

Correctness and Performance Charts

Peter Occil

1 Correctness and Performance Charts

This version of the document is dated 2025-09-30.

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.

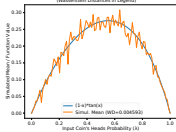
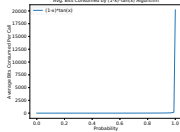
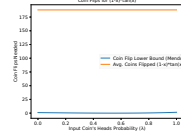
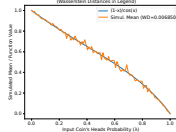
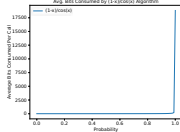
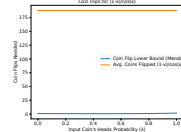
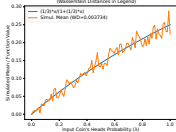
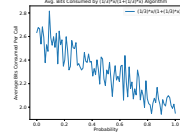
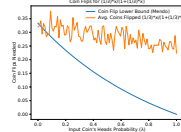
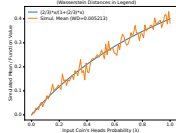
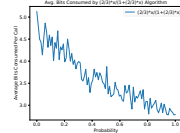
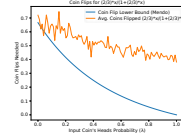
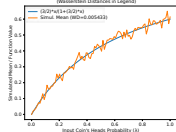
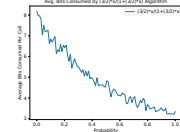
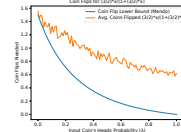
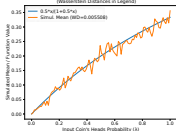
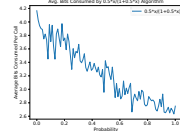
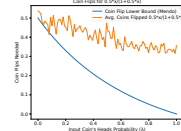
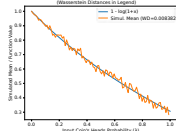
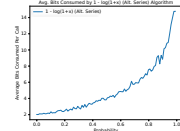
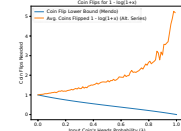
For each algorithm, if a single run was detected to use more than 5000 bits for a given λ , the entire data point for that λ was suppressed in the charts below.

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 specified 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—

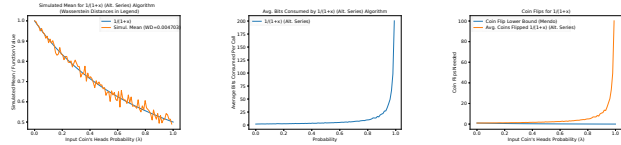
- an ϵ of $1 - (x + c) * 1.001$ was used (or 0.0001 if ϵ would be greater than 1), and
- an ϵ of $(x - c) * 0.9995$ for the subtraction variants.

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.

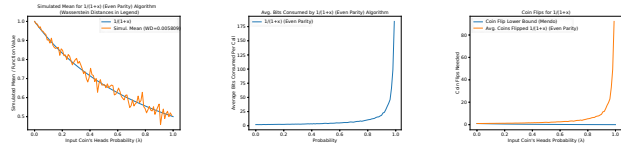
1.1 The Charts

Algorithm	Simulated Mean	Average Bits Consumed	Coin Flips
$(1-x)*\tan(x)$			
$(1-x)/\cos(x)$			
$(1/3)*x/(1+(1/3)*x)$			
$(2/3)*x/(1+(2/3)*x)$			
$(3/2)*x/(1+(3/2)*x)$			
$0.5*x/(1+0.5*x)$			
$1 - \ln(1+x)$ (Alt. Series)			

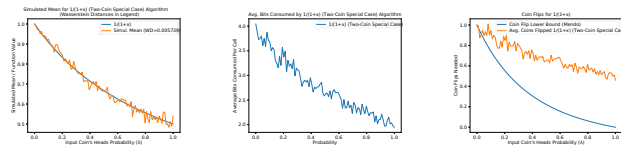
$1/(1+x)$ (Alt.
Series)



