

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 Algorithms](#)¹” 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—

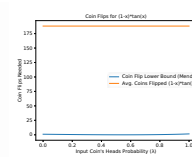
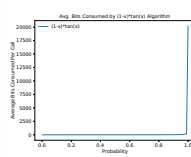
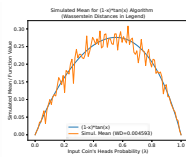
- an ϵ of $1 - (x + c) * 1.001$ was used (or 0.0001 if ϵ would be greater than 1), and
- an ϵ of $(x - c) * 0.9995$ for the subtraction variants.

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.

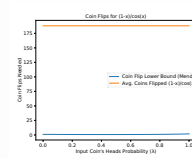
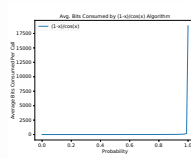
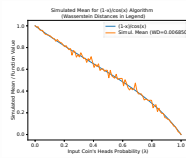
1.1 The Charts

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

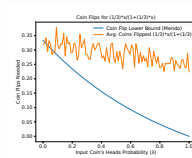
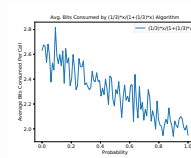
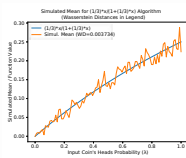
$$(1-x)*\tan(x)$$



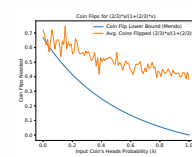
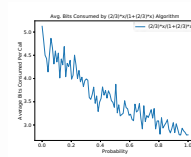
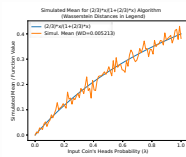
$$(1-x)/\cos(x)$$



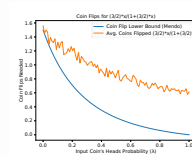
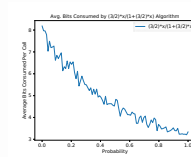
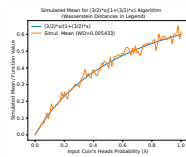
$$(1/3)*x/(1+(1/3)*x)$$



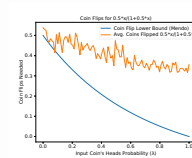
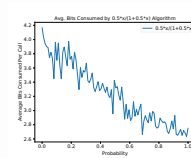
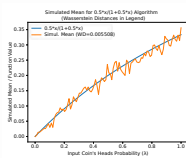
$$(2/3)*x/(1+(2/3)*x)$$



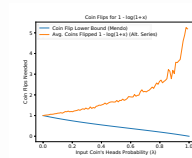
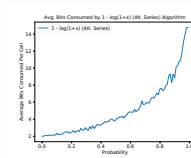
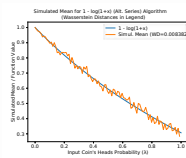
$$(3/2)*x/(1+(3/2)*x)$$



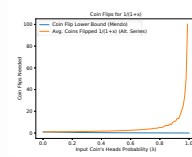
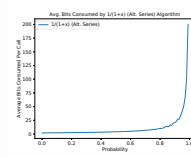
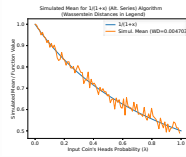
$$0.5*x/(1+0.5*x)$$



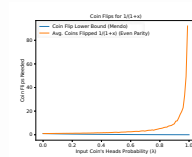
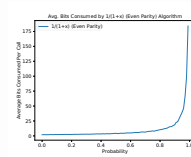
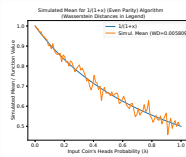
$$1 - \ln(1+x) \text{ (Alt. Series)}$$



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



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



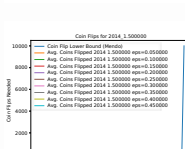
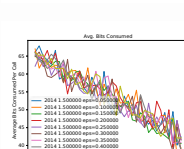
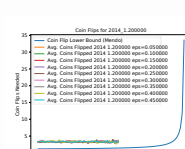
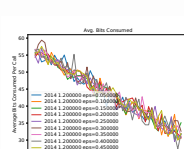
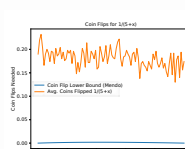
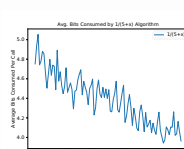
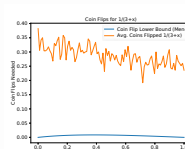
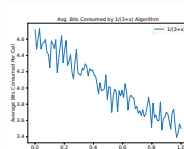
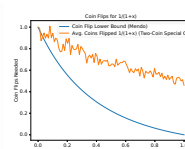
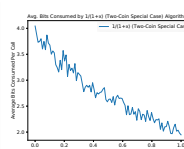


