

Correctness and Performance Charts

This version of the document is dated 2022-11-07.

The following charts show the correctness of many of the algorithms in "**Bernoulli Factory Algorithms**" 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—

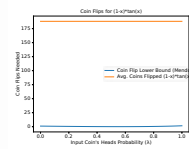
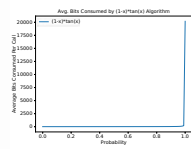
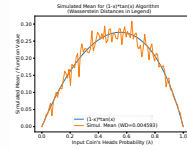
- an ϵ of $1 - (x + c) * 1.001$ was used (or 0.0001 if ϵ would be greater than 1), and
- an ϵ of $(x - c) * 0.9995$ for the subtraction variants.

Points with invalid ϵ values were suppressed. For the low-mean algorithm, an m of $\max(0.49999, x*c*1.02)$ was used unless noted otherwise.

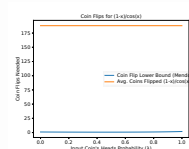
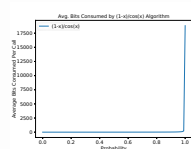
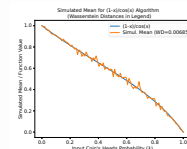
0.1 The Charts

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

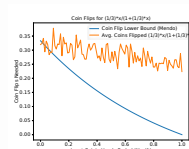
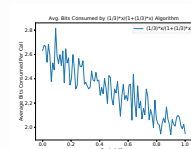
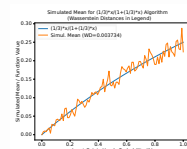
$$(1-x)*\tan(x)$$



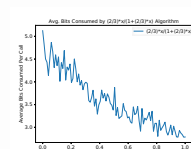
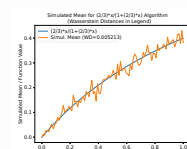
$$(1-x)/\cos(x)$$



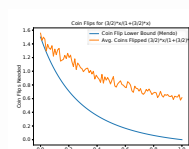
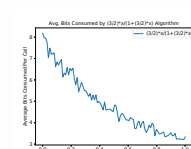
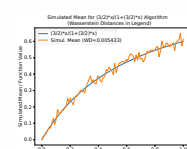
$$(1/3)*x/(1+(1/3)*x)$$



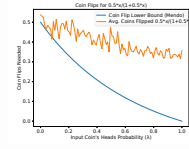
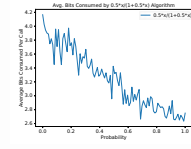
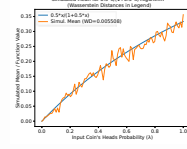
$$(2/3)*x/(1+(2/3)*x)$$



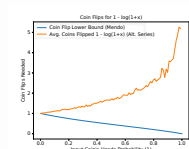
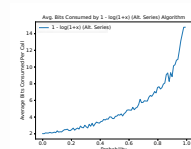
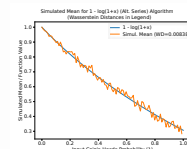
$$(3/2)*x/(1+(3/2)*x)$$



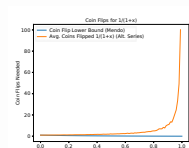
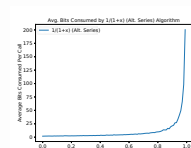
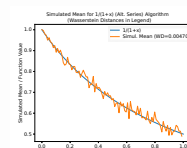
$$0.5*x/(1+0.5*x)$$



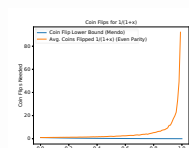
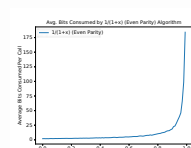
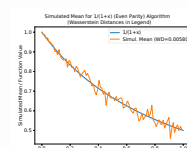
$$1 - \ln(1+x) \text{ (Alt. Series)}$$