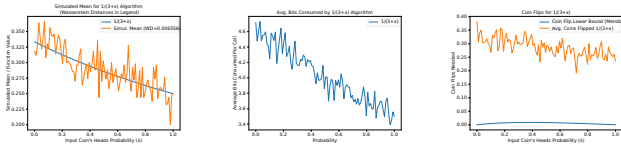
$1/(1+x)$ (Even
Parity)



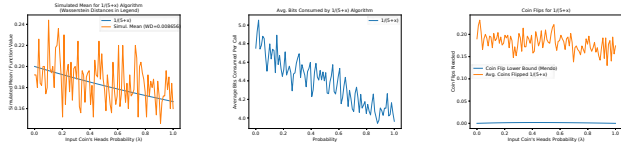
$1/(1+x)$ (Two-
Coin Special
Case)



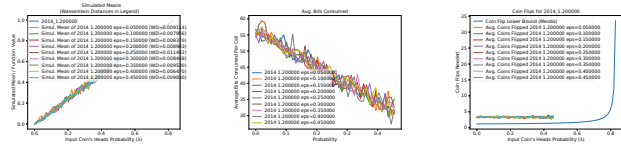
$1/(3+x)$



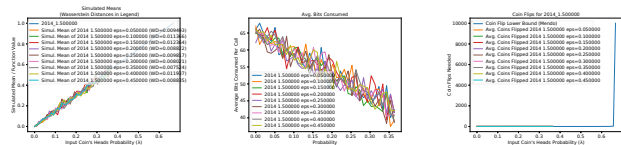
$1/(5+x)$



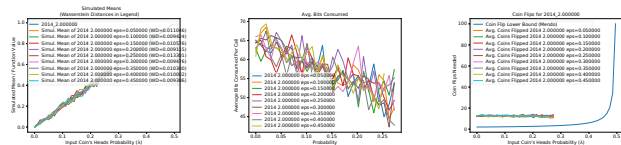
2014 1.200000
eps=0.050000



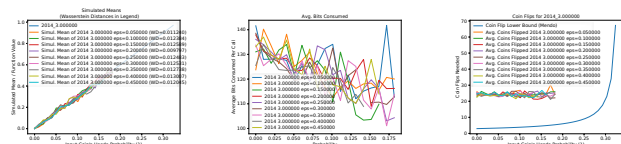
2014 1.500000
eps=0.050000



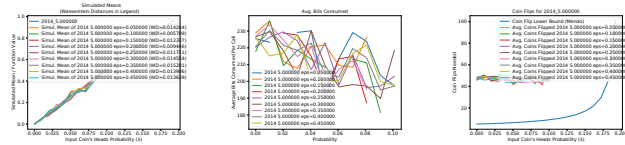
2014 2.000000
eps=0.050000



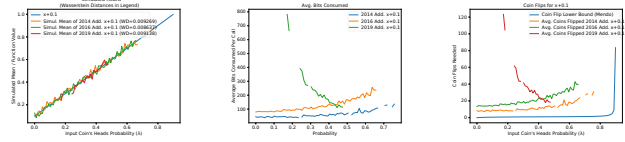
2014 3.000000
eps=0.050000



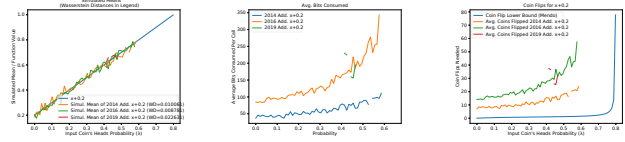
2014 5.000000
eps=0.050000



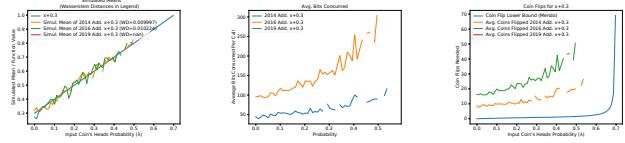
2014 Add. x+0.1



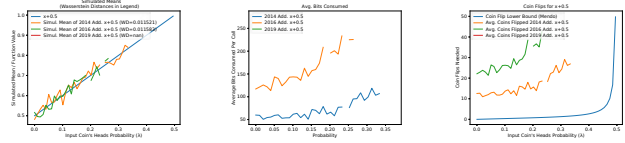
2014 Add. x+0.2



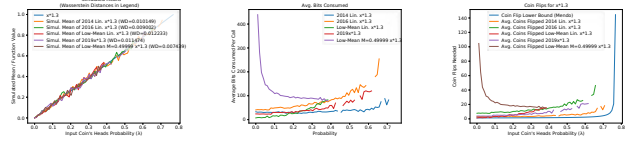
2014 Add. x+0.3



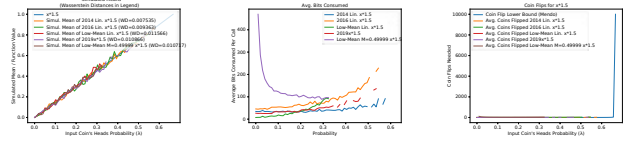
2014 Add. x+0.5



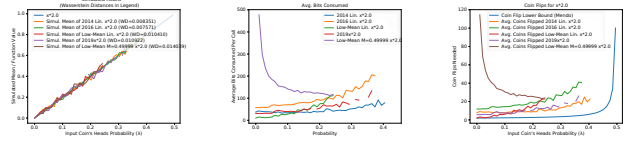
2014 Lin. x*1.3



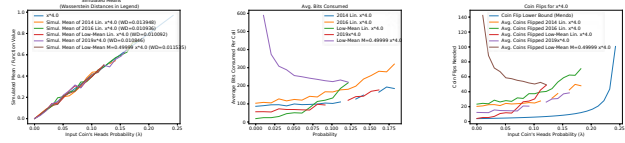
2014 Lin. x*1.5



2014 Lin. x*2.0



2014 Lin. x*4.0



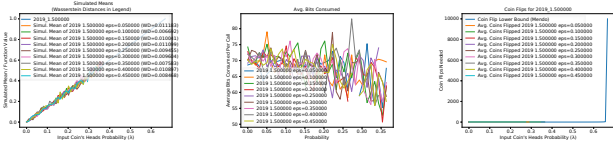
[illegible][illegible][illegible]

