

1 Correctness and Performance Charts

This version of the document is dated 2023-06-13.

The following charts show the correctness of many of the algorithms in “[Bernoulli Factory Algorithms¹](#)” and show their performance in terms of the number of bits they use on average. For each algorithm, and for each of 100 λ values evenly spaced from 0.0001 to 0.9999:

- 500 runs of the algorithm were done. Then...
- The number of bits used by the runs were averaged, as were the return values of the runs (since the return value is either 0 or 1, the mean return value will be in the interval $[0, 1]$). The number of bits used included the number of bits used to produce each coin flip, assuming the coin flip procedure for λ was generated using the `Bernoulli#coin()` method in *bernoulli.py*, which produces that probability in an optimal or near-optimal way.

For each algorithm, if a single run was detected to use more than 5000 bits for a given λ , the entire data point for that λ was suppressed in the charts below.

In addition, for each algorithm, a chart appears showing the minimum number of input coin flips that any fast Bernoulli factory algorithm will need on average to simulate the given function, based on work by Mendo (2019)[¹]. Note that some functions require a growing number of coin flips as λ approaches 0 or 1. Note that for the 2014, 2016, and 2019 algorithms—

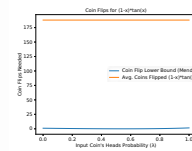
- an ϵ of $1 - (x + c) * 1.001$ was used (or 0.0001 if ϵ would be greater than 1), and
- an ϵ of $(x - c) * 0.9995$ for the subtraction variants.

Points with invalid ϵ values were suppressed. For the low-mean algorithm, an m of $\max(0.49999, xc1.02)$ was used unless noted otherwise.

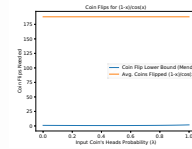
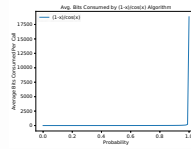
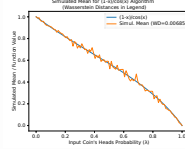
1.1 The Charts

Algorithm	Simulated Mean	Average Bits Consumed	Coin Flips
-----------	----------------	-----------------------	------------

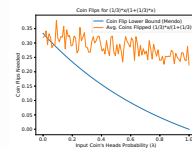
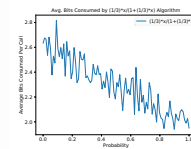
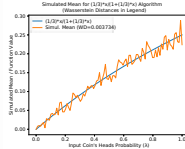
$$(1-x)*\tan(x)$$



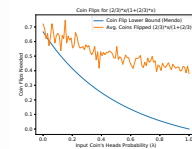
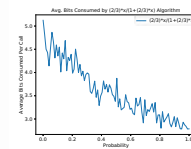
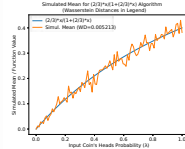
$$(1-x)/\cos(x)$$



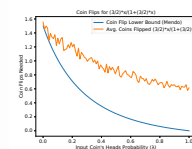
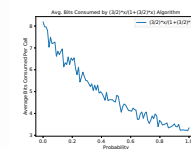
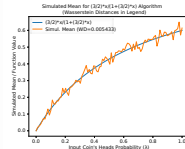
$$(1/3)*x/(1+(1/3)*x)$$



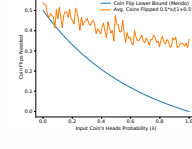
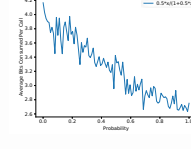
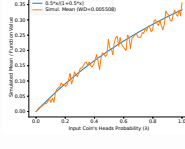
$$(2/3)*x/(1+(2/3)*x)$$



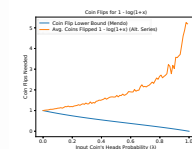
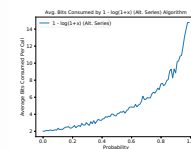
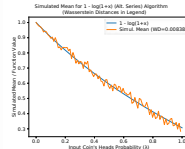
$$(3/2)*x/(1+(3/2)*x)$$



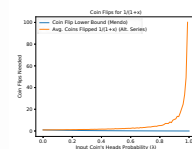
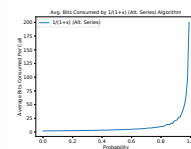
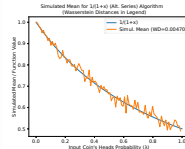
$$0.5*x/(1+0.5*x)$$



