July 2016. IEEE.

[Tan89] Ping-Tak Peter Tang. Table-driven implementation of the exponential function in IEEE floating-point arithmetic. *ACM Transactions on Mathematical Software*, 15(2):144–157, June 1989.

[ZH10] Yong-Kang Zhu and Wayne B. Hayes. Algorithm 908: Online Exact Summation of Floating-Point Streams. *ACM Transactions on Mathematical Software*, 37(3):1–13, September 2010.