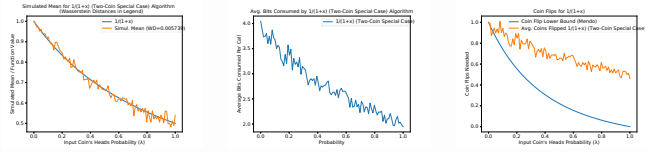
$$1/(1+x) \text{ (Alt. Series)}$$



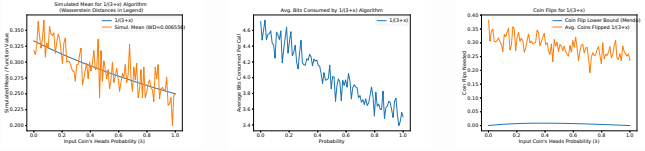
$$1/(1+x) \text{ (Even Parity)}$$



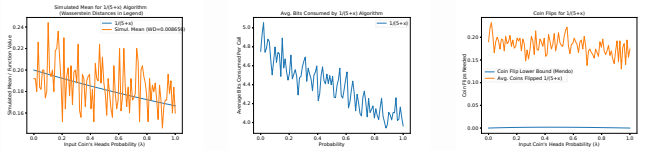
1/(1+x) (Two-Coin Special Case)



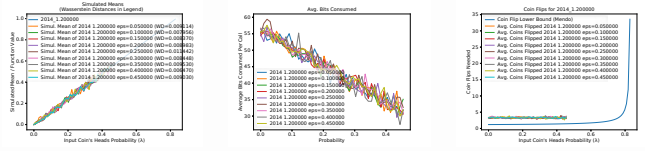
1/(3+x)



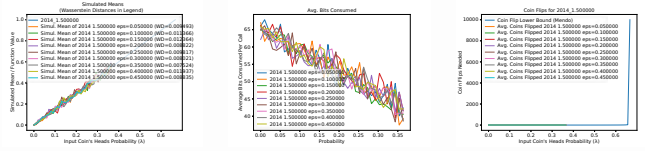
1/(5+x)



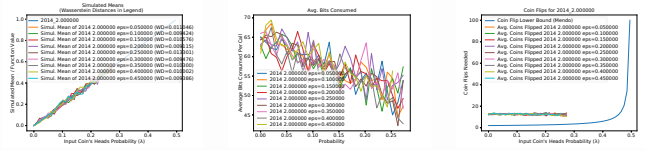
2014 1.200000
eps=0.050000



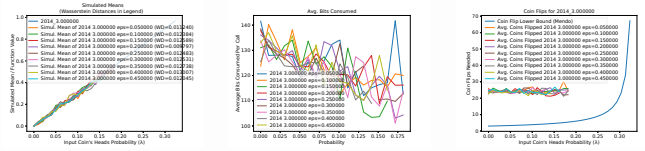
2014 1.500000
eps=0.050000



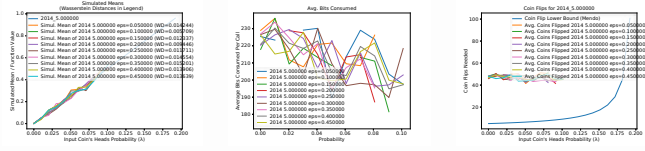
2014 2.000000
eps=0.050000



2014 3.000000
eps=0.050000



2014 5.000000
eps=0.050000



2014 Add. x+0.1

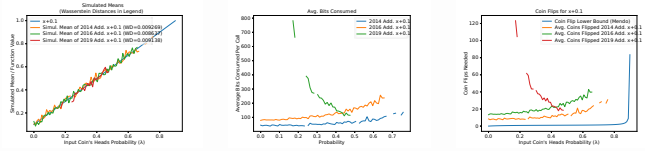


Figure 1 consists of three subplots. The left plot, titled 'Disaggregated Rain (Observations minus Forecast)', shows True Rain (mm) on the y-axis (0.0 to 1.0) versus Predicted Rain (mm) on the x-axis (0.0 to 1.0). It includes a blue diagonal line for perfect agreement and data points for 2014 (green), 2015 (orange), and 2016 (red). The middle plot, titled 'Avg. RMSE (mm) vs. Normalized Error', shows Average RMSE (mm) on the y-axis (0.0 to 0.05) versus Normalized Error on the x-axis (0.0 to 0.6). It includes data for 2014 (green), 2015 (orange), and 2016 (red). The right plot, titled 'RMSE (mm) vs. Normalized Error for 2017-2018', shows RMSE (mm) on the y-axis (0 to 80) versus Normalized Error on the x-axis (0.0 to 0.7). It includes data for 2017 (green), 2018 (orange), and 2019 (red).