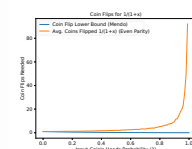
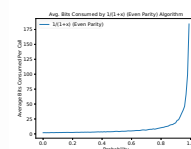
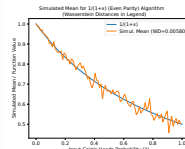
$$1 - \ln(1+x) \text{ (Alt. Series)}$$



$$1/(1+x) \text{ (Alt. Series)}$$



$$1/(1+x) \text{ (Even Parity)}$$



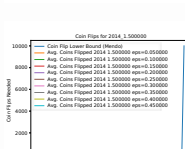
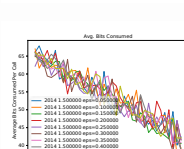
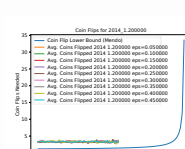
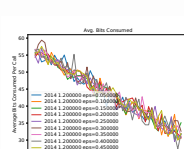
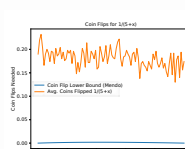
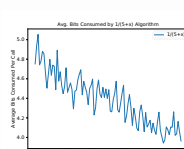
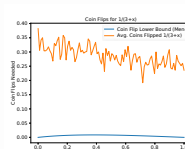
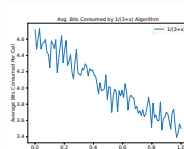
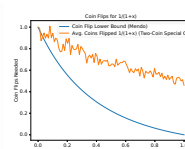
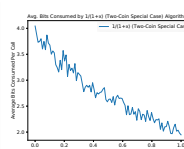


Figure 10 is a line graph titled "Simulated Means (Wasserman Distances in Legend)". The y-axis is labeled "Simulated Mean $r(x|y)$ on y-axis" and ranges from 0.0 to 1.0. The x-axis is labeled "Simulated Mean $r(x|y)$ on x-axis" and also ranges from 0.0 to 1.0. The graph displays 11 lines, each representing a different combination of year and speed. The lines are very close to the diagonal, indicating high accuracy. A legend on the right lists the parameters for each line:

- 2014, 2.000000
- Simul. Mean of 2014, 2.000000 speed=0.050000 (WID=0.0110)
- Simul. Mean of 2014, 2.000000 speed=0.100000 (WID=0.0094)
- Simul. Mean of 2014, 2.000000 speed=0.150000 (WID=0.0105)
- Simul. Mean of 2014, 2.000000 speed=0.200000 (WID=0.0093)
- Simul. Mean of 2014, 2.000000 speed=0.250000 (WID=0.0131)
- Simul. Mean of 2014, 2.000000 speed=0.300000 (WID=0.0094)
- Simul. Mean of 2014, 2.000000 speed=0.350000 (WID=0.0101)
- Simul. Mean of 2014, 2.000000 speed=0.400000 (WID=0.0100)
- Simul. Mean of 2014, 2.000000 speed=0.450000 (WID=0.0092)

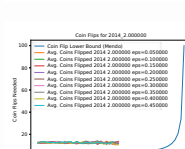
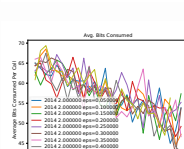
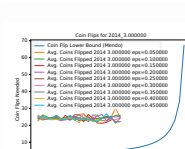
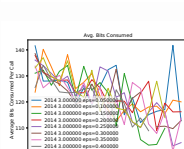
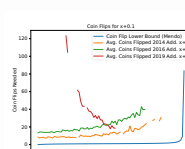
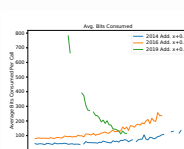
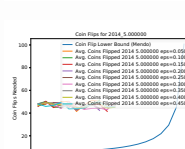
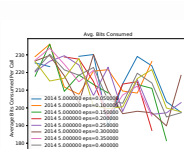
[illegible]

Figure 10: Simulated Means (Weissenstein Hires in Legend). The graph displays the relationship between the simulated mean function value and the simulated mean (Weissenstein Hires) for various simulation means. The Y-axis represents the Simulated Mean/Function Value, ranging from 0.0 to 1.0. The X-axis represents the Simulated Means (Weissenstein Hires in Legend), ranging from 0.000000 to 0.950000. The legend identifies 12 simulation means, each represented by a different colored line. The lines generally show an increasing trend, with the rate of increase slowing down as the simulated mean approaches 1.0.