Figure 10 is a line graph titled "Simulated Mean (Waterbody Distances in Legend)" on the x-axis and "Simulated Mean Function Value" on the y-axis. The x-axis ranges from 0.0 to 1.0, and the y-axis ranges from 0.0 to 1.0. The legend lists 11 simulated means for the year 2014, each with a specific waterbody distance and a corresponding R-squared value. The lines show a general upward trend, with some fluctuations, particularly at higher simulated mean values.

Simulated Mean (Waterbody Distances in Legend)	R-squared
2014 2.00000000	0.0110
Simul. Mean of 2014 2.00000000 apex=0.05000000	0.0110
Simul. Mean of 2014 2.00000000 apex=0.12000000	0.0094
Simul. Mean of 2014 2.00000000 apex=0.15000000	0.0105
Simul. Mean of 2014 2.00000000 apex=0.20000000	0.0092
Simul. Mean of 2014 2.00000000 apex=0.25000000	0.0133
Simul. Mean of 2014 2.00000000 apex=0.30000000	0.0094
Simul. Mean of 2014 2.00000000 apex=0.35000000	0.0100
Simul. Mean of 2014 2.00000000 apex=0.40000000	0.0107
Simul. Mean of 2014 2.00000000 apex=0.45000000	0.0092

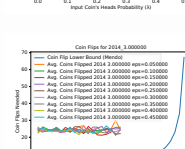
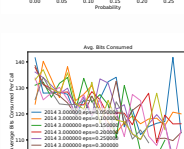
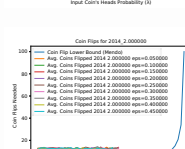
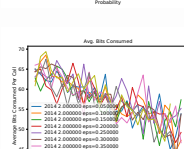
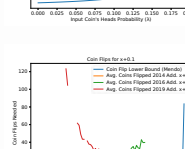
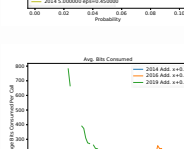
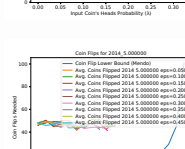
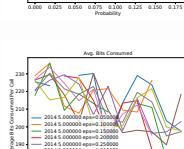
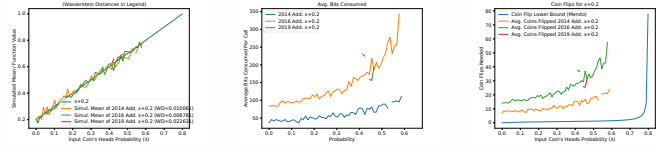


Figure 10 is a line graph titled "Simulated Means (Wasserstein Distances in Legend)". The x-axis is labeled "Input Coin's Heads Probability (X)" and ranges from 0.00 to 0.50. The y-axis is labeled "Simulated Mean Function Value" and ranges from 0.00 to 1.00. The legend lists 12 simulated means for the year 2014, each with a specific Wasserstein distance (W0) and a corresponding Mean Function Value. The lines show that as the input probability increases, the simulated means generally decrease, with some lines showing a sharp drop at 0.50.

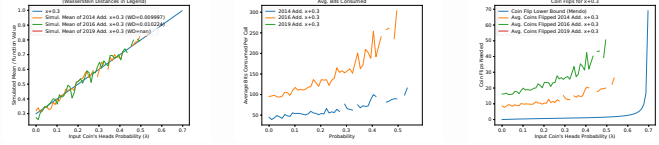
Simulated Mean	Wasserstein Distance (W0)	Mean Function Value
2014, 5.000000		0.000000
Simul. Mean of 2014 5.000000 $\alpha=0.050000$	0.050000	0.0142
Simul. Mean of 2014 5.000000 $\alpha=0.100000$	0.100000	0.0127
Simul. Mean of 2014 5.000000 $\alpha=0.150000$	0.150000	0.0091
Simul. Mean of 2014 5.000000 $\alpha=0.200000$	0.200000	0.0064
Simul. Mean of 2014 5.000000 $\alpha=0.250000$	0.250000	0.0117
Simul. Mean of 2014 5.000000 $\alpha=0.300000$	0.300000	0.0145
Simul. Mean of 2014 5.000000 $\alpha=0.350000$	0.350000	0.0125
Simul. Mean of 2014 5.000000 $\alpha=0.400000$	0.400000	0.0126
Simul. Mean of 2014 5.000000 $\alpha=0.450000$	0.450000	0.0126