Figure 1 consists of three subplots illustrating the performance of the proposed algorithm. The left subplot shows the 'Normalized Input Feature Size' (Y-axis, 0.0 to 1.0) versus 'Input Core's Needs Probability (ξ)' (X-axis, 0.0 to 0.7). It compares the performance of the proposed algorithm (blue line) with the baseline (orange line) for different data sets (2014, 2015, 2016) and aggregation levels ($a=0.5$ and $a=1$). The middle subplot shows the 'Average CPU's Consumed per User' (Y-axis, 50 to 200) versus 'Probability' (X-axis, 0.0 to 0.5). It compares the performance of the proposed algorithm (blue line) with the baseline (orange line) for different data sets (2014, 2015, 2016) and aggregation levels ($a=0.5$ and $a=1$). The right subplot shows the 'CPU's Consumed' (Y-axis, 0 to 100) versus 'Input Core's Needs Probability (ξ)' (X-axis, 0.0 to 1.0). It compares the performance of the proposed algorithm (blue line) with the baseline (orange line) for different data sets (2014, 2015, 2016) and aggregation levels ($a=0.5$ and $a=1$).

Figure 1 consists of three plots showing the performance of different methods (Cox, F1, Lasso, Ridge, and Reg) across varying input class-ness probabilities (0.0 to 0.5).

- Left Plot: Standard Mean F1 score**
 - Y-axis: Standard Mean F1 score (0.0 to 1.0)
 - X-axis: Input class-ness probability (0.0 to 0.5)
 - Legend:
 - Blue line: Cox
 - Orange line: F1
 - Green line: Lasso
 - Red line: Ridge
 - Pink line: Reg
 - Annotations:
 - at $\alpha = 0.5$: Cox (F1) = 0.95, Lasso = 0.95, Ridge = 0.95, Reg = 0.95
 - at $\alpha = 0.0$: Cox (F1) = 0.85, Lasso = 0.85, Ridge = 0.85, Reg = 0.85
- Middle Plot: Average BC Coefficient**
 - Y-axis: Average BC Coefficient (0.0 to 0.5)
 - X-axis: Input class-ness probability (0.0 to 0.5)
 - Legend:
 - Blue line: Cox
 - Orange line: F1
 - Green line: Lasso
 - Red line: Ridge
 - Pink line: Reg
 - Annotations:
 - at $\alpha = 0.5$: Cox (F1) = 0.45, Lasso = 0.45, Ridge = 0.45, Reg = 0.45
 - at $\alpha = 0.0$: Cox (F1) = 0.35, Lasso = 0.35, Ridge = 0.35, Reg = 0.35
- Right Plot: Cost Ratio**
 - Y-axis: Cost Ratio (0 to 60)
 - X-axis: Input class-ness probability (0.0 to 0.5)
 - Legend:
 - Blue line: Cox
 - Orange line: F1
 - Green line: Lasso
 - Red line: Ridge
 - Pink line: Reg
 - Annotations:
 - at $\alpha = 0.5$: Cox (F1) = 55, Lasso = 55, Ridge = 55, Reg = 55
 - at $\alpha = 0.0$: Cox (F1) = 10, Lasso = 10, Ridge = 10, Reg = 10

Figure 1 consists of four subplots arranged in a 2x2 grid. The top-left plot shows the Scaled Mean Squared Error (MSE) for 1000 trials on the x-axis (ranging from 0.0 to 0.6) versus the Scaled Mean Squared Error (MSE) for 1000 trials on the y-axis (ranging from 0.0 to 0.6). It includes a legend for 'Proposed' (red), 'ADMM' (blue), 'ADMM-VR' (green), 'ADMM-VR-1' (cyan), 'ADMM-VR-2' (magenta), 'ADMM-VR-3' (yellow), 'ADMM-VR-4' (black), 'ADMM-VR-5' (brown), 'ADMM-VR-6' (pink), and 'ADMM-VR-7' (grey). The top-right plot shows the Average MSE (log scale) on the y-axis (ranging from 10⁻¹ to 10¹) versus Epochs on the x-axis (ranging from 0.0 to 2.7). The bottom-left plot shows the Average MSE (log scale) on the y-axis (ranging from 10⁻¹ to 10¹) versus Epochs on the x-axis (ranging from 0.0 to 2.7). The bottom-right plot shows the Average MSE (log scale) on the y-axis (ranging from 10⁻¹ to 10¹) versus Epochs on the x-axis (ranging from 0.0 to 2.7). All plots show that the proposed algorithm (red line) converges much faster than the other algorithms.