Figure 1 consists of three subplots. Subplot (a) is a scatter plot titled 'Scatterplot Results' showing 'True Rank (x-axis)' vs. 'Estimated Rank (y-axis)'. The legend lists 10 data series for different years and iterations. Subplot (b) is a line plot titled 'Avg. Rank Comparison' showing 'Avg. Rank' vs. 'Iteration' for the same 10 series. Subplot (c) is a line plot titled 'Cost Ratio for 2016 & 201600000' showing 'Cost Ratio' vs. 'Iteration' for the same 10 series.

[illegible]

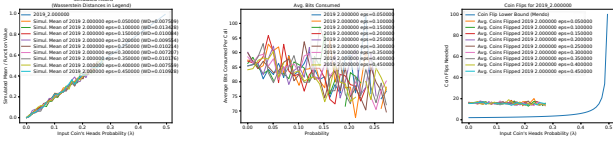
Figure 1 consists of three subplots. The left subplot is an ROC curve for the 2014-2015 dataset, showing the True Positive Rate (TPR) on the y-axis (0.0 to 1.0) versus the False Positive Rate (FPR) on the x-axis (0.00 to 0.90). The proposed model (red line) is the topmost curve, indicating the best performance. The middle subplot is an ROC curve for the 2015-2016 dataset, showing the True Positive Rate (TPR) on the y-axis (0.0 to 1.0) versus the False Positive Rate (FPR) on the x-axis (0.00 to 0.90). The proposed model (red line) is the topmost curve, indicating the best performance. The right subplot is a Cost Ratio plot for the 2014-2015 dataset, showing the Cost Ratio on the y-axis (0.00 to 1.00) versus the Cost Ratio on the x-axis (0.00 to 0.90). The proposed model (red line) shows a sharp increase in cost ratio as the cost ratio increases, while other models remain relatively flat.