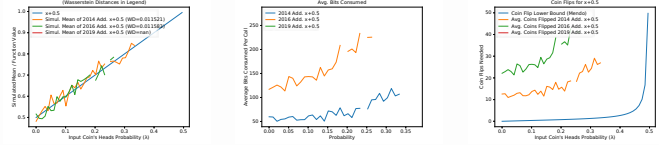
2014 Add. $x+0.2$



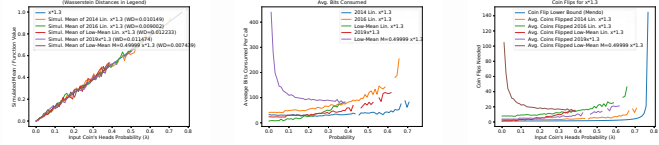
2014 Add. $x+0.3$



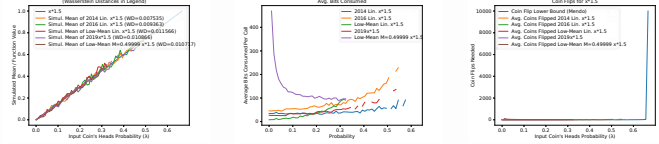
2014 Add. $x+0.5$



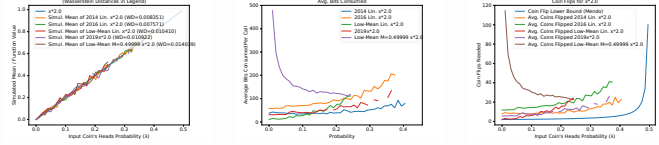
2014 Lin. $x*1.3$



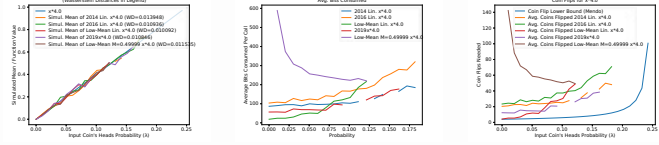
2014 Lin. $x*1.5$



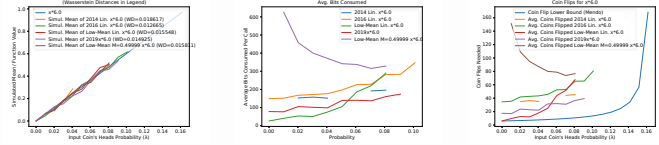
2014 Lin. $x*2.0$



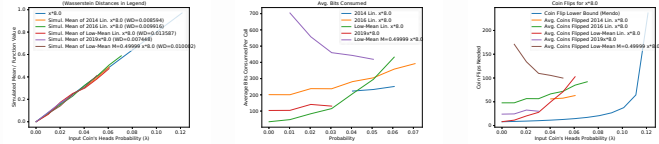
2014 Lin. $x*4.0$



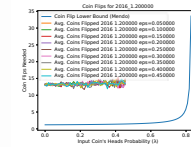
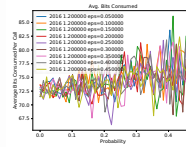
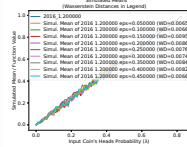
2014 Lin. $x*6.0$



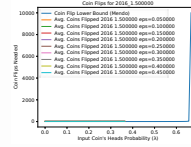
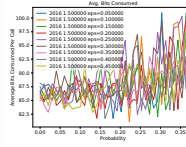
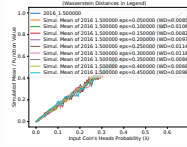
2014 Lin. $x*8.0$



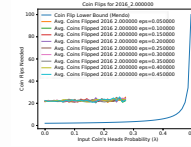
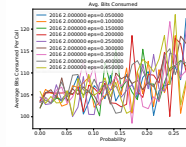
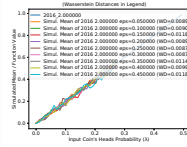
2016 1.200000
eps=0.050000



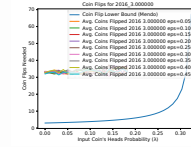
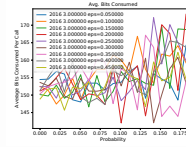
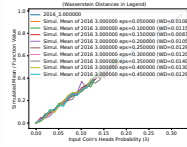
2016 1.500000
eps=0.050000