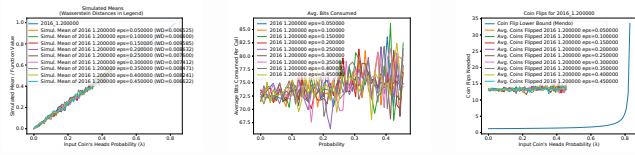
[illegible][illegible]

Figure 1 consists of three subplots. Subplot (a) is an ROC curve for the proposed model, 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 (0.0 to 0.25). The curve is a solid black line, and the area under the curve is shaded in light blue. Subplot (b) shows the Average ROC curves for the proposed model, with the y-axis labeled 'Average ROC' (0.0 to 1.0) and the x-axis labeled 'Probability' (0.00 to 0.15). It includes a legend for 'Avg. ROC Curves' with entries for '2018 Lm, $\eta = 0.0$ ', '2018 Lm, $\eta = 0.0$ ', '2018 Lm, $\eta = 0.0$ ', '2018 Lm, $\eta = 0.0$ ', and '2018 Lm, $\eta = 0.0$ '. Subplot (c) shows ROC curves for the proposed model with different input data, with the y-axis labeled 'ROC Curve' (0 to 140) and the x-axis labeled 'Input Data's Feature Probability (x)' (0.00 to 0.25). It includes a legend for 'ROC Curves for $\eta = 0$ ' with entries for '2018 Lm, $\eta = 0.0$ ', '2018 Lm, $\eta = 0.0$ ', '2018 Lm, $\eta = 0.0$ ', '2018 Lm, $\eta = 0.0$ ', and '2018 Lm, $\eta = 0.0$ '.

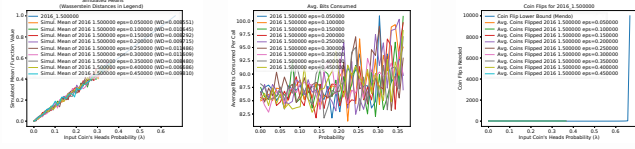
Figure 1 consists of three subplots. Subplot (a) is an ROC curve showing the True Positive Rate (Y-axis) versus the False Positive Rate (X-axis). The legend indicates the following AUC values: $\mu=0.6$ (0.61), $\mu=0.7$ (0.65), $\mu=0.8$ (0.69), $\mu=0.9$ (0.73), $\mu=0.95$ (0.77), $\mu=0.99$ (0.81), $\mu=0.999$ (0.85), $\mu=0.9999$ (0.89), and $\mu=0.99999$ (0.93). Subplot (b) shows the Average F1 score (Y-axis) versus Probability (X-axis). The legend indicates the following F1 scores: $\mu=0.6$ (0.61), $\mu=0.7$ (0.65), $\mu=0.8$ (0.69), $\mu=0.9$ (0.73), $\mu=0.95$ (0.77), $\mu=0.99$ (0.81), $\mu=0.999$ (0.85), $\mu=0.9999$ (0.89), and $\mu=0.99999$ (0.93). Subplot (c) shows the Gain Ratio (Y-axis) versus Input Class Means Probability (X-axis). The legend indicates the following Gain Ratios: $\mu=0.6$ (0.61), $\mu=0.7$ (0.65), $\mu=0.8$ (0.69), $\mu=0.9$ (0.73), $\mu=0.95$ (0.77), $\mu=0.99$ (0.81), $\mu=0.999$ (0.85), $\mu=0.9999$ (0.89), and $\mu=0.99999$ (0.93).