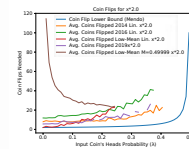
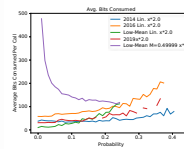
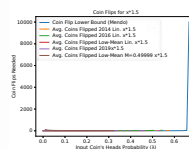
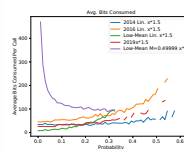
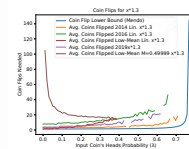
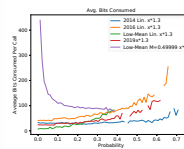
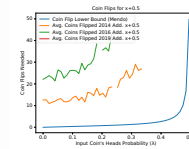
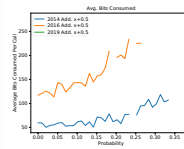
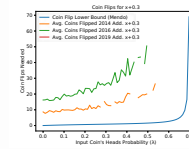
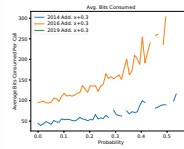
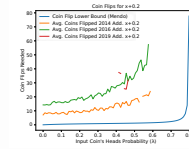
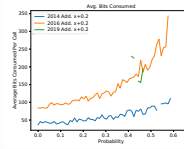
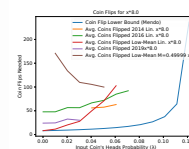
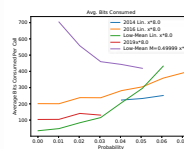
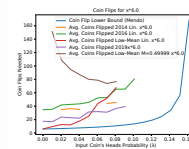
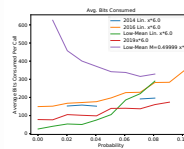
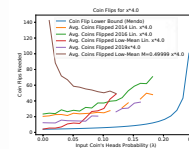
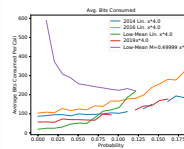


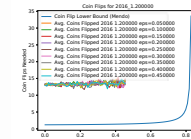
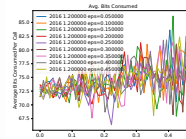
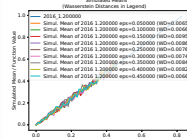
Figure 10 is a line graph titled "Simulated Means (Wasserstein Distances in Legend)". The y-axis is labeled "Simulated Mean (Function Value)" and ranges from 0.0 to 1.0. The x-axis is labeled "Input Color's Heads Probability (X)" and ranges from 0.00 to 0.25. The graph compares six simulated means against a reference line for $\mu = 4.0$. The legend indicates the following data series:

- Simul. Mean of 2014 Lin, $\mu=4.0$ (MSE=0.013948) - Blue line
- Simul. Mean of 2016 Lin, $\mu=4.0$ (MSE=0.010936) - Green line
- Simul. Mean of Low-Mean Lin, $\mu=4.0$ (MSE=0.010092) - Red line
- Simul. Mean of 2016 F4.0, $\mu=4.0$ (MSE=0.013948) - Orange line
- Simul. Mean of Low-Mean F4.0, $\mu=4.0$ (MSE=0.0115) - Grey line

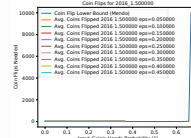
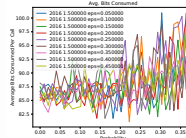
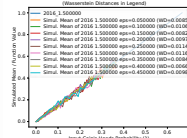
The graph shows that the simulated means for the 2014 and 2016 Lin models are very close to the $\mu = 4.0$ line. The Low-Mean Lin model shows a slight dip around $X = 0.15$. The 2016 F4.0 model is also close to the $\mu = 4.0$ line. The Low-Mean F4.0 model shows a more significant dip around $X = 0.15$, reaching a minimum value of approximately 0.6.