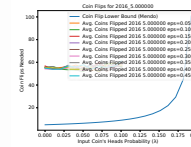
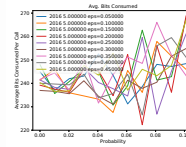
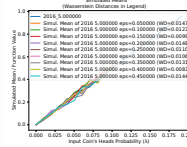
2016 2.000000
eps=0.050000



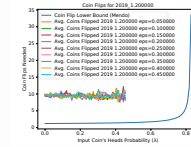
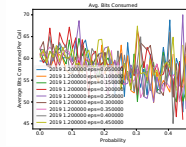
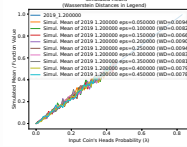
2016 3.000000
eps=0.050000



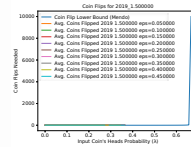
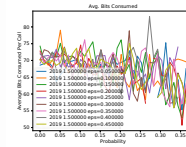
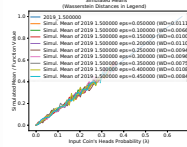
2016 5.000000
eps=0.050000



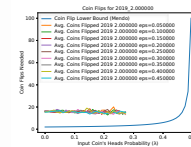
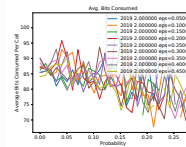
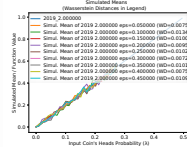
2019 1.200000
eps=0.050000



2019 1.500000
eps=0.050000

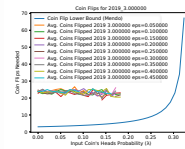


2019 2.000000
eps=0.050000

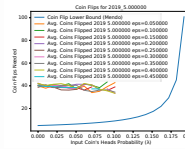


2019 3.000000

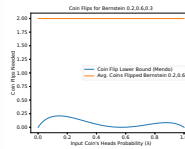
$\epsilon = 0.050000$



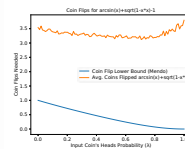
2019 5.000000
 $\epsilon = 0.050000$



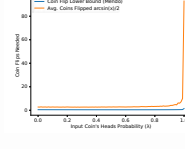
Bernstein
0.2,0.6,0.3



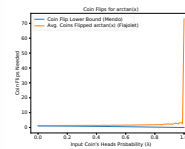
$\arcsin(x) + \sqrt{1 - x^2} - 1$



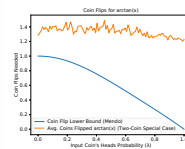
$\arcsin(x)/2$



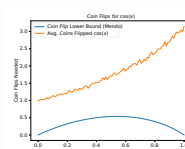
$\arctan(x)$
(Flajolet)



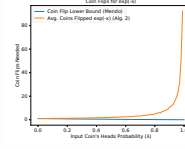
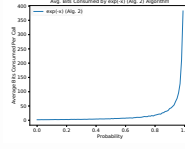
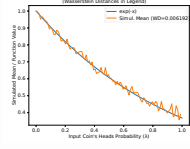
$\arctan(x)$ (Two-Coin Special Case)



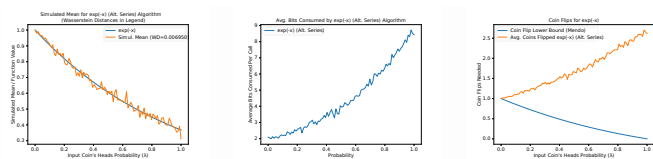
$\cos(x)$



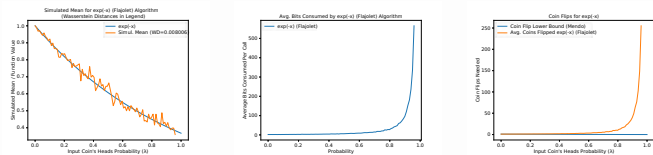
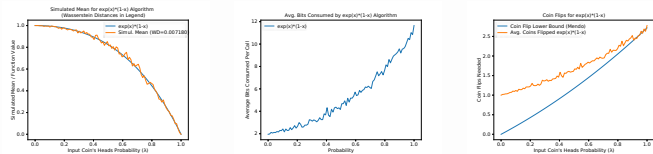
$\exp(-x)$ (Alg. 2)