eps=0.050000



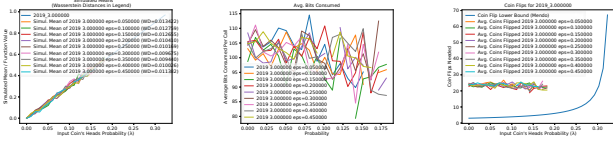
2019 2.000000

eps=0.050000



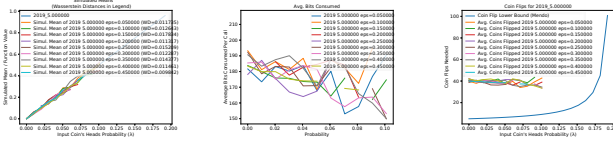
2019 3.000000

eps=0.050000



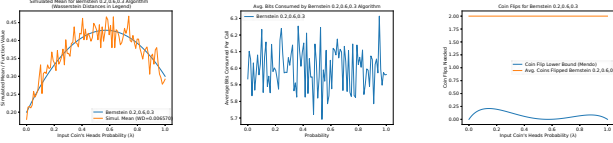
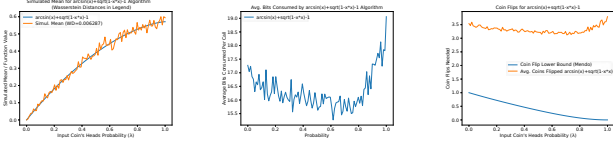
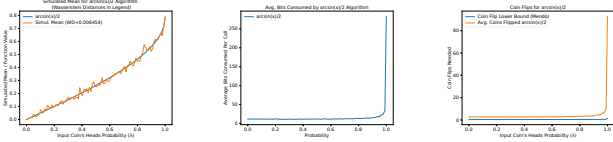
2019 5.000000

eps=0.050000

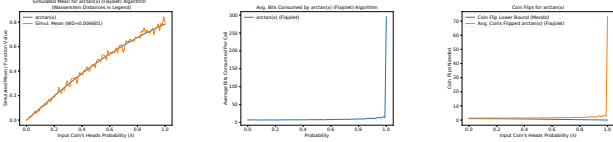


Bernstein

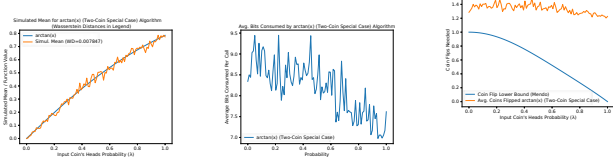
0.2,0.6,0.3


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

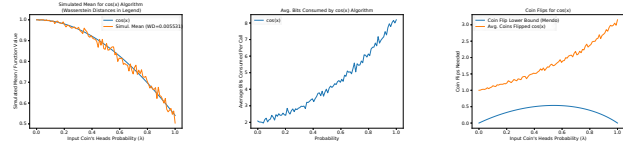
(Flajolet)



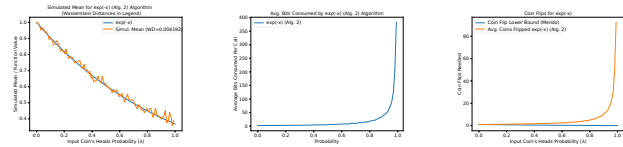
arctan(x) (Two-Coin Special Case)



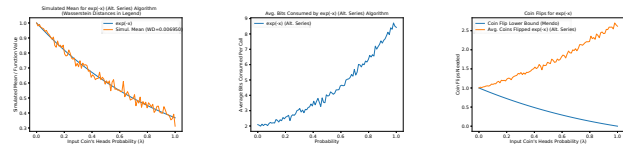
$\cos(x)$



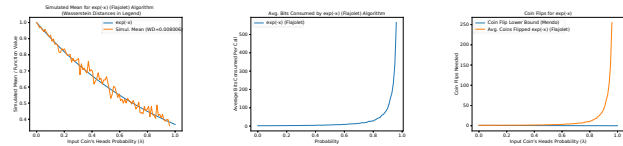
$\exp(-x)$ (Alg. 2)



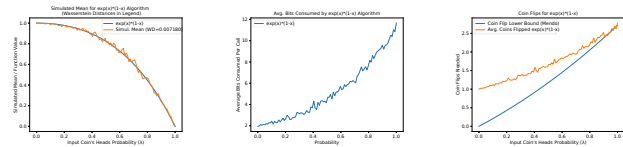
$\exp(-x)$ (Alt. Series)