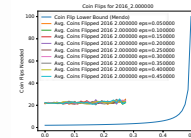
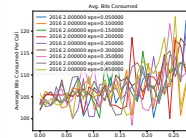
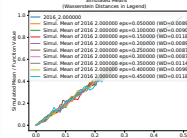
2016 1.200000
eps=0.050000



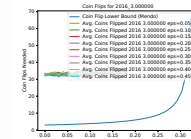
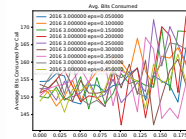
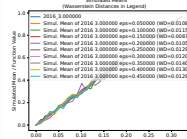
2016 1.500000
eps=0.050000



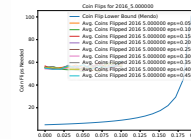
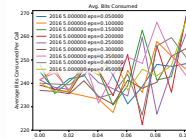
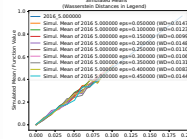
2016 2.000000
eps=0.050000



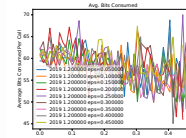
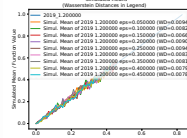
2016 3.000000
eps=0.050000



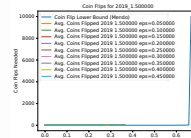
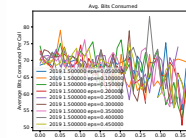
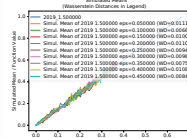
2016 5.000000
eps=0.050000



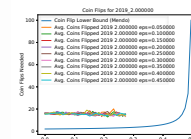
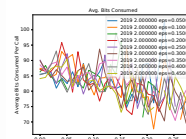
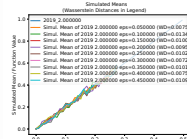
2019 1.200000
eps=0.050000



2019 1.500000
eps=0.050000

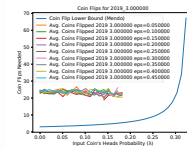


2019 2.000000
eps=0.050000

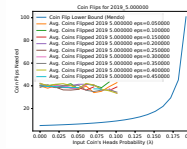


2019 3.000000

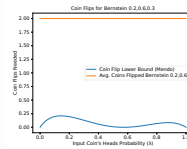
$\epsilon=0.050000$



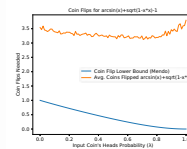
2019 5.000000
 $\epsilon=0.050000$



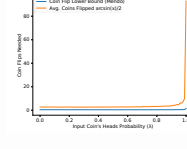
Bernstein
0.2,0.6,0.3



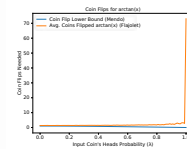
$\arcsin(x)+\sqrt{1-x^2}-1$



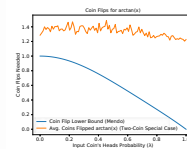
$\arcsin(x)/2$



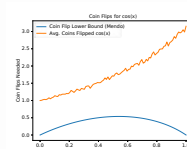
$\arctan(x)$
(Flajolet)



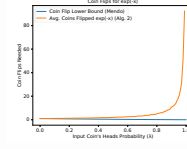
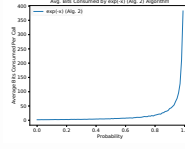
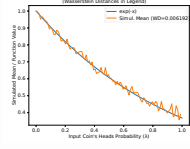
$\arctan(x)$ (Two-Coin Special Case)



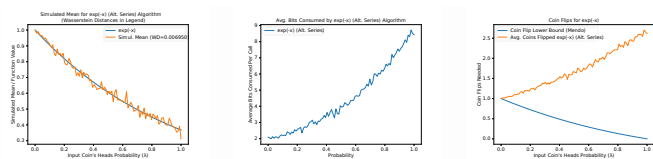
$\cos(x)$



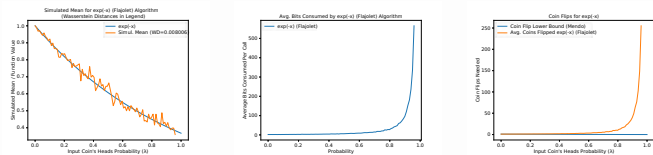
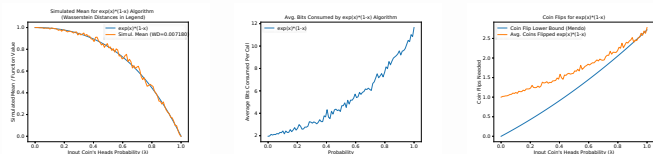
$\exp(-x)$ (Alg. 2)



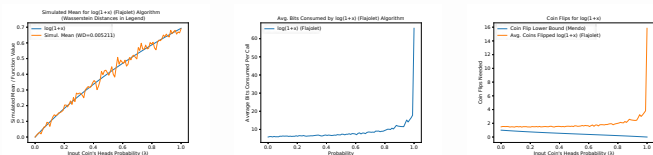
exp(-x) (Alt.
Series)