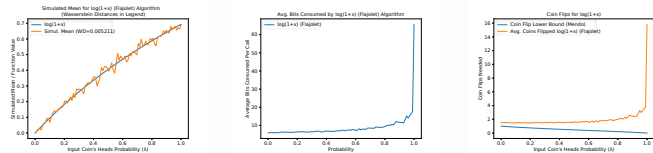
exp(-x) (Alt.
Series)



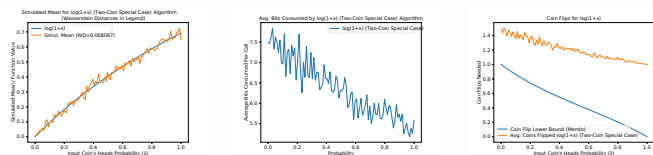
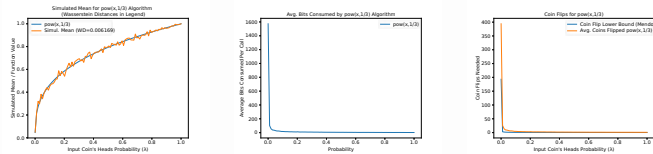
exp(-x) (Flajolet)


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


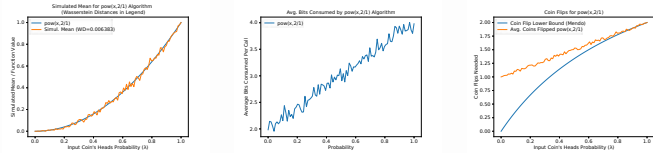
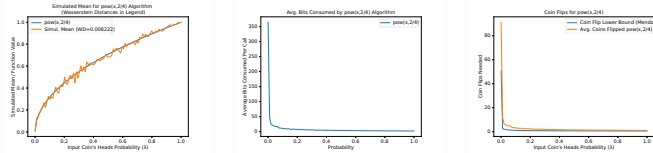
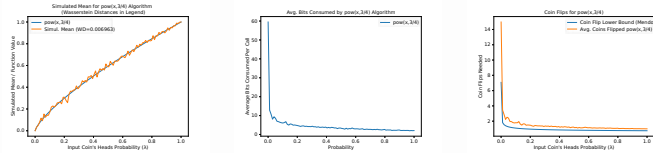
$\ln(1+x)$ (Flajolet)



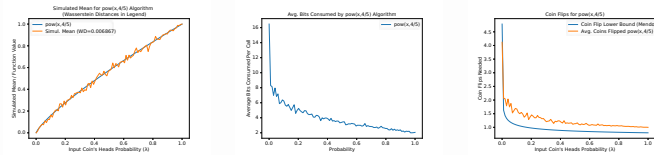
$\ln(1+x)$ (Two-Coin Special Case)

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

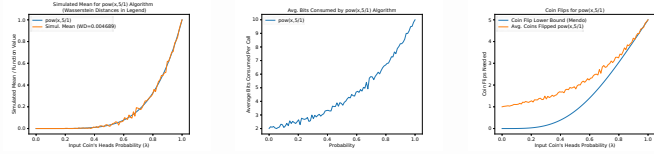
pow(x,2/1)

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

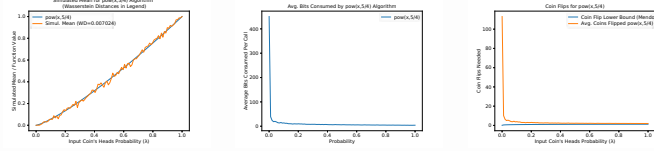
$\text{pow}(x, 4/5)$



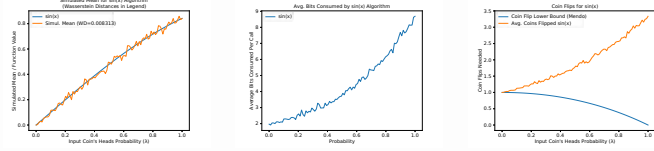
$\text{pow}(x, 5/1)$



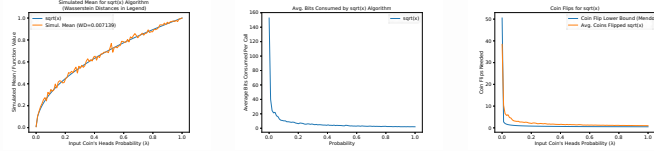
$\text{pow}(x, 5/4)$



$\sin(x)$



$\text{sqrt}(x)$



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