Figure 1 consists of two subplots. Subplot (a) is a Receiver Operating Characteristic (ROC) curve showing the Detection Rate (True Positive Rate) on the y-axis (ranging from 0.00 to 1.00) versus the Input SNR (Power Ratio) on the x-axis (ranging from 0.00 to 1.0). The legend indicates: 'Proposed' (red line), 'Cramér-Rao Bound' (blue line), 'ML' (green line), 'ML + 10% SNR Error' (orange line), 'ML + 20% SNR Error' (purple line), 'ML + 30% SNR Error' (brown line), 'ML + 40% SNR Error' (pink line), 'ML + 50% SNR Error' (grey line), 'ML + 60% SNR Error' (light blue line), 'ML + 70% SNR Error' (light green line), 'ML + 80% SNR Error' (light purple line), 'ML + 90% SNR Error' (light brown line), 'ML + 100% SNR Error' (light pink line), 'ML + 110% SNR Error' (light grey line), 'ML + 120% SNR Error' (light blue line), 'ML + 130% SNR Error' (light green line), 'ML + 140% SNR Error' (light purple line), 'ML + 150% SNR Error' (light brown line), 'ML + 160% SNR Error' (light pink line), 'ML + 170% SNR Error' (light grey line), 'ML + 180% SNR Error' (light blue line), 'ML + 190% SNR Error' (light green line), 'ML + 200% SNR Error' (light purple line). The proposed algorithm (red line) closely follows the Cramér-Rao Bound (blue line). Subplot (b) shows the Average Bit Error Rate (BER) on the y-axis (ranging from 0.00 to 0.08) versus Probability on the x-axis (ranging from 0.00 to 0.10). The legend indicates: 'Cramér-Rao Bound' (blue line), 'Proposed' (red line), 'ML' (green line), 'ML + 10% SNR Error' (orange line), 'ML + 20% SNR Error' (purple line), 'ML + 30% SNR Error' (brown line), 'ML + 40% SNR Error' (pink line), 'ML + 50% SNR Error' (grey line), 'ML + 60% SNR Error' (light blue line), 'ML + 70% SNR Error' (light green line), 'ML + 80% SNR Error' (light purple line), 'ML + 90% SNR Error' (light brown line), 'ML + 100% SNR Error' (light pink line), 'ML + 110% SNR Error' (light grey line), 'ML + 120% SNR Error' (light blue line), 'ML + 130% SNR Error' (light green line), 'ML + 140% SNR Error' (light purple line), 'ML + 150% SNR Error' (light brown line), 'ML + 160% SNR Error' (light pink line), 'ML + 170% SNR Error' (light grey line), 'ML + 180% SNR Error' (light blue line), 'ML + 190% SNR Error' (light green line), 'ML + 200% SNR Error' (light purple line). The proposed algorithm (red line) shows a lower BER than the ML algorithm (green line) and its variants with SNR estimation errors.