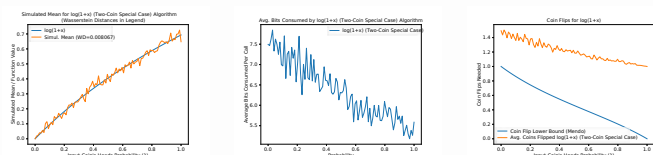
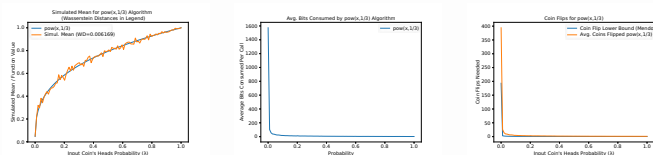
exp(-x) (Flajolet)


$$\exp(x) \cdot (1-x)$$


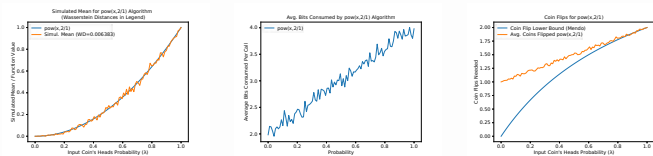
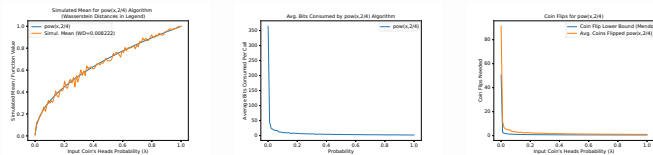
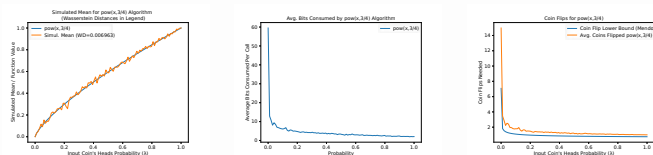
$\ln(1+x)$ (Flajolet)



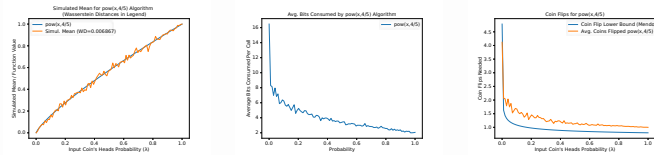
$\ln(1+x)$ (Two-Coin Special Case)

 $\text{pow}(x, 1/3)$ 

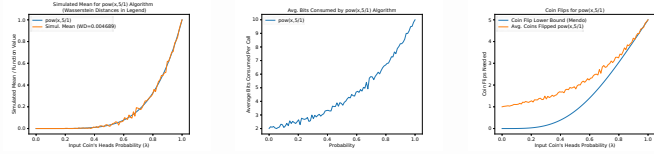
pow(x,2/1)

 $\text{pow}(x, 2/4)$  $\text{pow}(x, 3/4)$ 

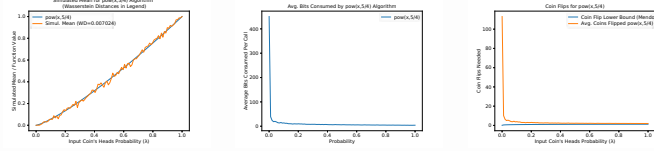
$\text{pow}(x, 4/5)$



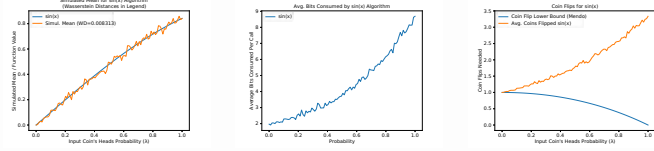
$\text{pow}(x, 5/1)$



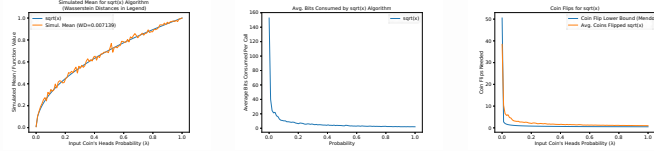
$\text{pow}(x, 5/4)$



$\sin(x)$



$\text{sqrt}(x)$



1. <https://peteroupc.github.io/bernoulli.md>