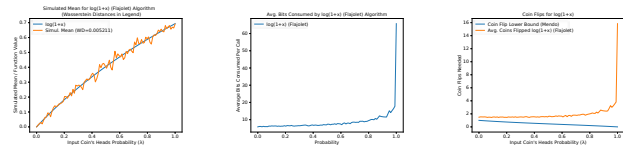
$\exp(-x)$ (Flajolet)



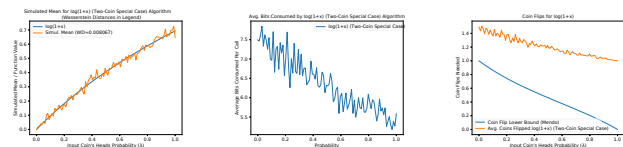
$\exp(x)*(1-x)$



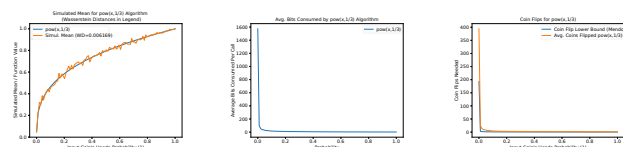
$\ln(1+x)$ (Flajolet)



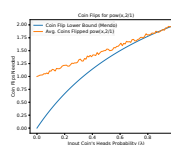
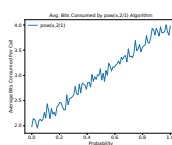
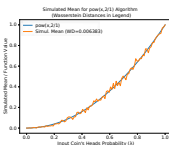
$\ln(1+x)$ (Two-Coin Special Case)



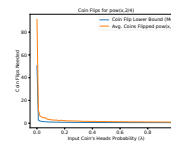
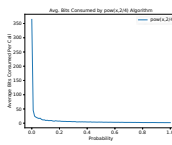
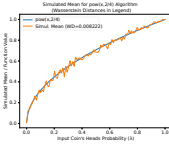
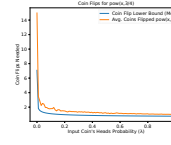
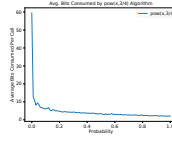
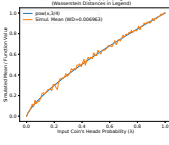
$\text{pow}(x, 1/3)$



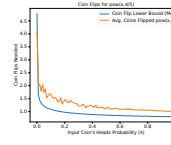
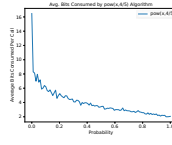
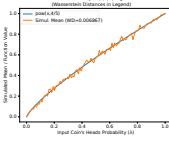
pow(x,2/1)



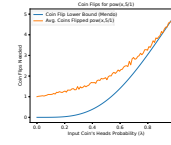
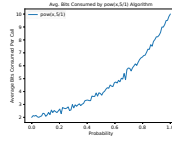
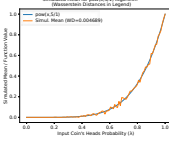
pow(x,2/4)

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

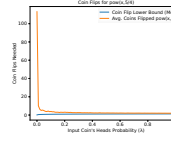
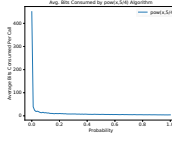
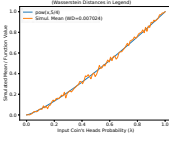
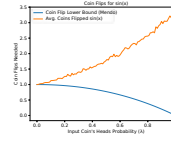
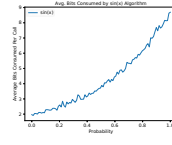
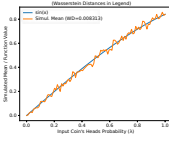
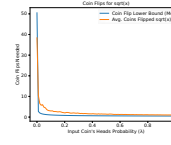
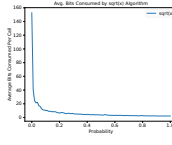
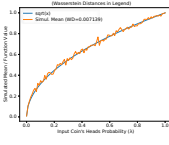
pow(x,4/5)



pow(x,5/1)



pow(x,5/4)

 $\sin(x)$  \sqrt{x} 

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