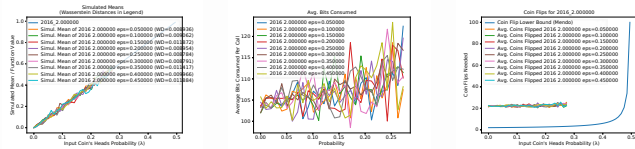
2016 1.200000
eps=0.050000



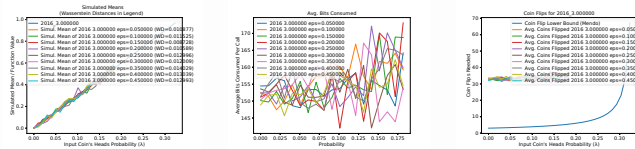
2016 1.500000
eps=0.050000



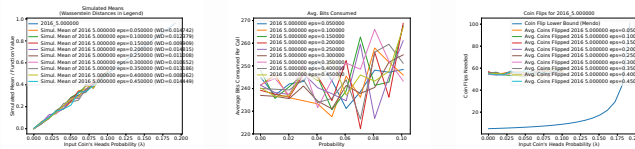
2016 2.000000
eps=0.050000



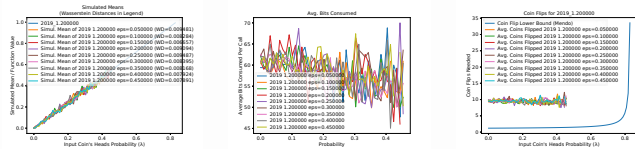
2016 3.000000
eps=0.050000



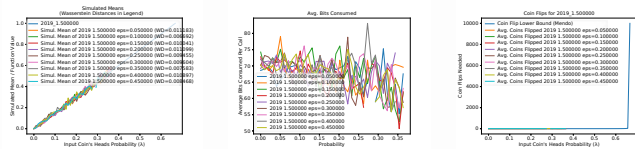
2016 5.000000
eps=0.050000



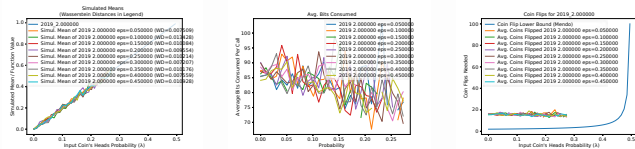
2019 1.200000
eps=0.050000



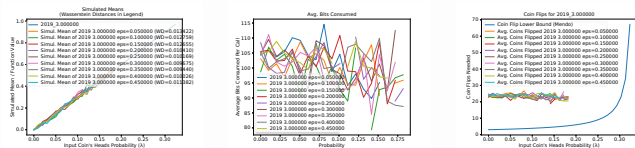
2019 1.500000
eps=0.050000



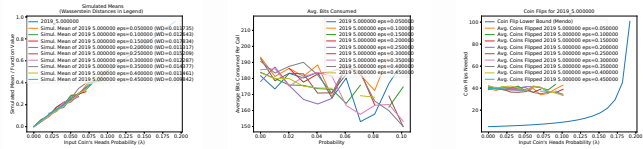
2019 2.000000
eps=0.050000



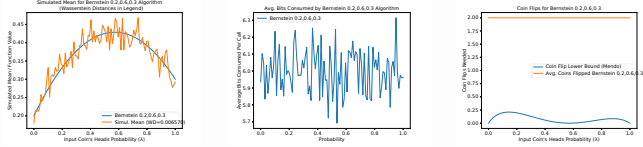
2019 3.000000
eps=0.050000



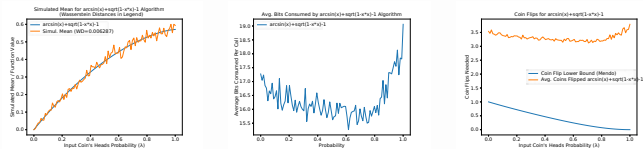
2019 5.000000
eps=0.050000



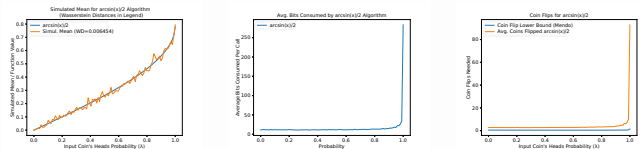
Bernstein
0.2,0.6,0.3



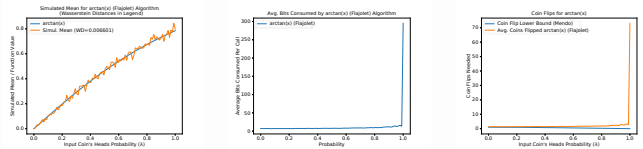
$\arcsin(x) + \sqrt{1-x^2} - 1$



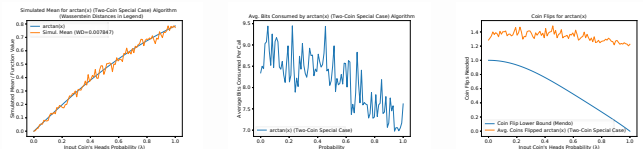
$\arcsin(x)/2$



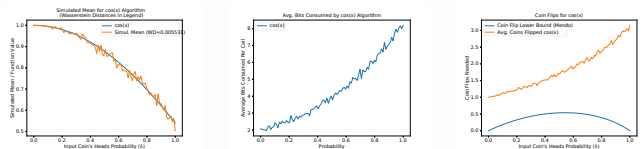
$\arctan(x)$
(Flajolet)



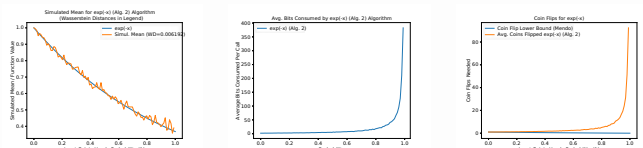
$\arctan(x)$ (Two-Coin Special Case)



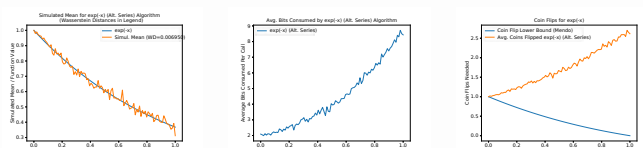
$\cos(x)$



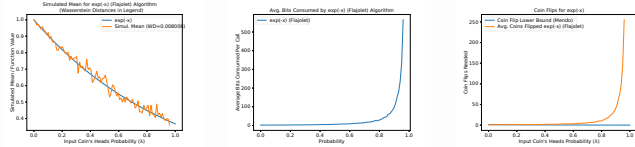
$\exp(-x)$ (Alg. 2)



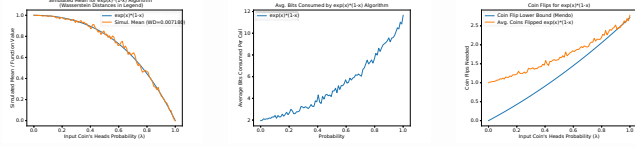
$\exp(-x)$ (Alt. Series)



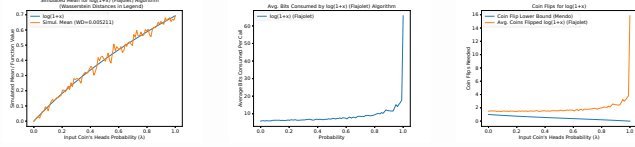
$\exp(-x)$ (Flajolet)



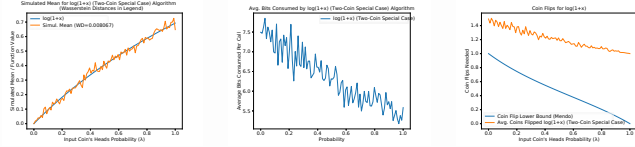
$\exp(x)*(1-x)$



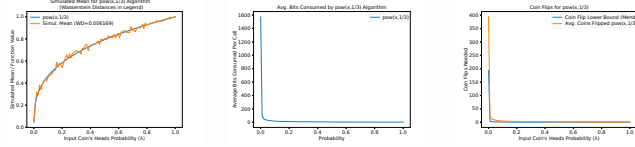
$\ln(1+x)$
(Flajolet)



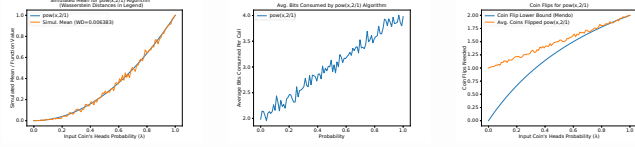
$\ln(1+x)$ (Two-Coin Special Case)



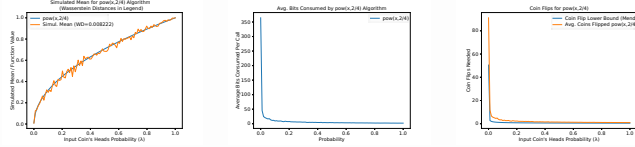
$\text{pow}(x, 1/3)$



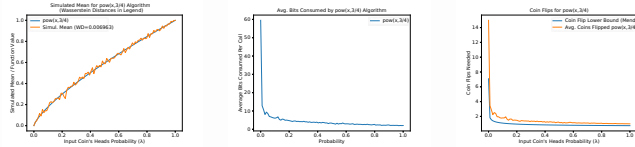
$\text{pow}(x, 2/1)$



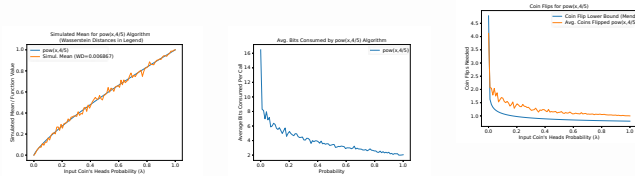
$\text{pow}(x, 2/4)$



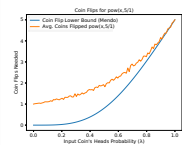
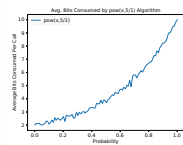
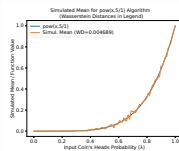
$\text{pow}(x, 3/4)$



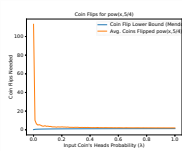
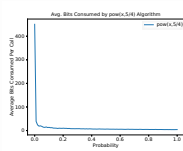
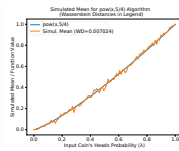
$\text{pow}(x, 4/5)$



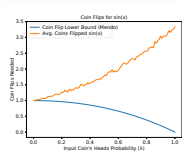
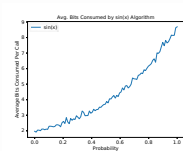
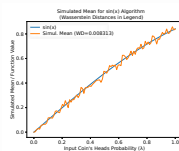
pow(x,5/1)



pow(x,5/4)



sin(x)



sqrt(x)

